

The European Conference on Mobile Robotics Paris, France 2017

September 6-8 2017

Proceedings

http://ecmr2017.ensta-paristech.fr





A Low-Cost Mobile Infrastructure for Compact Aerial Robots Under Supervision

Marc Lieser Henning Tjaden Robert Brylka Lasse Löffler Ulrich Schwanecke

***Abstract*— The availability of affordable Micro Aerial Vehicles (MAVs) opens up a whole new field of civil applications. We present an *Infrastructure for Compact Aerial Robots Under Supervision* (*ICARUS*) that realizes a scalable low-cost testbed for research in the area of MAVs starting at about $100. It combines hardware and software for tracking and computer-based control of multiple quadrotors. In combination with the usage of lightweight miniature off-the-shelf quadrotors our system provides a testbed that virtually can be used anywhere without the need of elaborate safety measures. We give an overview of the entire system, provide some implementation details as well as an evaluation and depict different applications based on our infrastructure such as an Unmanned Ground Vehicle (UGV) which in cooperation with a MAV can be utilized in Search and Rescue (SAR) operations and a multi-user interaction scenario with several MAVs.**

I. Introduction

Over the last years the quadrotor—a multirotor helicopter with four rotors—has emerged as the standard research platform for Micro Aerial Vehicles (MAVs) due to its affordability and mechanical simplicity. MAVs are utilized in numerous fields, e.g., first response, reconnaissance and surveillance, transportation and logistics, inspection of industrial facilities, measuring, construction and aerial photography or videography. In recent years, a lot of impressive research results have been published, such as quadrotors performing quite sophisticated feats with balls [1], [2], balancing inverted pendulums [3] and passing them to other quadrotors [4]. Further examples include quadrotors carrying out aggressive acrobatic maneuvers [5] as well as assembling structures with small construction elements [6].

The main prerequisite for autonomously controlling a quadrotor is the determination of its position and orientation in space, its pose. Outdoors, it can be determined based on information obtained from Global Positioning System (GPS) and on-board magnetometer and inertial measurement units. Indoors, i.e., in GPS-denied environments such as collapsed buildings, the pose has to be estimated using other sensors and algorithms. The most common one is an on-board camera in conjunction with Simultaneous Localization and Mapping (SLAM) algorithms as described in [7].

Research in the field of MAVs requires controlled indoor test environments, which at least consist of an (optical) outside-in pose estimation component as well as a radio control system to pilot the individual MAVs. Latest environments like the Flying Machine Arena (FMA) [8] or the General Robotics, Automation, Sensing & Perception (GRASP) testbed [9] are well-equipped with costly technology. Most test environments use multi-camera motion capturing systems, such as Vicon or OptiTrack for pose estimation, which settle in the six-figure US dollar range. A common used quadrotor is the Ascending Technologies Hummingbird, an MAV in the four-figure US dollar range of about $500\,\mathrm{g}$ weight.

The great expense of the aforementioned testbeds prevent them to become more widespread. But nowadays convenient hobby quadrotor platforms start at $10. Due to the size and weight of the latest consumer miniature quadrotors, elaborate security precautions which exceed the installation of fly screen are not necessary since these lightweight machines will not cause considerable damage to humans or components of the infrastructure. The hobby platforms are by far not as well equipped as professional quadrotors or even freely programmable. Therefore control tasks can not be accomplished directly on the MAVs but have to be done on the computer that remote controls them.

With *ICARUS*, we present an infrastructure for general research and application purposes consisting of a lightweight framework and software architecture resorting to affordable off-the-shelf hardware that is easily replicable. In its smallest expansion, our system is very portable and can be used for live demonstrations, e.g., as part of a lecture or exhibition. With *ICARUS*, research in the exciting field of multirotors gets started at overall hardware expenses of about $100 spent on a commercial multirotor platform and a radio control transmitter module, a Raspberry Pi in conjunction with its official camera and a tracking marker.

The following section gives an overview of our system and its components. Section III describes the software architecture. In Section IV the system's accuracy is discussed. Section V presents applications realized with *ICARUS*. Finally, Section VI summarizes our research and proposes future work.

II. System Components

An architectural overview of *ICARUS* is depicted in Fig. 1. As the main purpose of the presented system is to facilitate the realization of a low-cost testbed, we consequently use low-cost miniature off-the-shelf hobby quadrotors, which were slightly modified by attaching tracking markers. The quadrotors are tracked by an infrared-based monocular tracking system using at least one monochromatic camera. Thereby, the flight area can easily be expanded using multiple cameras. The quadrotors are piloted by a consumer notebook

The authors (`givenname.surname@cvmr.info`) are with the Computer Vision & Mixed Reality Group of RheinMain University of Applied Sciences, Wiesbaden, Germany, which supported this work by internal research funds.

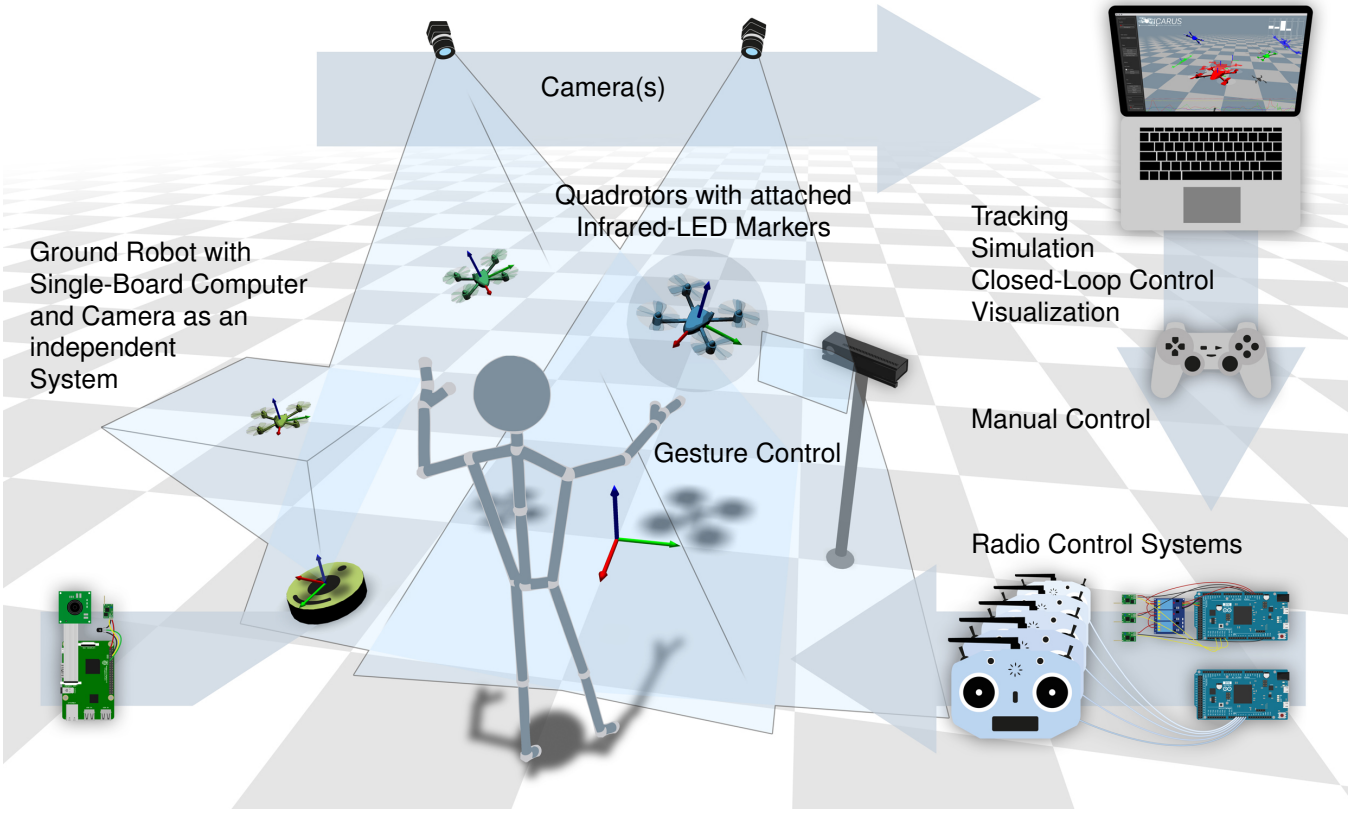


Fig. 1: Overview of *ICARUS*: After pose estimation of the individual quadrotors based on optical tracking the control variables are determined and sent via different radio control systems to the quadrotors.

connected to different radio control approaches. Override and manual control is enabled using a gamepad. The minimum setup can be realized for under $100 and therefore is affordable even for individual researchers and students. The complete system also runs independently on a ground robot as described later in Section V-A.

A. Optical Tracking

The main demands on the tracking of quadrotors are accuracy, frequency, robustness to rapid movement and the capability to distinguish multiple targets. In our system we use the High-Speed and Robust Monocular (HSRM)-Tracking system proposed in [10], a monocular outside-in tracking algorithm based on active infrared LED markers, which sufficiently meets all these demands while being very lightweight compared to multi-camera motion capturing systems used in other testbeds. One of the main advantages of this algorithm is, that it does not exploit any frame-to-frame strategies, guaranteeing prompt relocalization after temporary tracking losses, which is crucial in a MAV environment. Our prototype markers weigh about $4\,\mathrm{g}$, keeping the interference of the flight characteristics to a minimum.

Employing a monocular tracking algorithm, we can use one or multiple cameras, allowing us to easily scale the radius of action. The registration of multiple cameras to a common coordinate frame can be done by simply using one of the markers. Since each camera can estimate its relative pose to the marker, the relation between multiple cameras is instantly known whenever they see the same marker at the same time. The registration process determines the relative transformation between two cameras by minimizing the least mean squared error as proposed in [11] over a sequence of simultaneously estimated marker poses. Thus arbitrary multi-camera setups can quickly be calibrated incrementally when each camera's field of view overlaps with the field of view of at least one other camera. We fuse the measurements of all cameras over a parameterizable time window by linearly blending the marker poses in dual quaternion representation as described in [12]. Due to the monocular fashion of the tracking algorithm, multiple cameras with different frame rates can be utilized within the same setup.

Since the implementation of HSRM-Tracking is single-threaded, multi-camera setups can also easily be parallelized by simply processing each camera in a dedicated thread. Current quadcore consumer laptops could therefore process four high-speed cameras in parallel at full frame rate. Our most sophisticated implementation uses four high-speed Ximea MQ013MG-ON USB3 cameras in conjunction with Fujinon 1:1.2/6 mm DF6HA-1B lenses, operating at $150\,\mathrm{Hz}$ with 1280×1024 pixels resolution filming downwards from above the flight area. To be less dependent on ambient lighting, all cameras are equipped with infrared pass filters.

B. Quadrotor Control

We demonstrate the capability of our system using low-cost off-the-shelf miniature quadrotors from the hobby area. The only modification consists in the attachment of the aforementioned active tracking markers and connecting them to the quadrotors' power supply. All quadrotors possess on-board microcontrollers realizing an inner control loop to stabilize their horizontal position. Our system currently uses two approaches for the outer control loop: A simple hover controller which is tuned to maximize stiffness and a trajectory controller, both using a typical feedback control realized by multiple PID controllers. In section IV, we demonstrate that this simple control strategy leads to satisfactory results despite our tracking system refraining from using any prediction or estimation algorithms. Of course, these results can be further improved by the implementation of statistical filtering, such as an extended Kalman filter.

To apply the general PID control approach to our infrastructure, the terms of the PID controller have to be chosen and their gains have to be tuned. PID control is a compromise of different purposes and dependent on the demands to the system's dynamics. Generally simulations are helpful to test PID gains before applying them to the real system, thus reactions of the system to influences like changing the set point can be judged. All PID gains were determined experimentally.

In **hover control**, four independent PID controllers are used to approach the target pose. Thereby, each of the quadrotors input channels namely throttle, roll, pitch, and yaw are controlled the same way a human would do using a radio control system.

For the *throttle* control all terms of the PID controller are used. The height error is fed into the throttle controller. Next to the usual steady state error, the integral term also deals with the throttle needed to overcome gravity and compensates for the battery's dropping voltage. Relying on the integral term instead of a feed-forward term may result in inertia but yielded good experimental results. To avoid integrator windup, our system uses the conditional integration approach described in [13].

Due to our system's limitation of the quadrotors operating at a near hover state, the control errors for *roll* and *pitch* are calculated by projecting the quadrotors' positions onto the ground plane. For the quadrotor control to be independent of its yaw orientation the set point is transformed into the quadrotors coordinate system. Although a proportional and differential action would be sufficient for both the controllers, a low integral action was added to reduce the drift of hobby quadrotors, which would be leveled out by a hobby pilot by trimming the radio control.

Since our implementation is based on quaternions, the *yaw* control error can simply be calculated by multiplying the target orientation by the inverse of the quadrotor's heading and extracting the yaw Euler angle as the error fed into the control equation. As for the yaw controller a pure proportional controller proved to be effective.

The **trajectory controller** currently uses the approach proposed in [14], where next to the throttle and yaw PID controller described above two further PID controllers minimize errors along the path segment of the trajectory and orthogonal to it.

C. Radio Control Systems

In a regular handheld radio control, potentiometers pick up the positions of the levers and switches and convert them into electrical signals which are modulated to a high-frequency signal and transmitted to the on-board receiver of the quadrotor. After demodulation, the signal is converted into a proportional electrical signal controlling actuators such as electronic speed controllers or servos.

We substitute the handheld radio control system by a microcontroller connected to a computer. This microcontroller generates the required electrical signals to drive a transmitter based on control signals received from the computer. Thereby, the transmitter can be either a bare transmitter module (Arduino Triple Transmission (ATT) approach) or a handheld radio control connected to the microcontroller by its trainer port (Arduino Trainer Port (ATP) approach). Both approaches are open sourced at GitHub[1] and provide cheap solutions to computer-control multirotors or planes from the hobby area. The currently used hardware uses Spektrum DSMX modulation–which is widespread in the model flight community–and can be exchanged for devices using other hobby standards such as Graupner HoTT or FrSky ACCST. In addition to these approaches, *ICARUS* also supports the Bitcraze Crazyradio. All systems are enabled to be flown simultaneously. In the following we describe the implementation of both approaches based on the open source Arduino platform.

1) Arduino Triple Transmission: The ATT approach directly drives multiple transmitter modules removed from Spektrum budget radio controls via serial communication. As microcontroller we use an Arduino Due, which operates on $3.3\,\mathrm{V}$ matching the operating voltage of the transmitting units. Beside the USB interface it has three additional hardware serial pins allowing the control of up to three transmitter modules. To bind a module to a quadrotor's receiver the binding signal has to be applied before switching on the transmitter module. The required current of $30\,\mathrm{mA}$ cannot be supplied by the Arduino's pins, so some additional circuit based on a transistor or relay is required to switch the units on and off.

2) Arduino Trainer Port: The ATP approach employs the pulse-position modulated (PPM) trainer port signal and thus can be used with any radio control system equipped with a trainer port. In combination with an open source radio control, where manufacturer-dependent transmitters can simply be exchanged, there is no limitation to a particular radio protocol. As our system should be cost-efficient, we aim to drive as many control units as possible with a single microcontroller.

[1]Repository links listed at `cvmr.info/icarus/#opensource`.

As can be seen in Fig. 2, the frame of a typical trainer port signal takes 22 ms and encodes six channels: throttle, roll, pitch, yaw, and two additional channels (e.g., for switching flight modes). The length of a channel's high level signal depends on the according lever's position and takes 700 µs to 1530 µs. With a separation of each channel by a 400 µs low level signal, encoding all channels requires a period of 7000 µs to 11 980 µs. The rest of the frame is filled with a high level signal.

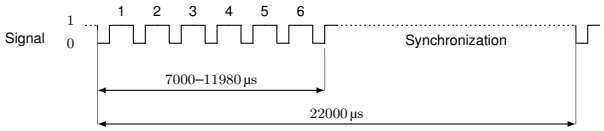


Fig. 2: Characteristic of a six channel trainer port signal.

A total number of 12 Bytes (2 Bytes per channel) are needed to encode control information within the signal. Limited by the maximum buffer size of 64 Bytes of the serial interface, up to five control units can be driven simultaneously by a single Arduino Due. A pulse-position modulated signal is typically created by using a sleep system call. Since these calls are implemented in a synchronized fashion, they cannot be used to create multiple signals in parallel on a single core architecture like Arduino. Our solution discretizes all five signals with a precision of one microsecond, in which within the synchronization phase a byte pattern mirroring the current signal states is created and then transferred to the Arduino's port registers at once. Due to the time needed for processing, we can drive all control units with a temporal accuracy of ±2 ms, which is sufficient for all control purposes.

III. Software Architecture

Our infrastructure is based on the libraries developed in [15] and was distributed and extended by the integration of an existing simulation environment in [16]. To achieve a maximum flexibility, the main demands of the *ICARUS* software architecture are scalability and platform independence in a distributed system. The developed software architecture connecting the parts of the infrastructure mentioned in the last section is displayed in Fig. 3 and uses a publish-subscribe messaging pattern to gain the desired flexibility. By providing loose coupling and a dynamic network topology the publish-subscribe pattern meets all requirements to the infrastructure. The following section describes the individual components of the software architecture.

The `ICARUS-COP` (Control Panel) is a web application which summarizes all important controls the user needs for interaction with the infrastructure. It enables the user to toggle radios and bind quadrotors and to execute predefined series of `Tasks` or to script new ones. The quadrotor's PID gains can be tuned with this interface as well. The origin of the reference coordinate system can be redefined by the current pose of a quadrotor. Beyond that, the user can manually control multirotors using a gamepad. The input generates `Commands` which then are sent to the corresponding radio control system. When using the `ICARUS-COP` on a mobile device its gyroscopes and accelerometers—if available—can be utilized for a more abstract control. The `Abstract Control` interface allows the user to pilot quadrotors by natural interaction. Currently this interface is implemented as gesture-based control and generates `Tasks` based on detected gestures. This interface is easily exchangeable for other interaction paradigms, e.g., a speech recognition component. The `Visualization` is a local OpenGL application providing visual feedback for the user by displaying the quadrotors' poses, their target poses, the control errors fed to the PID control as well as the camera poses, their frustums and images.

The `Flight Control` maintains synchronized task queues for each quadrotor. The PID control is encapsulated in a variety of yet implemented tasks, e.g., the hover task, which uses the tracking-determined pose of a quadrotor and calculates the control commands based on the PID control loop. The `Commands` then are sent to the radio bound to the according quadrotor. Beyond that, the `Flight Control` supervises each quadrotor and initiates emergency landing tasks in case a quadrotor leaves the radius of action.

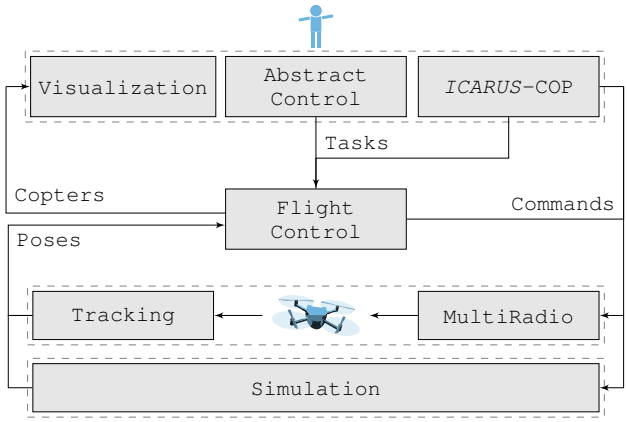


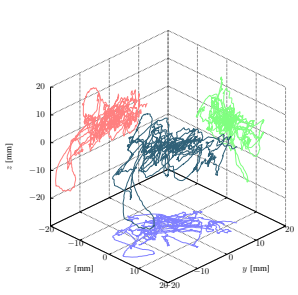
Fig. 3: The software architecture of *ICARUS*.

Due to the modularity in software architecture, components of the system can easily be replaced, e.g., pose estimation and radio control can be exchanged for a `Simulation` environment currently implemented by the Gazebo robotics simulator. The PID gains can be determined using the simulation, before applying them to the real quadrotors to prevent crashes. Using the same interfaces, simulated and real quadrotors can fly together in a mixed environment. Since the space of an outside-in controlled infrastructure is limited, the simulation environment additionally enables larger experiments, e.g., the generation of swarm scenarios which exceed the number of available quadrotors.

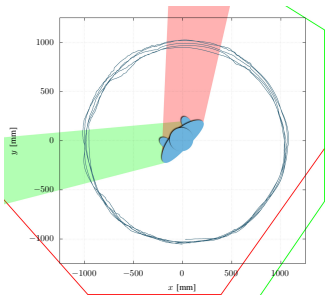
IV. Evaluation

This section evaluates the accuracy of our infrastructure in different flight modes. Videos of the experiments can be

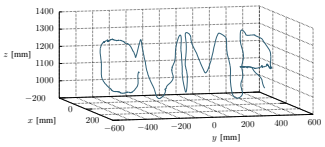
found online[2]. Evaluation periods of these experiments start with the arrival at the first waypoint and end with the arrival at the last waypoint. The poses provided by our tracking system were smoothed over a time window of 66.6 ms, which at a frame rate of 150 Hz corresponds to 10 poses in the single-camera-setup and 20 poses in the two-camera-setup. The precision of HSRM-Tracking (see [10] for details) has to be considered in the evaluation.



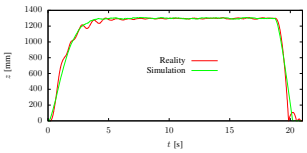
(a) 3D plot and 2D projections of the quadrotor's error in position during a 60 s hover flight.



(b) Flight of five circles with an obstacle at the center. The red and green lines depict the camera frustums with the red and green areas not visible for the according camera.



(c) Flight path following a prescribed trajectory.



(d) Comparison of a real quadrotor and its simplified model simulated by Gazebo.

Fig. 4: Results of different experiments.

A. Hovering

A flight in which the quadrotor was instructed to hover at the target height of 1300 mm for 60 s was carried out in a single camera environment. The target position of the quadrotor corresponded to a distance of approximately 1200 mm to the camera. The position error of the hover controller can be seen in Fig. 4a. The Root-Mean-Square Error (RMSE) of this experiment is 7.6 mm, the mean and standard deviations are $\bar{\mathbf{t}} = (0.6 \pm 4.4, -0.2 \pm 5.9, -2.3 \pm 4.1)$, and the maximum deviations are $\mathbf{d}_{\max} = (12.6, 18.3, 20.8)$.

B. Circular Trajectory

Using the trajectory controller, in this experiment a quadrotor was instructed to fly five circles with a diameter of 2000 mm at a speed of $0.5\,\mathrm{m\,s^{-1}}$. The distance between the waypoints was 25 mm. To be able to monitor this area, a two-camera-setup was used. To demonstrate the monocular fashion of our tracking algorithm, the quadrotor orbited a person occluding different areas of each camera's field of view, as can be seen in the long-exposure photography in Fig. 5c. In this experiment, the quadrotor flew with a mean velocity of $0.47\,\mathrm{m\,s^{-1}}$. The RMSE is 33.8 mm. The plot can be seen in Fig. 4b.

[2]Videos available at `cvmr.info/icarus/#videos`.

C. Skywriting

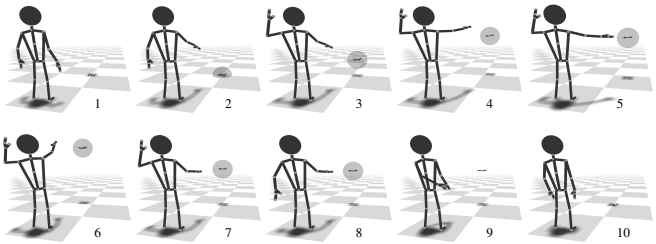
In this experiment, a quadrotor followed the trajectory of the letters CVMR with a total width of 1000 mm and a height of 292 mm using the hover controller. The baseline of the lettering was at a height of 1000 mm. A 3D plot of the quadrotor's trajectory is shown in Fig. 4c. The RMSE of this experiment is 22.2 mm.

D. Simulation

To compare a simulated and a real quadrotor against each other, a 15 s hover flight at the target height of 1300 mm was carried out in a two camera setup with both quadrotors using the same PID gains. The comparison of the heights depicted in Fig. 4d shows high consistency between simulation and reality.

V. Applications

As proof of concept, this section describes three applications based on *ICARUS*. These contain a quadrotor controlled by a ground robot, gesture-based human interaction with quadrotors and light paintings created by the quadrotors' trajectories.



(a) Gesture-based interaction: A user selects a quadrotor by pointing at it with the right arm (2). Lifting the left arm instructs the quadrotor to take off (3). While the left arm is up, the quadrotor imitates the motion of the user's right hand (4–7). When lowering the left arm, the quadrotor stays in hover mode and may be selected by another user (8). Bringing both hands together (9) lets the quadrotor perform a landing (10).



(b) MAV tracked and controlled by a mobile ground robot.



(c) Light paintings of the flights evaluated in Section IV.

Fig. 5: Different applications based on the presented infrastructure.

A. Flying Periscope

Our experimental Search and Rescue (SAR) application is based on a mobile ground platform and molds a flying periscope by collaborating with a quadrotor similar to the system proposed in [17]. A quadrotor controlled by the ground robot can be seen in Fig. 5b. Especially in the field of SAR operations the limited field of view of on-board sensors and cameras mounted on an Unmanned Ground Vehicle (UGV) can easily be expanded by collaborating with MAVs, which can investigate the surrounding area from a higher point of view. Furthermore the extended mobility of an MAV can be exploited. On the other hand, the UGV can serve the MAV as power supply platform to recharge its battery. For this project, we employed the whole infrastructure including HSRM-Tracking on the Raspberry Pi 3 using the Raspberry Pi camera module, operating at $50\,\mathrm{Hz}$ with 1280×720 pixels resolution.

B. Multi-User Interaction with MAVs

Since operating control levers is not very intuitive for the user, we extended *ICARUS* by a multi-user environment for gesture-based interaction and control of multiple quadrotors using a Microsoft Kinect. A typical user interaction sequence is shown in Fig. 5a.

C. Light Painting

The long-exposure photographies of the evaluation flights described in Section IV can be seen in Fig. 5c. This photographic technique is called light painting and provides a good intuition for the accuracy of our control system.

VI. Conclusion and Future Work

In this paper, we presented *ICARUS*, a least cost and portable infrastructure solely built on off-the-shelf hardware for the navigation and control of multirotors, especially of low-cost miniature quadrotors. We evaluated the components of the system with regard to pose accuracy and ported the infrastructure to a common mobile ground platform.

Future work regarding that mobile platform includes the integration into our main testbed to provide our quadrotors a landing platform equipped with inductive charging units in order to increase the radius of operation. Furthermore it is planned to extend our gesture-based control by exploring new human-robot-interaction paradigms such as tactile control of quadrotors as proposed in [18]. To enhance our infrastructure, we are currently working on the generation of feasible smooth trajectories and a model-predictive control approach to reduce the jerk of the quadrotors as described in [19]. We plan to make our implementation of the trajectory generation, quadrotor dynamics and control publicly available to the community by integrating it into ROS [20]. After that, it is our goal to escape the controlled indoor environment we are currently restricted to due to the use of an outside-in tracking approach. Thus each MAV will be equipped with a camera allowing for visual SLAM (e.g., ORB-SLAM proposed in [21], available in ROS), to be employed in order to obtain full autonomy.

Acknowledgement

The authors would like to thank Daniel Andres Lopez for his contribution to the Flying Periscope and Annalena Gutheil for implementing gesture-based interaction.

References

[1] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 5113–5120.

[2] R. Ritz, M. Mueller, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4972–4978.

[3] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Robotics and Automation (ICRA)*, 2011, pp. 763–770.

[4] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3472–3479.

[5] S. Lupashin, A. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[6] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler, "Aerial Robotic Construction: Towards a New Field of Architectural Research," *IJAC*, vol. 10, pp. 439–460, 2012.

[7] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

[8] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, 2014.

[9] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, Sept 2010.

[10] H. Tjaden, F. Stein, E. Schömer, and U. Schwanecke, "High-speed and robust monocular tracking," in *10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, May 2015.

[11] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, Apr 1991.

[12] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Transactions on Graphics*, vol. 27, no. 4, pp. 1–23, 2008.

[13] K. Åström and T. Hägglund, *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society, 2006.

[14] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, pp. 1–14.

[15] M. Lieser, "Outside-in Tracking-basierte Regelung von Multicoptern," Master's Thesis, RheinMain University of Applied Sciences, Wiesbaden, Germany, May 2014.

[16] L. Löffler, "Entwicklung einer Infrastruktur für die Steuerung und Simulation von Multicoptern," Master's Thesis, RheinMain University of Applied Sciences, Wiesbaden, Germany, December 2015.

[17] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared leds," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 907–913.

[18] A. Gomes, C. Rubens, S. Braley, and R. Vertegaal, "Bitdrones: Towards using 3d nanocopter displays as interactive self-levitating programmable matter," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. ACM, 2016, pp. 770–780.

[19] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec 2015.

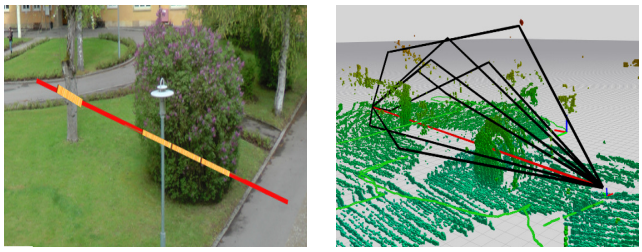
[20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

Outdoor Obstacle Avoidance based on Hybrid Visual Stereo SLAM for an Autonomous Quadrotor MAV

Radouane Ait-Jellal and Andreas Zell

***Abstract*— We address the problem of on-line volumetric map creation of unknown environments and planning of safe trajectories. The sensor used for this purpose is a stereo camera. Our system is designed to work in GPS denied areas. We design a keyframe based hybrid SLAM algorithm which combines feature-based stereo SLAM and direct stereo SLAM. We use it to grow the map while keeping track of the camera pose in the map. The SLAM system builds a sparse map. For the path planning we build a dense volumetric map by computing dense stereo matching at keyframes and inserting the point clouds in an Octomap. The computed disparity maps are reused on the direct tracking refinement step of our hybrid SLAM. Safe trajectories are then estimated using the RRT* algorithm in the SE(3) state space. In our experiments, we show that we can map large environments with hundreds of keyframes. We also conducted autonomous outdoor flights using a quadcopter to validate our approach for obstacle avoidance.**



(a) (b)

Fig. 1. (a) Our outdoor environment. The red line shows the straight line between the start and destination positions going through two obstacles (trees). (b) the 3D reconstruction using our stereo camera and a set of collision-free 3D paths generated by our system.

I. INTRODUCTION

Quadcopters are mobile robots which are suited for many robotic applications. Building inspection, agricultural fields surveillance and package delivery are some applications of quadcopters, just to name a few examples. Unlike fixed wing flying robots, quadcopters can achieve high manoeuvrability about all three axes and in all directions and they have the ability to hold the position (hovering). To achieve full autonomous flights, a quadcopter should rely only on its on-board sensors. Limited by the payload it can carry and by the real-time requirements for safe navigation of a quadcopter, many sensors are not suitable for autonomous quadcopters. Laser scanners which have sufficient frame rates and scan-lines are usually too heavy to be carried by a small quadcopter and they have high power consumption as well. The requirement to work in outdoor environments excludes the choice of using IR pattern based RGBD sensors as main sensors since, with substantial infra-red light from the sun, these sensors fail to estimate the depth images. Stereo cameras can provide data at high frame rates, they are lightweight, passive (do not illuminate the scene), energy efficient and customizable. Given enough data to build dense volumetric maps, the quadcopter can safely navigate in cluttered environments by avoiding obstacles. However, we need to run dense stereo matching on the on-board CPU to estimate the disparity map and use this disparity map to estimate the depth. To allow real time operation, the overhead, which is introduced by the on-board computation of the disparity, can be minimized by computing the disparity only at keyframes and by reducing the resolution. The volumetric mapping of large unknown space requires usage of SLAM (simultaneous localization and mapping) designed to operate at large scale and a memory efficient representation of the dense volumetric map. In this work, we design a hybrid SLAM that works in large scale environments based on the open-source ORB-SLAM2 [1] and for the memory efficient volumetric map we use Octomap [2] [3]. For efficient 3D path planning, we use the RRT* [4] algorithm. The output of the 3D planner is then set as desired way-points for our quadcopter micro aerial vehicle (MAV).

In this work, we propose a system for 3D outdoor obstacle avoidance using stereo. This system was tested on a quadcopter MAV and the quadcopter was able to run all software packages, shown in Fig. 2, in real-time. In addition to the module for 3D obstacle avoidance, we propose a hybrid visual stereo SLAM algorithm which combines a feature-based method and a direct image alignment method. Our choice is motivated by three reasons: first, we want to use more information from the images and not only abstract them to a set of features. Second, since we compute the relatively costly dense stereo matching for the keyframes, we want to benefit from that not only for building volumetric maps but also for refining the tracking poses. Third and finally, in some aspects feature-based methods and direct methods are complementary. So by combining them they compensate for each other's drawbacks. For example, when the stereo camera has moved over large distances between two frames the direct methods might diverge, since they need a good initial guess. In the best case they might need considerably more iterations to converge. This might make real time operation difficult to achieve. Using a coarse-to-fine approach and/or a

R. Ait-Jellal is with the Chair of Cognitive Systems, headed by Prof. A. Zell, Computer Science Department, University of Tübingen, Sand 1, D-72076 Tübingen, Germany {radouane.ait-jellal@uni-tuebingen.de, andreas.zell@uni-tuebingen.de}

motion model might help in some cases. On the other hand, the feature based approach can deal very efficiently with large camera movements. Having to deal with very sparse features, we can afford to enlarge the search domain for correspondences, while still keeping real-time capabilities. The benefit of using direct methods rather than feature based methods is in cases of degraded image quality due to camera defocus and motion blur. In these cases, feature extraction might fail, causing tracking loss while the image alignment would give a reasonably good motion estimation.

II. RELATED WORK

A basic capability that a mobile MAV should fulfill for safe navigation is obstacle avoidance. Heng et al. [5] have proposed a stereo approach for obstacle avoidance. They build an Octomap and use the anytime dynamic A* planner [6] to achieve collision-free path planning in 2D. For pose estimation they use either artificial landmarks or a Vicon tracking system. In our approach, we plan paths in full 3D-space using the efficient RRT* [4] algorithm and we run a hybrid SLAM on-board. The feature based part of our hybrid visual stereo SLAM system uses the popular ORB-SLAM2 [1] algorithm. The ORB-SLAM2 algorithm splits the tracking from the local mapping using two separate CPU threads. This architecture was initially proposed by Klein et al. [7] in their popular parallel tracking and mapping (PTAM) algorithm. The PTAM algorithm was then successfully used by many mobile robotic researchers [8] [9]. The ORB-SLAM2 algorithm starts by extracting a (sparse) set of ORB [10] features on both left and right images of the current stereo frame. The features are then matched on the previous stereo frame to establish 2D-2D correspondences. Using the map we get 2D-3D correspondences. A PnP solver is then used to optimize the initial pose. Using the co-visibility graph (a graph which connects keyframes which share enough map points) a local map is extracted and the pose gets refined by using more map points. The local map contains the keyframes and the map points observed by these local keyframes. The final pose we get from ORB-SLAM2 is estimated using only sparse features. Valuable information might be missed by abstracting the images to a set of ORB features. We propose to further refine the pose by using direct image alignment and use more data from the image. We do not use all of the pixels but only those which have valid depth (from dense stereo matching) and have an image gradient larger than a given threshold. Direct methods for SLAM are becoming popular, and efficient implementations exist [11] [12] [13]. The afore-mentioned implementations use the forward additive or forward compositional [14] variants of the Lucas Kanade [15] [14] algorithm. We use the more efficient inverse compositional alignment algorithm [16], which allows the pre-computation of the Jacobian (and Hessian). Forster et al. [17] introduced sparse direct image alignment where they use the inverse compositional algorithm to align a set of sparse features. They extend their approach for multiple cameras in the recent work [18]. Unlike [17] and [18] we build a hybrid SLAM and not only a visual odometry system.

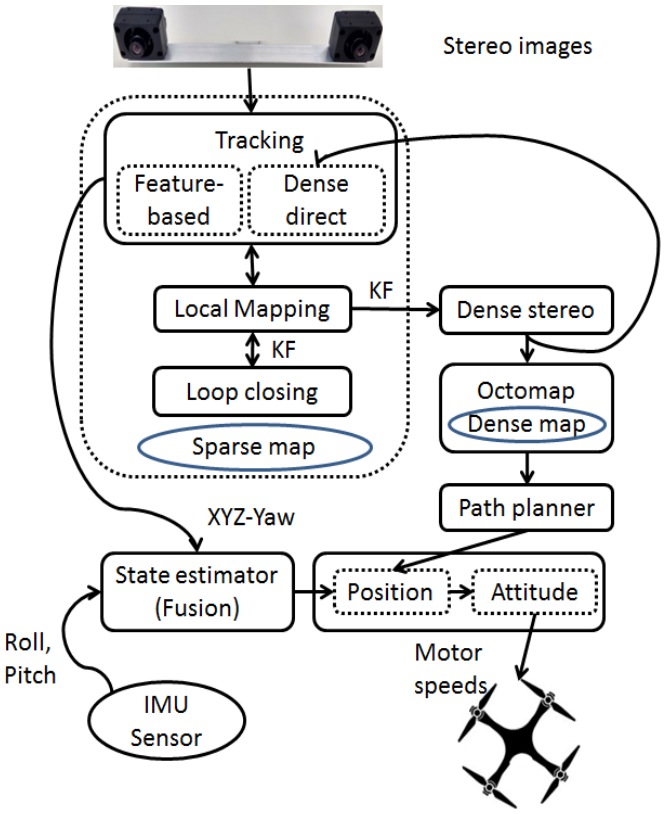


Fig. 2. System overview. The system includes a hybrid SLAM algorithm and a path planner. The tracking thread of the SLAM algorithm provides an estimate of the current pose to the controller. The path planner provides intermediate destinations to the controller. The Pixhawk position controller manages to fly smoothly through the intermediate way-points.

III. OVERVIEW

A. Block diagram of the proposed system

Fig. 2 shows an overview of our system. The hybrid stereo SLAM algorithm is in charge of keeping track of the quadcopter pose in real time while at the same time building a sparse map of the environment. At keyframes we compute dense stereo matching. The resulting disparity map is then used to create a 3D point cloud which is inserted in the occupancy grid map. The disparity map is also used by the tracking thread to refine the poses using direct image alignment. The planner gets an up to date binary occupancy grid map and the start (current) pose and destination pose and then plans a safe path for obstacle avoidance. The planner outputs smooth trajectory of way-points. The poses from the localization thread of the SLAM system are fused with the measurements from the on-board IMU using an Extended Kalman Filter (EKF) to estimate the quadcopter state. The controller gets the desired pose and the current pose and controls the motor speeds to fly to the desired position. The Pixhawk [19] controller is a cascaded controller which includes a high level position controller and a low level

attitude controller. The low level controller outputs the motor speeds.

B. Least squares problems

The least squares (LS) algorithm is used for optimizing non-linear cost functions in many parts of our system. We use least squares in the feature based tracking to optimize re-projection distances. Then, we use least squares in the direct image alignment to minimize photometric errors (intensity differences). On the local mapping thread, least squares are used in local bundle adjustment (LBA) to locally optimize the SLAM map. On the SLAM back-end, least squares are used to optimize a pose graph when loops are detected and afterwards to optimize the whole system (all keyframe poses and all map points) by global bundle adjustment (GBA). In general, given an error function $e(x) = [e_1(x), ..., e_m(x)]^T \in \mathbb{R}^m$, where the variable $x = [x_1, ..., x_n]^T \in \mathbb{R}^n$ is a vector of parameters. We seek to estimate the optimal solution x^* that minimizes the energy function $E(x)$ (Eq. 1).

$$x^* = \underset{x}{\operatorname{argmin}} E(x) \tag{1}$$

$$= \sum_{i=0}^{m} |e_i(x)|^2 \tag{2}$$

$$= e(x)^\intercal e(x) \tag{3}$$

In general, e is non-linear so we compute an approximation using the first order Taylor expansion and solve the optimization problem iteratively. Around the initial guess $\breve{x}$ we consider a small perturbation δx :
$e(\breve{x} + \delta x) \approx e(\breve{x}) + J\delta x$, here J is the Jacobian of $e(x)$ computed in $\delta x = 0$ (in $x = \breve{x}$). The elements E_i of the energy function $E(x)$ can then be approximated such that:

$$E(\breve{x} + \delta x) = e(\breve{x} + \delta x)^T e(\breve{x} + \delta x) \tag{4}$$

$$\approx (e(\breve{x}) + J\delta x)^T (e(\breve{x}) + J\delta x) \tag{5}$$

$$= E(\breve{x}) + 2e(\breve{x})^\intercal J\delta x + \delta x^\intercal J^\intercal J\delta x \tag{6}$$

$$= E(\breve{x}) + 2e(\breve{x})^\intercal J\delta x + \delta x^\intercal H\delta x \tag{7}$$

Where $H = J^\intercal J$ is the approximate Hessian. By computing the derivative with respect to δx and setting it to zero we can compute the increment δx^* which minimizes Eq. 7 and solves the following linear system (Eq. 8):

$$H\delta x^* = -e(\breve{x})^\intercal J \tag{8}$$

This linear system can be written as:

$$A\mathrm{x} = b \tag{9}$$

where: $A = H$, $b = -e(\breve{x})^\intercal J$ and $\mathrm{x} = \delta x^*$. In our implementation we use Cholesky factorization to solve the linear system in Eq. 8. The increment δx^* is then added to the initial guess $\breve{x}$.

$$x^* = \breve{x} + \delta x^* \tag{10}$$

For the inverse compositional direct alignment step of our algorithm, the Hessian and its inverse can be pre-computed (for all iterations). Thus, solving the linear system in Eq. 8 becomes trivial. The parameter update is also different since it involves inverted composition rather than forward additive increments (See Eq. 26).
The Gauss Newton solver iterates the following steps: first, it locally approximates the non-linear cost function with a linear function according to Eq. 5, then it computes in closed form the increment δx^* according to Eq. 8, and finally it updates the parameter vector according to Eq. 10. While the intermediate problem (linear approximation) is solved in closed form, the initial non-linear problem in Eq. 1 needs to be iteratively solved. In every iteration, we use the current updated parameter vector as an initial guess (linearization point). The iterative process continue until some termination criteria are met, e.g., when the algorithm converges or we reach a maximum number of iterations.

IV. The Hybrid SLAM system

We propose a hybrid visual stereo SLAM algorithm which combines a feature-based method and a direct image alignment method. As discussed on the introduction, feature-based approaches are generally fast and can handle large camera movements. We make use of this characteristic and design a hybrid visual stereo SLAM which uses the motion estimation from a feature-based visual SLAM as an initial guess for a direct image alignment method refinement step. Our hybrid stereo SLAM is based on the popular ORB-SLAM2 algorithm [1]. We modify the tracking thread such that we refine the poses using direct image alignment and we modify the mapping thread such that we compute dense binocular stereo matching at keyframes.

A. Notation

We use the following notation in this work:

- $\{I^l_{cur}, I^r_{cur}\}$ and $\{I^l_{ref}, I^r_{ref}\}$: the left and right images of the current frame and reference keyframe respectively.
- $x = (u, v)$: pixel with the coordinates (u, v).
- $p = (X, Y, Z)$: point in 3D space corresponding to the image pixel $x = (u, v)$.
- $T \in SE(3)$: a 3D rigid body transform. $T = \{R, t\}$ where $R \in SO(3)$ and $t \in \mathbb{R}^3$.
- $\xi \in se(3)$: minimal parametrization of the 3D rigid body transform in the Lie algebra.
- T_f: transform estimated by the feature-based approach.
- T_d: transform estimated by the direct image alignment approach using the inverse compositional algorithm.
- $W(x, \xi)$: 3D warp which maps pixels from the reference keyframe to the current frame.
- ∇I: image intensity gradients (Jacobians).
- π, π^{-1}: pinhole camera projection model and its inverse.
- f_x, f_y, c_x, c_y: camera intrinsic parameters.
- ρ: Huber robust cost function.
- e: error function.
- J and H: Jacobian and Hessian.
- $\breve{x}$: initial guess for the parameter x which we use to initialize the optimization.

- x^*: optimal value of the parameter x, which we get after the optimization.

B. Lie algebra parametrization for motion estimation

A rigid body transform g transforms as point $p \in \mathbb{R}^3$ in the 3D space to a point $g(p) \in \mathbb{R}^3$ in 3D space:

$$\begin{aligned} g &: \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ p &\rightarrow g(x) \end{aligned} \tag{11}$$

The rigid body transform g can be expressed by a 4×4 matrix $T \in SE(3)$, which is composed by a rotation matrix $R \in SO(3)$ and a translation vector $t = (t_x, t_y, t_z) \in \mathbb{R}^3$.

$$g(p) = g(p, T) = T.p = Rp + t \tag{12}$$

where

$$T = \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{13}$$

and

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{14}$$

We use the Lie group $SE(3)$ and its corresponding Lie algebra $se(3)$ to get a minimal parametrization of the 3D rigid body transforms. On the associated Lie algebra $se(3)$, the corresponding transform is a 6-dimentional vector $\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$. These coordinates are called the twist coordinates, where the first (ξ_1, ξ_2, ξ_3) are the linear velocities and (ξ_4, ξ_5, ξ_6) are the angular velocities. Here, we represent the rotation part of the transform with exactly three parameters (ξ_4, ξ_5, ξ_6) rather than nine parameters as in the rotation matrices representation. The use of the Lie group solves the singularities of the compact Euler-angles representation.
A rigid body transform T can be mapped to its corresponding ξ using the logarithmic map as follow:

$$\begin{aligned} log &: SE(3) \rightarrow se(3) \\ g &\rightarrow \xi = log(T) \end{aligned} \tag{15}$$

The inverse of this operation is the exponential map.

$$\begin{aligned} exp &: se(3) \rightarrow SE(3) \\ \xi &\rightarrow T = T(\xi) = exp(\xi) \end{aligned} \tag{16}$$

C. Tracking: Minimizing distances and intensity differences

Our system uses ORB-SLAM2 to estimate ξ_f^*, the pose of the current frame based on features. Any other feature-based stereo SLAM can be used. We compute T_f by minimizing the re-projection error (see Eq. 18) between predicted pixel locations and the observed pixels locations (see Eq. 17).

$$e_f(x_i) = x_i - \pi(RX_i + t) \tag{17}$$

$$T_f^* = \underset{T_f}{\operatorname{argmin}} \sum_{i \in \aleph} \rho\left(\|x_i - \pi(RX_i + t)\|_{\Omega_i}\right) \tag{18}$$

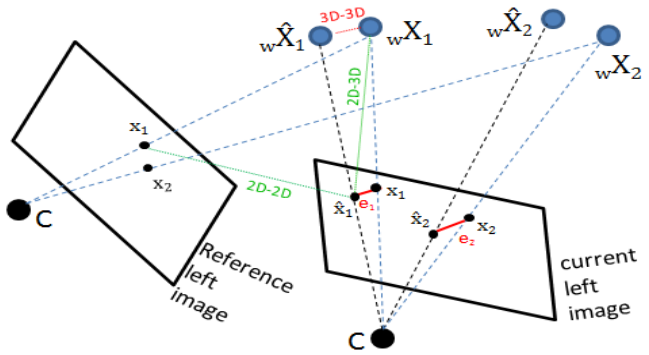


Fig. 3. Feature based motion estimation: we try to find the transform (pose of the current camera) for which the sum of all re-projection distances e_i (shown in red, the unit is Pixel) is minimal. Note that in a 3D-3D correspondence approach, distances (in Meter unit) between 3D points. Due to triangulation inaccuracy, these methods are generally less accurate than approaches based on 2D-3D correspondences.

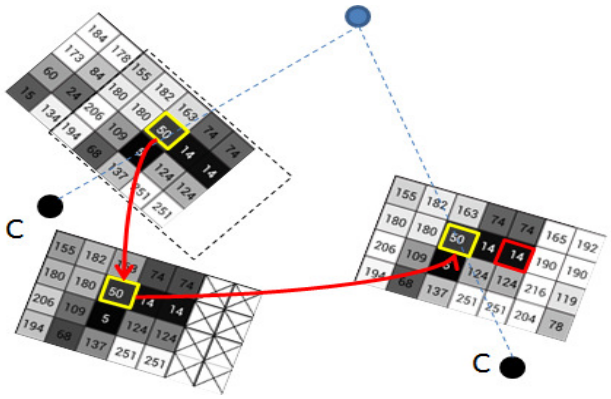


Fig. 4. Direct image alignment motion estimation: illustration of the inverse compositional algorithm. Here, we minimize the photometric errors. The color (gray values) of the squares encode the intensity value [0-255]. The search for the warp increment (image bottom left) is done in the reference image (top left) and then this warp is inverted and composed (Eq. 26) with the current warp of the target image (right image). The result, is a direct warp from the template image to the target image.

where Ω_i is a weighting (inverse covariance matrix) associated to the scale at which the feature was extracted. And ρ is the Huber robust cost function. We note that we minimize distances (in pixels) on the image plane. We illustrate this case in Fig 3. The final pose ξ_f^* we get with the feature based motion estimation is used as initial guess for our direct image alignment motion refinement. This is described in the following equation (Eq. 19):

$$\breve{\xi}_d = \xi_f^* \tag{19}$$

The 3D warp $W(x_i, \xi_d)$ which maps a pixel $x_i \in I_{ref}^l$ in the reference keyframe into its corresponding pixel in the image I_{cur}^l using the pinhole camera projection model π is given by:

$$\begin{aligned} W &: \mathbb{R}^2 \times \mathbb{R}^6 \rightarrow \mathbb{R}^2 \\ (x, \xi_d) &\rightarrow W(x, \xi_d) = \pi(\pi^{-1}(x, Z), g(T(\xi_d))) \end{aligned} \tag{20}$$

where

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{21}$$

and its inverse maps a 2D pixel $x = (u, v)$, with known depth Z, to its corresponding 3D point $p = (XYZ)$:

$$\pi^{-1}\left(\begin{bmatrix} u \\ v \\ Z \end{bmatrix}\right) = \begin{bmatrix} \frac{u-c_x}{f_x} Z \\ \frac{v-c_y}{f_y} Z \\ Z \end{bmatrix} \tag{22}$$

We compute T_d by minimizing the re-projection photometric error between the predicted intensities of the pixels and the observed intensities. We illustrate this case in Fig 4. We work with 8-bit gray-scale images with intensities in the interval [0-255]. The error is defined as follow (Eq. 23):

$$e_d(x_i) = I^l_{ref}(W(x_i, \delta\xi_d)) - I^l_{cur}(W(x_i, \xi_d)) \tag{23}$$

We optimize using pixels $x_i \in D_{ref}$ where D_{ref} is the set of pixels $x_i \in I^l_{ref}$ such that x_i has a valid depth and the magnitude of the intensity gradient at x_i is larger than a given threshold. In practice we use about 15% of the image pixels. Note that the domain D_{ref} here is defined in I^l_{ref} for the inverse compositional image alignment. For the forward additive algorithm and the compositional forward algorithm, the domain is defined on I^l_{cur}.
We iteratively optimize the following energy function to align I^l_{cur} with I^l_{ref}:

$$\xi^*_d = \underset{\xi_d}{\operatorname{argmin}} \sum_{x \in D_{ref}} \|I^l_{ref}(W(x_i, \delta\xi_d)) - I^l_{cur}(W(x_i, \xi_d))\|^2 \tag{24}$$

$$= \underset{\xi_d}{\operatorname{argmin}} E_d(\delta\xi_d) \tag{25}$$

Since the increment $\delta\xi_d$ is computed for the reference keyframe image (the template image not the target image) we need to invert it and and compose it with the current parameter estimate. The warp update at iteration $k+1$ is given by Eq. 26.

$$T^{k+1}_d = T^k_d * exp(-\delta\xi^*_d) \tag{26}$$

In the iterative optimization process, we linearize around $\delta\xi = 0$ at every iteration using Eq. 5 applied to the cost function in Eq. 24.

$$E_d \approx \sum_{x \in D_{ref}} \|I^l_{ref}(W(x, 0)) - I^l_{cur}(W(x, \xi_d)) + J_d \delta\xi_d\|^2 \tag{27}$$

The derivation of the Jacobians for our inverse compositional algorithm is similar to the forward compositional algorithm in [20]. However, Jacobians in our case are computed for the I^l_{ref} image and not for I_{cur} as in [20].

$$J_d(x, \xi_d) = \frac{\partial E_d}{\partial \xi_d} \tag{28}$$

The Jacobian can be computed using the chain rule.

$$\begin{aligned} J_d(x, \xi_d) =& J_I J_\pi J_g J_T \qquad (29) \\ =& \frac{\partial I^l_{ref}(W(x, \delta\xi_d))}{\partial \pi}\Big|_{x=\pi(g(p_i, T(0))=x_i} \cdot \\ & \frac{\partial \pi(p)}{\partial g}\Big|_{p=g(p_i,T(0))=p_i} \cdot \\ & \frac{\partial g(p,T)}{\partial T}\Big|_{\substack{T=T(0)=I^l_{cur}, \\ p=p_i}} \cdot \\ & \frac{\partial T(\xi_d)}{\partial \xi_d}\Big|_{\xi_d=0} \qquad (30) \end{aligned}$$

The individual Jacobians in Eq. 28 can be derived as follow. The intensity Jacobian J_I is evaluated at $x = \pi(g(p_i, T(0))$, where $T(0) = I_{4x4}$.

$$\begin{aligned} J_I =& \frac{\partial I^l_{ref}(W(x, \delta\xi_d))}{\partial \pi}\Big|_{x=\pi(g(p_i,T(0))=x_i} \qquad (31) \\ =& (\nabla I^l_{ref,u} \nabla I^l_{ref,v}) \qquad (32) \end{aligned}$$

The camera projection Jacobian is:

$$\begin{aligned} J_\pi =& \frac{\partial \pi(p)}{\partial g}\Big|_{p=g(p_i,T(0))=p_i} \qquad (33) \\ =& \begin{bmatrix} f_x \frac{1}{z} & 0 & -f_x \frac{x}{z^2} \\ 0 & f_y \frac{1}{z} & -f_y \frac{y}{z^2} \end{bmatrix} \qquad (34) \end{aligned}$$

The Jacobian of the function g is:

$$\begin{aligned} J_g =& \frac{\partial g(p,T)}{\partial T}\Big|_{\substack{T=T(0)=I^l_{cur}, \\ p=p_i}} \\ =& \begin{bmatrix} x.I_{3\times3} & y.I_{3\times3} & z.I_{3\times3} & I_{3\times3} \end{bmatrix} \qquad (35) \end{aligned}$$

The Jacobian of T is:

$$\begin{aligned} J_T =& \frac{\partial T(\xi_d)}{\partial \xi_d}\Big|_{\xi_d=0} \qquad (36) \\ =& \begin{bmatrix} & 0 & 0 & 0 \\ 0_{3\times3} & 0 & 0 & 1 \\ & 0 & -1 & 0 \\ & 0 & 0 & -1 \\ 0_{3\times3} & 0 & 0 & 0 \\ & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ 0_{3\times3} & -1 & 0 & 0 \\ & 0 & 0 & 0 \\ \\ I_{3\times3} & & 0_{3\times3} & \end{bmatrix} \qquad (37) \end{aligned}$$

Finally, we get the expression for our per-pixel Jacobian:

$$\begin{aligned} J_d =& J_I J_\pi J_g J_T \qquad (38) \\ =& (\nabla I^l_{ref,u} f_x, \nabla I^l_{ref,v} f_y) \cdot \\ & \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} & -\frac{xy}{z^2} & (1+\frac{x^2}{z^2}) & -\frac{y}{z} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} & -(1+\frac{y^2}{z^2}) & \frac{xy}{z^2} & \frac{x}{z} \end{bmatrix} \qquad (39) \end{aligned}$$

Note that the per-pixel Jacobian J_d is a 6-dimensional vector. By checking the dimensions of the different matrices which

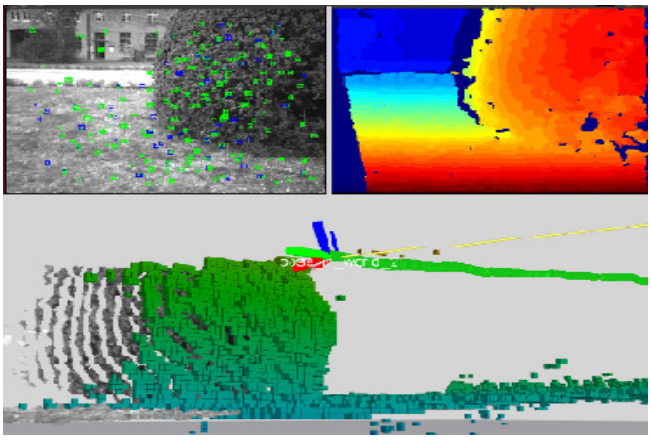


Fig. 5. This figure shows the features tracked on the current frame, the disparity map of the last keyframe (the color encode the disparity value) and a view of the volumetric reconstruction. The current point cloud which overlaps with the map is also drawn.

compose J_d, we get:

$$\underbrace{J_d}_{1\times 6} = \underbrace{J_I}_{1\times 2} \cdot \underbrace{J_\pi}_{2\times 3} \cdot \underbrace{J_g}_{3\times 12} \cdot \underbrace{J_T}_{12\times 6} \tag{40}$$

V. Outdoor 3D Obstacle Avoidance using Stereo

In this section we describe our stereo-based 3D path planning system. A volumetric map (Octomap) is maintained by the system. Stereo dense disparity maps are projected to 3D to create point clouds (Octomap scans). These measurements are then used to update the octree [21]. Collision-free trajectories can then be planned for safe navigation in 3D.

A. Occupancy grid map and dense stereo matching

The map built by ORB-SLAM2 is too sparse to be used for path planning. Thus, we build a dense volumetric occupancy grid map. For real-time purposes we update the occupancy grid map only at keyframes. When the tracking thread of the SLAM system decides to create a new keyframe we also store (with the keyframe) the intensity images (left and right at half resolution). At the mapping thread we estimate the disparity map by using the popular semi-global matching (SGM) dense stereo matching [22]. While there exist more efficient local stereo matching algorithms [23] [24], the need to estimate the disparity maps only at keyframes motivates us to use the globally more accurate stereo algorithm SGM.

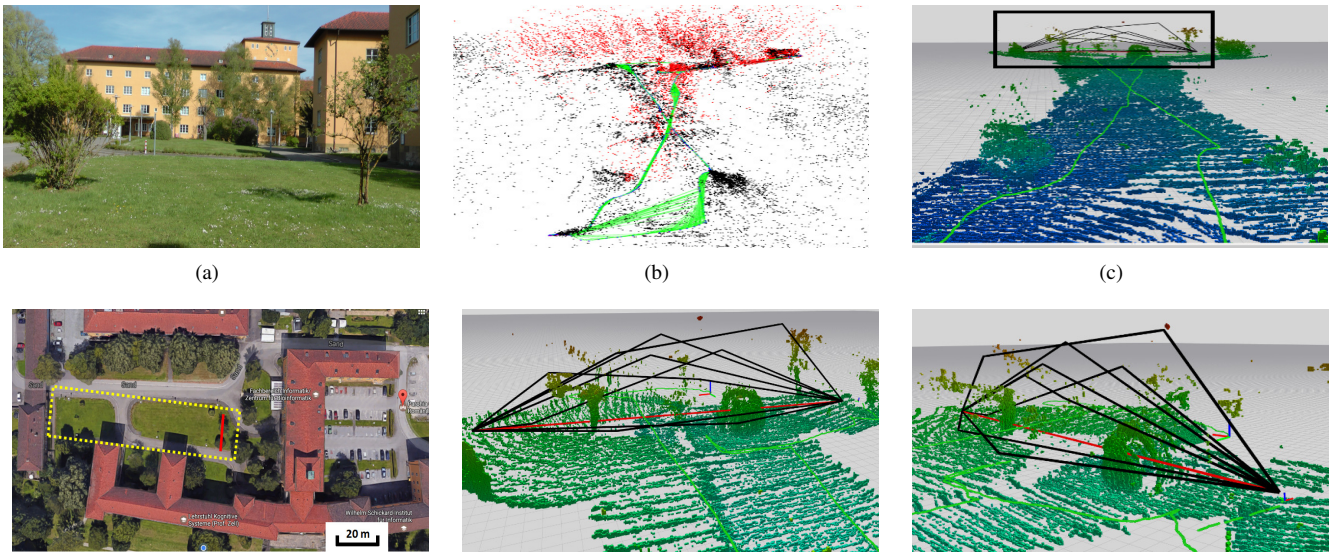
B. Path planning in 3D occupancy grid map

The 3D path planner gets a binary version of the up-to-date octree and updates its bounds using the current octree bounding boxes. The planning in 3D is more challenging than in 2D and efficient methods need to be used. Standard path planners such as A^* are not efficient for usage on-board a quadcopter. We choose to use a variant of the efficient rapidly-exploring random trees RRT [25]. We use the open-source library OMPL, which implements a set of planning algorithms. One of the algorithms used for path planning in this setup is RRT* [26]. It is a variant of the original RRT. In the following the choice of this algorithm is illustrated. RRT is a sampling-based algorithm and provides *probabilistic completeness* [26]. That means that the probability that the planner fails to find a solution, if there is one, goes to zero as the number of samples approaches infinity. Such a sampling-based planner requires a collision checking module as it does not represent obstacles explicitly [26]. A problem of RRT is that it doesn't provide any guarantees to an optimal solution. If we use RRT in this setup we get a valid solution very fast but as stated it is not necessarily optimal. Experiments show that it is very often not even close to optimal. But as we are planning paths for a Quadrotor optimal paths (or at least short paths) are important to not waste battery power. To achieve an optimization of the planned path RRT* is used. Basically the tree is constructed in the same manner as in normal RRT but not all feasible connections will be inserted. Normal RRT just inserts a connection between the new node and its nearest neighbor (considering the euclidean distance). The RRT* algorithm will check all nodes in the surrounding of a new node and only insert the shortest path to the new node, considering a cost function, into the tree. This cost function differs from the euclidean distance, because on the path from the root to the node, which will be connected to the new node, there might be some obstacle that increases the cost in comparison to a straight line connection. Therefore the nearest neighbor of the new node does not have to be on the shortest path to the new node. After that all the surrounding nodes are checked again whether there is a new shortest path to each of them using the newly inserted node. If that is the case the tree gets rewired to maintain it a tree structure. As stated in [26], RRT* is *probabilistically complete*, like the normal RRT, but moreover it is *asymptotically optimal*. That means that the returned solution converges almost surely to the optimal path [26]. This property made the RRT* algorithm a good choice for this path planning scenario. Moreover the algorithm is not limited to finding geometric shortest paths but can also optimize towards a mixed optimization objective taking different costs (e.g. avoiding power extensive maneuvers) into account. In the quadcopter online planning scenario it has to be considered that one has to make a trade-off between spending energy and time flying a non-optimal path vs. spending energy and time hovering too long to compute the optimal path plus flying this path. Once a path is planned the way-points are sent to the Pixhawk controller via a serial connection. The Pixhawk controller takes care of flying the quadcopter to the desired destination. The desired destination includes the position (XYZ) and the yaw angle.

VI. Results

A. Platform description

In our real experiments we used a custom-made quadcopter. It is shown in Fig. 7. The components of the quadcopter are as follows: a stereo camera with a pair of Point-Grey Firefly monochrome cameras with a resolution of 640x480 pixels. The baseline between the two camera is 22cm. The stereo camera delivers synchronized stereo pairs at 30Hz. The flight control is from the open-source project



(a) (b) (c)

(d) (e) (f)

Fig. 6. Thanks to the volumetric global map maintained by our algorithm, we can plan collision-free paths between any two positions on the global map. (a) a view of the outdoor environment in which we did the experiment. (d) a Google Maps view of this area. The yellow rectangle shows the mapped area and the red line inside it shows the straight line between the start and goal position for the planner. (b) the sparse map built by the SLAM algorithm. This map which is used for pose estimation and re-localization. (c) the 3D volumetric map (Octomap) which is used by the planner. The green dots show the robot trajectory. (e) and (f) show two views of a set of 3D paths between the start and destination positions. These views correspond to the black rectangle region of the map in (c).



Fig. 7. Quadcopter used for our outdoor real experiments with the forward looking stereo camera.

#Stereo pairs	9040
#Keyframes	1128
#Map points	30616
Feature based tracking (time)	52 ms
Direct alignment refinement (time)	21 ms
Dense stereo 320 × 240 pixels (time)	55 ms
Dense stereo 640 × 480 pixels (time)	183 ms
Octree update 320 × 240 pixels (time)	62 ms
Octree update 640 × 480 pixels (time)	191 ms
Octree memory (size)	360 MB
Binary Octree (size)	1.1 MB
Planning RRT* (time)	10 s
Average #WPs	86

TABLE I

SOME STATISTICS FROM THE OUTDOOR OBSTACLE AVOIDANCE EXPERIMENT.

Pixhawk [19]. The on-board computer is an Intel NUC mini-pc. It has an Intel core i7-5557U CPU with 2x3.1GHz, 4MB cache, 28 Watts thermal design power (TDP) and 8GB RAM.

B. Obstacle avoidance in outdoor environment

We performed outdoor experiments in an environment with vegetation (grass), trees, walls and asphalt. Some pictures of this environment can be seen in Fig. 1(a), 6(a) and 6(d). Fig. 6 shows the details of an outdoor experiment. It shows the sparse map and the volumetric map. A set of collision-free paths between two points are also shown. Table I shows some statics from the outdoor experiment. The running times are reported for the case where all software packages of our system (see Fig. 2) are running at the same time. The running times for performing dense stereo matching and octree update are given for two different resolutions (640×480 and 320×240). The planning time is set to 10 seconds. WPs means the number of intermediate way-points.

C. Results of the Hybrid SLAM on the Kitti Dataset

We evaluated our hybrid SLAM method on the Kitti odometry benchmark [27]. The Kitti datasets are recorded using a car with a stereo camera mounted on the top of the vehicle. The recorded trajectories have a total length of about 39 Km divided into 22 sequences. Ground truth trajectories are provided for the first 11 sequences. Some sequences include loop closures and dynamic objects (cars

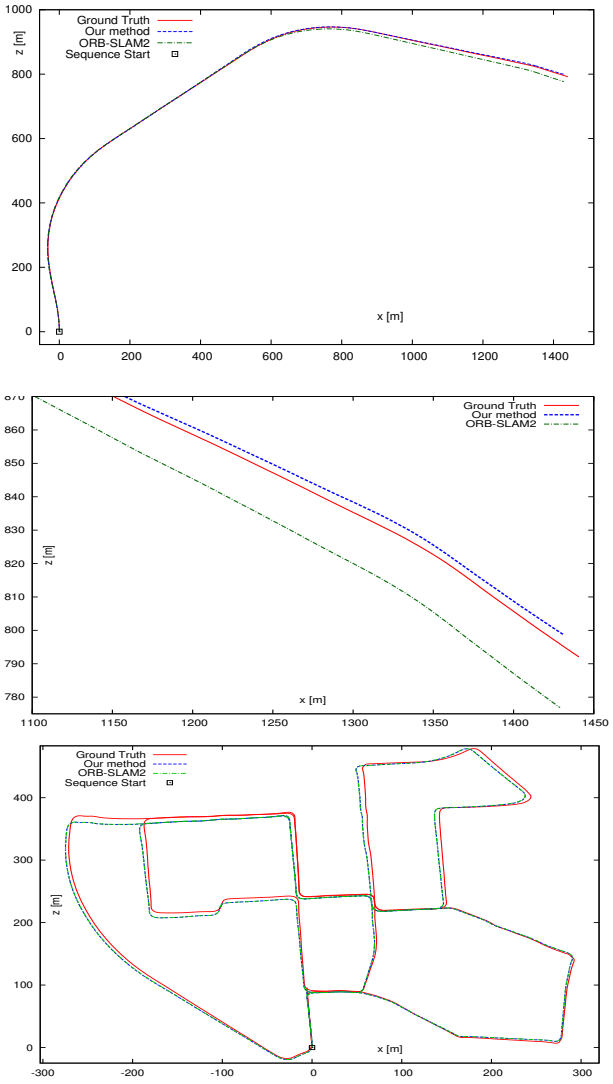


Fig. 8. Results on the Kitti odometry benchmark. Our algorithm (blue) is compared with the ORB-SLAM2 [1] (green). The ground truth is also drawn (red). Top: sequence 12. We achieve a considerable improvement. Our method accumulates less drift. Middle: for a better visualization we zoom in the last part of the trajectories of the previous image. Bottom: sequence 00 which has many loops. No significant improvement. The trajectories almost overlap.

driving around). Our results show that the refinement step (using the direct image alignment) improves the accuracy for sequences (e.g. sequence 12) with no loop closures. Fig. 8 shows the results obtained for the Kitti sequences 12 and 00. For the sequence 12 of the Kitti benchmark which has no loop closure the drift becomes larger with the travelled distance. Our refinement step helped to keep the drift very low. For sequences with many loop closures (e.g. sequence 00), we get almost the same results as the original algorithm. A loop closure detection and correction contribute to reduce the drift for the feature-based SLAM more than it does for our hybrid method. This can be confirmed by disabling the loop closure thread of the SLAM system.

REFERENCES

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[2] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and B. W., "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," *ICRA*, 2010.

[3] K. Schauwecker and A. Zell, "Robust and efficient volumetric occupancy mapping with an application to stereo vision," *ICRA*, pp. 6102–6107, 2014.

[4] "The open motion planning library. ompl.kavrakilab.org."

[5] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," *ICRA*, 2011.

[6] M. Likhachev, D. Ferguson, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," *International Conference on Automated Planning and Scheduling*, 2005.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," *ISMAR*, 2007.

[8] S. A. Scherer and A. Zell, "Efficient Onboard RGBD-SLAM for Fully Autonomous MAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo Big Sight, Japan, November 2013.

[9] S. A. Scherer, D. Dube, and A. Zell, "Using Depth in Visual Simultaneous Localisation and Mapping," in *IEEE International Conference on Robotics and Automation*, St. Paul, Minnesota, USA, May 2012.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf," *International Conference on Computer Vision, ICCV*, 2011.

[11] J. Engel, J. Stueckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[12] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[13] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, July 2016.

[14] B. Simon and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision, IJCV*, 2004.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981.

[16] S. Baker, "Aligning images incrementally backwards. carnegie mellon university. the robotics institute," 2001.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[18] C. Forster, Z. Zichao, G. Michael, W. Manuel, and D. Scaramuzza, "SVO 2.0: Semi-direct visual odometry for monocular and multi-camera systems," in *IEEE Transactions on Robotics*, 2016.

[19] "The pixhawk open-source autopilot project."

[20] C. Kerl, "Master thesis: Odometry from rgb-d cameras for autonomous quadrocopters," Nov. 2012.

[21] D. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer," in *Rensselaer Polytechnic Institute (Technical Report IPL-TR-80-111)*, October 1980.

[22] H. Hirschmueller, "Stereo processing by semi-global matching and mutual information," *IEEE TPAMI*, pp. 328–341, 2008.

[23] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ser. ACCV'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 25–38.

[24] R. Ait Jellal, M. Lange, B. Wassermann, S. Andreas, and A. Zell, "LS-ELAS: line segment based efficient large scale stereo matching," *ICRA*, 2017.

[25] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Segmentation of Depth Images into Objects Based on Local and Global Convexity*

Robert Cupec, *Member, IEEE*, Damir Filko, *Member, IEEE,* and Emmanuel K. Nyarko

***Abstract*— An approach for object detection in depth images based on local and global convexity is presented. The approach consists of three steps: image segmentation into planar patches, greedy planar patch aggregation based on local convexity and segment grouping based on global convexity. The proposed approach improves upon existing similar methods, which use convexity as a cue for object detection, by detecting convex objects represented by multiple spatially separated image regions as well as hollow convex objects. The presented method is experimentally evaluated using a publicly available benchmark dataset and compared to two state-of-the art approaches. The experimental analysis demonstrates improvement achieved by high-level segment grouping based on global convexity.**

I. Introduction

The ability to perceive a scene as a set of objects is very important for an intelligent robot which operates in complex and unstructured environments. Several successful approaches for segmentation of depth images obtained by an RGB-D camera into objects have been proposed [1]–[6]. Many cues are investigated in search for appropriate criteria which could be used to decide whether particular image regions belong to one object or to different objects. Since the probability that two randomly positioned objects form one convex surface is very low, convexity has been used as one of the most reliable cues for separating objects in the scene [4]–[6].

The research presented in this paper investigates the application of global convexity as a powerful cue for object detection in complex and cluttered scenes. The focus is on situations where an object is represented by multiple spatially separated image regions due to occlusion. An example is the box denoted by the yellow rectangle in Fig. 1. Such instances are referred to in this paper as *disconnected objects*. Another interesting situation involves *hollow convex objects* such as glasses, cups, bowls, open boxes and cans etc., which are often represented in the image by one convex and one concave surface separated by depth discontinuity, for example the bowl denoted by the green rectangle in Fig. 1. Methods, such as those presented in [4] and [5], which detect objects as connected convex surfaces result in oversegmentation, i.e. false segmentation in which such objects are represented by multiple convex segments. The method presented in this paper consists of three steps. The first step is segmentation of the input depth image into planar patches. The second step is grouping of planar patches using local convexity criterion, analogously to the approach proposed in [5]. An example of surfaces obtained by this second step is presented in the bottom left image of Fig. 1. It can be seen that some objects are represented by multiple segments. The final third step is high-level grouping of segments based on convexity of their union, which allows for the detection of disconnected objects and hollow convex objects as whole entities, as demonstrated in the bottom right image in Fig. 1. An efficient approach for determining approximate convexity of the union of two surfaces is proposed, which is based on representing the surfaces by planar patches.

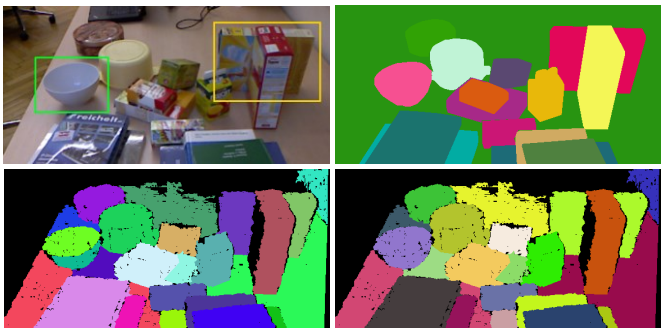


Figure 1. RGB-Image (top left), ground truth image (top right), segmentation based on local convexity (bottom left) and segmentation result after high-level grouping based on global convexity (bottom right). The example is selected from OSD [2].

II. Related Research

A number of approaches for segmenting RGB-D images into objects are proposed. A commonly used approach consists of two steps: (i) segmentation of image into smooth surface patches and (ii) aggregation of these patches into objects. The aggregation of surface patches is usually performed by forming a graph whose nodes are the surface patches, which is then segmented according to relations between neighboring patches. In this section, we review several methods based on this basic approach, which is also used in the research presented in this paper. The approach proposed in [1] segments a depth image into regions of smoothly curved surfaces bounded by sharp edges and then connects the detected smooth surfaces into objects. After image segmentation into smooth surfaces, an adjacency matrix is formed representing connectivity of adjacent surfaces. This adjacency matrix is used by a greedy algorithm

*This work has been fully supported by the Croatian Science Foundation under the project number IP-2014-09-3155.

R. Cupec, D. Filko and E. K. Nyarko are with the Faculty of Electrical Engineering, Computer Science and Information Technologies Osijek, J. J. Strossmayer University of Osijek, Kneza Trpimira 2B, 31000 Osijek, Croatia (phone: ++385 91 224 6019; fax: ++385 31 224 605; e-mail: robert.cupec@etfos.hr, damir.filko@etfos.hr, nyarko@etfos.hr).

which is based on the detection of triplets of adjacent surfaces.

In the approach proposed in [2][3], an RGB-D image is first segmented into planes and NURBS. Relations between two adjacent surface patches are described by feature vectors representing geometric and color properties of these two patches and the boundary which separates them. In addition to relations between adjacent patches, the discussed approach also considers the relations between spatially separated patches. Two different feature vectors are defined, one for neighboring and another for non-neighboring patches. These feature vectors are rather complex and they include 16 properties describing curvature of the border between the patches, variance of distances and color difference along the border, difference between size, color, texture and normal directions of the patches. Two SVM classifiers are then trained using an appropriate training dataset consisting of feature vectors describing the relations between two surface patches, where manually segmented images are used as the ground truth. One classifier is used for adjacent and the other for non-adjacent surface pairs. The SVM classifiers assign a value to each relation, which indicates the probability that the two surface patches corresponding to this relation belong to the same object. Image segmentation is based on graph representation of the image, where graph nodes represent surface patches. The relations between two surface patches are represented by edges connecting their corresponding nodes. Hence, each edge corresponds to a feature vector. The trained SVM classifiers are used to compute energy terms for the edges based on their corresponding feature vectors. Finally, a graph segmentation procedure based on the approach proposed in [7] is applied to the discussed graph. This procedure results in groups of nodes, each representing an object in the considered image.

This paper focuses on convexity as one of the most powerful cues for distinguishing objects from each other. An approach for hierarchical segmentation of triangular meshes into convex shapes is proposed in [8]. In [4], an efficient method for segmentation of depth images into approximately convex surfaces is proposed and its application for detecting simple objects is demonstrated. First, the image is segmented into triangles by a recursive approach based on Delaunay triangulation proposed in [9]. Then, the obtained triangles are grouped into approximately convex surfaces by a region growing procedure. An approach to object detection based on local convexity is presented in [5]. The image is first segmented into supervoxels using the VCCS method proposed in [10]. These supervoxels are then represented by a graph, where each node represents a supervoxel and nodes corresponding to adjacent supervoxels are connected by edges. A heuristic criterion based on convexity is used to decide whether or not two nodes connected by an edge belong to the same object. Then, a region growing procedure is applied to detect connected subgraphs of the discussed graph representing objects in the considered image. It is demonstrated that a simple criterion based on local convexity gives segmentation performance comparable to the much more complex method presented in [2] which combines many cues. The research presented in this paper supplements the research reported in [4] and [5] by investigating the improvement in object detection which can be achieved by grouping spatially separated image regions into objects using convexity cue only.

The method proposed in [6] also uses local and global convexity as cues for segmenting a 3D scene into objects. This method first decomposes the scene into a set of candidate mesh segments and then ranks each segment according to five intrinsic shape measures: compactness, symmetry, smoothness and local and global convexity as well as a recurrence measure (presence of similar objects in other scenes). To partition a scene mesh into segments, the mesh is represented by a graph and the graph based segmentation proposed in [7] is applied. In contrast to the approach presented in this paper, this method is designed for 3D scene models obtained by fusion of multiple views, while in this paper we investigate the ability of object detection from a single viewpoint. Furthermore, besides convexity, the approach presented in [6] uses other cues. On the other hand, in this paper, we investigate the object detection performance that can be achieved by using convexity as the only cue.

III. Depth Image Segmentation Based on Local Convexity

The approach investigated in this paper consists of three steps: (i) segmentation into planar patches, (ii) hierarchical planar patch clustering and (iii) high-level grouping of spatially separated segments into objects. The first two steps are explained in this section, while the high-level segment grouping, which is the focus of this paper, is described in Section IV

A. Segmentation into Planar Patches

A common approach to object detection is to first segment the input image into smooth surface patches. In [2] this is achieved by detection of planar patches and NURBS, in [1] and [6] by detection of low curvature regions using region growing and graph-based segmentation [7] respectively, while in [5] clustering into supervoxels is performed. In this paper, segmentation into planar patches based on region growing is used as the preprocessing step.

The approach used in the research presented in this paper has the same underlying principle as the segmentation method proposed in [11]. This approach is based on region growing and it requires a triangular mesh as input. The region growing procedure is initialized by creating a region consisting of a randomly selected seed point and growing this region by adding neighboring points which lie on the plane defined by the seed point and its normal within a predefined threshold. After the region growing procedure is completed, i.e. if no more points satisfying the planarity criterion can be added to the grown region, a new randomly selected mesh point, which currently isn't assigned to any planar segment, is used as the seed for the next region growing process. We improved this method by fitting the common boundary of adjacent patches to the intersection line between their supporting planes.

The input to our segmentation method is a triangular mesh. In the experiments reported in this paper, triangular

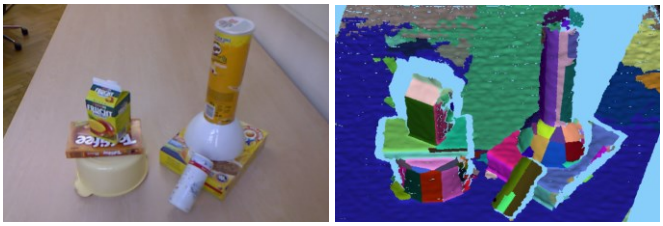


Figure 2. RGB-Image (left) and planar patches extracted from this image (right).

meshes are created from depth images by the mesh construction algorithm presented in [11], which is included in the Point Cloud Library (PCL) [12]. The approach used in this research performs region growing in two stages. In the first stage, region growing is performed analogously to the approach proposed in [11], which grows a region until reaching the boundaries of already existing patches. In the second stage, the grown region is expanded into adjacent patches by adding points belonging to these patches, which satisfy the planarity criterion of the currently grown region. The result of expansion of the currently grown region into an existing planar patch is a connected set of points which belong to both of these two regions. Then the minimum cost boundary between the grown region and the patch is searched for inside this common region, where the part of the boundary along the intersection line between the supporting planes of these two regions has zero cost. Thereby, the boundaries which are aligned with the intersection lines of adjacent planar patches are favored. Since the main topic of this paper is high-level grouping using global convexity criterion, only the basic idea of the applied planar patch extraction method is explained in this section, while a more detailed description of this method is given in [13]. An example of depth image representation by planar patches is shown in Fig. 2.

The result of the described procedure is a representation of the input image by a set of approximately coplanar subsets, referred to in this paper as planar patches C_i, $i = 1, \ldots, n_{pp}$. Each planar patch is assigned a normal $\mathbf{n}_i$, i.e. a unit vector perpendicular to the supporting plane of C_i, parameter d_i representing the distance of this plane w.r.t. the camera reference frame and a set V_{C_k} of boundary points in which three adjacent planar patches meet, referred to in this paper as *vertices*. Vertices play an important role in the high-level segment grouping based on global convexity.

The purpose of fitting the boundaries of adjacent planar patches to intersection lines between their supporting planes is to obtain a representation of the input depth image which can be regarded as an approximate polygonization of this depth image. A benefit of this representation is that the convex hull of the vertices of a set of planar patches obtained by the proposed method represents a good approximation of the convex hull of the union of all points of this set. This property is used in the third step of the proposed object detection described in Section IV.

B. Relations between Adjacent Planar Patches

Analogously to the approach presented in [5], the method considered in this paper relies on local convexity as the main

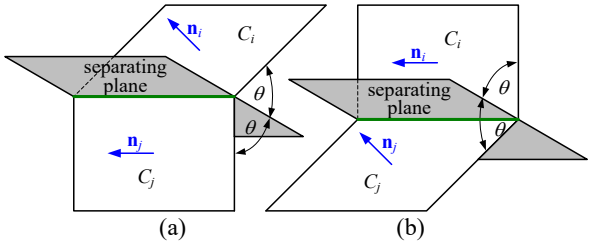


Figure 3. Two planar patches C_i and C_j separated by a convex (a) and concave (b) boundary. Green lines denote boundaries between patches.

cue for aggregation of planar patches in the second step. Planar patches obtained in the first step can be represented by a graph $\mathcal{G}$ whose nodes are these patches. Each two nodes in this graph, which represent adjacent planar patches C_i and C_j, are connected by an edge E_{ij}, representing the boundary separating these two patches.

We define the convexity relation between two adjacent planar patches C_i and C_j by introducing a *separating plane*, which contains the intersection line of the supporting planes of C_i and C_j and is oriented at the same angle w.r.t. these two planes, cf. Fig. 3. The separating plane of C_i and C_j is defined by normal $\mathbf{n}_{ij}$ and distance d_{ij} defined by

$$\mathbf{n}_{ij} = \frac{\mathbf{n}_i - \mathbf{n}_j}{\left\|\mathbf{n}_i - \mathbf{n}_j\right\|}, \qquad d_{ij} = \frac{d_i - d_j}{\left\|\mathbf{n}_i - \mathbf{n}_j\right\|}. \tag{1}$$

Let C_{ij}^+ be the set of all points $\mathbf{p} \in C_i$ such that

$$\mathbf{n}_{ij}^T \cdot \mathbf{p} - d_{ij} > 0 \tag{2}$$

and let $C_{ij}^- = C_i \setminus C_{ij}^+$. Furthermore, let C_{ji}^+ and C_{ji}^- be subsets of C_j defined analogously. The boundary between C_i and C_j is considered to be convex if $\left|C_{ij}^+\right|$ is significantly greater than $\left|C_{ij}^-\right|$ or $\left|C_{ji}^+\right|$ is significantly greater than $\left|C_{ji}^-\right|$, where $|\cdot|$ denotes the number of points in a particular set. This property can be formulated by introducing coefficients

$$\gamma_{ij}^+ = \max\left(\frac{\left|C_{ij}^+\right|}{\left|C_i\right|}, \frac{\left|C_{ji}^+\right|}{\left|C_j\right|}\right), \quad \gamma_{ij}^- = \max\left(\frac{\left|C_{ij}^-\right|}{\left|C_i\right|}, \frac{\left|C_{ji}^-\right|}{\left|C_j\right|}\right). \tag{3}$$

If $\gamma_{ij}^+ > \gamma_{ij}^-$, then the boundary between C_i and C_j is considered to be convex. Otherwise, the boundary is considered to be concave. Each edge E_{ij} is assigned a feature vector $\mathbf{f}_{ij}$ consisting of three components. The first two components are

$$f_1 = \operatorname{sign}\left(\gamma_{ij}^+ - \gamma_{ij}^-\right) \cdot \arccos\left(\mathbf{n}_i^T \cdot \mathbf{n}_j\right), \tag{4}$$

$$f_2 = \max\left(\gamma_{ij}^+, \gamma_{ij}^-\right). \tag{5}$$

These two components together represent a measure of convexity of the boundary separating two planar patches. The first component represents the angle between the supporting planes of the two patches, where positive angles represent convex and negative values concave boundary. The second component measures the property that one of the patches lies

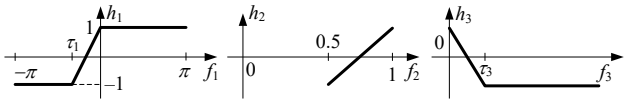


Figure 4. Classification functions defined for relation features.

mostly on one side of the separating plane. Intuitively, the larger the values f_1 and f_2, the higher is the probability that the two planar patches belong to the same object.

The third component f_3 of the feature vector $\mathbf{f}_{ij}$ is the average distance between the closest boundary points of the two planar patches. This component has large values along depth discontinuities, which usually separate planar patches belonging to different objects.

C. Greedy Planar Patch Aggregation

Feature vectors $\mathbf{f}_{ij}$ assigned to every edge E_{ij} are used to estimate the probability that two adjacent planar patches C_i and C_j belong to the same object. For that purpose, a *classification function* h is defined which maps vectors $\mathbf{f}_{ij}$ onto the interval [–1, 1]. Higher positive values of function h correspond to a higher probability that C_i and C_j belong to the same object, while negative values indicate that these two patches probably belong to two different objects. A function h_k, k = 1, 2, 3 is defined for each component f_k of vector $\mathbf{f}_{ij}$, which maps this component onto the interval [–1, 1] and the resulting classification function is computed as

$$h(\mathbf{f}) = \min\left(h_1(f_1), h_2(f_2), h_3(f_3)\right). \tag{6}$$

Functions h_k, shown in Fig. 4, implement a priori knowledge about correlation between the convexity and discontinuity properties of boundaries between planar patches and the probability that they belong to the same or different objects. It is a well-known fact, used in previous research on this topic [4]–[6] , that a convex surface probably represents a single object. Hence, $h_1(f_1) = 1$ for $f_1 > 0$. However, some objects can also have concave surfaces. Two planar patches with a small angle between them whose union is a concave surface probably belong to the same object. Hence, function h_1 falls proportionally for negative values of angle f_1 until this value reaches a predefined value τ_1. As explained in Section III.B, higher values of f_2 indicate a higher probability of two patches belonging to the same object. Value f_2 by its definition cannot be less than 0.5. Hence, function h_2 rises linearly from –1 to 1 on the interval [0.5, 1]. Function h_3 falls from 1 to –1 as the average distance between boundary points of the two patches increases from 0 to a predefined value τ_3, which implements a simple rule that larger depth discontinuities probably correspond to boundaries between different objects. Although the classification function is defined according to heuristic rules, these rules are very simple and require only two parameters τ_1 and τ_3 to be set.

Values $h(\mathbf{f}_{ij})$ computed for all edges E_{ij} are used to formulate the considered object detection problem as a graph energy minimization problem. Each edge E_{ij} is assigned the cost of the boundary between planar patches C_i and C_j, computed as

$$v_{ij} = w_{ij} \cdot h(\mathbf{f}_{ij}), \tag{7}$$

where w_{ij} is the boundary length. A higher cost of the boundary separating two planar patches indicates that this is not an object boundary, i.e. that these two patches belong to the same object. On the other hand, a negative cost indicates that this boundary is probably the object boundary separating two patches belonging to two different objects. The total energy of the graph $\mathcal{G}$ is defined as the sum of all edge costs. By removing the boundary between two planar patches C_i and C_j, the nodes representing these two patches are merged into a new node and a new graph is obtained. If C_i and C_j are both adjacent to a planar patch C_k, then the edges E_{ik} and E_{jk} are substituted in the new graph by a single edge whose cost is $v_{ik} + v_{jk}$. If the cost of the edge E_{ij} connecting C_i and C_j is $v_{ij} > 0$, then the graph obtained by removing this edge has a lower total energy.

We perform minimization of the total graph energy by applying a greedy iterative algorithm, which removes the boundary with the highest cost in each iteration. This process is repeated until all remaining boundaries have negative costs. This algorithm is referred to in this paper as *Greedy Planar Patch Aggregation* (GPPA). It is based on the approach proposed in [14] with the boundary weighting scheme adopted from [15]. A detailed description of this method can be found in [15]. The result of this algorithm is a set of segments obtained by aggregation of planar patches.

IV. Segment Grouping Using Global Convexity

In this section, the third step of the proposed object detection approach is described. The planar patch aggregation method, described in Section III, groups adjacent planar patches into connected sets, referred to in this paper as segments, according to local convexity and depth discontinuity cues. In order to detect disconnected and hollow convex objects, a convexity criterion is evaluated for all segment pairs and those which satisfy this criterion are considered as candidates for grouping.

Let's consider a segment S consisting of planar patches C_k, where each planar patch is assigned a set of vertices V_{C_k} defined in Section III.A. Furthermore, let V_S be the set of all vertices of all planar patches constituting S. This segment is convex if and only if for every vertex $P \in V_S$ and every planar patch $C_k \in S$

$$\mathbf{n}_k^T \cdot \mathbf{p} - d_k \leq \varepsilon \tag{8}$$

where $\mathbf{p} \in \mathbb{R}^3$ is a vector defining the position of P w.r.t. the camera reference frame and $\varepsilon = 0$. If we relax this condition by allowing ε to be a small positive constant, the definition of *approximately convex* surface is obtained.

This convexity criterion is used as the basis for the segment grouping criterion proposed in this paper. Two segments S_i and S_j are considered to represent one object if a high percentage of their supporting points belong to the same approximately convex surface. This property is evaluated by Algorithm 1.

Algorithm 1 *Convexity Measure*

Input: S_i, S_j, ε
Output: $\lambda(S_i, S_j)$

1 : $S_{ij} \leftarrow S_i \cup S_j$
2 : $Q \leftarrow$ list of planar patches contained in S_{ij} sorted in descending order
3 : $V \leftarrow \varnothing$
4 : $S_{conv} \leftarrow \varnothing$
5 : **Repeat**
6 : $C_k \leftarrow$ the first planar patch in Q
7 : Remove C_k from Q.
8 : **For** every vertex $\mathbf{p} \in V$
9 : **If** $\mathbf{n}_k^T \cdot \mathbf{p} - d_k > \varepsilon$, **then go to** line 18.
10 : **end for**
11 : **For** every $C_l \in S_{conv}$
12 : **For** every vertex $\mathbf{p} \in V_{C_k}$
13 : **If** $\mathbf{n}_l^T \cdot \mathbf{p} - d_l > \varepsilon$, **then go to** line 18.
14 : **end for**
15 : **end for**
16 : $V \leftarrow V \cup V_{C_k}$
17 : $S_{conv} \leftarrow S_{conv} \cup C_k$
18 : **until** Q is empty.
19 : Compute $\lambda(S_i, S_j)$ using (9).
20 : **Return** $\lambda(S_i, S_j)$.

For a given pair of segments S_i and S_j, this algorithm determines a set $S_{conv} \subseteq S_i \cup S_j$ which represents an approximately convex surface. The percentages of the supporting points of S_i and S_j belonging to S_{conv} are computed and the minimum of these two percentages is taken as the *convexity measure* of the segment pair (S_i, S_j).

Computation of the convexity measure $\lambda(S_i, S_j)$ can be described by equation

$$\lambda\left(S_i, S_j\right) = \min\left(\frac{\left|S_{conv} \cap S_i\right|}{\left|S_i\right|}, \frac{\left|S_{conv} \cap S_j\right|}{\left|S_j\right|}\right). \qquad (9)$$

Note that the computational complexity of the proposed method for measuring the convexity of the union of two surfaces depends on the number of planar patches belonging to these surfaces and their vertices, which is significantly lower than the total number of supporting image points of these surfaces.

The applied convexity criterion represents a rather weak constraint, which can result in many false objects. In order to reduce the probability of false segment grouping, the criterion of maximum object size is used in addition to the aforementioned convexity criterion. The bounding box of the union of two segments is determined. Only those segment pairs whose bounding box size is less than a predefined threshold a_{max} are considered for grouping. This additional constraint can be implemented by setting $\lambda(S_i, S_j)$ to zero for all segment pairs whose bounding box exceeds the threshold. In the experiments reported in this paper, a_{max} is determined according to the size of objects in a training set.

The described convexity measure is used in a greedy segment grouping procedure. The procedure starts by identifying the pair of segments (S_i, S_j) with the highest convexity measure $\lambda(S_i, S_j)$. If $\lambda(S_i, S_j) \geq \tau_{conv}$, where τ_{conv} is a predefined threshold, then these two segments are merged into a new segment S'. The procedure continues by identifying all pairs of segments, where one member of the pair belongs to S' and the other is not already grouped, and selecting the pair with the highest convexity measure. Let (S_p, S_q) be this pair and, without loss of generality, let's assume that $S_p \in S'$ and $S_q \notin S'$. If for all $S_r \in S'$, $\lambda\left(S_q, S_r\right) \geq \tau_{conv}$, then S_q is added to S'. This segment growing process stops when no more segments can be added to S'. After that, the pair of remaining segments with the highest convexity measure is identified and the described segment growing procedure is repeated. The segment grouping process is completed when there are no more segment pairs with convexity measure $\geq \tau_{conv}$.

In order to extend the proposed method to a wider class of objects, a preprocessing step, which identifies hollow convex objects is introduced. Before computing $\lambda(S_i, S_j)$, the convexity of segments S_i and S_j is examined. For every segment S_i, the ratio $\left|S_{conv} \cap S_i\right| / \left|S_i\right|$ is computed, as explained in Algorithm 1. In addition to this ratio, the same ratio is computed by assuming that the considered segment represents the inner surface of a hollow convex object. This is achieved by flipping the normals of all planar patches which constitute segment S_i. If the second ratio is greater, then $\lambda(S_i, S_j)$ is computed by evaluating the condition $-\left(\mathbf{n}_l^T \cdot \mathbf{p} - d_l\right) > \varepsilon$ instead of $\mathbf{n}_l^T \cdot \mathbf{p} - d_l > \varepsilon$ in lines 9 and 13 of Algorithm 1. This relaxed condition is applied to adjacent segments only.

In order to avoid small particles being falsely merged to distant objects, only segments with size greater than a predefined threshold τ_{group} are considered in the described segment grouping process.

At the end of this grouping process, the segments with size smaller than a threshold τ_{attach} are attached to the largest adjacent segment whose size is $\geq \tau_{attach}$. Such a postprocessing step is also applied in [5].

The method described in this section is referred to in this paper as *Global Convexity Segment Grouping* (GCSG). Two examples of detecting disconnected and hollow convex objects are shown in Fig. 5. Objects which are represented by multiple segments after GPPA step (top images) are connected after application of GCSG (bottom images).

V. Experimental Evaluation

The proposed segmentation method is implemented in C++ programming language using PCL [12] and OpenCV library [16]. The method is experimentally evaluated using Object Segmentation Database (OSD) [17], a publicly available dataset containing 111 RGB-D images of household objects placed on a table with manually annotated ground truth. This dataset is suitable for evaluation of the proposed method for several reasons. First, it has already been used by

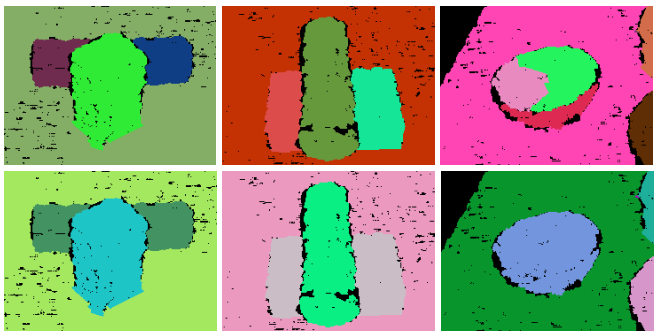


Figure 5. Segmentation before (top) and after high-level segment grouping based on global convexity (bottom).

other researchers [2][5] and, hence, we can make a clear comparison of our work with two state-of-the art methods. Furthermore, objects of simple shapes such as those used in OSD appear often in industry, household environments and medical institutions, and the method proposed in this paper is designed for improving the detection of such objects.

Each RGB-D image in OSD is assigned a ground truth image. Every pixel of a ground truth image is manually assigned a label of the object it belongs to. The segmentation results are compared to the corresponding ground truth data according to two performance measures – oversegmentation error and undersegmentation error, which are defined in the same way as in [2] and [5]. The oversegmentation error F_{os} and undersegmentation error F_{us} are defined as

$$F_{os} = 1 - \frac{N_{true}}{N_{total}}, \qquad F_{us} = \frac{N_{false}}{N_{total}}, \tag{10}$$

where N_{true} is the number of correctly assigned pixels, N_{false} is the number of falsely assigned pixels and N_{total} is the total number of pixels assigned to all objects in the corresponding ground truth image.

The values of the parameters of the proposed method, which are used in this analysis are given in Table I. The comparison of the proposed method with the LCCP method presented in [5] and the approach presented in [2] is given in Table II. The results of the segmentation obtained by the second step of our approach are denoted in Table II by GPPA, while the results obtained by the complete algorithm are denoted by GPPA + GCSG. Two variants of the approach proposed in [2] are presented in Table II, the one which considers only relations between adjacent surfaces (SVM_{nb}) and the one which connects spatially separated surfaces (SVM_{nnb}). The results denoted by LCCP [5] are taken from [5], while the results denoted by LCCP-PCL are generated by the implementation of LCCP included in PCL.

From the presented analysis it can be concluded that the GCSG step reduces the oversegmentation error by 32% (from 9.8% to 6.6%). Furthermore, the proposed method results in a 21% lower oversegmentation error in comparison to LCCP [5] and 38% lower oversegmentation error in comparison to LCCP-PCL. In the same time, the undersegmentation error is 12% higher in comparison to LCCP [5] and 10% higher in comparison to LCCP-PCL. Since LCCP relies on local convexity, it cannot detect disconnected objects and has

TABLE I. PARAMETER VALUES

τ_1	τ_2	ε	τ_{conv}	a_{max}	τ_{group}	τ_{attach}
$-\pi/6$	0.03 m	0.015 m	0.8	0.3 m	300	1000

TABLE II. COMPARISON OF SEGMENTATION METHODS

	F_{os}	F_{us}
Richtsfeld et al. [2], SVM_{nb}	4.5	7.9
Richtsfeld et al. [2], SVM_{nnb}	2.7	69.5
LCCP [5]	8.4	3.9
LCCP-PCL	10.7	4.0
GPPA	9.8	4.2
GPPA + GCSG	6.6	4.4

TABLE III. EXECUTION TIMES

	planar patches	GPPA	GCSG	Total
avg. (ms)	222.6	31.9	9.7	264.2
max. (ms)	360.6	54.6	13.0	408.0

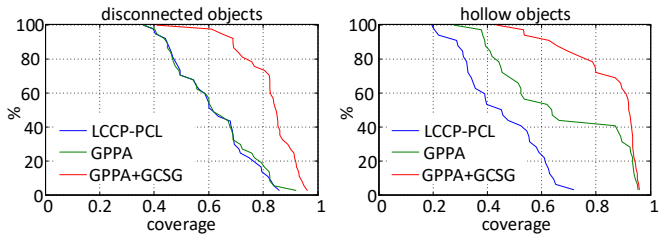


Figure 6. Coverage of ground truth objects with the dominant segment.

problems with detecting hollow objects. Our method, on the other hand, is designed as a postprocessing step which reduces this problem. In comparison to the approach presented in [2], our method has 47% higher oversegmentation error, but 45% lower undersegmentation error.

Since the disconnected and hollow objects contribute to the total number of object pixels in OSD with a relatively small percentage, the effect of convex-based segment grouping is not clearly visible in the analysis presented in Table II, which considers the complete database. In order to get a better insight into the improvement of detection of disconnected and hollow objects, we performed an analysis which focuses on these objects. All disconnected and hollow objects are manually selected from the ground truth images and for each of these objects the percentage of its pixels covered by the dominant segment obtained by the evaluated method is computed. This percentage is in the presented analysis referred to as *coverage* of a ground truth object. A total of 37 disconnected objects and 32 hollow objects are considered in this analysis. The graphs in Fig. 6 represent the percentage of the considered objects whose coverage is greater or equal to a particular coverage. It can be seen that for 80% of disconnected objects, the coverage achieved by GCSG is over 0.72 and for 80% of hollow objects the achieved coverage is over 0.74. Without application of GCSG, these values are 0.47 and 0.42. For LCCP-PCL these values are 0.47 and 0.31.

The computational efficiency of the proposed approach is analyzed in Table III. The first column represents the computation times needed for segmentation of depth images into planar patches. The second and third column represent execution times of GPPA and GCSG steps respectively. In the last column, the total execution time of the proposed method is presented. For each of the aforementioned steps, the average and maximum execution times are displayed. The algorithm is executed on a PC with Intel Core i7 processor and 8GB RAM. Computation of normals is performed using PCL and its computation time is not considered in Table III.

VI. Conclusion

The research presented in this paper is motivated by results of previous investigations in the field of object detection in depth images, which indicate: (i) that convexity and depth discontinuity are powerful cues for deciding whether or not two adjacent image regions belong to the same object and (ii) that by relying only on these two cues for detection of objects of simple shapes in cluttered scenes, object detection performance comparable to much more complex approaches can be achieved.

In order to confirm the aforementioned observation, an approach for segmentation of depth images based on convexity and depth discontinuity as the only cues is presented. Analogously to existing similar approaches, the proposed approach starts by segmenting the input depth image into surface patches, computes features characterizing relations between adjacent surface patches and then groups these patches using an appropriate graph segmentation algorithm. The proposed approach uses planar surface patches, features based on local convexity and depth discontinuity and a greedy graph segmentation algorithm. As the final step, the proposed method performs a high-level segment grouping based on global convexity in order to aggregate disconnected segments representing occluded objects as well as to merge adjacent convex and concave surfaces representing hollow convex objects.

The considered method is limited to the detection of simple convex and hollow convex objects. However, efficient detection of such objects is very useful since they are common in industry, household environments and medical institutions and we wanted to provide a practical tool for such applications. The presented experimental analysis using a publicly available benchmark dataset clearly demonstrates the improvement achieved by high-level segment grouping based on global convexity. Nevertheless, although the Object Segmentation Database (OSD) used to validate the proposed approach contains a certain number of images with disconnected and hollow convex objects, a thorough validation using an image dataset specific to the problem of detecting disconnected and hollow convex objects would be of great interest. Since, to the best of our knowledge, such a dataset is not publicly available, one possible direction of our future work is to create one.

Although convexity and depth discontinuity can be used for reliable detection of simple objects, which are ubiquitous in industry, household environments and medical institutions, detection of objects of complex shapes would by no means require more sophisticated cues. One approach is to measure the similarity of the union of two surfaces in a scene with the objects observed in the training phase and to decide whether or not these two surfaces represent the same object according to the similarity of their union with one or multiple observed objects. This will be the topic of our future research.

References

[1] A. Ückermann, R. Haschke and H. Ritter, "Real-time 3D segmentation of cluttered scenes for robot grasping," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012

[2] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4791–4796.

[3] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillic, and M.Vincze, "Learning of perceptual grouping for object segmentation on rgb-d data", *Journal of Visual Communication and Image Representation*, 2014, vol. 25, pp. 64–73.

[4] R. Cupec, E. K. Nyarko and D. Filko, "Fast 2.5D Mesh Segmentation to Approximately Convex Surfaces", *Proceedings of the European Conference on Mobile Robots (ECMR)*, 2011, Örebro, Sweden pp. 127–132.

[5] S. C. Stein, F. Wörgötter, M. Schoeler, J. Papon and T. Kulvicius, "Convexity based object partitioning for robot applications," *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3213–3220.

[6] A. Karpathy, S. Miller and L. Fei-Fei, "Object discovery in 3d scenes via shape analysis," *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[7] P. F. Felzenszwalb and D. P. Huttenlocher, "*Efficient graph-based image segmentation*," International Journal of Computer Vision, 2004, vol. 59, no. 2, pp. 167–181.

[8] M. Attene, M. Mortara, M. Spagnuolo and B. Falcidieno, "Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection," *Computer Graphics Forum*, 2008, vol. 27, no. 5, pp. 1323–1332.

[9] F. Schmitt and X. Chen, "Fast segmentation of range imagery into planar regions," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991, pp. 42–60.

[10] J. Papon, A. Abramov, M. Schoeler and F. Wörgötter, "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2027–2034.

[11] D. Holz and S. Behnke, "Approximate Triangulation and Region Growing for Efficient Segmentation and Smoothing of Range Images," *Robotics and Autonomous Systems*, vol. 62, no. 9, pp. 1282–1293, 2014

[12] R. B. Rusu, S. Cousins, "3D is here: Point Cloud Library (PCL)," *International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[13] R. Cupec, "3D Mesh Segmentation into Planar Segments Based on Region Growing and Plane Intersection," *Croatian Science Foundation, project IP-2014-09-3155*, *Technical Report TR3*, J. J. Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technologies Osijek, Croatia, 2016. [http://ferit.hr/ARP3D/TR3_v2.1.pdf]

[14] M. Garland, A. Willmott and P. S. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces," *ACM Symposium on Interactive 3D Graphics*, 2001

[15] R. Cupec, E. K. Nyarko and D. Slišković, "Efficient postprocessing of edge maps for image segmentation based on greedy correction cost minimization," *Journal of Electronic Imaging*, 2012, 21, 2; 023007-1-023007-13

[16] A. Kaehler and G. Bradski, *Learning OpenCV 3*, Sebastopol, CA: O'Reilly, 2017.

[17] A. Richtsfeld, The Object Segmentation Database (OSD), 2012. [http://www.acin.tuwien.ac.at/?id=289]

Kinematic Model based Visual Odometry for Differential Drive Vehicles

Julian Jordan[1] and Andreas Zell[1]

***Abstract*— This work presents KMVO, a ground plane based visual odometry that utilizes the vehicle's kinematic model to improve accuracy and robustness. Instead of solving a generic image alignment problem, the motion parameters of a differential drive vehicle can be directly estimated from RGB-D image data. In addition, a method for outlier rejection is presented that can deal with large percentages of outliers. The system is designed to run in real time on a single thread of a mobile CPU.**

The results of the proposed method are compared to other publicly available visual odometry and SLAM methods on a set of nine real world image sequences of different indoor environments.

I. INTRODUCTION

A precise and robust self localization is a crucial component for many robotic tasks. Although a variety of different approaches exist, most publicly available methods are designed to work with a forward looking sensor. For obstacle detection and avoidance, a downward facing camera is more suitable to perceive obstacles directly in front of the vehicle, and for reliable detection of negative obstacles. Having images mostly displaying the ground is not beneficial for visual odometry for several reasons, especially in indoor environments: it can have a very low contrast, quickly repeating texture, the ground plane does not provide geometric information and is close to the camera, increasing the effect of motion blur. Additionally, reflections and overexposure are more likely to occur. In order to achieve robust and reliable results with the described setup, an adapted visual odometry is required.

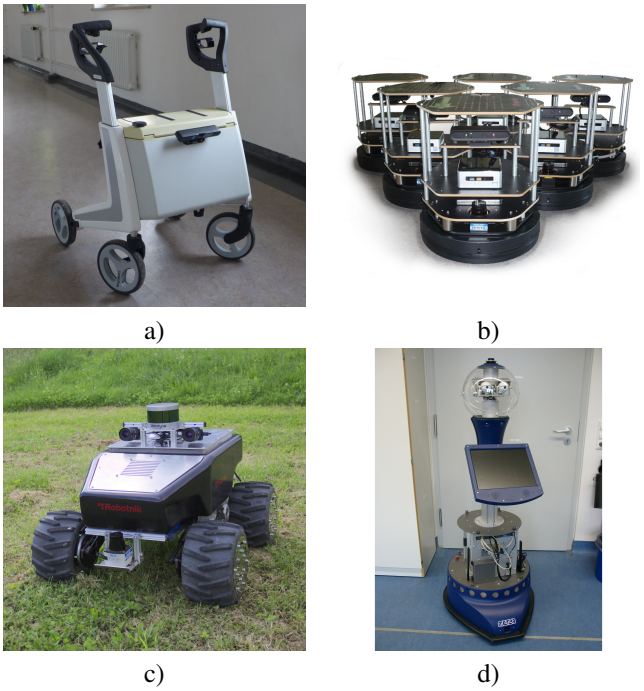
This work proposes a visual odometry method specifically designed to work with a downward facing camera, mounted on a differential drive vehicle. Due to the non-holonomic constraints of a differential drive, the motion can be described by two parameters. It is shown that estimating these two parameters directly from image data can significantly improve the accuracy of the estimated motion. Furthermore, an efficient outlier rejection scheme is presented. It operates in the model parameter space instead of the data space. This increases robustness against image noise and allows removal of large outlier areas. The performance of the proposed method is evaluated on nine real world datasets and compared to five state-of-the-art approaches and its predecessor. KMVO is able to outperform all six methods used for comparison, while being able to run at 60hz on a single thread of a mobile CPU.

[1]Julian Jordan and Andreas Zell are with the Faculty of Science, Dept. of Computer Science, University of Tuebingen, 72076 Tuebingen, Germany julian.jordan,andreas.zell@uni-tuebingen.de

The outline of the paper is as follows: Section II gives a review of related work. In Section III the hardware setup and the orthogonal projection preprocessing step are described. Section IV derives the image warping function for the differential drive model. The image alignment and outlier rejection are described in Section V. The evaluation method and evaluation results are presented in Section VI. Finally, a conclusion is given in Section VII.

The contributions of this paper are:

- A formulation that allows to estimate the motion parameters of the kinematic model directly from image data.
- An efficient motion parameter based outlier rejection scheme.
- Error handling for situations where the kinematic model cannot describe the actual motion.
- Performance comparison with six existing visual odometry and visual SLAM methods.



a) b) c) d)

Fig. 1. a) The Bemotec beActive+e electric wheeled walker used in this work. b) Kobuki Turtlebot. c) Robotnik Summit XL. d) Metralabs Scitos G5. The proposed method is suitable for all four vehicles, since they are all based on a differential drive.

II. RELATED WORK

Visual odometry and visual SLAM are important components in many mobile robotic applications, from service

robots to autonomous cars. Numerous methods exist employing different approaches to estimate camera motion. Typically they can be classified into two categories: feature based and direct methods. Feature based methods use descriptors extracted at key points to establish correspondences between images. This reduction of data to be processed greatly improves computational performance, but also induces the loss of potentially useful information. A popular descriptor is ORB [1], which is used in [2] and [3]. Both methods include loop closure using Bag-of-Words, pose graph optimization and RANSAC for outlier rejection. [2] additionally performs bundle adjustment to refine map point positions. Feature based methods are especially prone to motion blur: if the appearance of a key point changes too much, the correspondence fails. If the whole image is affected by motion blur this may cause loss of tracking.

Direct methods try to minimize the photometric error between two or more images, mostly employing a Lucas-Kanade method [4]. Several improvements and extensions of this method were presented: e.g. [5] or [6]. They can further be split into sparse methods and dense methods. Sparse methods select portions of the image, e.g. based on the amplitude of the gradient as in [7] and [8]. As for feature based methods, this information selection can discard useful information, although [8] argues that image data is highly redundant and the effect of additional pixels decreases fast. Dense methods like [9] process the whole image, mitigating the risk of loosing important information, but increasing computational requirements. For outlier rejection, [7] and [9] use weights that are based on image residuals. Different weighting functions are tested: Huber and Tukey in [7] and t-distribution and Tukey in [9]. Using these weights for iteratively re-weighted least-squares can decrease the influence of outliers up to a certain outlier ratio [9].

For wheeled robots it is possible to reduce the degrees of freedom to three since they should move in an at least locally planar environment. This constraint allows to perform scale free monocular odometry [10], [11] and [12]. All three methods are designed for vehicles with two motion parameters. But they use a three parameter representation for image warping. This may lead to image alignment results that contradict the vehicle's motion model. While [10] uses non-holonomic constraints for efficient recalibration, they are not employed directly in the image alignment process. The non-holonomic constraints of a wheeled vehicle can be exploited to improve motion estimates: a feature based approach for an Ackermann based vehicle is described in [13].

III. Setup

The locally planar environment in which a wheeled robot moves in allows to describe the vehicle motion with three instead of six degrees of freedom. Planarity also allows the use of a kinematic model of the vehicle, allowing to decrease the number of parameters required to estimate to two, if employed on a differential drive vehicle. The vehicle used in this work is an electric wheeled walker equipped with additional sensors to make it an intelligent personal mobility assistant. As such there are limitations in terms of computational resources, weight and sensor costs. Its main purpose is to increase the mobility of visual or cognitive impaired people by helping them avoiding obstacles. This requires a robust and accurate visual odometry which works in different environments under challenging conditions.

A. Robotic Platform

The prototype is based on a Bemotec beActive+e electric wheeled walker shown in Fig. 1. It is further equipped with a Sick TiM561 LIDAR, a U-Blox GPS Module, a Razor 9DoF IMU and an Asus Xtion Pro Live, which is the sensor used for recording the data required by the proposed method. Although it is a shared control vehicle, its motion can still be described by the kinematic model of a differential drive robot with two powered rear wheels and two caster-like front wheels. The maximum electrically supported velocity is $0.81m/s$, which is the maximum velocity used for evaluation.

B. Orthogonal Projection

In order to perform 3 degrees of freedom image alignment the images have to be projected onto the ground plane. Since the RGB-D sensor provides depth information along with an RGB image, it is possible to perform an orthogonal projection of the input data along the z-axis. In addition to the RGB or intensity image, a mask image is stored describing which pixels are valid. Invalid pixels are not included in the optimization process. This allows to perform visual odometry in environments which are not planar, like in Fig. 5 b). Only the vehicle motion must follow the planarity constraint. See [14] for details of this orthogonal projection. For monocular cameras, the input images can be projected onto the previously calibrated ground plane using a homography. However, as described in [11], every deviation from this ground plane will affect the estimated result negatively.

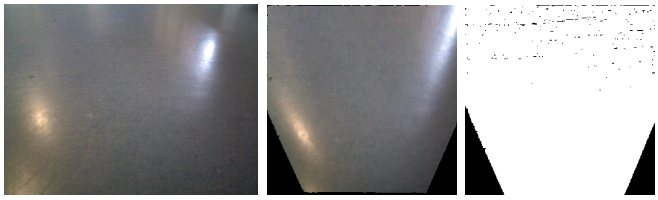


Fig. 2. Left: Original RGB image of a low contrast environment, Center: The orthogonal projection. Right: Mask image.

IV. Kinematic Model

In general the image alignment process for visual odometry consists of three components: (1) an optimization method for iteratively solving the non-linear problem, commonly Gauss-Newton or Levenberg-Marquardt are chosen. (2) a linearisation method e.g. Forward Compositional, Inverse Compositional or Efficient Second Order Minimization and (3) a warp parameter representation like $\mathfrak{se}2$ for 3DoF or $\mathfrak{se}3$ for 6DoF image alignment. See [5] and [6] for a more

detailed description of different methods for optimization, linearisation and warping. As a wheeled robot moves in an at least locally planar environment, the pose can be described by three parameters: x, y, θ. Given images depicting the ground plane, the visual odometry problem can be solved by finding the warp that minimizes the sum of squared differences between these images. Therefore the $\mathfrak{se}2$ parametrisation is an obvious choice for describing the image warp, as shown in [11] and [10]. Many wheeled vehicles are non-holonomic, like differential drive or Ackermann based vehicles, allowing to describe their motion by only two parameters while the robot itself moves in a 3DoF world. This over-parametrisation creates an ambiguity: Due to the locally linear character of small angle rotations it can be confused with a translation and vice versa. This problem increases with the distance from the image position to the center of rotation. This, especially in scenes with sub optimal image quality, can result in a motion estimate that minimizes the photometric error but describes a motion that is not feasible for the vehicle. Experiments have shown that for a differential drive vehicle with a camera mounted in the front and two powered wheels in the back, as depicted in Fig. 3, a distance from the rotation center to the sensor of about 1m is already large enough to observe this effect.

Conversely, if the vehicle performs a motion that cannot be described by the kinematic model, e.g. due to wheel slip, the proposed method detects this by comparing the values of the error functions of the $\mathfrak{se}2$ with the kinematic model alignment. If the error ratio exceeds a given threshold, the $\mathfrak{se}2$ alignment, as described in [14], is used.

A. Overview

The proposed method consists of the following processing steps:

1) Orthogonal projection of the RGB image and conversion to a gray scale image.
2) Full image alignment with $\mathfrak{se}2$.
3) If rotation and lateral motion are below a threshold go to 7).
4) Full image alignment with kinematic model.
5) Outlier rejection with kinematic model.
6) If kinematic model alignment is successful go to 8).
7) Outlier rejection with $\mathfrak{se}2$ parametrisation.
8) Update of the vehicle pose.

The full image alignment with $\mathfrak{se}2$ parametrisation is also used to reduce the number of iterations of the model based alignment by providing an initial estimate for r and $\Delta\theta$.

B. Differential Drive Model

A vehicle pose in a 2D world at time t is described by $\mathbf{p}_t = (p_{xt}, p_{yt}, p_{\theta t})$. The differential drive model describes the vehicle's motion with two parameters: The distance of the rear axis center to the center of rotation r and the rotation angle $\Delta\theta$, see Fig. 3. Since the vehicle body is rigid, all points on the vehicle perform a rotation around the same center by the same angle. This includes the position of the camera and its field of view, allowing to estimate the parameters $\mathbf{m}' = (r, \Delta\theta)$ directly from the image data. After estimating $\mathbf{m}'$ the robot pose is updated using:

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \begin{pmatrix} r(\sin(p_{\theta t} + \Delta\theta) - \sin(p_{\theta t})) \\ r(-\cos(p_{\theta t} + \Delta\theta) + \cos(p_{\theta t})) \\ \Delta\theta \end{pmatrix}. \quad (1)$$

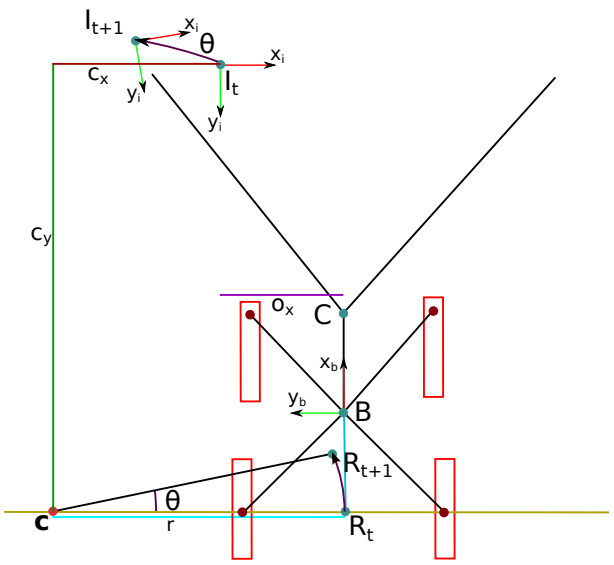


Fig. 3. Scheme of the differential drive model: B is the base link frame, R_t is the center of the rear axis at time t, $\mathbf{c}$ is the current center of rotation, C is the camera frame and I_t is the current image frame. The position of the rotation center in image coordinates is described by c_x and c_y. o_x is the known distance along the x-axis between the current image origin and the current rear axis center R_t.

In Fig. 3 two coordinate representations are used: The base link frame B is described in world coordinates with units $[m]$ and radians, all other frames are described in image coordinates with units pixels and radians. Therefore two conversion functions are required to convert world to image coordinates $q = f_{wi}(p)$ and vice versa $p = f_{iw}(q)$. f_{wi} is used for converting the fixed vehicle model parameters to image coordinates, $q = f_{wi}(p)$ is required to convert the image alignment results back to world coordinates.

C. Image Warp

To directly estimate the vehicle's motion parameters by minimizing the photometric error between two images, the image warp used for the optimization process has to be parametrised accordingly. Since the robot's motion is described by a rotation around a point on the rear axis, the same must hold for the images. The warping function is therefore parametrized with the rotation's center $c_x = f_{wi}(r) - o_x$ and angle $\theta = \Delta\theta$.

The rotation $\mathbf{R}_c$ around a point $\mathbf{c} = (c_x, c_y)$ is described by:

$$\begin{aligned} \mathbf{R}_c &= \mathbf{T}_c \mathbf{R}_\theta \mathbf{T}_c^{-1} \\ &= \begin{pmatrix} \cos(\theta) & -\sin(\theta) & (1-\cos(\theta))c_x - \sin(\theta)c_y \\ \sin(\theta) & \cos(\theta) & \sin(\theta)c_x + (1-\cos(\theta))c_y \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \quad (2)$$

where $\mathbf{T}_c$ is the translation matrix to the center and $\mathbf{R}_\theta$ is the rotation matrix around angle θ. Since the image coordinate system's z-axis is flipped, the rotational part is transposed. While c_y is known from the vehicle model, c_x and the rotation angle θ are free parameters. An image can be interpreted as function $I(\mathbf{i})$ of a point $\mathbf{i} = (i_x, i_y)$ that returns an intensity value, warping an image is equivalent to transforming the point positions by a function ω: $I(\omega(\mathbf{i}, \mathbf{m}))$

From [2] the function for warping an image point $\mathbf{i}$ by parameters $\mathbf{m} = (c_x, \theta)$ is:

$$\omega'(\mathbf{i}, \mathbf{m}) = \begin{pmatrix} \cos(\theta) i_x - \sin(\theta) i_y + (1 - \cos(\theta)) c_x - \sin(\theta) c_y \\ \sin(\theta) i_x + \cos(\theta) i_y + \sin(\theta) c_x + (1 - \cos(\theta)) c_y \end{pmatrix}. \quad (3)$$

Linearisation of $\omega'(\mathbf{i}, \mathbf{m})$ around $\theta = 0$ using a first order Taylor expansion gives:

$$\begin{aligned} \omega(\mathbf{i}, \mathbf{m}) &= \omega'(\mathbf{i}, \mathbf{m})|_{\theta=0} \\ &= \begin{pmatrix} i_x - i_y \theta - c_y \theta \\ i_x \theta + i_y + c_x \theta \end{pmatrix}. \end{aligned} \quad (4)$$

V. Image Alignment

Image alignment is done by minimizing the sum of squared differences between the previous image I_t and the current image I_{t+1} by finding the optimal warp parameters $\mathbf{m}$. The error function over all pixels is:

$$E(\mathbf{m}) = \sum_i (I_{t+1}(\mathbf{i}) - I_t(\omega(\mathbf{i}, \mathbf{m})))^2 \quad (5)$$

If the photo-consistency assumption holds, there are parameters $\mathbf{m}_{opt}$ for which $E(\mathbf{m}_{opt}) = 0$. $f_{iw}(\mathbf{m}_{opt})$ then describes the movement performed by the vehicle. To find the warp parameters $\mathbf{m}$ that minimize $E(\mathbf{m})$ between two consecutive images I_t and I_{t+1}, an iterative non-linear least squares method is used, as formulated in [15]:

$$\begin{aligned} \mathbf{m}_{n+1} =& \mathbf{m}_n - \mu_n (\mathbf{J}_n^T \mathbf{C}_D^{-1} \mathbf{J}_n + \mathbf{C}_M^{-1})^{-1} \\ & (\mathbf{J}_n^T \mathbf{C}_D^{-1} (I_{t+1} - I'_t)) + \mathbf{C}_M^{-1} (\mathbf{m}_n - \mathbf{m}_{prior})) \end{aligned} \quad (6)$$

where n is the current optimizer iteration, μ_n is the step width, $\mathbf{J}_n$ is the current Jacobian, $\mathbf{C}_D$ is the data covariance, $\mathbf{C}_M$ is the model covariance and $\mathbf{m}_{prior}$ is the model prior. The previous image I_t is transformed with the current parameter estimate $I'_t = I_t(\omega(\mathbf{i}, \mathbf{m}_n))$ after each iteration for updating the pixel-wise image differences $I_{t+1} - I'_t$ and the Jacobian $\mathbf{J}_n$.

Three termination criteria are used for the optimization: The maximum number of iterations $n > n_{max}$, a minimum error value $E(m_n) < \epsilon$ and a minimum step size $|m_{n+1} - m_n| < \delta$. Whenever one of these criteria is met, the optimizations stops.

In general the Jacobian $\mathbf{J}$ is the derivative of the warped image with regard to the model parameters, this derivative can be calculated by using the chain rule:

$$\mathbf{J} = \frac{\partial I(\omega(\mathbf{i}, \mathbf{m}))}{\partial \mathbf{m}} = \frac{\partial I(\mathbf{i})}{\partial \mathbf{i}} \frac{\partial \omega(\mathbf{i}, \mathbf{m})}{\partial \mathbf{m}} \quad (7)$$

The proposed method uses Efficient Second order Minimization (ESM) like formulation to calculate the Jacobian, see [16] and [5] for a detailed derivation of the ESM. ESM uses a Taylor expansion of the error function and the Jacobian to approximate the Hessian of the cost function, the ESM based Jacobian is:

$$\mathbf{J}_n = \frac{1}{2} \left(\frac{\partial I_{t+1}(\mathbf{i})}{\partial \mathbf{i}} + \frac{\partial I'_t(\mathbf{i})}{\partial \mathbf{i}} \right) \frac{\partial \omega(\mathbf{i}, \mathbf{m})}{\partial \mathbf{m}} \quad (8)$$

Image gradients can be obtained by using e.g. a Sobel operator:

$$\begin{aligned} \frac{\partial I'_t(\mathbf{i})}{\partial \mathbf{i}} &= \nabla I'_t = (\nabla_x I'_t, \nabla_y I'_t) \\ \frac{\partial I_{t+1}(\mathbf{i})}{\partial \mathbf{i}} &= \nabla I_{t+1} = (\nabla_x I_{t+1}, \nabla_y I_{t+1}) \end{aligned} \quad (9)$$

where $\nabla_x I$ and $\nabla_y I$ is the gradient in x resp. y direction. Deriving the warping function (4) with regard to the parameters $\mathbf{m}$:

$$\frac{\partial \omega(\mathbf{i}, \mathbf{m})}{\partial \mathbf{m}} = \begin{pmatrix} 0 & (-i_y - c_y) \\ \theta & (i_x + c_x) \end{pmatrix} \quad (10)$$

Plugging (9) and (10) into (8) results in the Jacobian for one pixel, which is one row of the $\mathbf{J}_n$ matrix:

$$(\nabla y \theta, \nabla x (-i_y - c_y) + \nabla y (i_x + c_x)) \quad (11)$$

with

$$\nabla x = \left(\frac{1}{2} (\nabla x I_{t+1} + \nabla_x I'_t \right) \quad (12)$$

$$\nabla y = \left(\frac{1}{2} (\nabla y I_{t+1} + \nabla_y I'_t \right) \quad (13)$$

The data covariance $\mathbf{C}_D$ in (6) is set to 1, all pixel intensities are independent and equally likely. The model covariance $\mathbf{C}_M$ has to be set appropriately to achieve fast and robust convergence of the optimisation. This is necessary because the ranges of the two model parameters differ by orders of magnitude: While c_x can have huge values, (in fact for a straight forward driving vehicle it is $\pm \inf$, θ has a typical range of $[-0.2, 0.2]$. During evaluation $\mathbf{C}_M$ was set to:

$$\mathbf{C}_M = \begin{pmatrix} 10^4 & 0 \\ 0 & 10^{-3} \end{pmatrix} \quad (14)$$

For the model prior $\mathbf{m}_{prior}$ the previously performed motion has shown to be a reasonable choice. If available, estimates from other sensors like wheel odometry or an IMU could also be used.

A. Outlier Removal

For visual odometry, outliers are image regions that do not reflect the actual motion of the camera. These must be excluded from the optimization process.

A popular way for outlier removal is the use of iteratively re-weighted least squares, as presented in [9] and [7]. After each optimizer iteration, a residual image is created and, based on these pixel wise residuals, a weight for each pixel is calculated. In the next optimizer iterations, these weights are multiplied with the corresponding Jacobian row to weight the pixel's influence on the optimization.

The proposed method uses an approach which takes into account the fact that outlier pixels usually have a spatial relation since effects like overexposure, reflections or moving objects are unlikely to appear only pixel wise.

The outlier rejection is performed after the alignment of the whole image terminates. The optimized parameters m_r are used as the initial estimate for the outlier rejection.

In order to find outlier regions, the image is split into blocks $\mathcal{B}$ of a fixed size, as shown in Fig. 4. For each block $\mathcal{B}_k$ the Jacobian $\mathbf{J}_k$ and the image difference $\mathbf{d}_k = (I_{t+1} - I'_t)$ of the contained pixels are calculated. With $\mathbf{J}_k$ and $\mathbf{d}_k$ for each block a single parameter update step $\mathbf{m}_k$ is estimated

$$\mathbf{m}_k = \mu_n(\mathbf{J}_k^T\mathbf{J}_k + \mathbf{C}_M^{-1})^{-1}(\mathbf{J}_k^T\mathbf{d}_k) + \mathbf{C}_M^{-1}(\mathbf{m}_r)). \quad (15)$$

Each $\mathbf{m}_k$ describes the direction in model space for which the blocks error $E(\mathbf{m}_k)$ would decrease. Without outliers and image noise, the values of all $\mathbf{m}_k$ should be similar. Blocks that contain a significant amount of outliers have a different gradient direction compared to blocks that correctly describe the vehicle's motion. Based on the estimated model parameters $\mathbf{m}_k$, a clustering in model space is performed by comparing the model space position of each block to all other blocks. For clustering a weighting function $W(v, \varepsilon)$ based on the Tukey weighting function [17] is used:

$$W(v,\varepsilon) = \begin{cases} 0 & \text{if } |v| > \varepsilon \\ (1 - (\frac{v}{\varepsilon})^2)^2 & \text{otherwise.} \end{cases} \quad (16)$$

Each model parameter c_x, θ has a separate weighting parameter $\varepsilon_x, \varepsilon_\theta$. The cluster weight of each block is:

$$\Phi(\mathbf{m}_k) = \sum_{j=0}^{k} W(m_{kx} - m_{jx}, \varepsilon_x) W(m_{k\theta} - m_{j\theta}, \varepsilon_\theta) \quad (17)$$

The parameters of the block with the highest $\Phi(\mathbf{m}_s)$ are selected as $\mathbf{m}_f$ and used as center for the cluster membership test. To get the new motion estimate, the weighted sums over the Jacobians and image differences are given by:

$$\mathbf{J}_f = \sum_{j=0}^{k} \mathbf{J}_j W(m_{fx} - m_{jx}, \varepsilon_x) W(m_{f\theta} - m_{j\theta}, \varepsilon_x) \quad (18)$$

and

$$\mathbf{d}_f = \sum_{j=0}^{k} \mathbf{d}_j W(m_{fx} - m_{jx}, \varepsilon_x) W(m_{f\theta} - m_{j\theta}, \varepsilon_x). \quad (19)$$

From $\mathbf{J}_f$ and $\mathbf{d}_f$ a new estimate for $\mathbf{m}_r$ is calculated as described in (15). The process can be repeated several times. This might be required if the initial estimate was too far from the optimum.

VI. Evaluation

A. Compared Methods

The proposed method was compared against five publicly available visual odometry or SLAM systems and its predecessor:

1) DoF3OR, the predecessor of the proposed method [14] without kinematic model constraints.

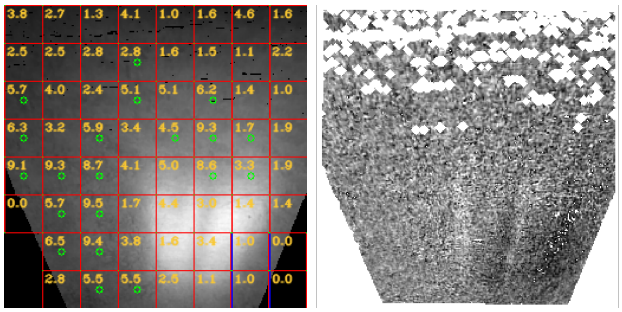


Fig. 4. Left: Visualization of the blocks used for outlier rejection, the number is $\phi(\mathbf{m}_k)$, green dots mark blocks that contribute to the selected cluster. Right: Residual image after performing the global image alignment. The white spots in the upper third of the image are pixels that were removed due to their proximity to invalid pixels.

2) DVO, a dense 6DoF visual odometry [9].
3) EDVO, a semi-dense 6DoF direct visual odometry method [7].
4) RTAB, a versatile feature-based SLAM system that includes loop closure and pose graph optimization [3]. The method was used with 3DoF localisation and non-holonomic constraints for motion estimation.
5) ORBSLAM2, a feature-based 6DoF SLAM system [2]. Two modes were tested: The full SLAM system (ORBSLAM) and the visual odometry mode with mapping disabled (ORBLOC).
6) DEMO, a feature based 6DoF visual odometry [20].

B. Data Acquisition

To compare the performance of the proposed method with other approaches, nine sequences in different indoor environments were recorded. Fig. 8 shows one of these sequences with ground truth trajectory and the resulting trajectories of all compared visual odometry systems. The total length of these nine trajectories is 527m and the total recorded time 17min. They were selected to cover different surface materials under different lighting conditions, see Fig. 5.



Fig. 5. Three different surface types were recorded: Tiles, wood and PVC (shown in Fig. 2). The images depict challenging situations the proposed method can handle: low light conditions and reflections, shadows and non planar environment.

The ground truth trajectories were created using a Sick TiM561 LIDAR for data acquisition and the Hector SLAM system [18] to integrate the LIDAR data into a global occupancy map with $0.05m$ resolution used for localisation. Every ground truth sequence was checked for map and

trajectory inconsistencies and discarded if any were found, leaving 9 out of 18 originally recorded sequences.

Each tested method was run on all nine sequences, recording the pose estimate after each frame. This yields a set $\mathcal{F}$ of poses for each method and each sequence.

C. Evaluation Method

For performance evaluation, the visual odometry evaluation method described in [19] was used. This evaluation method extracts sub paths of different lengths and calculates two error measures individually for each sub path: the rotation error and the translation error. Each error measure is normalized by the path length to be able to compare the results of sub paths of different lengths. The error measures, as given in [19], are:

$$E_{rot}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j \in \mathcal{F})} \angle[(\hat{p}_j \ominus \hat{p}_i]) \ominus (p_j \ominus p_i)] \quad (20)$$

$$E_{trans}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j \in \mathcal{F})} ||(\hat{p}_j \ominus \hat{p}_i]) \ominus (p_j \ominus p_i)||_2 \quad (21)$$

where i and j are the start and end indices of a subset of $\mathcal{F}$ for a given length l_p, $\hat{p}$ and p are the estimated resp. ground truth poses, $\ominus$ is the inverse compositional operator and $\angle$ denotes the rotation angle.

The path lengths for evaluation were l_p=(1m, 2m, 5m, 10m, 15m, 20m, 25m, 30m, 35m, 40m). For each path length, a sub path was extracted at every tenth ground truth frame.

D. Results

The mean translation error per path length over all sequences is shown in Fig. 6, the mean rotation error in Fig. 7. Table I shows the mean error weighted by the number of sub paths per path length. One major source of error are rotations, as seen in Fig. 8. Only the proposed method provides a proper estimate of all rotations contained in this sequence. Another problem are reflections, causing most methods to underestimate the distance travelled. This can be observed when turning to the right after approximately 16m.

Two methods are not suited to work with the provided data: DEMO has problems establishing correct feature correspondences. For EDVO the reason is not as clear, it may be caused by the information selection scheme which selects data with strongest Jacobians. In environments with low contrast texture, image noise and reflections tend to give larger gradients than the floor itself. Regardless, as errors in setting up theses systems cannot entirely be ruled out these results should be regarded with care.

ORBSLAM provides good results as long as the tracking remains intact. In cases where loop closure could be applied, it outperformed most other methods. When the tracking is lost, no further pose estimates are provided, causing the translation and rotation error to rise to maximum for the remaining trajectory. This happened in four out of nine sequences e.g. Fig. 8 and was caused either by fast rotation or heavy motion blur. To be comparable, it was required to use the visual odometry only version ORBLOC. The results of ORBSLAM are only shown for completeness.

As shown in Fig. 7, DVO provides reasonable rotation estimates for most sequences, but tends to underestimate the travelled distance. Several sequences contain reflections that cover significant parts of the image. As described in [9], this high outlier ratio can cause a drift.

RTAB was configured to perform 3DoF SLAM and use non-holonomic constraints, i.e. not allowing strife motions. Although the approach is lacking some features of ORB-SLAM, like local bundle adjustment, the 3DoF SLAM combined with model based constraints results in better performance on the given data.

DoF3OR without non-holonomic constraints gives results comparable to RTAB. The basic structure, performing image alignment on the orthogonal projection of the sensor data and performing block-wise outlier rejection, is similar to the proposed method.

KMVO performs best across all tests, especially rotation estimation accuracy could be improved compared to its predecessor and it is over two times better than RTAB, which also uses 3DoF and model based constraints.

Over all evaluated sequences the average processing time of the proposed method was 15.2ms per frame on a single thread of an Intel i5 4300U Mobile CPU with 1.9GHz. This makes the proposed method suitable for real time applications on a wide range of mobile robot hardware.

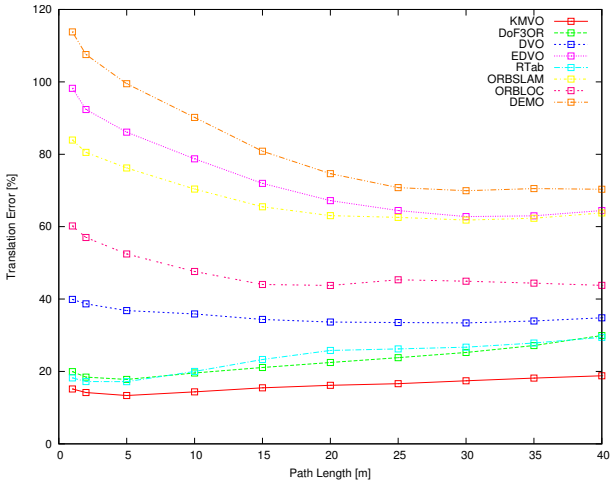


Fig. 6. Mean translation error by path length.

TABLE I

MEAN TRANSLATION AND ROTATION ERROR OVER ALL SUB PATHS.

Method	Trans (%)	Rot ($^\circ/m$)
KMVO	**15.48**	**1.69**
DoF3OR	21.44	2.80
DVO	36.00	4.10
EDVO	78.40	15.66
RTab	21.83	3.67
ORBSLAM	71.10	8.97
ORBLOC	49.80	7.20
DEMO	89.39	22.03

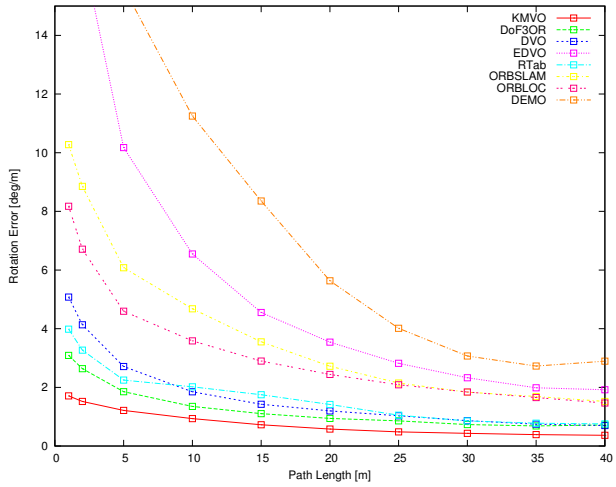


Fig. 7. Mean rotation error by path length.

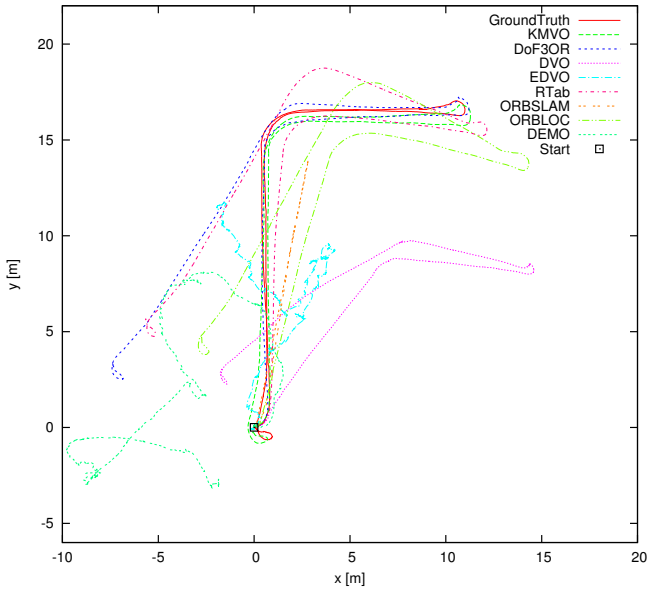


Fig. 8. Example Trajectory recorded in the environment shown as left image in Fig. 2.

VII. Conclusion

This work presented KMVO, a method for estimating motion parameters of a differential drive vehicle directly from ground plane images. The reduction from three to two parameters in the optimization process significantly improves the localization accuracy, especially with regard to the rotational part. Furthermore, an outlier rejection scheme was presented that can handle large outlier areas and is computationally efficient. The complete system was evaluated on nine real world data sets and compared to six other methods. In the tested scenarios, it outperformed all six methods it was compared to. It is also shown that the system is fast enough to run in real time on a single thread of a mobile CPU. Future work includes porting the system to monocular cameras and the inclusion of a pose graph optimization to further improve the accuracy.

Including the kinematic model into the visual odometry system is not constrained to differential drive vehicles. It can be similarly applied to other non-holonomic vehicles with two degrees of freedom, e.g. Ackermann steered vehicles.

ACKNOWLEDGMENT

This work was supported by the BMBF project MobilAssist under grant number 01IS15049A.

References

[1] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[2] R. Mur-Artal and J. D. Tardos, "Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[3] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *Intelligent Robots and Systems (IROS), 2014*, Sept 2014, pp. 2661–2666.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981, pp. 674–679.

[5] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 1843–1848.

[6] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[7] S. Klose, P. Heise, and A. Knoll, "Efficient compositional approaches for real-time robust direct visual odometry from rgb-d data," in *Intelligent Robots and Systems (IROS), 2013*. IEEE, 2013, pp. 1100–1106.

[8] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, July 2016.

[9] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Robotics and Automation (ICRA), 2013*. IEEE, 2013, pp. 3748–3754.

[10] J. Zienkiewicz and A. Davison, "Extrinsics autocalibration for dense planar visual odometry," *Journal of Field Robotics*, pp. 803–825, 2014.

[11] S. Lovegrove, A. Davison, and J. Ibanez-Guzman, "Accurate visual odometry from a rear parking camera." IEEE, 2011, pp. 788–793.

[12] B. M. Kitt, J. Rehder, A. D. Chambers, M. Schonbein, H. Lategahn, and S. Singh, "Monocular visual odometry using a planar road model to solve scale ambiguity," in *Proc. European Conference on Mobile Robots*, September 2011.

[13] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *Robotics and Automation (ICRA), 2009.* IEEE, 2009, pp. 4293–4299.

[14] J. Jordan and A. Zell, "Ground plane based visual odometry for rgbd-cameras using orthogonal projection," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 108–113, 2016.

[15] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

[16] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Intelligent Robots and Systems, 2004. (IROS).*, vol. 1, Sept 2004, pp. 943–948.

[17] P. Huber, *Robust Statistical Procedures*. Society for Industrial and Applied Mathematics, 1996.

[18] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[19] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, 2012.

[20] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Intelligent Robots and Systems (IROS), 2014*. IEEE, 2014, pp. 4973–4980.

Managing a Fleet of Autonomous Mobile Robots (AMR) using Cloud Robotics Platform

Aniruddha Singhal, Prasun Pallav, Nishant Kejriwal, Soumyadeep Choudhury, Swagat Kumar and Rajesh Sinha
TCS Research, Tata Consultancy Services, New Delhi, India 201309.

***Abstract*—In this paper, we provide details of managing a fleet of autonomous mobile robots (AMR) using Rapyuta Cloud Robotics Platform. While the robots are themselves completely autonomous in its motion and obstacle avoidance capability, the target destination for each robot is provided by a global planner which itself may receive goals from an Enterprise Resource Planning (ERP) system. The global planner and the ground vehicles (robots) constitute a multi agent system (MAS) which communicate with each other over a wireless network. The complexities involved, and the corresponding benefits of implementing such a cloud based system is explained by comparing with two other implementations based on the standard distributed computing and communication framework of Robot Operating System (ROS). The working of the complete system is demonstrated through a real world experiment with physical robots in a laboratory setting. Through these implementations, the limitations of the current cloud framework is identified and critical suggestions are made for its improvement which, in turn, forms the future direction for this work.**

***Index Terms*—Fleet Management System, Multi-AMR control, Rapyuta, Cloud Robotics Platform, Robot Operating System, MAS**

I. Introduction

The industries and warehouses of the future will employ a fleet of autonomous robots working together to accomplish a common goal set by enterprise system with least human intervention. Such collaboration will be enabled by a cloud robotics platform [1] [2] which will allow the robots to communicate with other robots and humans over network, including Internet. A cloud infrastructure allows off-loading computationally expensive tasks onto a remote server thereby reducing the on-board power and computing requirements. This will reduce the cost of robots being deployed thereby making it possible to launch viable business solutions based on "Robotics-as-a-Service (RaaS)" framework. In this paper, we demonstrate one such use-case where a cloud robotics platform is used to manage a fleet of autonomous mobile robots (AMR) for carrying goods within a factory or a warehouse premises. The objective is to understand the underlying challenges of implementing such a multi-agent system using cloud robotics platform and suggest ways to address them.

A simplified version of a fleet management system for autonomous mobile robots (AMRs) is shown in Figure 1.

The authors are with the TCS Research, Tata Consultancy Services, New Delhi, India 201309.

E-mail:{aniruddha.singhal, prasun.pallav, nishant.kejriwal, soumyadeep.choudhury, swagat.kumar, rajesh.sinha}@tcs.com

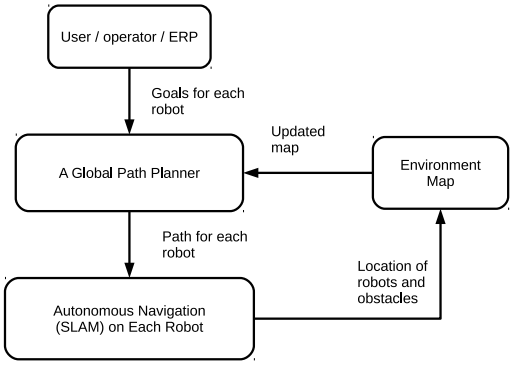


Fig. 1. Block diagram of a simplified fleet management system for autonomous vehicles.

It consists of a global planner which receives the target destination for each robot from an user or an operator or an ERP system. The planner generates paths from its current location to its destination using a path planning algorithm. The individual mobile robots receive the next target location from the global planner and uses its on-board SLAM algorithm to reach it destination. The autonomy of each robot is governed by the navigation module that implements SLAM (Simultaneous localization and mapping) as well as obstacle avoidance capabilities. The main focus of this paper is to understand the underlying software framework necessary for implementing such systems. To be specific, we provide details of three implementation in this paper. First two make use of the distributed control and communication framework of Robot Operating System (ROS) and the last implementation uses Rapyuta cloud robotics engine [1]. A comparative analysis of these approaches are carried out through simulation as well as real experiment which provides an insight into the underlying challenges, which if addressed, may increase the usability of the platform. Such an implementation based on ROS and Rapyuta is a novel contribution which has not been reported so far. These analyses help us in identifying critical flaws with existing systems and we suggest methods to overcome them, which forms the future direction for this work.

The rest of this paper is organized as follows. An overview of related work is provided in the next section. The three approaches of implementing fleet management system is described in Section II. The comparative performance analysis of these systems for simulation and actual experiments are

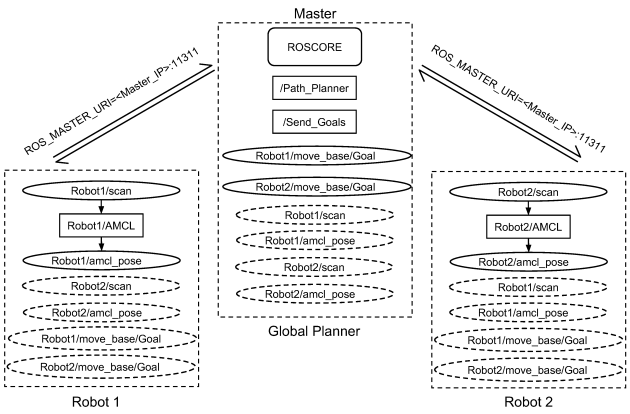


Fig. 2. The figure shows two robots connected to a third system which is the master running ROSCORE. The master also runs the global planner. Rectangular boxes show nodes, solid oval shows topics on the machine, and dashed oval shows topics available for subscription from other machines.

provided in Section III followed by conclusion in Section IV.

II. The Methods

In this section, we provide details of our implementation of a simplified fleet management system as shown in Figure 1. It primarily consists of four modules: (1) a user, an operator or an ERP system that provides goals or target destination for each robot, (2) a global planner that computes the path to be taken by each robot based on the current state of the environment (3) Autonomous Mobile Robots (AMR) having capability for autonomous navigation and obstacle avoidance; and (4) an environment map which could be updated with the information of new obstacles detected by the robots. The user is also free to update the availability of routes for any robot by creating obstacles in the environment map.

The above fleet management system is implemented using three methods: (1) single-master system, (2) multi-master system and (3) Cloud Robotics platform. The first two methods make use of the distributed computing and communication architecture of Robot Operating System (ROS) [3] while the last methods uses Rapyuta cloud robotics framework [1]. The details of each implementation and their respective pros and cons are presented next in this section.

A. Single Master System

In a single master system, `ROSCORE` runs on one machine which is called the master. Other nodes work in a distributed fashion on different machines. The nodes can run anywhere on the network except the driver nodes, which runs on the system that is directly connected to the hardware. All the nodes need to connect to the master. They connect via `ROS_MASTER_URI` which can be set in `.bashrc` file of the respective machines as shown below. All the machines in the network have a bidirectional connection with each other. Also, the host IP and the master IP will be same in case of the master machine.

Some of the common tasks like localization, mapping etc. runs on every robot resulting in nodes with same name under `ROSCORE`. A single launch file cannot be used to launch the nodes as it will create a conflict and the previous running node will be overridden with the new instance of the same name. This problem is resolved by introducing *namespace* and `tf_prefix` tags in the launch file.

The single master system can be set up by the following the steps: (1) Setup `.bashrc` in each robot as shown above. (2) Append suitable namespace and `tf_prefix` to the nodes corresponding to each robot. (3) Run `roscore` on the master. (4) Launch each individual robot.

A single master system is handy for quick testing of algorithms on a single robot because of its simple setup process. Its simplicity, however, does not provide much advantage as the number of robots increase in the environment. A schematic diagram of a working instance of single master system is shown in Figure 2. It shows one master running `roscore` and two client robots connected to the master over LAN. As one can see, all the topics from one robot is available for subscription by the all other robots as well. These topics are shown as dotted ellipse. The topics generated by the robot is shown as solid ellipses. Making topics available to everyone all the time may lead to some security concern as one would like to have some control over who can access which topics. In other words, this would require additional overhead to restrict access to the topics of a given robot by the other. Secondly, the bandwidth requirement for a single master system with multiple robots is comparatively higher as all the topics are available over the network for subscription. Moreover, having a single master makes the whole system vulnerable because if `roscore` dies, service based communication between the nodes get stopped. Topic based communication can still work because once a connection between nodes is established via topics, `roscore` is no longer needed, but new topics cannot be created without `roscore` running. Also, as the number of robots increase, it becomes increasingly cumbersome to deal with conflict among similar topics and namespace resolution.

B. Multi Master System

Many of the limitations of a single master system can be overcome by having multiple masters running their own independent `roscore` as shown in Figure 3. This makes the system robust as the failure of one will not lead to the failure of the complete system. Since the visibility of topics is limited to the scope of each `roscore` environment, there are no namespace conflict with topics in a multi-master system. All the nodes and services are local to that robot. However, it is possible to share a minimum number of topics with other robots through remapping as and when required. Since only a limited number of topics are shared, the bandwidth required in a multi-master system is less compared to that in a single master system for the same task.

To implement a multi-master System, a package called `multimaster_fkie` is needed and can be easily installed as shown below. This allows two important processes,

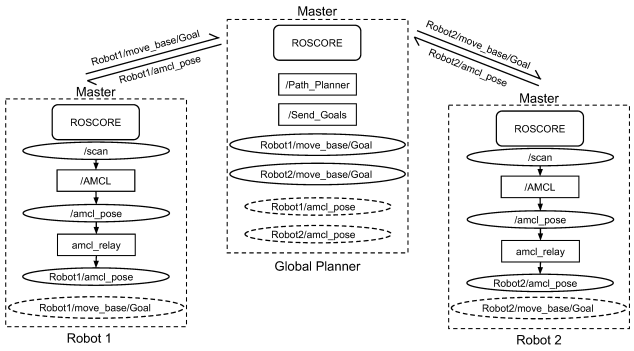


Fig. 3. A schematic view of a multi master system. The figure shows multiple `roscore`s running on different machines. In this configuration, there is no conflict among the topics with similar names as their visibility is limited to the machine running its own `roscore`.

`master_discovery` and `master_sync` to run simultaneously. The function of `master_discovery` is to send multicast messages to the network so that all `roscore` environments become aware of each other. It also monitors the changes in the network and intimates all ROS masters about these changes. The other process called `master_sync` enables us to select which topics can shared between different `roscore`. Without `master_sync` node, no information can be accessed by other `roscore`s.

It is to be noted that the host and master IPs are same on each machine. This is unlike the single-master case where these two IPs could be different for a given machine. The namespace conflict in multi-master system can be avoided using a *relay* node. The use of relay node can be understood in the context shown in Figure 3. The global planner needs to access pose data from Robot 1 and 2 for carrying out path planning. Each of these two robots publish pose data to a topic called `/amcl_pose` under their respective `roscore`s. To avoid conflict, one has to relay the `/amcl_pose` of Robot 1 to the topic `/Robot1/amcl_pose` and that of Robot 2 to `/Robot2/amcl_pose` respectively.

As shown in the above figure, the global planner can now access these new topics called `/Robot1/amcl_pose` and `/Robot2/amcl_pose` for obtaining their respective pose data.

Even though multi-master system saves us from several problems encountered in a single master system, it still does not provide solution to some other problems such as scalability, load balancing and lower computation power. As number of robots increase, one needs to reconfigure system files manually for each robot to enable multi-casting. It does not make efficient use of the processing power available because, by default, the processes are not distributed such that load on each machine is balanced. Bandwidth usage in multi-master system is still high compared to a cloud-based system due to the difference in network protocols used by different machines. In a multi-master system, each machine has a limited onboard computational hardware which can not be augmented to accommodate for higher demand in the run time. This limits the usability of multi-machine system.

C. Cloud Robotics System

Many of the limitations of a multi-master system can be solved by having a cloud infrastructure to which the robots can offload computationally heavy tasks. In this paper, Rapyuta cloud robotics engine [1] [4] is used for implementing the fleet management system. As discussed earlier, it is a Platform as a Service (PaaS) framework suitable for developing robotic applications. It includes four main components: (1) a cloud server which includes both software as well as hardware infrastructure; (2) Physical or simulated Robots and their working environment. (3) an user interface for interacting with the system and (4) an operator or an ERP system to provide goals for the system.

The inner working of this cloud-based implemented could be better understood by studying the Figure 4 that provides a process level overview of the system showing nodes, topics and interconnection pathways among various modules of the fleet management system. The figure shows a five agent system implemented using four physical machines (three robots and a server). Each robot runs processes for localization and autonomous navigation through nodes `/amcl` and `/move_base` respectively. The processes related to Rapyuta cloud robotics engine runs on the server machine. It also runs processes for global planner which generates paths for the robots. In a general scenario, the global planner and all related optimization algorithms can run on a separate physically machine on the network. Hence, it is shown as a separate block in the Figure 4 similar to the blocks corresponding to robots.

As shown in this figure, the global planner publishes data into two types of topics. The first topic is `/goalNodesList` which provides paths generated by the planner in the form of an array of grid block numbers. Each robot subscribes to its corresponding `goalNodeList` to know the cell locations that it needs to traverse. The second topic, called `/cancelGoal`, is a binary number which indicates whether the current goal locations received from the global planner is to be discarded by the robot or not. The binary value for the topic `/cancelGoal` for a given robot is set if a cell on its path is blocked either by an user or by an obstacle detected by the robot sensors. The grid cells could also be blocked by an ERP (Enterprise Resource Planning) system indicating non-traversable regions in the environment. Whenever the value for `/cancelGoal` is set, the robot discards previously received goal locations and uses new values available at the corresponding `/goalNodesList` topic. These topics are subscribed by the respective `move_client` nodes on the cloud which, in turn, publish necessary topics for use subscription by the physical robots.

Before going further, a brief understanding of Rapyuta organization will be useful for understanding the configuration steps described later. Rapyuta has the following four main components [1]. (1) *Computing environments* are the Linux

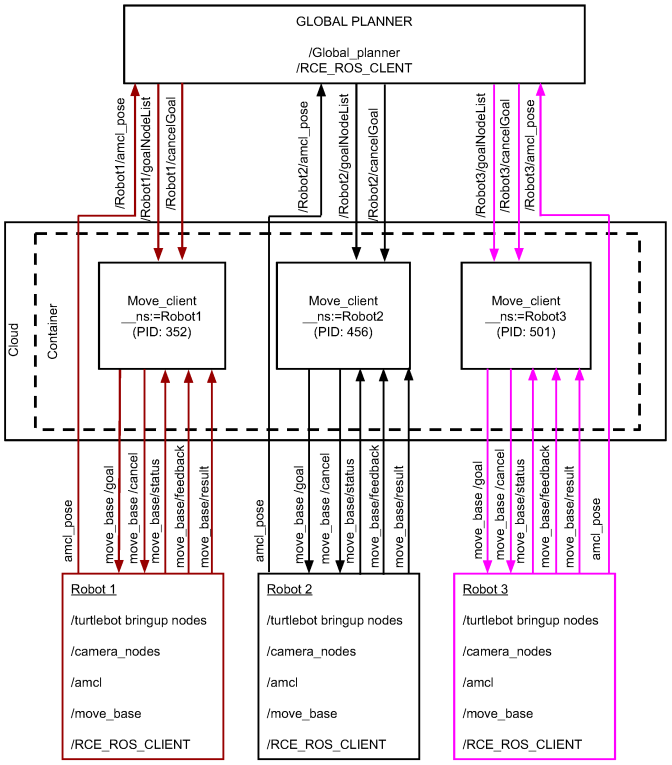


Fig. 4. The process nodes and topics required for implementing the fleet management system using Rapyuta cloud robotics engine. The system shows four agents (three robots and one global planner) interacting with each other through a cloud server. In this implementation, only a single container is used to execute all relevant processes. The arrow heads show the direction of information flow through topics between different nodes.

containers [5] [6] used for running various ROS based robot applications; (2) *Communication protocols:* are the standard protocols used for internal and external communication between cloud, container and robot processes. (3) *Core Task Set:* for managing all process and tasks. They are further divided into three groups, namely, robot task set, environment task set and container task set. (4) *Command Data Structures:* are the necessary formats used for various system administration activities.

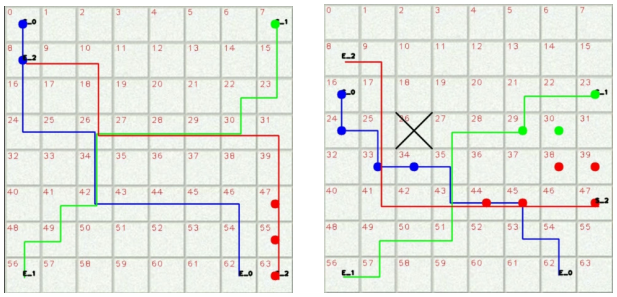
The setup process for the cloud robotics based fleet management system involves two main step:

- Create configuration files providing details of interaction between cloud and robots.
- Launch these files using system commands on server as well as robot clients.

In the remaining part of this section, we provide the details of configuration on server as well as the clients.

D. Global Planner

As discussed earlier, the global planner is responsible for generating paths for robots between their current locations and the target destinations provided by the operator. It receives the location information from each of the robots, the destination information for these robots from the operator and, uses the



(a) (b)

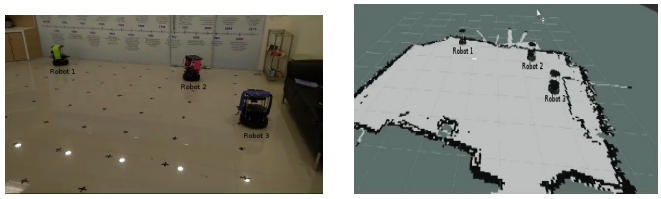
Fig. 5. Paths generated by Global Planner: (a) Paths for three robots obtained without any obstacles. Circular dots show the location of the robot as it traverses this path. (b) Shows new paths generated by the global planner once the user blocks the cell number 26. Grid cells can also be blocked when a robot detects an obstacle.

latest map to generate necessary paths for the robots. In its simplified form, it implements a Dijkstra algorithm [7] on a grid map to find shortest path between two cells as shown in Figure 5. In this figure, the robots are represented by filled circles. The start and end destinations of these robots are represented by the symbol pair $\{S_i, E_i\}, i = 1, 2, \ldots, N$ where $N = 3$ in this case. The Figure 5a shows the case when no obstacles are present in the map. As soon as the path information is transmitted to the robots, they start following their respective paths as shown by the trail of circular dots on their paths. The Figure 5b shows the case when an obstacle is created (or detected) in the cell number 26 at any time during this motion. This results in generation of new paths by the global planner. In a simulated environment, the robots can react instantaneously to this change. However, the robots may take some in a real world scenario due to factors like communication delay and inertia of motion as shown in this figure. The global planner may also include several other factors such as, battery life of robots, additional on-board sensor or actuator on robots (in case of a heterogeneous scenario) and other environmental conditions to solve a multi-objective optimization problem to generate these paths. Our purpose in this paper has been to demonstrate the working of a complete fleet management system which invariably requires such a centralized planner for task allocation and towards this end, we pick up the simplest path planner as an example. Readers are free to explore other planners in the same context.

III. Experiments

The details of simulation and real world experiment is discussed in this section. The simulation is carried out by spawning mobile robot models into a Gazebo environment. The Gazebo environment runs on one machine while the individual robot processes are made to run on other machines connected to each other over a wireless LAN. In the real world experiment, the robot models are replaced with actual Turtlebots in a lab environment as shown in Figure 6. The map of the environment is created by using Gmapping SLAM algorithm available with ROS. The map generated is shown in

Figure 6b. Each of the robots run AMCL-based localization algorithm to locate themselves in the map. Compared to existing methods that use markers embedded in the environment to guide themselves, we are considering a fleet of fully autonomous systems. It also runs an obstacle avoidance algorithm that uses on-board Kinect depth range information to locate obstacles on the path and avoid them. The map is divided into equispaced 8×8 grid to match with the grid up used by the global planner shown in Figure 5. For this lab experiment, we have selected the server and client machines with similar configurations with an Intel i7 processor with 8 GB on-board RAM. The complete video of the experiment [8] as well as the source codes [9] are made available online for the convenience of users. A more detailed version of this paper is published on Arxiv [10] for the benefit of readers.



(a) (b)

Fig. 6. (a) The actual experimental setup for fleet management system using AMRs. (b) Simulation environment showing the map generated using on-board SLAM algorithm.

A. Performance Analysis

In order to assess the relative performance of each of these three modes of implementation, the following experiment is performed. The experiment uses two physical machines in the network connected to each other through Wireless LAN. One of these machines publish images onto a topic which is subscribed by the other machine. The other machine simply echoes this data on a console. The second machine subscribing to the image publishing topic is considered as the server as it either runs a `roscore` process in the single master mode or a Rapyuta engine in the cloud robotics mode of operation. The relative performance of the machines is analyzed and compared in terms of CPU usage and network bandwidth usage as shown in Figure 7. The network usage is almost same in all the three cases as all of them use the same publishing rate and there are no other processes / nodes that generate additional network traffic. However, there is a difference in the CPU usage in these implementations. It is highest in Cloud Robotics mode of operation both on client as well as server side. This could be attributed to the additional computational overhead needed for running cloud processes. The multi-master system has the second highest CPU usage owing to the additional computation needed for running `master_discovery` processes and `master_sync` processes. Since none of these additional processes are there in the single master mode, the CPU usage is least in this case. These observations are in sync with our understanding of the systems as explained in the previous sections.

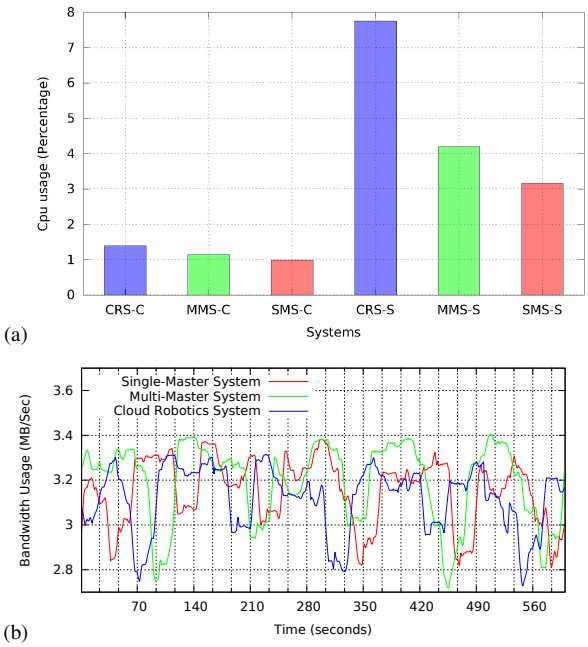


Fig. 7. (a) CPU usage and (b) network bandwidth usage for client and server in each of the three modes of operation.

B. Limitations and Future Work

The single-master and the multi-master ROS systems implement network robotics model based on Robot-to-Robot (R2R) communication framework. The problem of a single point of failure present in single-master mode is removed in the multi-master mode with a slight increase in the CPU and network usage. Both of these modes of operation suffer from limitation of resource and communication constraint as the on-board hardware capabilities can not be easily upgraded once deployed. They also suffer from scalability constraint as the performance deteriorates with increasing number of robots in the fleet. Many of these limitations are overcome in the cloud based PaaS systems such as Rapyuta, which implements Robot-to-Cloud (R2C) model. It still has several limitations as discussed below:

- In its current form, it does not offer *high availability* [11] for Rapyuta Master taskset and its failure leads to collapse of the whole system. This needs remediation by infrastructural mechanisms in combination with checkpoint-restart utilities [12].
- Of the five key characteristics of Cloud Services, the current implementation of Rapyuta PaaS lacks one, namely, the *elasticity*. It uses a cannibalized approach for all containers on a host to access compute, storage and network resources on the host machine and does not offer ability to allocate and resize these containers in the runtime to meet the varying workload demands. The utilities for monitoring the resource consumption are rudimentary and do not offer advice for migration of containers from one host to another or resizing.
- In the current implementation of the cloud platform,

there are no provisions for managing communication bandwidth to cater to different traffic situations. In practical scenarios for fleet management, having a logical segregation of communication bandwidth between control and data signals will improve the responsiveness of the R2C system. This is a concern when a remote teleoperation is required for an impaired mobile robot in a data centric network environment. Ability to leverage Multi-Path TCP [13] can also improve the transfer rates with R2C communication as it can make use of multiple interfaces to compensate for congestion in one of the channels.

- In a large warehouse of several thousand square feet area, it is possible that all mobile robots may not always have access to Cloud through the Cloud Access point. But with alternate communication modalities like Bluetooth, Zigbee or Wifi Direct - they may have connectivity to nearby robots which, in turn, may have access to the Cloud infrastructure. In such a scenario, a proxy-based [14] compute topology will be useful where one robot functions as a group leader to bridge the interaction between the set of nearby out-of-coverage robots and the cloud. The current Rapyuta implementation does not provide this topology and would require extensive changes to enable this. However, the other topologies such as clone-based or peer-based models are easier to implement with the current implementation and may be used along with ROS single-master or multi-master mode to simulate proxy-based systems.
- In the current implementation of Rapyuta framework - the partitioning of data and compute across three options - onboard compute on robot itself or robotic R2R network and/or Cloud execution has to be decided upfront and is usually static. Depending on the task with deadline, whether it is a SLAM, Navigation or Grasping task in warehouse, it would be useful to have a framework that can allocate these tasks to suitable compute resources (on edge / fog / cloud) in the run-time. Use of energy-efficient optimization algorithms [15] for task allocation and subsequent path planning and coordination have to be added on the top of Rapyuta platform for warehouse fleet management.

The directions for future work, therefore, include remediation of these limitations by developing additional layers and modules to support these functionalities.

IV. Conclusion

This paper presents the details of implementation of a fleet management system for a group of autonomous mobile robots (AMR) using three configurations: single-master, multi-master and cloud robotics platform. The mobile robots are completely autonomous as far as their navigation capabilities are concerned. These robots are required to traverse paths provided by a global planner. The global planner implements a basic path planning algorithm to generate paths between the current robot locations and the desired goal locations set by the operator, taking into account the obstacles which could be created dynamically in run time. The whole system can be controlled or monitored through a web-based user interface. The details of implementation for both simulation as well as actual experiment is provided which will be useful for students and practicing engineers alike. These details provide an insight into the working of each of the these modes of operation allowing us to identify the strengths and weaknesses of each one of them. These insights are further corroborated by analyzing parameters such as, network usage and CPU load. We also identify critical limitations of current cloud robotics platform and provide suggestions for improving them which forms the future direction for our work.

References

[1] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 481–493, 2015.

[2] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 398–409, 2015.

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[4] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 438–444.

[5] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support PaaS," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 610–614.

[6] A. M. Joy, "Performance comparison between linux containers and virtual machines," in *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*. IEEE, 2015, pp. 342–346.

[7] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2005, pp. 156–165.

[8] N. Kejriwal and P. Pallav, "Demonstration of cloud based fleet management system," https://www.youtube.com/watch?v=QMA6dnBweE0.

[9] Simulation Code, "Cloud robotics based fleet management system," https://gitlab.com/prasun2712/Cloud_Robotics_Simulated_Demo.

[10] A. Singhal, N. Kejriwal, P. Pallav, R. S. Soumyadeep Choudhury, and S. Kumar, "Managing a fleet of autonomous mobile robots (amr) using cloud robotics platform," *arXiv [cs.RO]*, no. 1706.08931, June 2017.

[11] J. Gray and D. P. Siewiorek, "High-availability computer systems," *Computer*, vol. 24, no. 9, pp. 39–48, 1991.

[12] O. Laadan and S. E. Hallyn, "Linux-cr: Transparent application checkpoint-restart in linux," in *Linux Symposium*. Citeseer, 2010, pp. 159–172.

[13] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath tcp development," Tech. Rep., 2011.

[14] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *Network, IEEE*, vol. 26, no. 3, pp. 21–28, 2012.

[15] A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, F. Leali, and S. Biller, "Modeling and optimization of energy consumption in cooperative multi-robot systems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 423–428, 2012.

Local Contextual Trajectory Estimation with Demonstration for Assisting Mobile Robot Teleoperation

Ming Gao and J. Marius Zöllner

***Abstract*— We focus on assisting mobile robot teleoperation in a task-appropriate way, where we model the user intention as an action primitive to perform a contextual task, *e.g.* doorway crossing and object inspection, and provide motion assistance according to the task recognition. This paper contributes to formulating motion assistance in a data-driven manner. With the motion clusters obtained in our previous report [1], we apply a fast online Gaussian Mixture Regression (GMR) approach to the most probable motion cluster classified during operation, to estimate the local trajectory the human user intends to follow in the short term for the corresponding task execution with the recognized contextual information. To regulate the estimation accuracy, we compute the Mahalanobis distance of each estimated trajectory way point. By thresholding the distance, we can achieve the trajectory estimation within a predefined tolerance bound regarding the regression outliers. The experimental results from both the qualitative and quantitative tests using the real data confirmed the effectiveness and real-time performance of the proposed approach.**

I. INTRODUCTION

The goal of our research work is to devise a shared autonomy system to assist mobile robot teleoperation, such as remote infrastructure inspection. In our previous work [1], we argued that the robot is supposed to assist the human operator in a *task-appropriate* way, and we proposed to learn the motion patterns of the human operator performing various contextual tasks, *e.g.* doorway crossing and object inspection, from demonstrations in an unsupervised manner into motion clusters. During operation, the learned mixture of motion experts are classified to recognize the motion behavior of the human user with the contextual information, *i.e.* the user inputs and the semantic components of the environment (*e.g.* candidate doorways to cross and objects for inspection).

To step further towards a contextual-task aware shared autonomy system, this report focuses on formulating motion assistance based on the motion pattern recognitions achieved with our previous study. As the major contribution of the paper, we apply a fast online GMR approach to the most probable motion cluster classified during operation, to estimate the local trajectory the human user intends to follow in the short term for the corresponding task execution with the associated contextual information (*i.e.* user inputs and the recognized target doorway or object). In this way, the motion assistance is formulated in a data-driven manner instead of being devised manually. To regulate the estimation accuracy, we compute the *Mahalanobis distance* of each estimated trajectory way point, since each motion cluster is *normally* distributed. By thresholding the distance, we can achieve the trajectory estimation within a predefined tolerance bound regarding the regression outliers. The estimated trajectory for the corresponding contextual task execution is *local*, because we model the user intention as an *action primitive* to perform a contextual task, aiming to provide immediate efficient motion assistance to the human operator *locally*. Moreover, contextual action primitives are more flexible and meaningful in describing the human behaviors than reactive actions on the lower level, which mostly pertains to either *obstacle avoidance* or *collision stopping*. Finally, the estimated local trajectory is to be provided to the state-of-art mobile robot motion controller as the *reference trajectory* to generate safe motion commands, which are blended with the user inputs according to the classification confidence, to achieve proactive motion assistance in a task-appropriate way. Fig.1 briefly illustrates our concept.

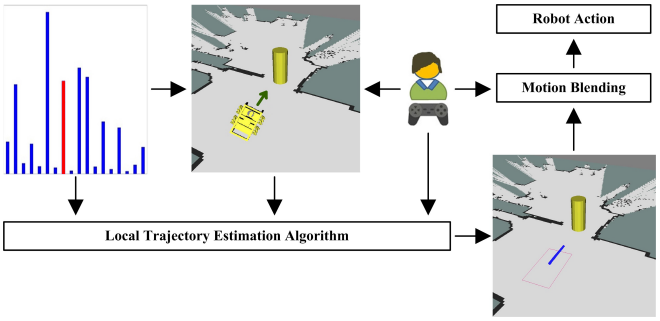


Fig. 1: In an example scenario, the robot is being operated to cross a doorway in front (middle), where the target doorway is recognized with high confidence indicated by the bar height. The proposed approach does regression on the classified motion cluster (the red bar) with the user input and the recognized semantic target to estimate the local trajectory (in blue) the human operator intends to follow in the next steps for the corresponding task execution (right). The estimated trajectory is to be employed by the state-of-art mobile robot motion controller to generate safe motion command, which is blended with the user input according to the classification confidence to achieve an adaptive task-aware motion assistance.

A very short version of this work was previously published in [2] as a two-page abstract. This paper significantly extends it by presenting the proposed approach and the related literature more thoroughly, and conducting more comprehensive evaluations including both qualitative and quantitative

Ming Gao (email: gao@fzi.de) and J. Marius Zöllner (email: Zoellner@fzi.de) are with the group of Technical Cognitive System (TKS), FZI Research Center for Information Technology, 76131 Karlsruhe, Germany.

experiments to investigate the performance of the proposed approach.

The remainder of this study is organized as follows. Sec. II introduces the related work. Sec. III details the methodology of the proposed approach. The experimental results and discussions are presented in Sec. IV. Finally, a conclusion and outlook section closes the report.

II. RELATED WORK

To assist mobile robot teleoperation, it is a major research topic to employ force/haptic devices to interact with the human operator [3], yet delicate control models have to be devised to efficiently merge user and robot motion inputs during operation. Another important research direction is on designing novel user interfaces to improve the situational awareness of the human operator [4]. In contrast, we propose to recognize the on-going tasks the human operator performs with the mobile robot using a mechanical joystick in a data-driven manner, and provide motion assistance to the user in a task-appropriate way.

Regarding providing task-aware motion assistance, work done in [5] reports a shared control system to assist the inspection of vertical infrastructure with a quadrotor. Due to the on-board sensing and partial autonomy of the robot, an unskilled user is able to safely operate it in close proximity to target structures. Work done in [6] introduces a shared autonomy system to actively assist the control of the flippers of a tracked vehicle based on the continuous three-dimensional terrain scanning. Both works consider only single task type for assistance, *i.e.* inspection of vertical infrastructure or rough terrain traversal. In contrast, our approach is able to recognize multiple contextual task types through learning the human motion patterns for various task executions from demonstrations. Work done in [7] presents a task inference and motion planning system to assist the teleoperation of a 6D robot manipulator using a 2D mouse. It proposes to employ data-driven approaches to learn and infer human motion patterns for task execution, which shares similarity with our approach. However, such work focuses on recognizing *freeform* tasks compared with our problem of estimating *contextual* tasks, which rely on the semantic components extracted from the environment.

Learning from demonstration is widely applied to formulate assistance for mobile robot operation (in both remote and local[1] situations) [8] [9], due to its intuitiveness and the availability of the user demonstration data. Work done in [10] proposes a shared control system to assist wheelchair driving by inferring the global trajectory the users intend to track from learning their operations. It assumes a set of candidate trajectories leading to certain semantic targets in the environment *beforehand*, and the inference afterwards is fixed to these candidates. In contrast, our approach focuses on estimating the local trajectory the human operator intends to follow in short terms from the demonstration data with the contextual information, without making any assumptions regarding the property of the trajectory *a priori*.

[1]Such as works proposed to assist wheelchair driving.

III. METHODOLOGY

This section begins with the introduction of the notations used in the following sections, which are closely related to our previous work. Then the proposed approach for local contextual trajectory estimation will be presented in detail.

A. Notation

$\mathbf{p} = (x, y, \theta)$ denotes the localized pose of the robot in the involved scenario, which also describes the state of the robot, since we focus on the ground mobile robot operated in indoor scenarios.

To utilize data-driven approach to learn human motion patterns for various task executions, we introduce *task feature* (denoted as $\mathbf{q}$) to describe a contextual task. $\mathbf{q}$ is computed with environmental information and user input $\mathbf{u}$. We use a mechanical joystick to issue $\mathbf{u}$ to control a ground holonomic mobile robot. Hence $\mathbf{u}$ is in the form of translational velocities along x and y axes, and rotational velocity around z axis in the local coordinate frame of the robot: $\mathbf{u} = (v_x, v_y, \omega)$. The environmental information is encoded with the *intentional target point* $\mathbf{s}$, which is extracted from the semantic components of the environment, and computed in the coordinate frame fixed on the robot center.

During demonstration, we obtain a set of task features $\mathbf{Q} = \{\mathbf{q}_n\}$ computed from the groundtruth semantic components of the scenario as the training data. The motion patterns are then learned into a set of motion clusters $\{\mathbf{Q}_c | c = 1, ..., M\}$, where $\mathbf{Q}_c = \{\mathbf{q}_n^c | n = 1, ..., N_c\}$ represents that the c-th motion cluster consists of N_c training task features, and M is the number of the clusters.

During operation, we classify each query task feature[2] to the learned motion clusters. By taking the maximal predictive probability across all clusters and query task features, we obtain both the classification result (*i.e.* the assigned most probable motion cluster c_{max}) and the associated semantic component. We use $p_{c_{\text{max}}}$ to denote the classification confidence[3].

The next part will detail the proposed data-driven algorithm for local contextual trajectory estimation to formulate the motion assistance.

B. Local Contextual Trajectory Estimation with Fast Online GMR

The fast online GMR approach was initially presented in our previous work [11], but it was aimed for contextual task recognition in a supervised learning framework instead of trajectory estimation by regression. In this report, we apply the approach to estimate the local trajectory the human operator intends to follow in the short term for corresponding task execution by regression with associated contextual information. Moreover, we extend the approach by considering the regulation of the estimation accuracy to suit our application.

[2]Each query task feature is computed with each candidate semantic component of the environment.

[3]Bold characters denote a vector, while un-bold ones represent a scalar.

For the purpose of presentational completeness, the fast online GMR approach is briefly summarized as follows. Given the learned training dataset $\mathbf{D}$ and the query value of the regressors $\mathbf{x}_t$ at time t, the state-of-art Fast Approximate Nearest Neighbor (FANN) algorithm is firstly applied to $\mathbf{D}$ with $\mathbf{r}_t$ to obtain a *small* and *local* database $\mathbf{D}(\mathbf{r}_t)$ consisting of the k points closest to $\mathbf{r}_t$. $\mathbf{D}(\mathbf{r}_t)$ is then employed by the Expectation-Maximization (EM) algorithm to train a Gaussian Mixture Model (GMM) with a very few number of the mixture components (denoted as h, which is usually 2 or 3). Finally, an online GMR model is derived accordingly from the trained GMM to predict the value of the dependent variables $\mathbf{y}_t$. The approach is listed in Alg. 1.

Algorithm 1 Fast Online GMR Algorithm

1: Given learned trained dataset $\mathbf{D}$
2: Given query value of the regressors $\mathbf{x}_t$ at time t
3: $\mathbf{D}(\mathbf{r}_t) \leftarrow$ by applying FANN algorithm to $\mathbf{D}$ with $\mathbf{x}_t$
4: $\mathrm{GMM}_{\mathbf{r}_t} \leftarrow$ by applying EM algorithm to $\mathbf{D}(\mathbf{r}_t)$
5: $\mathrm{GMR}_{\mathbf{r}_t} \leftarrow \mathrm{GMM}_{\mathbf{r}_t}$
6: $\mathbf{y}_t \leftarrow \mathrm{GMR}_{\mathbf{r}_t}(\mathbf{r}_t)$

With the fast online GMR approach, the proposed local trajectory estimation algorithm is listed in Alg. 2. In each iteration of the estimation, we firstly compute the query intentional target point $\mathbf{s}_*$ with the associated semantic component and the robot pose. Then we apply the fast online GMR approach to $\mathbf{Q}_{c_{\max}}$ with $\mathbf{s}_*$, to obtain the predictive motion command for the robot (*i.e.* the user input $\mathbf{u}_*$). The next way point the human operator intends to drive the robot to reach is computed by applying the robot kinematic model (RKM) with $\mathbf{u}_*$, the simulation time Δ_t and the current pose of the robot.

Although such estimation procedure can be iterated forever, to achieve certain prediction accuracy, we ought to terminate it reasonably. At each iteration, the corresponding task feature $\mathbf{q}_{\text{estimated}}$ is computed with the query intentional target point $\mathbf{s}_*$ and the estimated user input $\mathbf{u}_*$. Since each motion cluster is assumed to be *normally* distributed, its mean vector and covariance matrix can be easily computed beforehand. Thus we can obtain the Mahalanobis distance for $\mathbf{q}_{\text{estimated}}$ with respect to the assigned motion cluster $\mathbf{Q}_{c_{\max}}$ at each iteration. We use $\bar{\mathbf{q}}_{c_{\max}}$ and $\mathbf{\Sigma}_{c_{\max}}$ to denote the mean vector and the covariance matrix of the assigned motion cluster $\mathbf{Q}_{c_{\max}}$ respectively, the Mahalanobis distance d_{mh} for $\mathbf{q}_{\text{estimated}}$ is computed according to its definition as:

$$d_{\text{mh}} = \sqrt{(\mathbf{q}_{\text{estimated}} - \bar{\mathbf{q}}_{c_{\max}})^T \cdot \mathbf{\Sigma}_{c_{\max}}^{-1} \cdot (\mathbf{q}_{\text{estimated}} - \bar{\mathbf{q}}_{c_{\max}})}. \quad (1)$$

Mahalanobis distance [12] is a measure of the distance between a query point and a distribution. It is *unitless* and *scale-invariant*, and takes into account the correlations of the data set. It is a usual measurement to detect outliers in regressions. Hence by thresholding d_{mh}, we can terminate the trajectory prediction iteration within a predefined tolerance bound regarding the regression outliers. The choice of the fast online GMR approach for regression is advantageous because of its outstanding performance over the batch GMR algorithm (will be evaluated in the experimental section) and simplicity for hyper-parameters (k and h) tuning.

Algorithm 2 Local Trajectory Estimation with Associated Contextual Information

1: Given the classified motion cluster with the highest confidence $\mathbf{Q}_{c_{\max}}$
2: Given the semantic component associated with the most probable query task feature
3: Given the current robot pose $\mathbf{p}_r$
4: Given the RKM$(\mathbf{v}_t, \Delta_t, \mathbf{p}_t)$
5: initialize $\mathbf{p}_{\text{est}} \leftarrow \mathbf{p}_r$
6: initialize Traj_Pred $\leftarrow \{\}$
7: initialize $d_{\text{mh}} \leftarrow 0$
8: **while** $d_{\text{mh}} < d_{\text{mh_thres}}$ **do**
9: $\quad \mathbf{s}_* \leftarrow$ with the associated semantic component and $\mathbf{p}_{\text{est}}$
10: $\quad \mathbf{u}_* = (v_x^*, v_y^*, v_\omega^*) \leftarrow$ by applying the fast online GMR approach (Alg. 1) to $\mathbf{Q}_{c_{\max}}$ with $\mathbf{s}_*$
11: $\quad \mathbf{q}_{\text{estimated}} \leftarrow \mathbf{s}_*$ and $\mathbf{u}_*$
12: $\quad d_{\text{mh}} \leftarrow$ Eq. 1 with $\mathbf{q}_{\text{estimated}}$
13: $\quad \mathbf{p}_t \leftarrow \mathbf{p}_{\text{est}}$
14: $\quad \mathbf{p}_{\text{est}} \leftarrow \text{RKM}(\mathbf{u}_*, \Delta_t, \mathbf{p}_t)$
15: $\quad$ Traj_Pred $\leftarrow$ Traj_Pred $\cup \{\mathbf{p}_{\text{est}}\}$
16: **end while**
17: **return** Traj_Pred

We aim to provide the estimated trajectory to the state-of-art mobile robot motion controller (*e.g.* work done in [13]) as the *reference trajectory*, to obtain safe motion commands (denoted as $\mathbf{g}_t$) towards task execution. $\mathbf{g}_t$ is blended with $\mathbf{u}_t$ according to the classification confidence $p_{c_{\max}}$ to achieve proactive motion assistance:

$$\mathbf{v}_t = (1 - p_{c_{\max}})\mathbf{u}_t + p_{c_{\max}}\mathbf{g}_t. \quad (2)$$

Consequently, the level of autonomy is seamlessly switched between *manual control* (when $p_{c_{\max}}$ is extremely low) and *autonomous control* (when it is tremendously high) in a task-appropriate way during operation, realizing an adaptive contextual-task aware shared autonomy system for assisting mobile robot teleoperation.

IV. EXPERIMENTAL RESULTS

A. Test Configuration and Datasets

In this section, the evaluation focus is the performance of the proposed approach on estimating local trajectories with demonstrations and associated contextual information. Regarding the investigation of the effect of motion assistance in mobile robot teleoperation, we set it as our future work.

We employed the same sensor-equipped holonomic mobile robot as in [1] to evaluate our approach in indoor scenarios. It uses a 2D laser scanner to perceive the environment. We adopted a Logitech F710 wireless gamepad to control the robot. The data sampling rate was 20Hz, which was the

sampling rate of the joystick. The average speed of the robot during the evaluations was approximately $0.3m/s$.

For the convenience of providing demonstrations and data analysis (*e.g.* to label the groundtruth semantic targets), without the loss of generality of the proposed approach, we built the maps of the involved scenarios using a state-of-art SLAM implementation within ROS framework, and processed them to extract the required semantic components beforehand employing the works presented in [14], including: the center points of the candidate doorways, and the surface points of the candidate object and wall segments, respectively.

We employed the motion clusters obtained in [1], and trained the classifier to classify the motion clusters and recognize the associated semantic components along the test trajectories according to [1]. There were three sources to obtain the test trajectories, and all the test trajectories were recorded from test participants manually driving the robot for certain predefined (yet unknown to the robot) contextual tasks with random initial poses.

The first source was the *nine* trajectories collected from the human operator sequentially performing three contextual tasks for nine times: firstly following the wall segment, then inspecting an object, finally crossing a doorway. Such source contained the four contextual task types used for clustering, *i.e.* Doorway Crossing, Object Inspection, Wall Following and Object Bypass, and it was sampled from the evaluations made in [11]. The second source was the *six* trajectories collected from the human operator performing each of the three contextual tasks: Wall Inspection, Robot Docking and Gap Crossing, for two times, and it was sampled from the evaluations made in [1]. These three task types were not demonstrated for clustering, but they share motion similarity with the learned ones. Hence the aim of this source was to evaluate the generalizability of the proposed approach. The last source was the *five* trajectories collected in the same scenario as that used in the second source. They were recorded from two volunteers manually driving the robot[4] to execute a set of contextual tasks along the way in the cluttered scenario, *e.g.* crossing a narrow gap, inspecting an object or a wall segment, bypassing an object and docking into a table, with the aim of comprehensively evaluating the proposed algorithm.

Totally, there were *twenty* test trajectories, consisting of 15575 way points obtained from different human operators controlling the robot to perform seven contextual task types[5] in different scenarios. Especially, they contained the samples collected from executing certain task types in alternative ways, *e.g.* object inspection (in either clockwise or counter-clockwise direction regarding the target object during inspection), wall following (to follow the wall segment on either side of the robot), and object bypass (to pass the target object by either side of the robot). Thus these twenty trajectories are the appropriate test datasets to evaluate the *scalability* and *effectiveness* of the proposed approach.

Regarding the parameters of the proposed approach, throughout the following evaluations, the simulation time for extrapolation was set $\Delta_t = 0.05s$ (*i.e.* 20Hz) to be the same as the sampling period of the dataset, and the threshold of the Mahalanobis distance of each estimated way point was set $d_{\text{mh_thres}} = 3.0$ empirically. The fast online GMR approach used $h = 2$ Gaussian components for regression, and employed diagonal covariance matrix for each Gaussian component to simplify computation. Its training data size was $k = 10$.

To comprehensively evaluate the proposed local contextual trajectory estimation approach, the following tests are grouped into two parts: the qualitative evaluations and the quantitative ones.

B. Qualitative Evaluations

In the qualitative evaluations, we apply the proposed approach to certain way points of several test trajectories, to estimate local trajectories originating from them. We draw together the estimated and the groundtruth trajectory segments starting from the same poses and possessing approximately the same length[6], aiming to provide the visual comparisons between the estimations and the groundtruth results. In the following comparison figures, apart from the two trajectory segments with different colors, a bar is put on the most probable semantic target recognized by the classifier at the applied way point, with its height indicating the classification confidence. The footprint of the robot is also denoted with a pink polygon in these figures.

Firstly, we show the performance of the proposed approach on estimating the four motion patterns (including their possible variations) used for training. Fig. 2 illustrates the four groundtruth trajectories adopted for these qualitative evaluations.

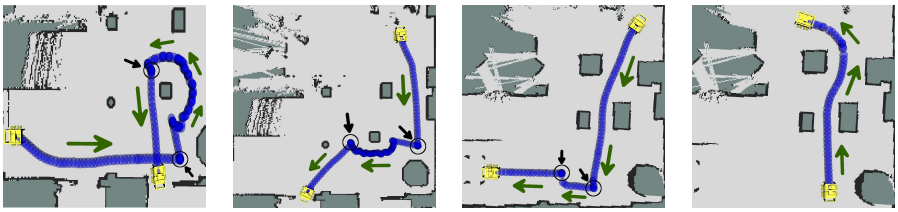
For *Doorway Crossing*, Fig. 3 depicts the comparison done at one way point of the trajectory in Fig. 2(a). As can be noticed, the estimated trajectory matches the groundtruth one quite well, and the correct semantic target is also recognized with high confidence, demonstrating qualitatively the performance of the proposed approach in estimating this motion pattern.

For *Object Inspection*, after approaching the target object, there are two operational directions to inspect it, *i.e.* in either clockwise or counter-clockwise direction regarding the target object. Fig. 4(a) compares the groundtruth trajectory segment (from the trajectory in Fig. 2(a)) with the one estimated at the way point where the human operator was driving the robot to approach the target object. Since the motion pattern of *approaching object* has been learned from demonstrations, the execution of it can be captured with high confidence and accuracy. With respect to the inspection phase, Fig. 4(b) and Fig. 4(c) draw the comparisons done at the two way points

[4]Two test trajectories were obtained from one volunteer, while the rest three were collected from the other one.

[5]Doorway Crossing, Object Inspection, Wall Following, Object Bypass, Robot Docking, Wall Inspection and Gap Crossing

[6]The length of a trajectory is obtained by accumulating the Euclidean distances (computed with x and y coordinates) between its ordered pairs of way points.



(a) (b) (c) (d)

Fig. 2: The groundtruth trajectories for qualitatively evaluating the proposed approach on estimating the four motion patterns used for training, where the robot was manually operated to execute various contextual tasks in cluttered scenarios. The black arrows denote the manually labeled split points of the sequentially performed tasks (when applicable), while the green ones indicate the movement directions of the robot. (a) A sequence of tasks were executed: the robot followed wall segment on its right side at first, then inspected target object in counter-clockwise direction, finally crossed target doorway. (b) Similarly, the robot followed wall segment on its right side at first, then inspected target object in clockwise direction, finally crossed target doorway. (c) The robot moved among obstacles at first, then inspected wall segment in one direction, finally docked into a table. (d) The robot moved among obstacles to reach the other side.

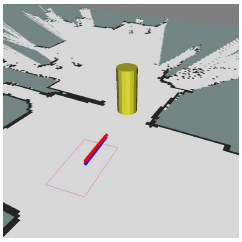
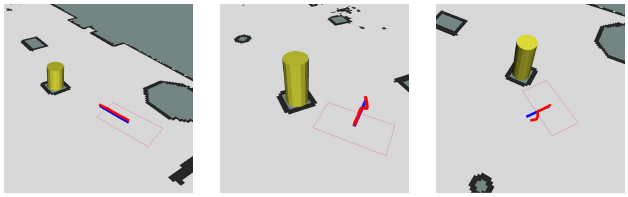


Fig. 3: The estimated trajectory (blue) is compared with the groundtruth segment (red), where the robot was being operated to cross doorway. The pink polygon represents the footprint of the robot. A bar is put on the most probable semantic target estimated by the classifier at this way point, with its height indicating the estimation confidence.

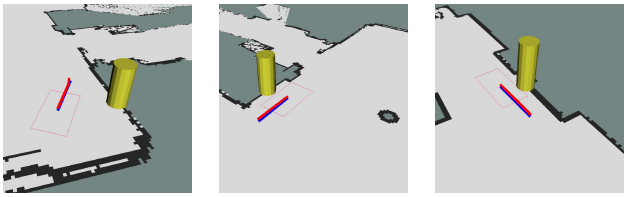


(a) (b) (c)

Fig. 4: The estimated trajectories are compared with the groundtruth segments, where the human operator was driving the robot to execute object inspection task. The color coding is the same as Fig. 3.

(from the trajectories in Fig. 2(a) and Fig. 2(b)) where the robot was moving along the target object while facing it in counter-clockwise and clockwise directions respectively. In both situations, the human motion patterns are correctly interpreted and estimated with high confidence, despite the jitters of the recorded groundtruth way points resulting from the strong motion drifts of the robot when moving sideways on the uneven floors of the scenarios.

Regarding *Wall Following*, the robot can follow the target wall segment on either right or left side of it. Fig. 5(b)



(a) (b) (c)

Fig. 5: The estimated trajectories are compared with the groundtruth segments, where the human operator was driving the robot to perform wall following task. The color coding is the same as Fig. 3.

and Fig. 5(c) depict the comparisons done at the two way points (from the trajectories in Fig. 2(a) and Fig. 2(b)) where the robot was following the target wall segment on its right and left sides respectively. As displayed in both figures, the results of the semantic target recognition and the local trajectory estimation qualitatively demonstrate that, the human motion intentions are correctly interpreted and predicted with high confidence in both situations. When executing Wall Following, the human operator usually controls the robot to approach the target wall segment while aligning the robot with its surface, especially when the robot is noticed to be not close enough to the wall. Such motion pattern is learned in the motion clusters from the demonstrations. Hence its execution can be correctly recognized during operation, and the corresponding local trajectory is estimated with high accuracy, as displayed in Fig. 5(a).

Regarding *Object Bypass*, the human operator can execute it in alternative directions, *i.e.* to pass the target object by either left or right side of the robot. Fig. 6(a) illustrates the comparison (applied to the trajectory in Fig. 2(c)) where the robot was avoiding the object on its right side, while Fig. 6(b) depicts the comparison (applied to the trajectory in Fig. 2(d)) where the robot was bypassing the object on its left side. As can be viewed, the task motions of the human operator in both situations are correctly interpreted and predicted

with high accuracy. Interestingly, at some way points when performing Object Bypass, the estimated trajectories are noticed to aim to guide the robot further away from the nearby obstacle then the groundtruth trajectory segments, as shown in Fig. 6(c) and Fig. 6(d), respectively. We deduce that, by learning from demonstrations, the proposed approach interprets such situations more conservatively than the human operator actually executing this movement from the perspective of robot safety. How such action *deviation* between human and robot will influence the human-robot interaction within a shared autonomy system in reality remains an attractive point for further investigation.

The above qualitative evaluation results comprehensively confirm the performance of the proposed approach on estimating the motion patterns (including their possible variations) used for training. The following qualitative evaluations are focused on investigating the performance of the proposed approach on estimating the motion patterns (including their possible variations) not employed for training, yet sharing motion similarity with the learned ones. Fig. 7 depicts the four groundtruth trajectories used for the following qualitative evaluations.

For *Gap Crossing*, Fig. 8(a) draws the comparison (applied to the trajectory in Fig. 7(a)) where the robot was being controlled to cross a gap between two obstacles. As can be seen, at this way point, the classifier correctly recognizes the semantic target with high confidence, and the estimated local trajectory matches the groundtruth one quite well. Fig. 8(b) illustrates another comparison applied to the same trajectory. Although the recognized semantic target is incorrect, the estimated trajectory still matches the groundtruth one quite well. We further examine the motion cluster assigned to the starting way point, and find that it consists of the demonstrations from Wall Following and Object Bypass, hence we deduce that the proposed approach interpreted the motion pattern at this way point as to bypass the target object, which is reasonable for this situation.

For *Robot Docking*, Fig. 9 depicts the comparison (applied to the trajectory in Fig. 7(b)) where the robot was docking into a table in front. The semantic target is correctly recognized with high confidence, and the local trajectory is estimated with high accuracy. Moreover, after checking the motion cluster assigned to the starting way point, we find that it mainly consists of the demonstrations from Doorway Crossing. Therefore, we posit that the proposed approach interprets the motion pattern at this way point by generalizing from the learned motion pattern of Doorway Crossing.

Regarding *Wall Inspection*, it can be executed in alternative directions after the robot approaching the target wall segment. The comparisons depicted in Fig. 10(b) (applied to the trajectory in Fig. 7(d)) and Fig. 10(c) (applied to the trajectory[7] in Fig. 7(c)) verify the very accurate estimation results of the proposed approach in both situations. Meanwhile, the human motion pattern of driving the robot to approach the target wall segment is also estimated with high confidence and accuracy, as displayed in Fig. 10(a).

The above qualitative evaluation results prove the performance of the proposed approach in generalizing to the task types (including their possible variations) not used for training. In summary, the overall qualitative results in this subsection verify the scalability and effectiveness of not only the proposed local trajectory estimation approach, but also the obtained motion clusters and classifier. The next subsection will present the quantitative evaluation results with the baseline approaches.

C. Quantitative Evaluations

To further quantitatively evaluate the performance of the proposed approach, we employ the batch GMR and the Locally Weighted Projection Regression (LWPR) algorithm [15] as the baseline regression approaches. LWPR is a popular machine learning algorithm seeking to provide incremental, real-time inference and prediction for high-dimensional input-output function approximation. Sharing the similar aim, it is poised to compete with the fast online GMR approach. We use the library proposed in [16] for its implementation, and tune its (hyper-)parameters according to the suggestions introduced in [17].

The batch GMR model uses full covariance matrix, and it is trained by the EM method with up to ten components on each learned motion cluster beforehand. For each motion cluster, the optimal model is selected among the candidate ones based on the Bayesian Information Criterion (BIC). The LWPR model is also trained for each motion cluster in advance, and during training, $\omega_{\text{gen}} = 0.2$, an initial setting of $d_* = 2.0$ and blending is enabled. The estimations made by both algorithms are also thresholded with $d_{\text{mh_thres}} = 3.0$.

The three approaches are applied to estimate the local trajectory at each way point of all test trajectories respectively. To characterize the estimation error in a straightforward manner[8], at each way point for examination, we compute the *pairwise Euclidean distance* between the estimated trajectory and its corresponding groundtruth one having the same number of way points[9]. Finally, the means and the standard deviations of such error measurement are obtained for the three approaches over all test trajectories respectively. They are shown in Fig. 11. The results indicates that the fast online GMR approach outperforms the baseline approaches in this test.

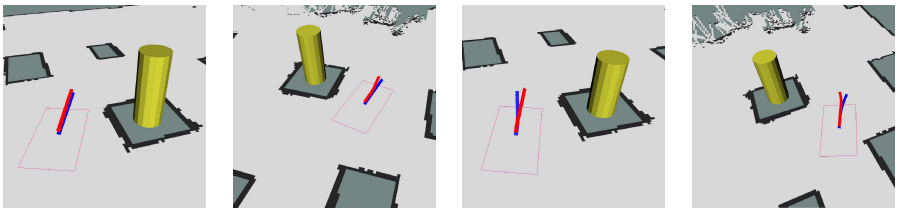
In addition to the estimation accuracy, the estimation speed of the proposed approach is also our concern, since it performs the learning and estimation online during operation. In the application of this study, the required process speed[10] corresponds to $\sim$ 30Hz. Because in real applications, the estimation is made after the motion cluster classification

[7]Please notice the jitters of the recorded way points in this trajectory due to the strong motion drifts of the robot when moving sideways on the uneven floor of the scenario.

[8]For more information regarding trajectory similarity comparison, please refer to [18].

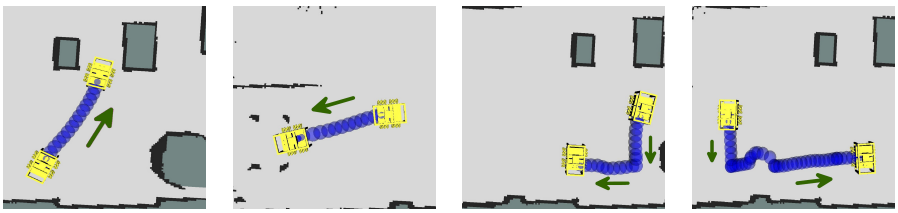
[9]The distance between two way points is computed with $\mathbf{p} = (x, y, \theta)$

[10]At least, it should be higher than the data sampling period which is 20Hz



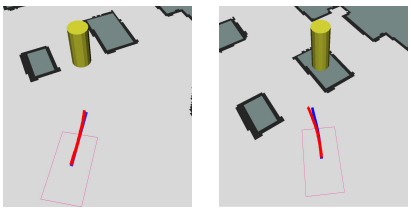
(a) (b) (c) (d)

Fig. 6: The estimated trajectories are compared with the groundtruth segments, where the human operator was controlling the robot to perform object bypass task. The color coding is the same as Fig. 3.



(a) (b) (c) (d)

Fig. 7: The groundtruth trajectories for qualitatively evaluating the proposed approach on estimating the three motion patterns not demonstrated during training. The color coding is the same as Fig. 2. (a) The robot was crossing a gap between objects. (b) The robot was docking into a table. (c) The robot approached wall segment at first, then inspected it in one direction. (d) The robot inspected wall segment in another direction after approaching it.



(a) (b)

Fig. 8: The trajectories estimated at two way points are compared with the groundtruth segments, where the robot was being operated to cross a gap between obstacles. The color coding is the same as Fig. 3.

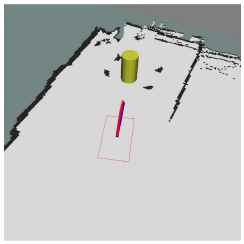


Fig. 9: The estimated trajectory is compared with the groundtruth segment, where the robot was being controlled to dock into a table. The color coding is the same as Fig. 3.

and semantic target recognition, it is meaningful to evaluate the process speed of the whole pipeline (*i.e.* motion cluster classification, semantic target recognition and local trajectory estimation). Towards this aim, the three regression approaches combined with the classifier are applied to each way point of all test trajectories respectively.

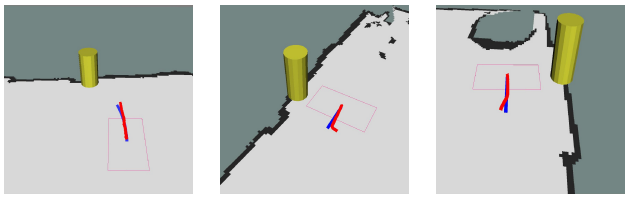
At each way point for evaluation, the process time of each combination consists of the time for motion cluster classification and semantic target recognition by the classifier and the time for trajectory estimation by the corresponding regression approach[11]. Such process time of each combination is then recorded at each test way point respectively. In the end, we obtain the means and the standard deviations of the process time of the three combinations per way point over all test trajectories respectively, as shown in Fig. 12. The results denote that the combination using the proposed online GMR approach achieves the fastest process speed (approximately 55Hz). Though it is not considerably faster than the baseline approaches, no pre-training is needed to employ the proposed online GMR approach. Such results further verify that the winning combination satisfies the real-time requirement of the considered application.

In summary, the above quantitative evaluation results confirm the performance of the proposed approach in local trajectory estimation, especially its real-time property in data processing to efficiently learn and make predictions with high accuracy.

V. CONCLUSIONS

This paper reported a novel approach to estimate the local trajectory the human operator intends to follow in the short term for certain contextual task execution, with the aim of assisting mobile robot teleoperation by sharing autonomy in

[11]The classifier, the batch GMR model and the LWPR model are trained *off-line* on each motion cluster, thus the training time of them is not considered in the process time.



(a) (b) (c)

Fig. 10: The estimated trajectories are compared with the groundtruth segments, where the human operator was driving the robot to execute wall inspection task. The color coding is the same as Fig. 3.

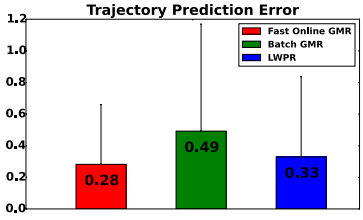


Fig. 11: The means and the standard deviations of the trajectory estimation errors for the three regression approaches.

a task-appropriate way. We proposed to apply a fast online GMR approach to the most probable motion cluster classified during operation, to estimate the trajectory way points by regression with the recognized contextual information. In this way, the motion assistance is formulated in a data-driven manner instead of being devised manually. To calculate and threshold the Mahalanobis distance computed with each estimated way point, it is able to achieve the trajectory estimation within a predefined tolerance bound regarding the regression outliers. The experimental results from both the qualitative and quantitative tests using the real data confirmed the effectiveness and real-time performance of the proposed approach.

The estimated trajectory is to be used by the state-of-art mobile robot motion controller as the reference trajectory to generate safe motion commands, which are blended with the user inputs based on the motion classification confidence to realize a contextual-task aware shared autonomy framework. This is the ultimate goal of our research work. Therefore, a comprehensive user study to investigate the overall performance of such framework is our significant future work afterwards.

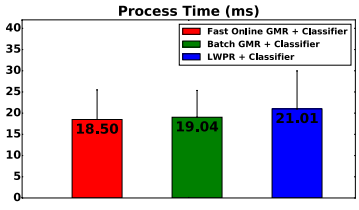


Fig. 12: The means and the standard deviations of the process time of the three prediction combinations at each test way point.

ACKNOWLEDGEMENTS

The authors are grateful for all test participants, and would like to thank the reviewers for their constructive comments which help improve the quality of this report.

REFERENCES

[1] M. Gao, R. Kohlhaas, and J. M. Zöllner, "Unsupervised Contextual Task Learning and Recognition for Sharing Autonomy to Assist Mobile Robot Teleoperation," in *Informatics in Control, Automation and Robotics (INCO), 2016 13th International Conference on*, 2016.

[2] M. Gao, R. Kohlhaas, and J. M. Zöllner, "Contextual learning and sharing autonomy to assist mobile robot by trajectory prediction," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 274–275.

[3] N. Yu, K. Wang, Y. Li, C. Xu, and J. Liu, "A haptic shared control algorithm for flexible human assistance to semi-autonomous robots," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5241–5246.

[4] P. Marion, M. Fallon, R. Deits, A. Valenzuela, C. Pérez D'Arpino, G. Izatt, L. Manuelli, M. Antone, H. Dai, T. Koolen *et al.*, "Director: A user interface designed for robot operation with shared autonomy," *Journal of Field Robotics*, vol. 34, no. 2, pp. 262–280, 2017.

[5] I. Sa, S. Hrabar, and P. Corke, "Inspection of pole-like structures using a visual-inertial aided vtol platform with shared autonomy," *Sensors*, vol. 15, no. 9, pp. 22 003–22 048, 2015.

[6] Y. Okada, K. Nagatani, K. Yoshida, S. Tadokoro, T. Yoshida, and E. Koyanagi, "Shared autonomy system for tracked vehicles on rough terrain based on continuous three-dimensional terrain scanning," *Journal of Field Robotics*, vol. 28, no. 6, pp. 875–893, 2011.

[7] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, vol. 35, no. 4, pp. 241–254, 2013.

[8] T. Witzig, J. M. Zollner, D. Pangercic, S. Osentoski, R. Jakel, and R. Dillmann, "Context aware shared autonomy for robotic manipulation tasks," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5686–5693.

[9] H. Soh and Y. Demiris, "Learning assistance by demonstration: Smart mobility with shared control and paired haptic controllers," *Journal of Human-Robot Interaction*, vol. 4, no. 3, pp. 76–100, 2015.

[10] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin, "User-adapted plan recognition and user-adapted shared control: A Bayesian approach to semi-autonomous wheelchair driving," *Autonomous Robots*, vol. 24, no. 2, pp. 193–211, 2008.

[11] M. Gao, T. Schamm, and J. M. Zöllner, "Online and Incremental Contextual Task Learning and Recognition for Sharing Autonomy to Assist Mobile Robot Teleoperation," in *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*. IEEE, 2015.

[12] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[13] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[14] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1019–1026.

[15] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[16] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *Journal of Machine Learning Research*, vol. 9, no. Apr, pp. 623–626, 2008.

[17] S. Klanke and S. Vijayakumar, "Lwpr supplementary documentation," Tech. Rep, Tech. Rep., 2008.

[18] K. Toohey and M. Duckham, "Trajectory similarity measures," *SIGSPATIAL Special*, vol. 7, no. 1, pp. 43–50, 2015.

Controllers Design for Differential Drive Mobile Robots based on Extended Kinematic Modeling

Julio C. Montesdeoca Contreras[1, 2], D. Herrera [2], J.M. Toibero[2], R. Carelli[2]

***Abstract*—This paper presents the simulation results of the controllers design for differential-drive mobile robot (DDMR) using a novel modeling method, which is based on the inclusion of the sideway velocity into the kinematic modeling, in order to obtain a holonomic-like model. Next, non-holonomic constraint is introduced assuming that the sideway slipping is measurable. The controller design considers a variable position for the point of interest and takes into account also the robot constrained inputs. The obtained inverse kinematics controller is differentiable, time invariant and naturally incorporates the sideway slipping which is considered measurable. Moreover, the proposed controller can be used for both: trajectory tracking and path following by setting appropriate desired values at the planning stage. The Lyapunov theory is used to prove the stability of the control system. Simulator includes a robot dynamics module that supports physics engines. Obtained simulations results show a high performance for both tasks.**

I. INTRODUCTION

Nowadays, the non-holonomic wheeled mobile robots (WMRs) have an important role in applications such as, rescue, agriculture, exploring, healthy care, indoor simultaneous localization and mapping. A non-holonomic system means that the controllable variables number are less than the states of system number [1] [2]. Differential-drive mobile robots (DDMRs) have two independient drivable wheels mounted along a common axle. As it is well known, the classical model of DDMR is based on the non-holonomic constraints of pure rolling and non-slipping [3] [4].

In this context, the bibliographic review presents several DDMR kinematic controllers using this classical non-holonomic model. For instance, Lafferriere and Sussmann [5] present a general solution for motion planning based on the Lie algebra and considering a chained-form system. Samson and Ait-Abderrahim [6] present the trajectory tracking solution with controller restrictions when the reference model is motionless. Kanayama *et. al* [7] propose using a reference model to generate the desired control inputs actions for the trajectory tracking problem; they provide a Lyapunov based stability proof. On the other hand, Jiang and Nijmeijer [8] and Chwa [9] present a stable time-variant backstepping based controller considering a reference model. Moreover, Chwa includes constrained control inputs into the controller design. In all the aforementioned works, the point of interest is located at the robot center of mass (CoM) in the middle point of the wheels axle.

[1] Universidad Politécnica Salesiana, Cuenca, Ecuador, jmontesdeoca@ups.edu.ec

[2] Instituto de Automatica, CONICET / Universidad Nacional de San Juan, San Juan, Argentina, {jmontesdeoca,dherrera,mtoibero,rcarelli}@inaut.unsj.edu.ar

Next, Diaz and Kelly [10] present the classical kinematic model but considering the point of interest displacement from the CoM into the robot's body. The obtained controller addresses the trajectory tracking problem including the associated stability analysis proving its asymptotic stability based on linear system theory. This work, however, does not consider constrained control inputs.

None of the aforementioned works present a generalized time-invariant controller with a variant position of the point of interest, that solve both, the following path and the trajectory tracking problem; neither the constrained control inputs are taking into account, using the same controller.

Therefore, this paper proposes a novel controller for the non-holonomic DDMR using the extended kinematic model (EKM). Hence, the objective of the EKM is to obtain a *holonomic-like* model for the non-holonomic DDMR, which will be useful to propose a differentiable, time-invariant and easy to implement controller based on inverse-kinematics. Since real WMRs have constrained control inputs, continuous saturation functions are included in the auxiliary control law to get the appropriate control inputs for the DDMR. Another important characteristic of this proposal relays in the fact that naturally includes the sideway slipping into the controller. In this way, the obtained controllers can be used for both trajectory tracking or path following by selecting appropriate desired references at the planning stage. At present, many DDMR applications require that the point of interest (ip) has a desired position, not necessary at the robot CoM, taking this into account, this work presents the EKM considering the point of interest in an arbitrary position, due to a mathematical indetermination appears when the desired position of the point of interest is over the sideway axle, another EKM is proposed to solve this special case. Finally, the asymptotic stability proof is reported by considering Lyapunov theory.

This document is structured as follows: Section II presents the extended kinematic modeling of differential-drive mobile robot considering the point of interest ip in an arbitrary position. Section III describes the design of an auxiliary control law. Next, section IV describes the obtained closed-loop system and the associated Lyapunov based stability proof. Section V presents the methodology to propose the desired values in order to have a trajectory tracking behavior or a path following behavior. Section VI presents simulation results using the virtual robot experimentation platform (v-rep) [11], the section VII proposes future work and finally, discussions and conclusions are stated in section VIII.

II. Extended Kinematic Modeling

The extended model includes the sideway velocity into of kinematic model, wherewith it be obtaining an holonomic-like model for the DDMR. Please notice that *the sideway slipping velocity is bounded and measurable variable in order to guarantee the non-holonomic constraint*. Considering this, the sideway slipping velocity is included into the EKM as a *virtual control input*.

A. Point of interest at an arbitrary position

Figure 1 shows the structure of DDMR used to propose the extended kinematic modeling, where (x^w, y^w) specifies the fixed 2D cartesian world's frame; d is the distance between wheels, the point of interest is at (x, y) and can be represented by a and α. Robot's orientation with respect to x^w positive axis is given by ϕ. The generalized coordinates of DDMR are given by $\boldsymbol{q} = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$.

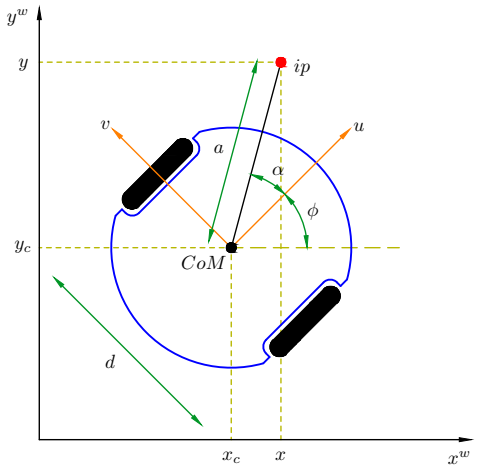


Fig. 1. DDMR with point of interest located at $ip = (x, y)$

Then, the EKM of a DDMR with point of interest at an arbitrary position $(a > 0,\ \alpha > 0)$ is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & -a\sin(\phi+\alpha) \\ \sin\phi & \cos\phi & a\cos(\phi+\alpha) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega_\phi \end{bmatrix}. \tag{1}$$

In (1), u and v are the translational and sideway velocities respectively, whereas ω_ϕ is the robot rotational velocity. Considering (1), it can be expressed on its compact form as $\dot{\boldsymbol{q}} = \boldsymbol{A}(\boldsymbol{q})\,\boldsymbol{u}$, where $\dot{\boldsymbol{q}}$ are the robot generalized velocities. Note that $\boldsymbol{A}(\boldsymbol{q})$ is invertible. Next, (1) can be inverted as

$$\begin{bmatrix} u \\ v \\ \omega_\phi \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & a\sin\alpha \\ -\sin\phi & \cos\phi & -a\cos\alpha \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}. \tag{2}$$

A convenient way to express (2) is $\boldsymbol{u} = \boldsymbol{A}(\boldsymbol{q})^{-1}\dot{\boldsymbol{q}}$. Now, based on inverse kinematics structure, the controller is defined as $\boldsymbol{u}_c = \boldsymbol{A}(\boldsymbol{q})^{-1}\boldsymbol{\eta}$, where $\boldsymbol{u}_c = \begin{bmatrix} u_c & v & \omega_{\phi_c} \end{bmatrix}^T$ is the pseudo control inputs vector and $\boldsymbol{\eta} = \begin{bmatrix} \eta_x & \eta_y & \eta_\phi \end{bmatrix}^T$ is the auxiliary control law vector. Notice that virtual and real control inputs may be written as

$$u_c = \eta_x\cos\phi + \eta_y\sin\phi + \eta_\phi a\sin\alpha \tag{3}$$
$$v = -\eta_x\sin\phi + \eta_y\cos\phi - \eta_\phi a\cos\alpha \tag{4}$$
$$\omega_{\phi_c} = \eta_\phi \tag{5}$$

where the real control inputs are: u_c, the driving velocity control input and ω_{ϕ_c}, the angular velocity control input and the virtual control input is v, the sideway slipping velocity. Since v in (4) is a measurable variable, then it is possible to solve for η_ϕ, leading to

$$\eta_\phi = (a\cos\alpha)^{-1}(\eta_y\cos\phi - \eta_x\sin\phi - v). \tag{6}$$

Here, a mathematical indetermination appears when $a = 0$ (the point of interest is on the axis between wheels) or when $\alpha = 90^o$ (the point of interest is over the sideway axis). The following subsection addresses these two situations in order to complete the kinematic model.

B. Point of interest on the sideway axis ($\alpha = 90^o$)

If the desired position of the point of interest is over the sideway velocity axis with $a > 0$, the EKM is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & -a\cos\alpha \\ \sin\phi & \cos\phi & -a\sin\alpha \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega_\phi \end{bmatrix}, \tag{7}$$

with a compact representation given by $\dot{\boldsymbol{q}} = \boldsymbol{A}(\boldsymbol{q})\,\boldsymbol{u}$, where $\boldsymbol{A}(\boldsymbol{q})$ is also an invertible matrix. Therefore, by solving for the control inputs

$$\begin{bmatrix} u \\ v \\ \omega_\phi \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & a \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}. \tag{8}$$

where, (8) it is possible to express as $\boldsymbol{u} = \boldsymbol{A}(\boldsymbol{q})^{-1}\dot{\boldsymbol{q}}$; then, the controller is proposed as $\boldsymbol{u}_c = \boldsymbol{A}(\boldsymbol{q})^{-1}\boldsymbol{\eta}$, where its expanded representation is given by

$$u_c = \eta_x\cos\phi + \eta_y\sin\phi + \eta_\phi a \tag{9}$$
$$v = -\eta_x\sin\phi + \eta_y\cos\phi \tag{10}$$
$$\omega_{\phi_c} = \eta_\phi \tag{11}$$

where the sideway velocity v in (10) is a measurable variable; hence, solving for ϕ results

$$\phi_d = \arctan\left(\frac{\eta_y}{\eta_x}\right) - \arcsin\left(\frac{v}{\sqrt{\eta_x^2+\eta_y^2}}\right). \tag{12}$$

Notice that for this new EKM, the controller does not have any math indetermination. It is even possible to use this controller when the point of interest is located at CoM ($a = 0$). This case is reported in the bibliographic widely and it is solved with complex techniques.

III. Auxiliary Control Law

According to the previous notation, the desired generalized coordinates $\boldsymbol{q}_d = \begin{bmatrix} x_d & y_d & \phi_d \end{bmatrix}^T$ and the desired generalized velocities $\dot{\boldsymbol{q}}_d = \begin{bmatrix} \dot{x}_d & \dot{y}_d & \dot{\phi}_d \end{bmatrix}^T$ are denoted, which are differentiable, smooth and bounded. Next, the generalized coordinates error are defined as $\tilde{\boldsymbol{q}} = (\boldsymbol{q}_d - \boldsymbol{q})$, allowing to properly define the control objective as

$$\lim_{t\to\infty} \tilde{\boldsymbol{q}} = 0.$$

Figure 2 shows the system block diagram. The controller mode is established at the process planning stage according to desired values (see section V).

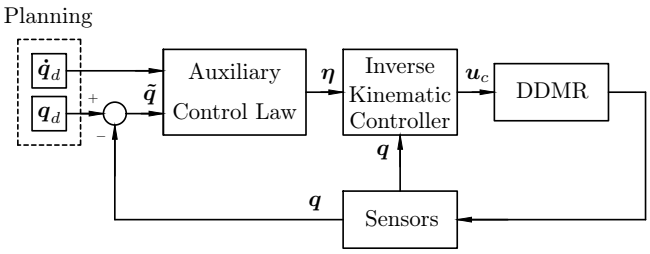


Fig. 2. Closed-loop system block diagram representation

Selection of auxiliary control laws depends on the point of interest position. Next, these cases are separately addressed. Furthermore, adequate smooth saturation functions are aggregated in order to deal with the physically constrained character of each physical control input.

A. Point of interest at an arbitrary position

Here, η_ϕ is obtained directly from (6), whereas η_x and η_y are designed according to

$$\eta_x = \dot{x}_d + l_x \tanh\left(k_x l_x^{-1} \tilde{x}\right) \tag{13}$$
$$\eta_y = \dot{y}_d + l_y \tanh\left(k_y l_y^{-1} \tilde{y}\right) \tag{14}$$

where, $l_x > 0$ and $l_y > 0$ correspond to the saturation constrains of kinematic control inputs, these values depend on the WMR actuator saturation; and $k_x > 0$, $k_y > 0$ are design parameters. Errors are defined as follows $\tilde{x} = x_d - x$ and $\tilde{y} = y_d - y$.

B. Point of interest on the sideway axis

When the point of interest is located on the sideway axis, control laws (13) and (14) for η_x and η_y respectively remain the same. However, in this case it is now necessary to design η_ϕ according to

$$\eta_\phi = \dot{\phi}_d + l_\phi \tanh\left(k_\phi l_\phi^{-1} \tilde{\phi}\right) \tag{15}$$

where $l_\phi > 0$ is the angular velocity saturation value, $k_\phi > 0$ is a design parameter. On the other hand, the angular error is defined by $\tilde{\phi} = \phi_d - \phi$, where ϕ_d is obtained from (12). Notice that, this auxiliary control law and the proposed controller above have no mathematical singularities.

IV. Stability Proof

It is first considered the closed-loop system, which varies depending on the point of interest position. Next, once the equilibrium are defined, the system stability is analyzed according to Lyapunov theory.

A. Closed-Loop System

If it is considered a perfect velocity tracking: $\boldsymbol{u} = \boldsymbol{u}_c$, the compact representation of closed-loop systems results

$$\dot{\boldsymbol{q}} = \boldsymbol{A}(\boldsymbol{q})\,\boldsymbol{A}(\boldsymbol{q})^{-1}\,\boldsymbol{\eta} = \boldsymbol{\eta}. \tag{16}$$

By following the procedure the closed-loop system is proposed for both cases to the point of interest positions, as follows.

1) Point of interest at an arbitrary position: In correspondence with (16), and taking into account that η_ϕ is not computed by the controller, the closed-loop equations are cited below:

$$\dot{\tilde{x}} = -l_x \tanh\left(k_x l_x^{-1} \tilde{x}\right) \tag{17}$$
$$\dot{\tilde{y}} = -l_y \tanh\left(k_y l_y^{-1} \tilde{y}\right) \tag{18}$$

2) Interest Point on the sideway axis: in this case, (15) must be considered in order to complete the closed-loop equations. This way the third equation looks like

$$\dot{\tilde{\phi}} = -l_\phi \tanh\left(k_\phi l_\phi^{-1} \tilde{\phi}\right). \tag{19}$$

Additionally, in order to simplify the notation the closed-loop system is expressed as $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_d + \boldsymbol{L}\tanh\left(\boldsymbol{K}\boldsymbol{L}^{-1}\tilde{\boldsymbol{q}}\right)$, being $\mathbf{L} = \text{diag}\,(l_x,\ l_y,\ l_\phi)$ the saturation constraints matrix and $\mathbf{K} = \text{diag}\,(k_x,\ k_y,\ k_\phi)$ the associated design parameters matrix. By introducing $\dot{\tilde{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}$, the closed-loop equation for the point of interest over the sideway axis is

$$\dot{\tilde{\boldsymbol{q}}} = -\boldsymbol{L}\tanh\left(\boldsymbol{K}\boldsymbol{L}^{-1}\tilde{\boldsymbol{q}}\right). \tag{20}$$

B. Stability Proofs

For stability proof, firstly the equilibrium are defined, then the system stability of both cases aforementioned are analyzed according to Lyapunov theory.

1) Point of interest at an arbitrary position: In this case the equilibrium point is composed by the states $\tilde{x}$ and $\tilde{y}$ only: $\boldsymbol{\kappa_1} = \begin{bmatrix} \tilde{x} & \tilde{y} \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. The Lyapunov Function Candidate (LFC) is proposed as follows:

$$V_1 = \frac{1}{2}\left(\tilde{x}^2 + \tilde{y}^2\right). \tag{21}$$

Then, considering its time-derivative of the LFC in (21) along the system trajectories, this leads to:

$$\dot{V}_1 = \tilde{x}\dot{\tilde{x}} + \tilde{y}\dot{\tilde{y}}, \tag{22}$$

next, by replacing the closed-loop equations (17) and (18) into (22), it results

$$\dot{V}_1 = -l_x \tanh\left(k_x l_x^{-1}\tilde{x}\right)\tilde{x} - l_y \tanh\left(k_y l_y^{-1}\tilde{y}\right)\tilde{y} < 0, \tag{23}$$

In other words, $\tilde{x} \to 0$ and $\tilde{y} \to 0$ when $t \to \infty$, in consequence, the asymptotic stability of the equilibrium point is proved. Moreover, from (13) and (14) it can be concluded also that $\eta_x \to \dot{x}_d$ and $\eta_y \to \dot{y}_d$ when $t \to \infty$ respectively. There is not information about $\dot{\phi}$, but taking into consideration these last results and replacing them on $\dot{\phi} = \eta_\phi$ it is obtained the following equation

$$\dot{\phi} = \omega_{\phi_c} = (a\cos\alpha)^{-1}\left(\dot{y}_d\cos\phi - \dot{x}_d\sin\phi - v\right).$$

It can be seen that $\dot{\phi}$ is bounded, provided that $\dot{\tilde{x}}_d$, $\dot{\tilde{y}}_d$ and v are all bounded. The other control action, the linear velocity u_c, after the same considerations is given by

$$u_c = \dot{x}_d\cos\phi + \dot{y}_d\sin\phi + \eta_\phi a,$$

which is also bounded.

2) Point of interest on the sideway axis: In this case, the equilibrium point is given by the compact form by $\tilde{\boldsymbol{q}} = 0$ including the three states of the DDMR. Then the LFC is proposed in (24)

$$V_2 = \frac{1}{2}\left(\tilde{\boldsymbol{q}}^T\tilde{\boldsymbol{q}}\right). \tag{24}$$

Next, the time-derivative of (24) along system trajectories is given by (25)

$$\dot{V}_2 = \tilde{\boldsymbol{q}}^T\dot{\tilde{\boldsymbol{q}}},$$

by replacing the closed loop equation (20) it is possible to conclude about the equilibrium point asymptotic stability

$$\dot{V}_2 = -\tilde{\boldsymbol{q}}^T\boldsymbol{L}\tanh\left(\boldsymbol{K}\boldsymbol{L}^{-1}\tilde{\boldsymbol{q}}\right) < 0.$$

In other words, $\tilde{\boldsymbol{q}}(t) \to \boldsymbol{0}$ when $t \to \infty$; therefore, the asymptotic stability of the equilibrium point is proved.

V. Controller Modes

This section describes how to define the desired parameters in order to set the controller behavior i.e. trajectory tracking or to path following.

A. Trajectory Tracking Mode

In order to obtain a trajectory tracking behavior, the desired values are defined in cartesian coordinates at the planning stage. The references are defined as differentiable, smooth and bounded time-variant parametric functions

$$x_d = x_d(t)$$
$$y_d = y_d(t)$$

Considering their time-derivatives, the desired cartesian velocities $\dot{x}_d(t)$ and $\dot{y}_d(t)$ are obtained. In consequence the desired angular velocity is given by

$$\dot{\phi}_d = \frac{\ddot{y}_d\dot{x}_d - \dot{y}_d\ddot{x}_d}{\dot{x}_d^2 + \dot{y}_d^2}.$$

B. Path Following Mode

The path following mode requires a geometric path. If this path is s-parametrized in meters $[m]$, the desired cartesian position for each axis is a differentiable and smooth function given by $x_d = x_d(s)$ and by $y_d = y_d(s)$. The angle over the desired path is given by

$$\phi_p(s) = \arctan\left[\frac{\frac{d}{ds}y_d(s)}{\frac{d}{ds}x_d(s)}\right]. \tag{25}$$

According to notation in section III, the desired angular velocity is given by $\dot{\phi}_d \sim \frac{d}{ds}\phi_p(s)$, the desired cartesian velocities over the path now is given by $\dot{x}_d \sim v_d\cos\phi_p(s)$ and $\dot{y}_d \sim v_d\sin\phi_p(s)$.

VI. Simulation Results

According to auxiliary control law detailed in section III-A and section III-B, different parameters are considered in simulation results for each case.

A. Trajectory Tracking Mode

Figure 3 shows the DDMR simulation with the point of interest located at CoM. The parameters are setting as follows: $l_x = l_y = l_\phi = 0.8$, $k_x = k_y = 0.8$, $k_\phi = 1.5$, the initial generalized position is given by $\boldsymbol{q}(0) = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T$.

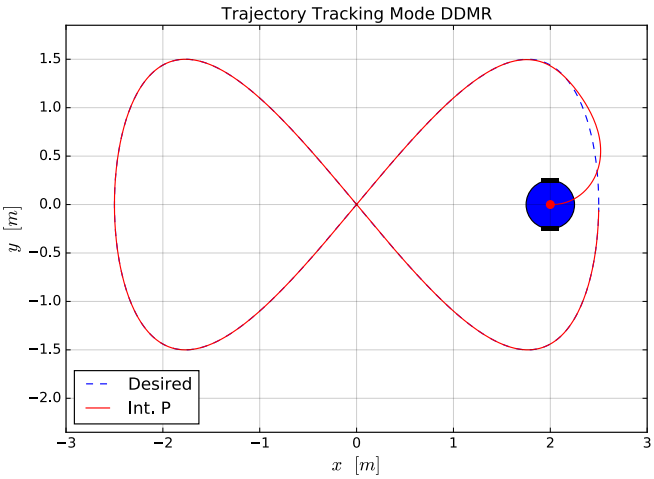


Fig. 3. Trajectory tracking of DDMR when ip located at CoM

Figure 4 shows the driving velocity and angular velocity applied to DDMR based on Fig. 3 statements, in this case the sideway slipping velocity is $v = 0$, according to nonholonomic constraint and the cartesian position errors

The figure 5 shows the DDMR simulation considering following parameters $a = 0.4$ and $\alpha = 30^o$, $k_x = k_y = 0.7$, $l_x = l_y = l_\phi = 0.8$, and $k_\phi = 1.5$, the initial generalized position is given by $\boldsymbol{q}(0) = \begin{bmatrix} 1.5 & 0 & 0 \end{bmatrix}^T$.

Figure 6 shows the control inputs applied to DDMR considering Fig. 5 statements, the sideway slipping velocity is $v = 0$ according to nonholonomic constraint and the cartesian error in euclidean norm, in this case the point of interest is in $a = 0.4$ and $\alpha = 30^o$.

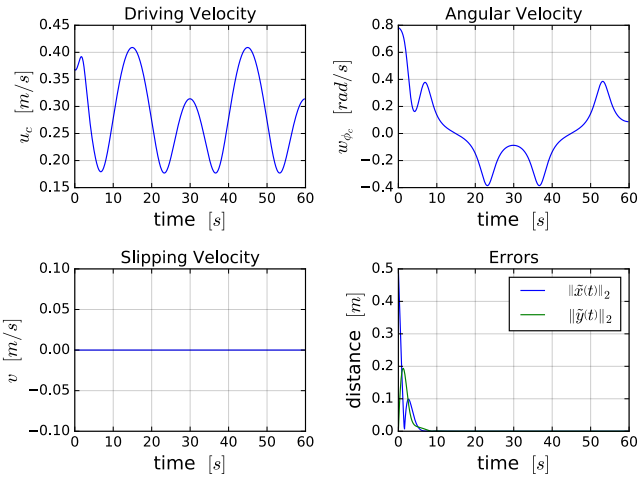


Fig. 4. Trajectory tracking control inputs with ip located at CoM

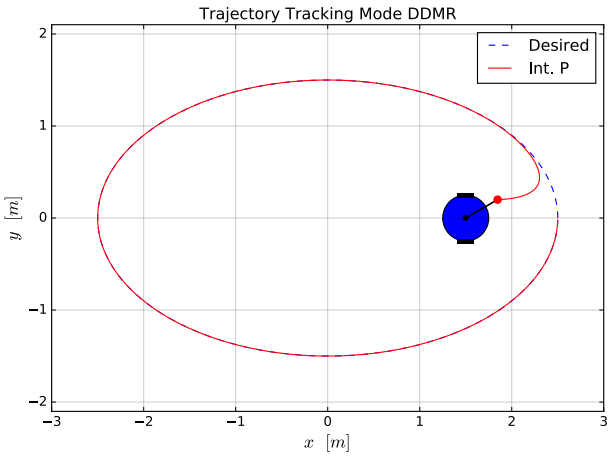


Fig. 5. Trajectory tracking of DDMR with $a = 0.4$ and $\alpha = 30^o$

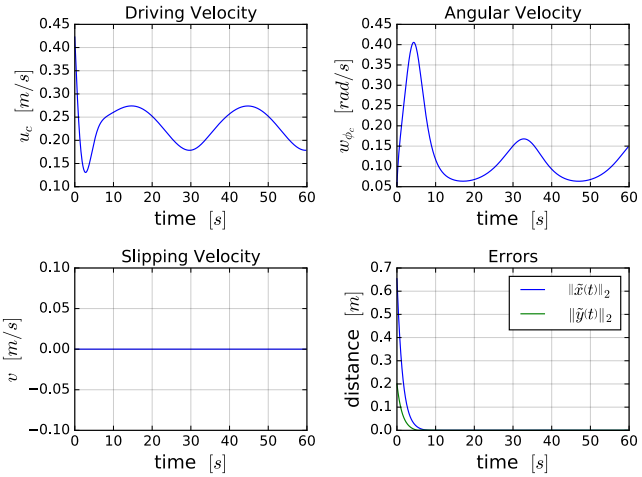


Fig. 6. Trajectory tracking control inputs with $a = 0.4$ and $\alpha = 30^o$

B. *Path Following Mode*

Figure 7 shows the simulation results when the controller is used in path following mode. The point of interest is located at CoM. According to the notation of section V-B the desired velocity required over he path is $v_d = 0.3$. The parameters are setting as follows: $k_x = k_y = 0.8$ and $k_\phi = 6$, $l_x = l_y = l_\phi = 0.8$, the initial state of generalized coordinates are given by $\boldsymbol{q}(0) = \begin{bmatrix} -0.5 & -1 & 0 \end{bmatrix}^T$.

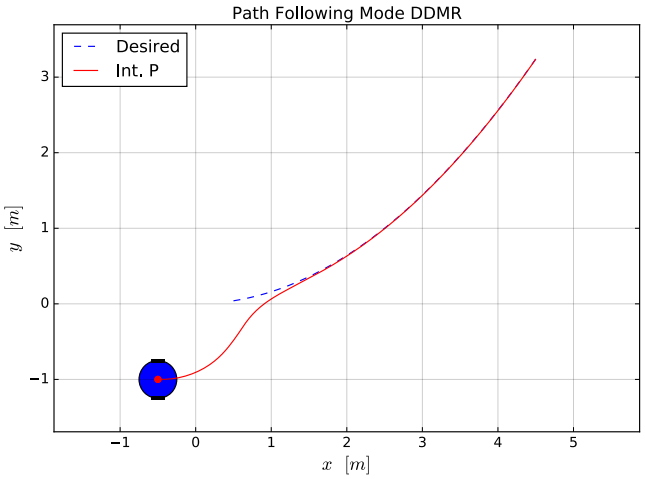


Fig. 7. Path following simulation with the point of interest located at CoM.

Figure 8 shows the control inputs applied to DDMR based on Fig. 7 statements, the cartesian error is expressed in euclidean norm and sideway slipping velocity is $v = 0$.

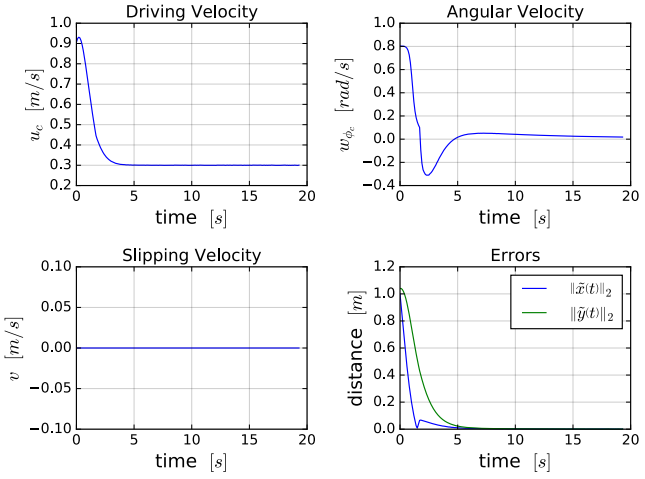


Fig. 8. Results of path following with the point of interest located at CoM.

Figure 9 shows the simulation results with a desired velocity over he path of $v_d = 0.25$. The parameters are setting as follows: $k_x = k_y = 0.8$ and $k_\phi = 6$, $l_x = l_y = l_\phi = 0.8$, $a = 0.4$, $\alpha = -30^o$, the initial state of generalized coordinates are given by $\boldsymbol{q}(0) = \begin{bmatrix} -0.5 & -0.5 & 0 \end{bmatrix}^T$.

In this case the sideway slipping velocity is different to zero; in this context, it is included with a value of $v = 0.15\ m/s$ in a time interval.

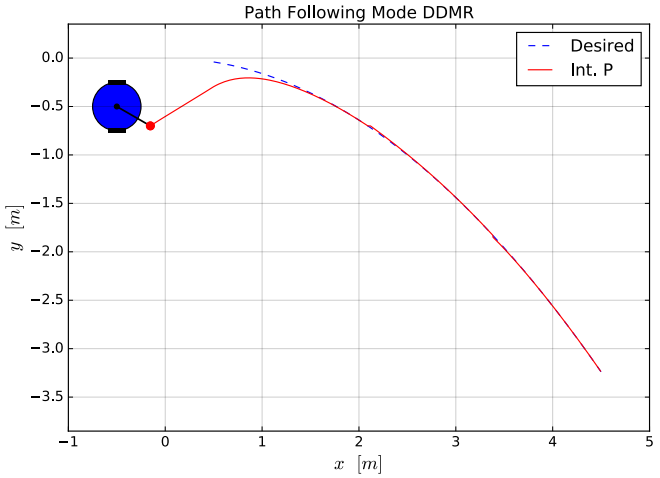


Fig. 9. Path following simulation of DDMR with $a = 0.25$, $\alpha = -30^o$ and including the sideway slipping velocity.

Figure 10 shows the control inputs according to Fig. 9 statements. Notice that the sideway slipping velocity changes but the cartesian errors tend to zero, the cartesian errors are expressed in euclidean norm, also notice that that both driving velocity and angular velocity are according to sideway slipping velocity inclusion.

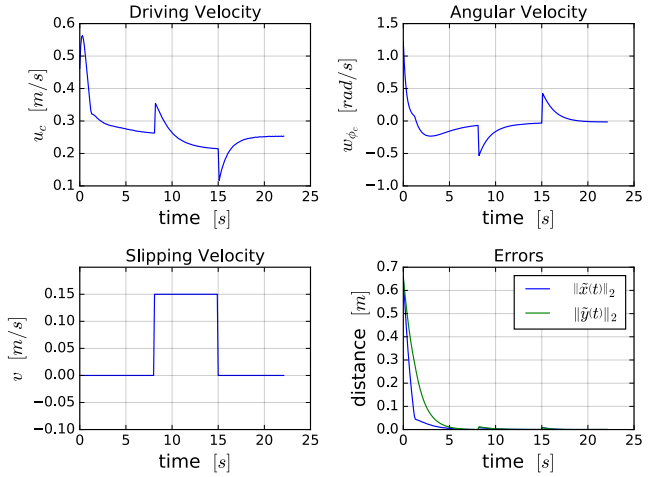


Fig. 10. Results of path following mode including the sideway slipping velocity.

VII. Future Work

It is now being studied the extended kinematic model applied to car-like mobile robots, and experiments are being conducted for both kind of mobile robots.

VIII. Discussion and Conclusions

Using the EKM to model the non-holonomic WMR a differentiable, time-invariant controller is obtained for both tasks path following and trajectory tracking, where the point of interest position is defined by a and α.

Based on the Lyapunov stability proof, it is concluded that when the point of interest is located at CoM the desired generalized coordinates of WMR are guaranteed, but when the point of interest is located at an arbitrary position only the cartesian position (x, y) of ip is guaranteed.

Notice that, the sideway slipping velocity could be any measurable and bounded value, including the zero value (if this is not verified, v is set to zero value in the design) without lose generality in the controller structure.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[2] G. Oriolo, A. D. Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, Nov 2002.

[3] B. d'Andrea Novel, G. Bastin, and G. Campion, "Modelling and control of non-holonomic wheeled mobile robots," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Apr 1991, pp. 1130–1135 vol.2.

[4] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, Feb 1996.

[5] G. Lafferriere and H. Sussmann, "Motion planning for controllable systems without drift," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Apr 1991, pp. 1148–1153 vol.2.

[6] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, Apr 1991, pp. 1136–1141 vol.2.

[7] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, May 1990, pp. 384–389 vol.1.

[8] Z.-P. JIANGdagger and H. NIJMEIJER, "Tracking control of mobile robots: A case study in backstepping*," *Automatica*, vol. 33, no. 7, pp. 1393 – 1399, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109897000551

[9] D. Chwa, "Tracking control of differential-drive wheeled mobile robots using a backstepping-like feedback linearization," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 6, pp. 1285–1295, Nov 2010.

[10] D. Diaz and R. Kelly, "On modeling and position tracking control of the generalized differential driven wheeled mobile robot," in *2016 IEEE International Conference on Automatica (ICA-ACCA)*, Oct 2016, pp. 1–6.

[11] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

A QR-code Localization System for Mobile Robots: Application to Smart Wheelchairs

Luca Cavanini[1], Gionata Cimini[2], Francesco Ferracuti[1], Alessandro Freddi[1], Gianluca Ippoliti[1], Andrea Monteriù[1] and Federica Verdini[1]

***Abstract*— A Smart Wheelchair (SW) is an electric powered wheelchair, equipped with sensors and computational capabilities, with the general aim of both enhancing independence and improving perceived quality of life of the impaired people using it. SWs belong to the class of semi-autonomous mobile robots, designed to carry the user from one location to another of his/her choice. For such systems, the localization aspect is of utmost concern, since GPS signal is not available indoors and alternative sensor sets are required. This paper proposes a low-cost artificial landmark-based localization system for mobile robots operating indoor. It is based on Quick Response (QR) codes, which contain the absolute position of the landmark: a vision system recognizes the codes, estimates the relative position of the robot (i.e., displacement and orientation) w.r.t. the codes and, finally, calculates the absolute position of the robot by exploiting the information contained in the codes. The system has been experimentally validated for self-localization of a smart wheelchair, and experimental results confirm that navigation is possible when considering an high QR-code density, while QR-code low density conditions permit to reset the cumulative odometry error.**

***Index Terms*— Smart wheelchair, Localization, Mobile robots, Assistive robotics**

I. Introduction

Traditional wheelchairs are among the most common assistive devices, and represent one of the first solution to the problem of mobility for impaired users [1]. Recently they have been turned into Smart Wheelchairs (SWs) through the integration of sensors and computational capabilities [2]. At the moment the market offers SWs with several functionalities, i.e., autonomous behaviours, interaction with domestic environment and parameters measurements [3], [4] and self learning functionalities [5] for users affected by severe disabilities reducing their interaction possibility.

Despite of the attractive additional features, the main task of a SW is the autonomous navigation. Several solutions have been proposed over the years, using different approaches and exploiting several sensors, but autonomous navigation still needs research efforts [6], [7]. The requirements can dramatically change according to the structure of the environment, to the available computational power and, most of all, according to the user needs, and often a trade-off is needed as the approaches have complementary strengths and weaknesses [8], [9].

As a fundamental part of navigation, localization of SWs plays a key role, especially since GPS is not available indoors and alternative sensor sets are required. Commonly used solutions based on proprioceptive sensors which can be employed indoors in conjunction with well known localization techniques, e.g., Least Mean Square or Dead Reckoning, have been demonstrated to be poor in resolution, accuracy and applicability, at least when no other sensors are involved [10]. Due to this problem, in this work, the authors focus their attention on the development of a vision system for indoor localization of mobile robots based on artificial landmarks. This kind of system can be classified in the category of Visual Servoing (VS), namely indicating those techniques which control mobile robots by using feedback information extracted from a visual sensor [11]. Several modern algorithms are based on VS platforms, which include both elementary tasks and several visual features; different features can be combined together to obtain tracking of visual cues at video rate [12]. In this paper, the authors propose a landmark-based approach which is computationally cheap and reliable avoiding the need of additional measurements.

Although natural landmarks have been successfully explored in the literature [13], the authors have considered artificial landmarks since they grant better image recognition performances, which are necessary to obtain an accurate and precise localization, as required by SW systems. About this, in [14] simple colored artificial landmark patterns have been used, whereas in [15] the color's information are supported also by depth analysis: these approaches require a powerful visual sensor, even catadioptric. Among the artificial landmarks, Quick Response (QR) codes are easy to detect indoors and several efficient libraries already exist for their interpretation [16], which require simple camera sensors. In addition, the power of QR-codes consist in the possibility to embed coded information. The problem of mobile robot localization using QR-codes has been tackled only by few researchers in the literature, and obtained results do not allow to evaluate the real effort introduced by different approaches [17], [18], [19], [20], [21] to the study of localization and navigation problems.

In this paper, the authors exploit QR-codes as artificial landmarks, and focus their attention on the localization problem. The authors provide a detailed description of the localization system, and present qualitative and quantitative results on the localization error which should be expected

[1]Luca Cavanini, Francesco Ferracuti, Alessandro Freddi, Gianluca Ippoliti, Andrea Monteriù and Federica Verdini are with the Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via Brecce Bianche, 60131 Ancona, Italy {l.cavanini, f.ferracuti, a.freddi, g.ippoliti, a.monteriu, f.verdini}@univpm.it

[2]Gionata Cimini is with ODYS S.r.l., Milano, Italy gionata.cimini@odys.it

when using a QR-code based localization algorithm. The algorithm has been tested with a commercial electric powered wheelchair, equipped with a cheap webcam. The software structure has been realized using the Robotic Operating System (ROS) framework [23]. The good performance of the developed localization system have been confirmed by experimental tests, in which a stereophotogrammetric system characterized by six high-resolution infrared-sensitive cameras has been adopted as the pose estimation ground truth. This last system is usually adopted to measure body segments movements where the required accuracy is low (under 1 cm) [24]. An additional test regarding the orientation estimation of the vehicle has been performed: in this case the results have been compared with those provided by a high performance IMU.

Although the results have confirmed that the method is accurate and precise, and can be used as the only path planning support, it can be seen as the base for more powerful Simultaneous Localization and Mapping (SLAM) algorithms [25]. The validation here reported also confirms the usefulness of QR-codes as a low cost solution for providing exteroceptive auxiliary measurements in GPS-denied environments, as already qualitatively stated in [26], therefore resulting a viable solution for indoor navigation of service robots.

The paper is organized as follows. Section II details the localization strategy which is at the base of the proposed system, whereas Section III describes the hardware structure and the software system architecture. In Section IV the experimental results are provided. Finally Section V provides a brief critical analysis of the proposed system.

II. The Localization Algorithm

The main core of the proposed localization algorithm is based on the ViSP middleware, a C++ library for the design of visual tracking and visual servoing algorithms [26], [27]. The tool has been recently used in different application fields, from structural properties monitoring to robot pose control [28], [29]. The aim of vision-based control schemes is to minimize the matching error, defined as

$$\boldsymbol{e}(t) = \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}) - \boldsymbol{s}^* \tag{1}$$

where the vector $\boldsymbol{m}(t)$ is a set of image measurements (e.g., the coordinates of interest points or the coordinates of the object's centroid). These measurements are used to compute a vector of k visual features $\boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a})$ in which $\boldsymbol{a}$ is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3-D models of objects). The vector $\boldsymbol{s}^*$ contains the desired values of the features. The degree of freedom of the servoing scheme is the design of visual features. The proposed approach is based on the ViSP KLT (Kanade – Lucas -Tomasi) point tracker algorithm [30]. The KLT algorithm is based on the gradient method, used to compare a template with an acquired image, also in case of very different images dimension. To estimate the displacement of the tracked object w.r.t. the template image, the tracking method monitors the position of significant points in different sequential images. The tracking is realized by minimizing the image dissimilarity, defined as

$$\sum_{\boldsymbol{x}} \left[I(\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p})) - T(\boldsymbol{x}) \right]^2 \tag{2}$$

where

- $\boldsymbol{x}$ is the column vector containing the image coordinates $[x \ y]^T$;
- $T(\boldsymbol{x})$ is the template (reference) image;
- $\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p})$ is the set of available warps, which for translations becomes $\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p}) = [x + p_1 \ y + p_2]^T$;
- $\boldsymbol{p}$ is the displacement;
- $I(\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p}))$ is the input (acquired) image;

The objective is then to find $\boldsymbol{p}$ which minimizes the image dissimilarity, which can be expressed as:

$$\sum_{\boldsymbol{x}} \left[I(\boldsymbol{W}(\boldsymbol{x}, \boldsymbol{p} - T(\boldsymbol{x}) \right]^2 \tag{3}$$

With ViSP it is also possible to track an object by using its model, which is useful in the case of artificial landmarks such as QR-codes. The model-based variant of the KLT tracker can be used alone, or in combination with the model-based edges tracker. This hybrid approach, which is the one used in the proposed system, is especially appropriate to track textured objects with visible edges [31].

Once the QR-code has been detected and tracked, it is possible to estimate the robot position. In detail, let us consider the pose $\boldsymbol{P}$ of the QR-code with respect to the reference frame of the camera

$$\boldsymbol{P} = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T \tag{4}$$

and the quaternion of its orientation

$$\boldsymbol{O} = \begin{bmatrix} O_x & O_y & O_z & O_w \end{bmatrix}^T = \begin{bmatrix} a & b & c & d \end{bmatrix}^T \tag{5}$$

then the rotation matrix to obtain the position of the camera $\boldsymbol{P'}$ (and consequently of the vehicle) with respect to the landmark such that

$$\boldsymbol{P'} = \begin{bmatrix} P'_x & P'_y & P'_z \end{bmatrix}^T = \boldsymbol{M} \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T \tag{6}$$

is

$$\boldsymbol{M} = \begin{bmatrix} 1 - 2c^2 - 2d^2 & 2bc + 2ad & 2bd - 2ac \\ 2bc - 2ad & 1 - 2b^2 - 2d^2 & 2cd + 2ba \\ 2bd + 2ac & 2cd - 2ba & a - 2b^2 - 2c^2 \end{bmatrix}. \tag{7}$$

Finally, the global position of the vehicle $\bar{\boldsymbol{P}}'$ respect to the global reference frame is calculated by considering the encoded global position $\bar{\boldsymbol{P}}$ of the QR-code, i.e.,

$$\bar{P}'_x = \bar{P}_x + P'_x \tag{8}$$

$$\bar{P}'_x = \bar{P}_y + P'_y \tag{9}$$

$$\bar{P}'_z = K \tag{10}$$

where K is a fixed quantity. The flowchart of the algorithm is reported in Figure 1.

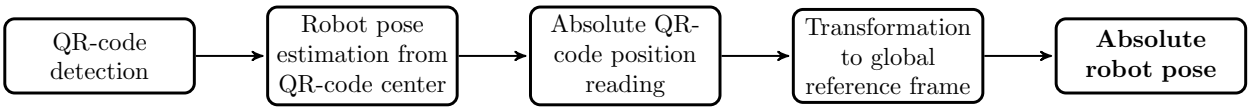


Fig. 1. Flowchart of the localization algorithm: starting from the QR-code detection, the absolute position embedded within the code is read, corrected by the robot pose estimation, and finally transformed into the global reference frame coordinates

III. Hardware and Software System Setup

The developed system is composed by a simple hardware configuration, consisting in an USB Logitech HD Webcam C525, a Sunrise Quickie Salsa R2 electric powered wheelchair, a set of QR codes and a laptop with Linux Ubuntu 12.04 OS running ROS framework. The vision sensor is cheap and characterized by a maximum resolution of 1280x720 pixels. The lower resolution of 640x480 pixels was selected, in order to reduce the amount of data to transmit and obtain a more efficient communication. The wheelchair is characterized by two driving rear wheels and two free castor wheels. The vision sensor is mounted behind the user seat, at the middle of the rear wheelbase, to eliminate rotation axis. With the described hardware setup, real-time elaboration is achieved for video sampling frequencies up to 10 Hz

QR-codes are placed on the ceiling, with a standard orientation respect a global fixed orientation frame (see Figure 2). This characteristic makes it suitable to domestic and civil environments and also non-domestic environments, like hospitals, care facilities, etc. In this work, we assume that the distance between the camera and the ceiling is fixed: this reasonable assumption can improve the pose estimation. In the developed system, each QR code contains the coordinates of the landmark in the working environment, structured as a string with the format:

#xx.xx#yy.yy#zz.zz#rr

where the variables *#xx.xx*, *#yy.yy* and *#zz.zz* indicate the coordinates in the global reference frame, expressed in meters. The variable *#rr* could be used to indicate the room where the landmark is located, or a particular room area in case of larger spaces.

The developed system is based on the ROS framework, so the software structure is organized by the ROS architecture philosophy, in nodes (algorithms) and topics (communication channels) [23]. The software structure is organized in three modules, each one representing a data elaboration process. The first module, called `/usb_cam`, consists in the camera driver. It acquires image data from the sensor and sends them to the `/ViSP_auto_tracker` module. The communication channels are the topics `/usb_cam/camera_info`, containing information about the camera data format, and `/usb_cam/image_raw`, containing the data information from the camera. The second module, called `ViSP_auto_tracker`, realizes the QR code data conversion, producing a decoded data string indicating the landmark position coordinates. The last module, the node `/evaluate`, represents the localization algorithm providing the robot pose estimation respect to the position of the QR code inside the sensor's cone of vision.



(a) Frontside view picture (b) Topside view picture

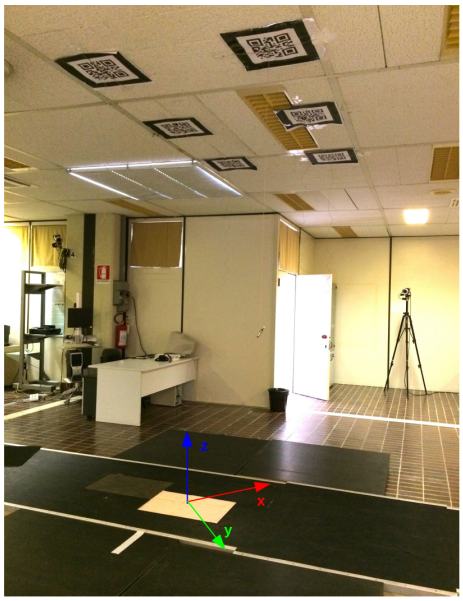
(c) backside view picture

Fig. 2. Smart wheelchair test setup

IV. Experimental Results

A. Experimental Setup

A stereophotogrammetric system characterised by six high-resolution infrared-sensitive cameras is adopted as ground truth. The cameras are positioned and calibrated to cover a measurement volume of 2.5 m (length) x 3.5 m (depth) x 2 m (height). The movement of the vehicle is measured by placing 12 markers reflectors in key points (see Figure 3) of the vehicle, i.e., 2 on the back (markers $1-2$), 2 on each arm (markers $3-6$), 2 on the rear (markers $7-8$), 1 on each longitudinal frontal bar (markers $9-10$) and finally 2 on the footrest (markers $11-12$). The trajectory of the markers is recorded by the acquisition system with respect to the laboratory reference system, oriented as shown in Figure 3 (red: x axis, green: y axis, blue: z axis). Each QR-code is 42×42 cm and it is placed at an height of 3 m from the floor (z axis) and 1.5 m far from the others.



(a) Test environment and QR-codes setup



(b) Smart wheelchair and positions of the markers reflectors

Fig. 3. The test environment and the smartwheelchair with marker points

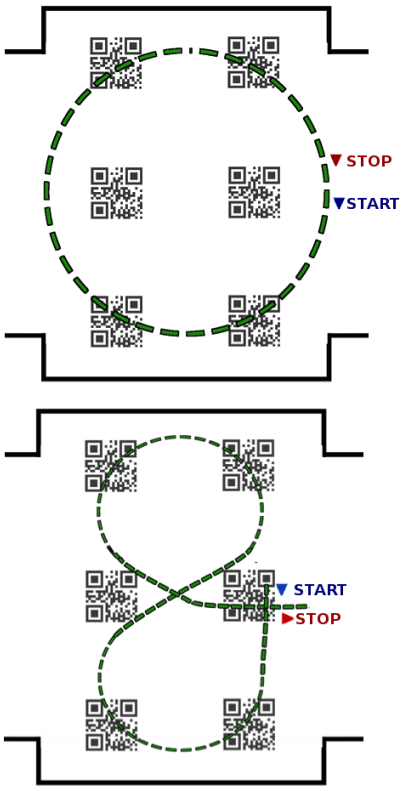


Fig. 4. Test trajectories with QR-codes positions

TABLE I

EXPERIMENTAL RESULTS OF POSITION AND ORIENTATION ESTIMATION: ERROR MEAN VALUE AND STANDARD DEVIATION

Test	$E(e)$	σ_e
Circular Path	0.2101 m	0.0095 m
8-Shape Path	0.14085 m	0.0089 m
Orientation Test	0.267 rad	0.0120 rad

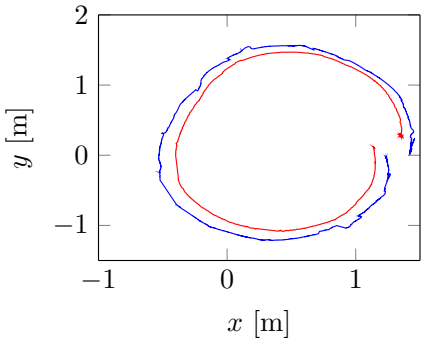
B. *Experimental tests*

The proposed localization system has been evaluated on two standard trajectories: a circle path and an 8-shape path. In Figure 4, the two trajectories are shown together with the position of the QR-codes in the test environment. The graphical results of the two test trajectories are collected in Figures 5 and 6, respectively, showing the comparison of the measurements provided by the stereophotogrammetric system, which can be taken as the actual path covered by the vehicle, and the estimation provided by the proposed algorithm. In both cases, the SW has been guided by a user, thus the actual trajectories are slightly different with respect to the reference trajectories provided in Figure 4.

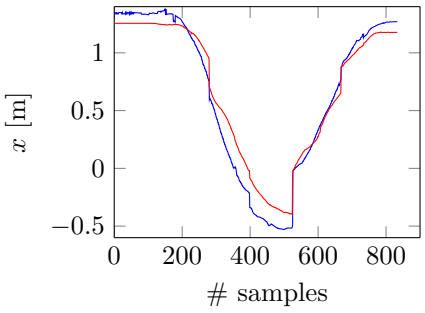
In Table I, the experimental results are numerically evaluated in terms of mean error and standard deviation from the reference vision system along the whole path: in this way the proposed visual system can be easily compared with similar systems in the literature, as long as they provide the error measurements w.r.t a ground truth on a curvilinear trajectory. In the proposed case, the visual localization system performs well and the position of the vehicle is estimated within an acceptable absolute error.

At the best of the authors knowledge, the only work in the literature which clearly validates a localization system based on QR-codes is [22]: in their work the authors employ a smartphone-based mobile robot with Android, rather than a ROS-based system for assistive scenarios; moreover they provide results for a single trajectory made of four linear segments, while we provide the results for two trajectories based on curvilinear segments. The results are similar, and also in line with the other works already mentioned in Section I, namely [17], [18], [19], [20], [21], which however do not provide enough details on how the errors were computed and on which trajectories.

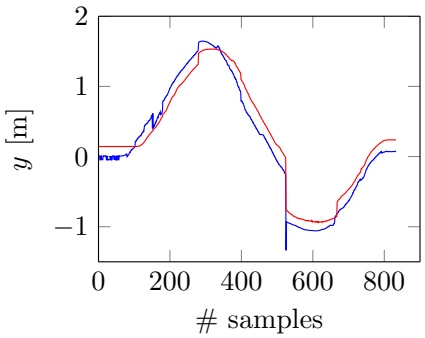
In indoor navigation for impaired people, a major importance is assumed also by the orientation at the target point, as often it represents a place where the user has to perform a task. For this reason, a test for the orientation accuracy has also been performed. In a structured environment, like user's



(a) Path acquisitions

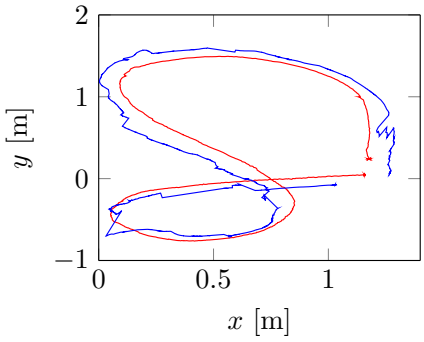


(b) x comparison

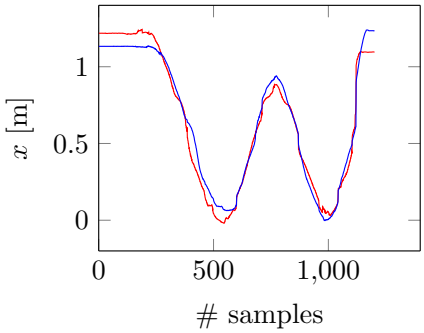


(c) y comparison

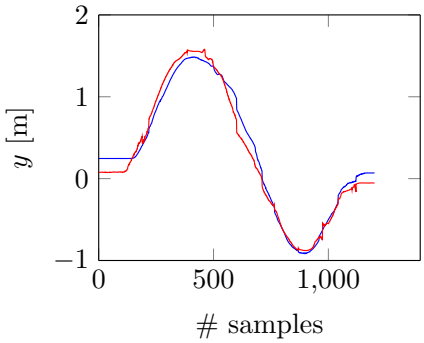
Fig. 5. Circle path test results: in red the path measured by the camera system, in blue the path estimated by the proposed system



(a) Path acquisitions



(b) x comparison



(c) y comparison

Fig. 6. 8-shape path test results

home, the target points are well known and in the design phase a good rule of thumb to follow is to place a QR-code in proximity of these target points to maximize the accuracy. As several particular situations could happen, the authors have performed a test in proximity of a QR landmark but in a not centered position to consider the worst case performance. The benchmark consists in a series of estimations placing the SW in different orientations. For these tests, a Microstrain 3DM-GX3-25 IIMU has been employed as ground truth. The adopted sensor is an high precision IMU, characterized by a gyroscope with a low drift ($< 12{\cdot}10^{-3}$ deg/s). The algorithm estimation has been compared to the IMU acquisitions and the heading error has been evaluated in Table I in terms of mean error and variance. As well as for the pose estimation, the orientation test has proved the suitability of the proposed approach and the results are comparable with those in the recent literature [22].

It is worth noticing that the proposed approach has been employed as the only pose estimation source, without the fusion of other systems, commonly used in literature. The provided results can be achieved whenever the QR code density is high (typically $>$ 1 QR-code$/m^2$): in this reduced framework the results are encouraging and comparable with those found in the literature. Worse results should be expected for lower densities, however when high density can not be achieved, it is possible to employ the proposed method for providing auxiliary measurements, crucial for resetting the cumulative and unavoidable error generated by inertial systems.

V. Conclusion

The lack of a mature technology for indoor navigation has moved the interest into developing low-cost pose es-

timation systems, applicable in general for mobile robots. In particular, a landmark-based visual localization system for a smart wheelchair has been proposed in this work. The approach relies on QR-codes landmarks, detected by a webcam oriented to the ceiling. The cost of the system is very low due to the nature of the landmarks, which are cheap and permit the use of a low resolution camera to be detected. In fact, the particular pattern avoid the false detection of similar objects in indoor environments and, additionally, the information about the absolute position in the global reference frame can be encoded in the QR-code. The entire algorithm has been implemented in ROS environment, exploiting ViSP middleware. In order to evaluate the performance of the approach, an high resolution stereophotogrammetric system and a high performance IMU have been used as reference gold standards. The results are good and encouraging, both in different path following scenarios and in orientation estimation, and show that QR-codes can be used as the only mean of localization in case of high density. Whenever this is not possible, i.e. in case of low QR-codes density, the proposed method can be used to provide auxiliary measurements to limit the drift caused by inertial systems.

REFERENCES

[1] F. Benetazzo, F. Ferracuti, A. Freddi, A. Giantomassi, S. Iarlori, S. Longhi, A. Monteriù and D. Ortenzi (2015) "AAL Technologies for independent life of elderly people". *ser. Biosystems & Biorobotics, Ambient Assisted Living: Italian Forum 2014, Springer International Publishing*, 11:329–343.

[2] S. Fioretti, S, T. Leo and S. Longhi (2000) "A navigation system for increasing the autonomy and the security of powered wheelchairs". *IEEE Transactions on Rehabilitation Engineering*, 8(4):490–498.

[3] R. Simpson, D. Poirot and F. Baxter (2002) "The hephaestus smart wheelchair system". *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(2):118–122.

[4] F. Leishman, V. Monfort, O. Horn and G. Bourhis (2014) "Driving assistance by deictic control for a smart wheelchair: The assessment issue". *IEEE Transactions on Human-Machine Systems*, 4(1):66–77.

[5] J. Kim, X. Huo, J. Minocha, J. Holbrook, A. Laumann and M. Ghovanloo (2012) "Evaluation of a smartphone platform as a wireless interface between tongue drive system and electric-powered wheelchairs". *IEEE Transactions on Biomedical Engineering*, 59(6):1787–1796.

[6] L. Armesto, G. Ippoliti, S. Longhi and J. Tornero (2008) "Probabilistic self-localization and mapping - An asynchronous multirate approach". *IEEE Robotics and Automation Magazine*, 15(2):77–88.

[7] A. D'Amico, G. Ippoliti and S. Longhi (2006) "A multiple models approach for adaptation and learning in mobile robots control". *Journal of Intelligent & Robotic Systems*, 47(1):3–31.

[8] A. Freddi, S. Longhi and A. Monteriù (2012) "A Coordination Architecture for UUV Fleets". *Journal of Intelligent Service Robotics*, 5(2):133–146.

[9] G. Ippoliti, L. Jetto and S. Longhi (2005) "Localization of mobile robots: Development and comparative evaluation of algorithms based on odometric and inertial sensors". *Journal of Robotic Systems*, 22(12):725–735.

[10] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza (2011) "Introduction to Autonomous Mobile Robots". *Intelligent Robotics and Autonomous Agents series, The MIT Press*.

[11] S. Hutchinson, G. D. Hager and P. I. Corke (1996) "A tutorial on visual servo control". *IEEE Transactions on Robotics and Automation*, 12(5):651–670.

[12] E. Marchand, F. Spindler and F. Chaumette (2005) "Visp for visual servoing: a generic software platform with a wide class of robot control skills". *IEEE Robotics & Automation Magazine*, 12(4):40–52.

[13] R. Luo, S. C. Lin and C. C. Lai (2008) "Indoor autonomous mobile robot localization using natural landmark". In: *Proceedings of the 34th Annual Conference of IEEE Industrial Electronics (IECON)*, Nov. 2008, Orlando (USA), pp. 1626–1631.

[14] H. Zhang, L. Zhang, and J. Dai (2012) "Landmark-based localization for indoor mobile robots with stereo vision". In: *Proceedings of the Second International Conference on Intelligent System Design and Engineering Application (ISDEA)*, Jan. 2012, Sanya (China), pp. 700–702.

[15] G. Jang, S. Kim, J. Kim and I. Kweon (2005) "Metric localization using a single artificial landmark for indoor mobile robots". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Aug. 2005, Edmonton (Canada), pp. 2857–2862.

[16] I. Szentandrási, A. Herout and M Dubská (2013) "Fast detection and recognition of QR codes in high-resolution images". In: *Proceedings of the 28th Spring Conference on Computer Graphics*, May 2012, Smolenice (Slovakia), pp. 129–136.

[17] S. Tansuriyavong, K Higa and C. Boonmee (2011) "Development of Guide Robot by Using QR Code Recognition". In: *Proceedings of the Second TSME International Conference on Mechanical Engineering*, Oct. 2011, Krabi (Thailand).

[18] H. Kobayashi (2012) "A new proposal for self-localization of mobile robot by self-contained 2d barcode landmark". In: *Proceedings of the SICE Annual Conference*, Aug. 2012, Akita (Japan), pp. 2080–2083.

[19] M. Rostkowska, M. Topolski (2015) "On the Application of QR Codes for Robust Self-localization of Mobile Robots in Various Application Scenarios". In: *Progress in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing*, 351:243–252. Springer, Cham.

[20] H. Zhang, C. Zhang, W. Yang and C. Y. Chen (2015) "Localization and navigation using QR code for mobile robot in indoor environment". In: *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2015, Zhuhai (China), pp. 2501–2506.

[21] K. Okuyama, T. Kawasaki, and V. Kroumov (2011) "Localization and position correction for mobile robot using artificial visual landmarks". In: *Proceedings of the International Conference on Advanced Mechatronic Systems (ICAMechS)*, Aug. 2011, Zhengzhou (China), pp. 414–418.

[22] S.J. Lee, G. Tewolde, J. Lim and J. Kwon (2015) "QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument". *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Busan (Korea), 2015, pp. 965-970.

[23] Ros (2009) [Online]. Available: http://www.ros.org/. Accessed 12 July 2017.

[24] L. Chiari, U. Della Croce, A. Leardini and A. Cappozzo (2005) "Human movement analysis using stereophotogrammetry. Part 2: instrumental errors". *Gait & Posture*, 21(2):197–211.

[25] L. Cavanini, F. Benetazzo, A. Freddi, S. Longhi and A. Monteriù (2014) "SLAM- based Autonomous Wheelchair Navigation System for AAL Scenarios". In: *Proceedings of the IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Senigallia (Italy), Sep. 2014, pp. 1–6.

[26] G. Cimini, F. Ferracuti, A. Freddi, S. Iarlori and A. Monteriù (2013) "An inertial and qr code landmarks-based navigation system for impaired wheelchair users". *Ambient Assisted Living: Italian Forum 2013, Springer International Publishing*, pp. 205–214.

[27] E. Marchand, F. Spindler, and F. Chaumette (2005) "Visp for visual servoing: a generic software platform with a wide class of robot control skills". *IEEE Robotics Automation Magazine*, 12(4):40–52.

[28] H. Jeon, W. Myeong, J. U. Shin, J. W. Park, H. J. Jung and H. Myung (2014) "Experimental validation of visually servoed paired structured light system (visp) for structural displacement monitoring". *IEEE/ASME Transactions on Mechatronics*, 19(5):1603–1611.

[29] D. Agravante, J. Pages and F. Chaumette (2013) "Visual servoing for the reem humanoid robot's upper body". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, Karlsruhe (Germany), pp. 5253–5258.

[30] S. Birchfield (2007) "Klt: An implementation of the kanade-lucas-tomasi feature tracker". [Online]. Available: https://www.ces.clemson.edu/ stb/klt/. Accessed 12 July 2017.

[31] Inria Lagadic (2015) "Markerless 3D model-based tracker module overview". [Online]. Available: https://visp.inria.fr/mbt/. Accessed 12 July 2017.

Fast Autonomous Landing on a Moving Target at MBZIRC

Marius Beul, Sebastian Houben, Matthias Nieuwenhuisen, and Sven Behnke

***Abstract*— The ability to identify, follow, approach, and intercept a non-stationary target is a desirable capability of autonomous micro aerial vehicles (MAV) and puts high demands on reliable target perception, fast trajectory planning, and stable control. We present a fully autonomous MAV that lands on a planar platform mounted on a ground vehicle, relying only on onboard sensing and computing.**

We evaluate our system in simulation as well as with real robot experiments. Its resilience was demonstrated at the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) where it worked under competition conditions. Our team NimbRo ranked third in the MBZIRC Challenge 1 and – in combination with two other tasks – won the MBZIRC Grand Challenge.

I. Introduction

Fast autonomous landing of micro aerial vehicles (MAV) on moving platforms is challenging. One of the three tasks during the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 consisted of landing a flying robot on an artificial pattern on top of a golf cart, moving with $15\,\mathrm{km\,h^{-1}}$ on a figure eight course (cf. Fig. 1). The arena for this task had a size of $90\,\mathrm{m} \times 60\,\mathrm{m}$. The entire competition was executed under challenging outdoor conditions with temperatures of up to $38\,^{\circ}\mathrm{C}$ and strong gusts of wind. Although the vehicle would slow down after a given amount of time, a team had to land the robot in the first minutes, autonomously, and without any damage in order to receive a high score for its run. In fact, the teams ranked highest were able to complete the task in less than $30\,\mathrm{s}$ after takeoff, including the time needed to search for the moving target.

The MBZIRC task may seem like a toy example with no immediate application, but landing on a moving platform is an important step towards operating MAVs in more complex dynamical environments in the future. Not only does the task require precise state estimation but also low-latency, accurate detection and prediction of the target and MAV movements.

In this paper, we present our integrated MAV system designed for landing on a moving platform, including

- tailored hardware design,
- robust perception and fast tracking of a landing pattern,
- state estimation for the moving target, and
- fast, analytical trajectory generation for interception.

We evaluate our approach in both simulated and robot experiments, and report results from the MBZIRC competition.

This work has been supported by a grant of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC), and grants BE 2556/7-2 and BE 2556/8-2 of the German Research Foundation (DFG).

The authors are with the Autonomous Intelligent Systems Group, Computer Science VI, University of Bonn, Germany mbeul@ais.uni-bonn.de



Fig. 1. Final approach only seconds before a successful landing on a moving vehicle at the MBZIRC Grand Challenge.

II. Related Work

For reasons of brevity, in this overview, we cover lines of research performing the landing without cooperation between the robot and the ground vehicle. Lee et al. [1] demonstrated the viability of the task by using visual servoing in order to maneuver above the moving pattern, and relying on a motion-capture system for external state estimation. A similar system was demonstrated by Serra et al. [2] who also use visual servoing but do not rely on a vision-based distance estimation to the target. In comparison to our system, both approaches are evaluated with a slow or even static target. Borowczyk et al. [3] use a system of two cameras and filter the detections together with an IMU and GPS receiver mounted at the target. They report landing velocities of up to $50\,\mathrm{km\,h^{-1}}$. Landing indoors on an inclined plane was achieved by Vlantis et al. [4] who designed an adapted model-predictive controller to optimize the local trajectory in real time. However, due to the computational demand, optimization was done on an external base station and, thus, required a stable network connection.

Fast real-time trajectory planning and control is an active area of research. Ezair et al. [5] compare polynomial trajectory generation algorithms regarding the order, state constraints, and constraints on initial and final conditions. In their works [6] and [7], Mueller et al. present a trajectory generator similar to our work. It is also capable of approaching the full target state (position, velocity, and acceleration) and is real-time capable. Analogue to our approach, they use jerk (respectively the rotational velocity ω) as system input, but the convex optimization problem is solved numerically. Generated trajectories are not time-optimal.

From other MBZIRC participants, we want to cite the early work by the team of the Korea Advanced Institute

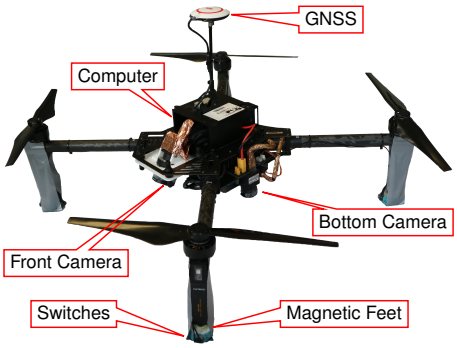


Fig. 2. Design of our MAV equipped with two cameras, magnetic feet, and a lightweight but fast onboard PC (shielding removed for better view).

of Science and Technology [8] where landing on a larger platform at a velocity of $0.75\,\mathrm{m\,s^{-1}}$ was demonstrated, but the visual detection was still simplified by a marker reflecting infrared light.

In contrast to prior works, the setup of the MBZIRC challenge required most of the processing (i.e., state estimation, control, sensor processing, mission- and trajectory planning) to take place onboard the system. With the exception of a ground station for GPS where a reference positioning signal is provided over WiFi if and when a connection is available, no external resources are used in our system.

III. System Setup

Our MAV, shown in Fig. 2, is based on the DJI Matrice 100 platform. It is equipped with a small but fast Gigabyte GB-BSi7T-6500 onboard PC with an Intel® Core™ i7-6500U CPU running at $2.5/3.1\,\mathrm{GHz}$ and $16\,\mathrm{GB}$ of RAM. The landing pattern is perceived by two Point Grey BFLY-U3-23S6M-C greyscale cameras with 2.3 MP. The first camera—equipped with a wide-angle lens with an apex angle of $195° \times 195°$—points downwards. To facilitate the detection of a far-away pattern and to keep it in the field of view (FoV) during descent on a glide path, the second camera—with an apex angle of $69° \times 85°$— points $30°$ into forward direction. Both cameras capture 40 frames per second, resulting in 80 frames per second in total.

We replaced the landing feet of the MAV with strong magnets with a total rated force of $860\,\mathrm{N}$ to keep it in place on the moving target with ferromagnetic surface after landing. A successful landing is detected by eight micro switches attached to the landing feet. The switches are individually connected to an Arduino Nano v3.0 that serves as bridge to our onboard computer.

For allocentric localization and state estimation, we employ the filter onboard the DJI flight control that incorporates GNSS (global navigation satellite system) and IMU data. To avoid electromagnetic interference between components—in particular USB 3.0 and GPS—the core of our MAV is wrapped in electromagnetic shielding material. This increases the system stability significantly. Fig. 3 gives an overview of the information flow in our system. We use the robot operating system (ROS) as middleware on the MAV and the ground control station. We communicate over WiFi

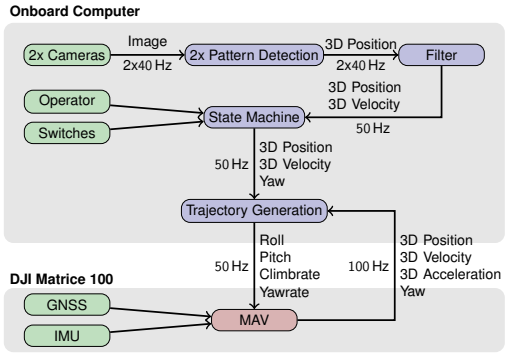


Fig. 3. Structure of our method. Green boxes represent external inputs like sensors, blue boxes represent software modules, and the red box indicates the MAV hardware. All software components use ROS as middleware. Position, Velocity, Acceleration and Yaw are allocentric.

with a robust UDP protocol, developed for connections with low-bandwidth and high-latency [9].

IV. Landing Pattern Perception

When detecting the pattern with a camera, one must consider two main objectives:

- the detection process itself should be low-latency and yield accurate results and
- the detection range should be as large as possible.

A. Landing Pattern Detection

We developed a multi-stage detection pipeline (cf. Fig. 4): The camera image is transformed to a bird's eye representation (cf. Fig. 4 (c))[1], a segmentation step detects line-like structures within the image (cf. Fig. 4 (d)) that are processed via a circular Hough transform in order to generate a number of hypotheses and their respective confidence.

Let r be the radius of the landing pattern in meters. In order to maximize the detection range the MAV's flying

[1]Please note that a) the camera setup is not aligned to the ground plane and b) the camera may have an arbitrary orientation during rapid manoeuvres, thus, a prior image transform is necessary.

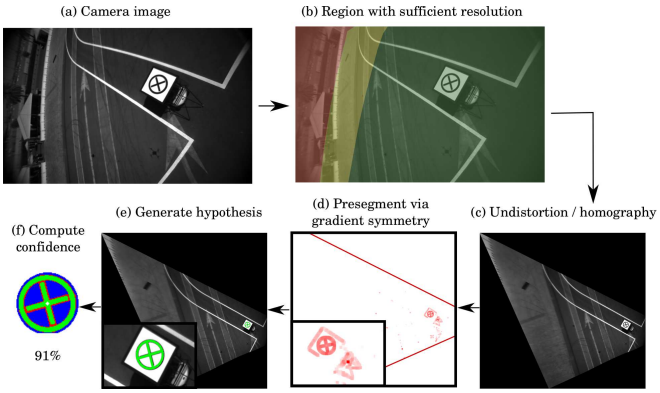


Fig. 4. Landing pattern detection from the front camera during the competition: (a) original camera image, (b) image regions with sufficient resolution for pattern detection (green), insufficient resolution (yellow), and regions above the ground plane (red), (c) bird's eye representation from a region with (mostly) sufficient resolution, (d) results from symmetry segmentation shown dilated for better visibility, (e) initial hypothesis in green, (f) confidence computation via pattern-detection overlay: green and blue denote correct pixels, red incorrect ones.

height h, obtained by relative barometric measurements, and its attitude, represented by the IMU gravitational vector $\hat{r}_z$ within camera coordinates, is taken into account. Then the rotation matrix

$$\begin{aligned} \hat{R} &:= (\hat{r}_x, \hat{r}_y, \hat{r}_z) \\ \text{with } \hat{r}_x &:= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T \times \hat{r}_z, \\ \hat{r}_y &:= \hat{r}_z \times \hat{r}_x \end{aligned}$$

describes the rotation from the camera frame into a camera frame where the image plane is aligned with the ground plane, i.e., the matrix

$$M := K_g \hat{R} K_c^{-1}$$

with accordingly chosen camera matrices K_g, K_c describes a pixel coordinate transform into a bird's eye representation via homogeneous coordinates. Finally, taking the lens distortion into account, we arrive at

$$(u, v) \mapsto M \begin{pmatrix} f(u, v) \\ 1 \end{pmatrix} \tag{1}$$

with an invertible radial-tangential lens undistortion function f operating on the image coordinates (u, v). K_c is given by the camera intrinsics, K_g is set to

$$K_g := diag\left(\tfrac{2r \cdot \rho}{h}, \tfrac{2r \cdot \rho}{h}, 1\right)$$

where ρ is the desired resolution of the pattern (for ρ we choose 60 pixels or lower, depending on the maximum possible resolution of the pattern in the original camera image). It remains to compute those image regions that yield at least the resolution ρ_{min} required for detection (chosen as 20 pixels) when transformed by M. To this end, a point grid with a fixed stride of k pixels in the original image is mapped via M and the distance between neighboring points is computed. If this distance is below $\frac{2rk}{\rho_{min}}$, the resolution of the the corresponding image patch is too low for detection. The maximum rectangular region in the camera image containing all grid points with sufficient resolution after mapping (i.e., the maximum rectangle enclosing only green points in Fig. 4 (b)) is computed via a heuristic approach and the resulting region of the camera image is subsequently transformed. In order to efficiently execute this transform, please note that f is static such that a pixel-wise lookup table can be precomputed. The second part of the mapping in Equation (1) is linear-projective and can be computed very efficiently, in particular on rectangular regions when not the entire image has to be traversed.

For segmentation, fast symmetry detection is performed. Local pairs of pixels with approximately converse gradients are identified and their center is segmented if their distance matches the expected line width of the pattern ($10\,\text{cm}$). The procedure is detailed in [10]. A circular Hough transform—the approximate diameter of the circle in image dimensions is known—provides a fixed number of hypothesis that are subsequently confirmed if two lines with a central perpendicular intersection are detected within. Finally, a confidence measure is computed by thresholding the potential region of the intensity image with the expected quantile of dark versus white pixels and computing the ratio of the correct pixels with an artificial overlay (cf. Fig. 4 (f)).

In order to meet the required latency, after a sufficiently confident detection only a rectangular image region around the previous position is considered in the following iterations, reducing the algorithm to steps (c) – (f) from Fig. 4. In order to follow the pattern as long as possible, the requirement on minimum image resolution of the pattern is ignored during this tracking phase.

B. Landing Pattern Tracking

To predict the movement of the landing target in a world frame, we use a simpler version of our onboard state estimation filter presented in previous work [11]. We estimate position and velocity of the target in an allocentric frame with a constant velocity assumption in the prediction step. We consider detections from both cameras as independent observations and the filter merges them to a coherent world view. The pose of the MAV and the projection of the landing target perceptions into the allocentric frame are subject to the same localization error. Thus, the allocentric estimate of the target is consistent with the egocentric control of the MAV. Since we do not make any assumptions about the path of the landing target, e.g., moving in an eight pattern, our method is applicable to arbitrary pattern motions and independent from exact absolute MAV localization.

V. Landing Control

Since the total time to land during the challenge is crucial, we employ our time-optimal trajectory generation method described in [12] with the extensions from [13].

A. MAV Model

We assume the MAV to follow rigid body dynamics and simplify it as a point mass with jerk j as system input. Following Newton's second law, the system is a triple integrator in each dimension (x, y, z) with position p, velocity v, acceleration a, and jerk j:

$$\dot{p} = v, \qquad \dot{v} = a, \qquad \dot{a} = j. \tag{2}$$

Thus, the three-dimensional allocentric state of the MAV $\boldsymbol{x}$ can be expressed by

$$\boldsymbol{x} = \begin{pmatrix} p_x & p_y & p_z \\ v_x & v_y & v_z \\ a_x & a_y & a_z \end{pmatrix}. \tag{3}$$

We assume jerk j to be the direct control input to the linear system. Without loss of generality, we define the z-axis to be collinear to the gravity vector. Furthermore, we define the origin to be the middle of the arena and the xy-plane equal with the ground plane. We do not model

- moment of inertia,
- drag,
- yaw dynamics, and
- coupling of the axes that occurs in non-hover conditions,

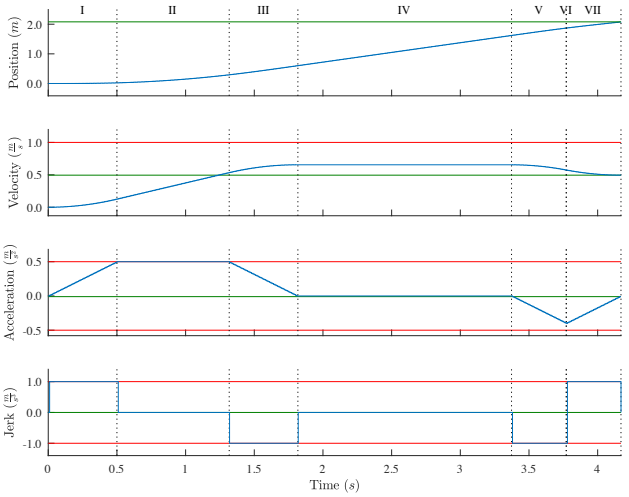


Fig. 5. This time-optimal trajectory was generated with our method. Starting from state ($p_x = 0\,\mathrm{m}, v_x = 0\,\mathrm{m\,s^{-1}}, a_x = 0\,\mathrm{m\,s^{-2}}$), it brings the MAV to the target state ($p_x = 2.08\,\mathrm{m}, v_x = 0.5\,\mathrm{m\,s^{-1}}, a_x = 0\,\mathrm{m\,s^{-2}}$). The trajectory satisfies constraints $v_{max} = 1\,\mathrm{m\,s^{-1}}$, $a_{max} = 0.5\,\mathrm{m\,s^{-2}}$, and $j_{max} = 1\,\mathrm{m\,s^{-3}}$. The calculated switching times are $t_1 = 0.5\,\mathrm{s}$, $t_2 = 0.82\,\mathrm{s}$, $t_3 = 0.5\,\mathrm{s}$, $t_4 = 1.55\,\mathrm{s}$, $t_5 = 0.4\,\mathrm{s}$, $t_6 = 0.0\,\mathrm{s}$, and $t_7 = 0.4\,\mathrm{s}$. The trajectory corresponds to the x-axis in Fig. 7. It is suboptimal (maximum velocity not reached), since this axis is slowed down to synchronize with the slower y-axis.

but rely on fast replanning to account for model uncertainties and unmodeled effects instead. Since our model is parameterless, our approach generalizes to all multicopters and no cumbersome parameter tuning is required.

B. Time-optimal Control

Based on the simple triple integrator model, our method analytically generates third-order time-optimal trajectories that satisfy input ($j_{min} \leq j \leq j_{max}$) and state constraints ($a_{min} \leq a \leq a_{max}$, $v_{min} \leq v \leq v_{max}$). The planned trajectory consists of up to seven phases of constant jerk ($j = j_{max} \vee j = 0 \vee j = j_{min}$), resulting in a third-order bang-zero-bang trajectory.

Fig. 5 shows a 1-dimensional trajectory. We synchronize all axes to arrive at the target at the same time. By doing so, the MAV flies on a straight glide path towards the landing position.

Furthermore, we use the ability of our trajectory generation method to calculate an optimal interception point, based on the current velocity of the target. We predict the target motion and do not fly to the current target location, but to the position, the MAV can intercept the target assuming a constant velocity motion and respecting the MAVs constraints. Although the assumption of constant velocity may not be justified in the curved parts of the figure eight since the acceleration is relatively large ($|a_{target}| \approx 1\,\mathrm{m\,s^{-2}}$), we found the error to be compensable by fast replanning.

C. MPC Application

We use the above mentioned trajectory generation method as an MPC (Model Predictive Controller), running in a closed loop with 50 Hz. Our hardware does not support directly sending jerk commands. We therefore assume pitch and roll to directly relate to $\theta = \mathrm{atan2}(a_x, g)$ and $\phi = \mathrm{atan2}(a_y, g)$.

TABLE I
PARAMETERS USED AT MBZIRC

Param	Axis	Value	Param	Axis	Value
v_{max}	x,y	$8.33\,\mathrm{m\,s^{-1}}$	v_{max}	z	$1.00\,\mathrm{m\,s^{-1}}$
a_{max}	x,y	$4.73\,\mathrm{m\,s^{-2}}$	a_{max}	z	$10.0\,\mathrm{m\,s^{-2}}$
j_{max}	x,y	$5.00\,\mathrm{m\,s^{-3}}$	j_{max}	z	$50.0\,\mathrm{m\,s^{-3}}$
Δt_{setp}	x,y	0.15 s	Δt_{setp}	z	0.50 s

So, instead, we send smooth pitch θ and roll ϕ commands for horizontal movement and smooth climb rates v_z in z direction. Due to the linearization, the acceleration constraint relates to an attitude constraint with $\theta_{max} = \mathrm{atan2}(a_{max}, g)$.

Our method plans the whole trajectory to the target instead of relying on a small constant lookahead commonly found in MPCs. Since the whole future trajectory is known at every replanning time step, one can choose from which future time Δt_{setp} to send the commands to the system. If the value is small (e.g., $\Delta t_{setp} = 0\,\mathrm{s}$), the system will react slowly. This is because small lookahead values will render the setpoint changes from the current state to be small and thus the underlying control loops slow. If the lookahead value is too large, the system can become unstable or perform suboptimal. Also communication delay has negative impact on the system and is compensated by choosing an appropriate lookahead. We experimentally determined that the values found in Tab. I offer good performance.

In Sec. V-A we report that we model the MAV in three orthogonal axes with the z-axis collinear to the gravity vector. The rotation about the z-axis α however is not defined. We define the rotation to be the allocentric angle of the current position to the target position $\alpha = \mathrm{atan2}(p_{y_{wayp}} - p_y, p_{x_{wayp}} - p_x)$. By doing so, we project the per-axis velocity constraint to lie in the axis of the dominant motion. Otherwise, the global horizontal velocity constraint would result in being $v_{max} = \sqrt{v_{max_x}^2 + v_{max_y}^2}$ and thus violating the maximum allowed velocity at the MBZIRC of $30\,\mathrm{km\,h^{-1}}$.

D. Yaw Control

Although an arbitrary number of axes can be synchronized, we do not consider the yaw-axis to be synchronized with the x, y and z-axis. For simplicity, we use proportional control for the yaw-axis Ψ. The yaw rate setpoint $\dot{\Psi}_{setp} = K_p \cdot (\Psi_{setp} - \Psi)$ is sent to the MAV. A couple of different yaw behaviors can be selected by the state machine described in Sec. VI, depending on the current situational requirement. The MAV can point towards

- a defined allocentric yaw angle,
- the current target,
- the optimal interception point,
- forward direction (current MAV horizontal velocity vector), and
- direction of target motion (current target horizontal velocity vector).

VI. Mission Control State Machine

The behavior of the MAV is controlled by a state machine that serves as a generator for position, velocity, and yaw

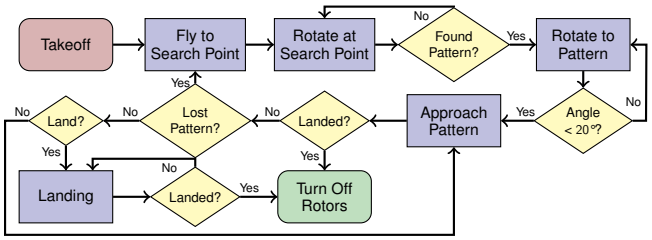


Fig. 6. The flowchart of our state machine. Besides the basic behavioral control, it features strategies to recover from failed landing approaches.

setpoints for the lower layers. Fig. 6 shows a flowchart of our state machine. After takeoff, the MAV flies with maximum velocity to a search point 8 m above the field, already exploring the arena through rotations. When the landing pattern is first detected, we rotate the front camera approximately into pattern direction before starting the descent.

We restrict the descent rate based on the distance to the target to ensure good perceivability of the pattern in the cameras. The state machine transfers the yaw authority to the low-level trajectory generation during pattern following. Depending on the distance to the target, the MAV yaws towards the target or into flight direction, to ensure that the pattern is visible in either the bottom camera or—in stable forward flight without fast rotations—the front camera.

The final landing decision is based on relative orientation, distance and relative height to the pattern, and visibility in the cameras. If the landing decision has been taken, the MAV descents until ground contact is detected by the switches at its feet. This is necessary as the pattern cannot be reliably tracked during the landing due to its proximity to the MAV.

To prevent unstable behavior while manoeuvring in the vicinity of the fast moving landing pattern or in corner cases for the perception, the descent is completely aborted and the landing procedure restarts from the initial search point above the field when the pattern is lost during following. For safety, we also detect premature landings in the pattern following state and turn off the rotors. Since the state machine is the only subsystem which the operator interfaces with during flight, we built a distinct GUI for situational awareness of the operator.

VII. Evaluation

We evaluate our system in simulation as well as with a real MAV. Videos of our evaluation can be found on our website[2].

A. Evaluation in Simulation

Landing on a target moving at relatively high speed imposes a risk for the MAV and persons who move the target. Thus, we tested all individual software components and the integrated software system in simulation.

Fig. 7 shows our Matlab simulation for the optimal interception of a moving landing pattern. We model the MAV as pictured in Sec. V-A. It can be seen that the MAV lunges to eliminate any velocity difference to the target when arriving at the interception point.

[2]http://www.ais.uni-bonn.de/videos/ECMR_2017_Beul

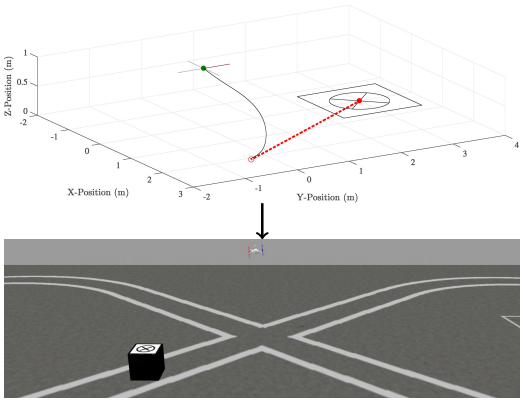


Fig. 7. Landing simulation in Matlab and Gazebo. We first simulate the interception of the target with a simplified linear model. The MAV is marked with a green dot. The target is marked with an solid red dot. The predicted target trajectory is marked with a dashed line, ending in the interception point (red ring). Subsequently, we modeled the MBZIRC arena including the moving target in Gazebo. The MAV can be simulated with HIL, employing a complex motion model and challenging environmental conditions.

To achieve a high level of realism, we also modeled the MBZIRC arena and the moving target in the RotorS simulator [14]. In addition to the physics-based MAV simulation of RotorS, we implemented a hardware in the loop (HIL) bridge, employing the DJI simulator. Here, the flight control is connected to the DJI Assistant 2 software via USB. Instead of controlling the motors of the real MAV, the flight control firmware sends control commands to the simulator, where MAV dynamics and sensors like IMU and GPS are simulated and sent back to the flight control. Thus, for our ROS middleware it is undeterminable if the real or the simulated MAV is used.

B. Real-robot Evaluation at MBZIRC

Our system was used to compete in the MBZIRC. We were able to place third in Challenge 1 and – in combination with two other tasks – first in the Grand Challenge of the total 24 resp. 14 competitors. During the first run in Challenge 1, we first experienced a hardware problem with the USB 3.0 connection of the front camera and were forced to restart. After fixing this issue, we were able to successfully land in 34 s—measured from spinning up the rotors to landing on the pattern. In total, the time from the start of the challenge to landing—including fixing the MAV—was 112 s, resulting in the third place in the final ranking. In order to fix the connection, we attached more shielding for the second trial. Unfortunately, this shielding negatively affected the compass of the MAV so that it went into failsafe mode directly after the start. We canceled the second trial since we could not fix this issue fast enough to improve our time from the first challenge run.

In the first trial of the Grand Challenge, we were able to land in 42 s. Fig. 8 shows the trajectory and detections of this trial. After reaching the center of the field, the MAV searched for another 11.9 s for the target because the cart was in a disadvantageous position. In the second trial, we could not improve our landing time and canceled this trial after 42 s.

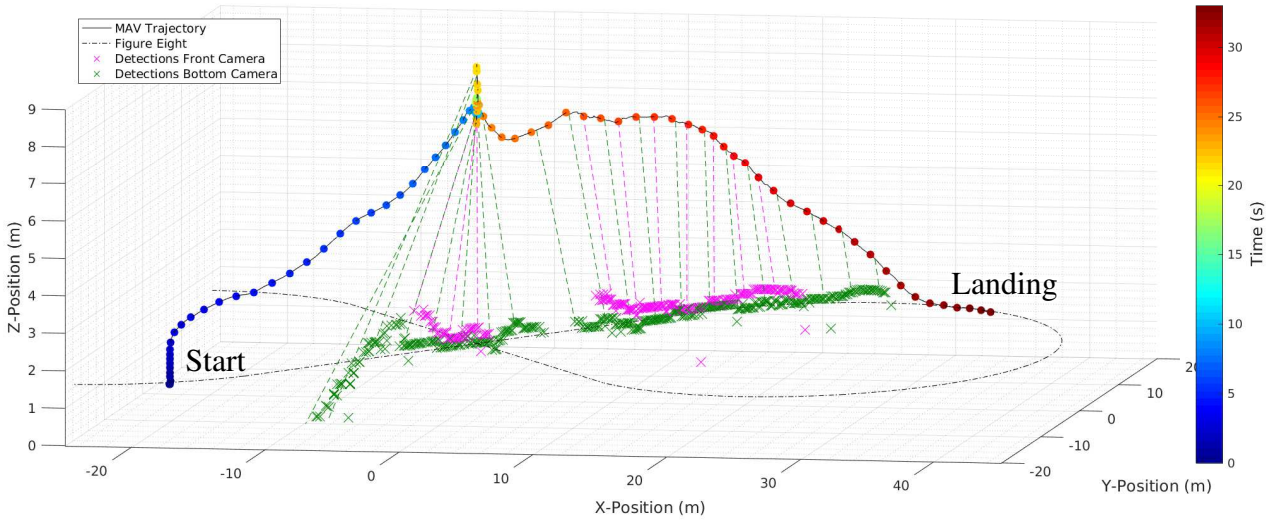


Fig. 8. Landing in MBZIRC Grand Challenge. First, the MAV starts in the middle of the left circle and flies straight up to a height of 1.5 m to not collide with near objects. Next, it flies to the center of the field with a height of 8 m to search for the landing target. The total ascent takes 8.5 s. After 20.4 s, the landing target is first detected in the bottom camera. Immediately, the MAV begins to descent while tracking the target in both cameras. The descent only takes 11.6 s, resulting in a total completion time of 32 s. Due to the fast motion of the target, the MAV cannot descent fast enough to reach the target on the straight segment and has to land in the curved segment of the figure eight. The challenge completion time from start signal to landing is 42 s. Colored markers are placed every 250 ms on the trajectory. Every 20th of all 585 detections is indicated with the corresponding viewpoint on the trajectory.

VIII. Conclusion

We have provided detailed insight into our robust MAV setup for quickly landing on a fast moving target. The viability of this approach has been demonstrated in an outside scenario with minimum preparation time during the MBZIRC where the MAV consistently performed the landing as one of the fastest among all competitors.

In particular the adaptive and fast trajectory replanning combined with a high-frequency pattern detection turned out to reliably match direction and velocity with the moving target. Furthermore, the use of two cameras in combination with an adaptive yawing strategy enabled us to track the target pattern under fast manoeuvres and in close proximity.

We believe that our contribution, but in general all experience from the MBZIRC landing challenge, will facilitate new ideas of how to operate flying robots in dynamic—and hence real-world—environments.

References

[1] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[2] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," *IEEE Trans. on Robotics*, vol. 32, no. 6, pp. 1524 – 1535, 12 2016.

[3] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *ArXiv e-prints*, vol. 1611.07329, 11 2016.

[4] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[5] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *The Int. J. of Robotics Research*, vol. 33, no. 6, pp. 898–916, 2014.

[6] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *Proc. of the European Control Conference (ECC)*, 2011.

[7] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasability verification," in *Proc of IEEE/RSJ Int Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[8] H. Lee, S. Jung, and D. H. Shim, "Vision-based uav landing on the moving vehicle," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016.

[9] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, S. Schüller, D. Schwarz, A. Topalidou-Kyniazopoulou, and S. Behnke, "NimbRo Rescue: Solving disaster-response tasks through mobile manipulation robot Momaro," *Journal of Field Robotics*, vol. 34, no. 2, pp. 400–425, 2017.

[10] S. Houben, M. Neuhausen, M. Michael, R. Kesten, F. Mickler, and F. Schuller, "Park marking-based vehicle self-localization with a fisheye topview system," *J. of Real-Time Image Processing*, pp. 1–16, 2015.

[11] M. Beul, N. Krombach, Y. Zhong, D. Droeschel, M. Nieuwenhuisen, and S. Behnke, "A high-performance mav for autonomous navigation in complex 3d environments," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 6 2015.

[12] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2016.

[13] ——, "Fast full state trajectory generation for multirotors," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2017.

[14] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS–a modular gazebo MAV simulator framework," in *Robot Operating System (ROS): The complete reference*, A. Koubaa, Ed., 2016, vol. 1, ch. 23, pp. 595–625.

Path tracking of a four-wheel steering mobile robot: A robust off-road parallel steering strategy

Mathieu Deremetz[1], Roland Lenain[1], Adrian Couvent[1], Christophe Cariou[1] and Benoit Thuilot[2,3]

Abstract— **In this paper, the problem associated with accurate control of a four-wheel steering mobile robot following a path, while keeping different desired absolute orientations and ensuring different desired lateral deviations, is addressed thanks to a backstepping control strategy. In particular, the control of each steering angle is investigated through a new parallel steering approach based on an extended kinematic model of a bicycle-model robot assuming that the two front steering angles are equal and likewise for the two rear ones. Two control laws are then proposed to ensure a suitable path following according to orientation and position conditions. In order to balance the lateral effects, notably the sideslip angles, an observer has been used to estimate the sliding. This estimation permits to feed the proposed control laws appropriately, enabling an accurate path tracking and orientation keeping along the trajectory. This new point of view permits to achieve difficult manoeuvres in narrow environments such as a parallel parking or sharp turns. Previous approaches have focused on the control of four-wheel steering mobile robots with respect to the trajectory but do not combine path following with independent heading angle control and slippery conditions.**

I. INTRODUCTION

Initiated by W. Grey Walter thanks to *Elmer* and *Elsie* robots in [5], mobile robotics constitutes today a major economic and scientific topic. One of the best examples is the progress achieved on autonomous vehicles, notably cars [11] which are now close to the market. The transport sector is not the only field that benefits from mobile robotics. Indeed, since the beginning of the twenty-first century, research has focused its interest on the automation of high-speed vehicles in off-road context. Thus, sectors such as agriculture [2] or even defence and space [15] are considering autonomous ground vehicles in order to achieve painful tasks, submitted to very harsh and variable conditions.

Over the last few years, it can be noticed that mobile robotics developments are not only linked to automated manned vehicles but also to totally autonomous adaptable robots [12] that have more and non-standard mobility to be piloted. Concerning guidance, firstly based on the front-steering model used for cars, these mobility levels have indeed evolved through adding a rear steering axle, skid-steering configuration or using holonomic wheels. Another option, investigated in this paper, is the use of four independent steering wheels. Because of its high mobility, this configuration, compared to the latest, is well appropriate for performing tricky moves on road or in off-road contexts. Such a robot, equipped with four steering wheels, is then appropriate to achieved path tracking requiring suitable control of both heading and orientation.

Several approaches have been proposed in the framework of path following. Firstly developed for front-steering [13] or skid-steering [4], [8] mobile robots, classical approaches are designed using pure rolling without sliding assumption, as developed by [13] or [16]. Nevertheless, if the robot has to follow a trajectory at medium speed (beyond $2-3\,m/s$) the accuracy of the tracking may be seriously damaged, especially in off-road context. Then, sliding has to be taken into account to preserve the accuracy of the path tracking. This approach is well demonstrated for front-steering mobile robots in [9]. However, this approach only ensures the control of the lateral deviation of the robot w.r.t. the trajectory. Indeed, a crab angle between the orientation of the robot and the tangent to the path appears when the robot moves on a slope. To avoid this phenomena another control strategy has been proposed in [3] for four-wheel steering mobile robots in order to take advantage of both front and rear steering actuations to control both the lateral deviation and the angular deviation of the robot w.r.t. the trajectory during the path tracking. This latter approach is notably designed for autonomous vehicles with limited steering angles because of Ackermann steering geometry. Moreover, the orientation of the robot is controlled w.r.t. the trajectory. Another method in [17] proposed a parallel steering with equal front and rear steering angles but did not suggest a robust method to avoid the skid and the deviation from the desired path when grip conditions are poor.

With the raise of unmanned adaptable robots equipped with four independent steering wheels having a huge range of movement (beyond $\frac{\pi}{2}$ or even π radians) such as the *Thorvald platform* [6] or the agricultural robot proposed in [7], new kinds of movement during path tracking can be achieved. Indeed, if there is no passenger, the angular error of the robot w.r.t. the trajectory does not have to reach zero anymore. From this point of view, one can then imagine controlling a robot with a desired absolute orientation during the path tracking, which permits to substantially reduce the manoeuverability space.

In this paper, the autonomous following of a path with a desired absolute angle by a four-wheel steering mobile robot is investigated. More precisely, a backstepping approach has been developed in order to compute the front steering angles and the rear steering angles based on an extended kinematic

[1] Irstea, Technologies and Information Support System Research Unit, 9 avenue Blaise Pascal, CS 20085, 63178 Aubière, France firstname.lastname@irstea.fr

[2] Clermont Université, Université Blaise Pascal, Institut Pascal, BP 10448, 63000 Clermont-Ferrand, France firstname.lastname@univ-bpclermont.fr

[3] CNRS, UMR 6602, Institut Pascal, 63178 Aubière, France

model. To compensate for the effects of the grip conditions, an absolute observer, based on prior work [10], estimates the sideslip angles and feeds the proposed control laws in terms of sliding.

This paper is decomposed as follows. First, the extended kinematic model is recalled. The second part details the observer used to estimate the grip conditions of the robot wheels, while the third section describes the proposed control laws based on a backstepping strategy. Finally, simulations are conducted to demonstrate the results and the efficiency of the proposed approach and are illustrated through a few agricultural contexts.

II. MODELING

A. Mobile robot modeling

The objective of the proposed application is to allow for a four-wheel steering mobile robot to follow a previously defined trajectory (i.e., computed, previously learned or coming from a structure to follow). This trajectory may include turns and ground variations which may lead to sliding and then create grip condition modifications. To ensure accurate tracking by using an observer and a control algorithm, it is appropriate to use models that take into account the kinematics of the robot.

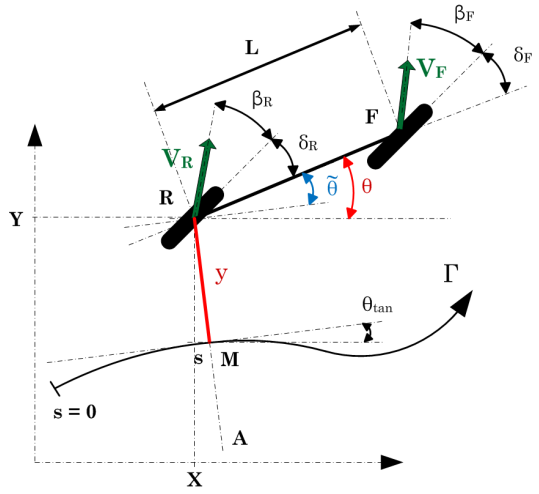


Fig. 1: Extended kinematic model of the robot with respect to the reference trajectory Γ and in the absolute frame defined by the X and Y coordinates.

In this paper, a four-wheel steered and drive robot is considered. As is commonly assumed, the robot is viewed as a bicycle, with an equivalent front steering angle δ_F, an equivalent rear steering angle δ_R and a wheelbase L (Fig. 1). The robot's speed is defined thanks to the speed of the front axle v_F and the speed of the rear axle v_R. The bicycle model is commonly used in the field of mobile robotics, but it is generally supposed that the rolling without sliding condition is satisfied. In an off-road context or at high speed, however, this assumption cannot be satisfied and leads to a lateral deviation of the robot during the tracking when control based on such a model is implemented, as pointed out in [1]. To overcome this phenomenon and take into account the non-ideal grip conditions, two additional variables, β_F and β_R, are added. These variables, called sideslip angles, denote the front and rear angles, respectively, between the tire orientation and the current speed vector orientation at the contact points F and R.

Thus, to follow the previously defined trajectory Γ and estimate the grip conditions β_F and β_R, state variables of the robot are defined as follows:

- s, the robot curvilinear abscissa. It is the curvilinear distance along Γ of point M, the point on Γ that is the closest to R. The curvature of Γ at point M is denoted $c(s)$,
- y, the robot tracking error or lateral deviation. It is the algebraic distance between R and M,
- X, the abscissa value of the center of the robot rear axle R in the absolute frame,
- Y, the ordinate value of the center of the robot rear axle R in the absolute frame,
- $\tilde{\theta}$, the robot angular error or angular deviation. It is the angle between the absolute robot heading, denoted θ and the orientation of the tangent to the trajectory at point M, denoted θ_{tan}.

In this paper, additional notations will be added in the next sections to define the desired offsets applied to the robot moves when tracking. Thus, in the following, y_d denotes the desired lateral deviation of the robot w.r.t. the trajectory while θ_d denotes the desired absolute orientation of the robot in the absolute frame.

B. Motion model in the absolute frame

According to classical kinematic analyses, such as those presented in [14], derivatives of the robot kinematic state variables (1) in the absolute frame can be written as follows:

$$\begin{cases} \dot{X} &= v_R \cos\left(\theta + \delta_R + \beta_R\right) \\ \dot{Y} &= v_R \sin\left(\theta + \delta_R + \beta_R\right) \\ \dot{\theta} &= \dfrac{v_F \sin\left(\delta_F + \beta_F\right) - v_R \sin\left(\delta_R + \beta_R\right)}{L} \end{cases} . \quad (1)$$

The derivative of the state variable $\dot{\theta}$ is highly recommended in its given form which has no singularity and allows the continuity of the algorithms. This formula is used both in Secs. III and IV to define the evolution model of the proposed observer and the second part of the control algorithm.

C. Motion equations with respect to a reference trajectory

According to classical kinematic analyses, such as those presented in [9], derivatives of the robot kinematic state variables (2) w.r.t. the reference trajectory can be written as follows:

$$\left\{ \begin{array}{rcl} \dot{s} & = & v_R \, \dfrac{\cos\left(\tilde{\theta} + \delta_R + \beta_R\right)}{1 - c(s)\, y} \\ \dot{y} & = & v_R \, \sin\left(\tilde{\theta} + \delta_R + \beta_R\right) \\ \dot{\tilde{\theta}} & = & \dfrac{v_F \, \sin\left(\delta_F + \beta_F\right) - v_R \, \sin\left(\delta_R + \beta_R\right)}{L} \\ & & -v_R \, \dfrac{c(s) \, \cos\left(\tilde{\theta} + \delta_R + \beta_R\right)}{1 - c(s)\, y} \end{array} \right. . \quad (2)$$

This model is defined if the $1 - c(s)\, y \neq 0$ condition is true. This means that the center of curvature A should not be superimposed with the center of the rear axle R. However, if the robot is properly initialized, such a case is never reached in practice.

III. ESTIMATION OF SIDESLIP ANGLES

To ensure a suitable convergence of the tracking and angular errors along the trajectory when facing poor grip conditions, it is compulsory to know accurately the values of the sideslip angles. Because there is no direct perception system allowing to measure such variables, it has been shown in [10] that an observer may be built to estimate the two sideslip angles with a satisfying accuracy. To achieve this estimation, the robot must be equipped with on-board sensors that permit to measure the position, the orientation and the velocity of the robot at a given point, here R. Moreover, according to the model (1), defined above, the velocity of the robot at F, has to be measured.

A. Observer state

In this paper, the proposed observer in [10] built for a two-wheel steering mobile robot is extended to a four-wheel steering one, according to the equations of the model (1). Because this approach is a version close to [10], only a short presentation is proposed in this section.

Hereafter, ξ should be consider as an effectively measured variable, while $\hat{\xi}$ should be considered as an observed or estimated variable. $\tilde{\xi}$ is defined as the observation error $\tilde{\xi} = \xi - \hat{\xi}$.

To achieve the indirect estimation of the sideslip angles, let us consider the state space ξ defined as follows:

$$\xi = \left[\begin{array}{c} \xi_{pos} \\ \xi_{\beta_i} \end{array} \right] , \quad (3)$$

where ξ is split into two sub-states:

- $\xi_{pos} = [X \;\; Y \;\; \theta]^T$, which constitutes the pose (position and orientation) of the robot in the absolute frame,
- $\xi_{\beta_i} = [\beta_F \;\; \beta_R]^T$, which is composed of the sideslip angles, to be estimated.

Its evolution model is based on (1), and may be written as follows:

$$\dot{\xi} = \left[\begin{array}{c} \dot{\xi}_{pos} \\ \dot{\xi}_{\beta_i} \end{array} \right] = \left[\begin{array}{c} f(\xi_{pos}, \xi_{\beta_i}, v_F, v_R, \delta_F, \delta_R) \\ 0_{2\times 1} \end{array} \right] , \quad (4)$$

where $f(\xi_{pos}, \xi_{\beta_i}, v_F, v_R, \delta_F, \delta_R)$ is directly deduced from the three equations of model (1).

B. Observer equations

Equations for the observer are as follows:

$$\dot{\hat{\xi}} = \left[\begin{array}{c} \dot{\hat{\xi}}_{pos} \\ \\ \dot{\hat{\xi}}_{\beta_i} \end{array} \right] = \left[\begin{array}{c} f(\xi_{pos}, \hat{\xi}_{\beta_i}, v_F, v_R, \delta_F, \delta_R) + \ldots \\ \ldots \alpha_{pos}(\tilde{\xi}_{pos}) \\ \alpha_{\beta_i}(\tilde{\xi}_{pos}) \end{array} \right] , \quad (5)$$

where $\alpha_{pos}(\tilde{\xi}_{pos})$ and $\alpha_{\beta_i}(\tilde{\xi}_{pos})$ are functions of the observation error attached to the pose part of the state ξ, and defined as follows:

$$\left\{ \begin{array}{rcl} \alpha_{pos}(\tilde{\xi}_{pos}) & = & K_{pos}\, \tilde{\xi}_{pos} \\ \alpha_{\beta_i}(\tilde{\xi}_{pos}) & = & K_\beta \left[\frac{\partial f}{\partial \xi_{\beta_i}} (\xi_{pos}, \hat{\xi}_{\beta_i}, v_F, v_R, \delta_F, \delta_R) \right]^T \tilde{\xi}_{pos} \end{array} \right. , \quad (6)$$

where:

- K_{pos} is 3×3 positive diagonal matrice,
- K_β is a positive scalar.

As shown in [10] and extended to a four-wheel steering mobile robot, the observer defined by (5) and (6) allows the convergence of the whole observed state $\hat{\xi}$ to the actual one ξ. As a consequence, the values for the sideslip angles, $\hat{\beta}_F$ and $\hat{\beta}_R$ are obtained and can feed a control law appropriately in order to follow a path while keeping the same absolute orientation.

IV. CONTROL ALGORITHMS

A. Proposed strategy

Previous approaches about parallel steering assume that the front and the rear steering are equal ($\delta_F = \delta_R$) such as in [17]. However, if the grip conditions are not sufficient the vehicle will skid and deviate from the desired path.

Such a control strategy for each steering angle is not sufficient to ensure a suitable parallel steering. Indeed, even if the grip conditions are taken into account the robot will skid because its number of mobility degree is lower than the number of mobility required to ensure the tracking and the absolute orientation. In this paper, the strategy proposes to add a degree of mobility by independently controlling the rear and the front angles. The rear steering will ensure the convergence of the robot to the desired trajectory while the front steering will ensure in addition the convergence of the robot orientation to the desired one. This additional mobility

will allow to correct the absolute orientation preventing the skid while following the path. Following subsections detail the backstepping strategy for each axle to reach this objective.

B. Control algorithm for the rear steering angles

The objective of this first step is to find a mathematical expression for δ_R that ensures the convergence of the rear lateral deviation y to a desired lateral deviation y_d.

It has been chosen to set a convergence distance instead of a settling time to design a control strategy independent from time and the speed of the robot for the convergence of the lateral deviation. To obtain this kind of convergence it is compulsory to use derivatives of the state vector w.r.t. the curvilinear abscissa. Hereafter, x' should be consider as the derivative of x w.r.t. the curvilinear abscissa s, such that $x' = \frac{\mathrm{d}\,x}{\mathrm{d}\,s}$.

Then from the product of the first equation of the model (2) with the inverse of the second one, the derivative of y w.r.t. the curvilinear abscissa, denoted y', can be written as follows:

$$y' \;=\; \tan\left(\tilde{\theta} + \delta_R + \hat{\beta}_R\right)\left(1 - c(s)\,y\right). \tag{7}$$

By denoting the error on the lateral deviation such that $e_y = y - y_d$, a way to ensure the convergence of this error to zero is to use a differential equation such that:

$$e'_y \;=\; y' - y'_d \;=\; k_y\, e_y\,, \tag{8}$$

with k_y a negative scalar defining the convergence distance for the exponential convergence of y to y_d imposed by (8).

By injecting in this condition the expression of y' introduced in (7), one can then reformulate the previous conditions and obtain the following equation for each rear steering angle δ_R:

$$\delta_R \;=\; \arctan\left(\frac{k_y(y-y_d)+y'_d}{1-c(s)\,y}\right) - \tilde{\theta} - \hat{\beta}_R\,. \tag{9}$$

This expression allows to ensure the differential Eq. (8) on lateral deviation y, implying its convergence to y_d. Limits for this expression are similar as those defined for expression (2).

C. Control algorithm for the front steering angles

The objective of this second step is to find a mathematical expression for δ_F that ensures both the convergence of the lateral deviation of the robot y towards its desired value y_d and the convergence of the absolute orientation of the robot θ towards its desired value θ_d.

By denoting the error on the absolute angular deviation such that $e_\theta = \theta - \theta_d$, a way to ensure the convergence of this error to zero is to use a differential equation such that:

$$\dot{e}_\theta \;=\; \dot{\theta} - \dot{\theta}_d \;=\; k_\theta\, e_\theta\,, \tag{10}$$

with k_θ a negative scalar defining the settling time for the exponential convergence of θ to θ_d imposed by (10).

By injecting in this condition the expression of $\dot{\theta}$ introduced in (1), one can then reformulate the previous conditions and obtain the following equation for each front steering angle δ_F:

$$\delta_F \;=\; \arcsin\left(\frac{L\left(k_\theta(\theta-\theta_d)+\dot{\theta}_d\right)+v_R\sin\left(\delta_R+\hat{\beta}_R\right)}{v_F}\right) - \hat{\beta}_F\,. \tag{11}$$

After injecting in (11) the expression for δ_R (9) previously introduced, one can finally obtain the following equation for each front steering angle δ_F:

$$\delta_F \;=\; \arcsin\left(\frac{L\left(k_\theta(\theta-\theta_d)+\dot{\theta}_d\right)}{v_F} + \frac{v_R\sin\left(\arctan\left(\frac{k_y(y-y_d)+y'_d}{1-c(s)\,y}\right)-\tilde{\theta}\right)}{v_F}\right) - \hat{\beta}_F\,, \tag{12}$$

provided that $v_F \neq 0$ and limits similar as those defined for expression (9). In practice, the velocity is always positive when path tracking and consequently non null. This expression allows then to ensure the differential Eq. (8) on lateral deviation y, implying its convergence to y_d. In addition, this expression also ensures the convergence of the absolute orientation θ to its desired value θ_d.

Finally, by using expressions (9) and (12) for the rear and the front angles coming from the proposed backstepping strategy and by settling the desired lateral deviation and orientation, y_d and θ_d, the objective introduced by this paper (i.e. control the absolute orientation of the robot and its lateral position w.r.t. the trajectory) is achieved.

D. Predictive curvature servoing

To anticipate trajectory overshoots due to the actuator settling time, a predictive curvature servoing has been implemented. Only a few details about these control algorithms are given in this section. Nevertheless, it follows the same methodology as the one detailed in [3].

Knowing the evolution of the reference path (i.e. computed or previously learned), the reference path curvature can be anticipated. To achieve this, the future curvature of the reference path is computed w.r.t. a prediction time denoted t_{pred}. This future curvature is then injected in the steering laws to ensure the equality between the trajectory and vehicle curvature.

V. SIMULATION RESULTS

Making use of the proposed strategy based on the observer, detailed in Sec. III, and the control laws, detailed above,

algorithms have been tested on a MATLAB/ADAMS co-simulator with the simulated vehicle depicted in Fig. 2. This vehicle is four-wheel drive and has four independent steering wheels without steering range. Its features are described in Table I. These simulations are realistic w.r.t. technologies met in current industries. Indeed, the vehicle is equipped with a simulated IMU and a simulated RTK-GPS. The sampling rate for these trials has been set at $10\,Hz$ as in common real-time mobile robotic applications.

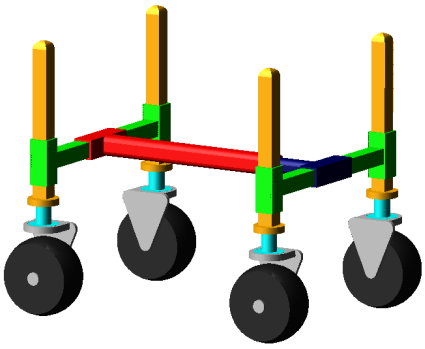


Fig. 2: 4WS mobile robot simulator.

TABLE I: Vehicle features.

Weight	$900\,kg$
Wheelbase (L)	$2\,m$
Trackbase	$1.2\,m$
Steering angle response time	$0.24s$

For the following simulations, the values of the gains have been chosen as follows: $k_y = -0.55$ and $k_\theta = -0.8$. The robot speed has been set at $2\,m/s$ according to the work speeds usually used in many agricultural contexts.

Successive robot positions (seen from above: ⊢⊣) have been added in the following tracking figures to illustrate the motion of the robot when tracking.

A. *Straight line tracking*

The objective of this trial is the verification of the proposed control strategy for parallel steering during straight line tracking when the robot is given different desired lateral deviations and absolute orientations. For this we have imposed to the vehicle the reference path depicted in black in Fig. 3 and Fig. 4.

As illustrated in Fig. 3, the vehicle is first located at an initial distance to the trajectory y_{init} equal to $0.5\,m$ (①). At the beginning ($t = 0\,s$), the robot is given a desired lateral deviation equal to zero ($y_d = 0\,m$) and a desired absolute orientation equal to zero ($\theta_d = 0$ degrees) (②). Then, at $t = 10\,s$, the lateral deviation is set at $0.3\,m$ (③) and then set, at $t = 15s$, at zero again ($y_d = 0\,m$) (④). At $t = 20\,s$, the robot is given a desired absolute orientation equal to 25 degrees ≈ 0.436 radians (⑤). Then, the desired lateral deviation is set at several values while keeping this absolute orientation. Then, at $t = 30\,s$, the desired lateral deviation is set at $0.4\,m$ (⑥), then, at $t = 40\,s$, the desired lateral deviation is set at $-0.2\,m$ (⑦) and finally, at $t = 45\,s$, the desired lateral deviation is set to zero ($y_d = 0\,m$) (⑧).

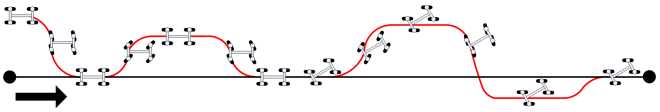


Fig. 3: Reference path (black), expected path (red) and successive robot positions (⊢⊣).

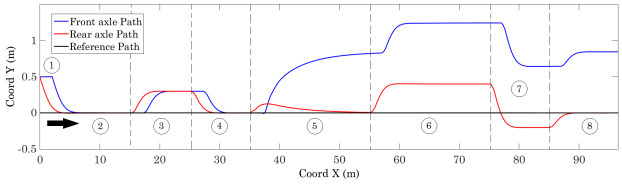


Fig. 4: Reference path (black) and tracking (blue and red).

The results for the lateral deviation y and the absolute orientation θ of the robot are depicted in Figs. 5 and 6.

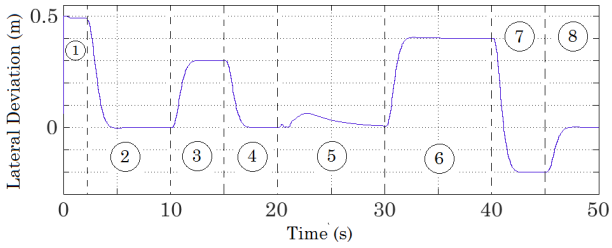


Fig. 5: Lateral deviation results.

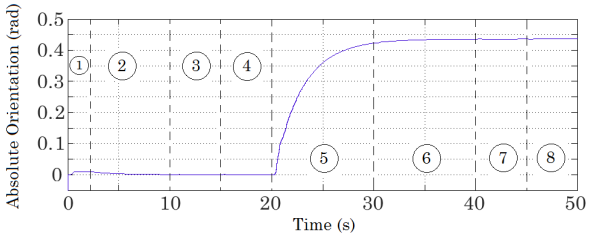


Fig. 6: Absolute orientation results.

It is apparent in Fig. 5 that the proposed parallel steering strategy ensures the convergence of the robot towards different desired lateral deviations during straight line tracking whatever the absolute orientation of the robot. Moreover, as depicted in Fig. 6 the proposed strategy ensures the convergence of the absolute orientation of the robot to different desired orientations while ensuring the convergence of the lateral deviation.

B. Curve tracking

The objective of this trial is the verification of the proposed control strategy for parallel steering during curve tracking when the robot is given different desired lateral deviations and absolute orientations. For this we have imposed to the vehicle the trajectory depicted in black in Fig. 7.

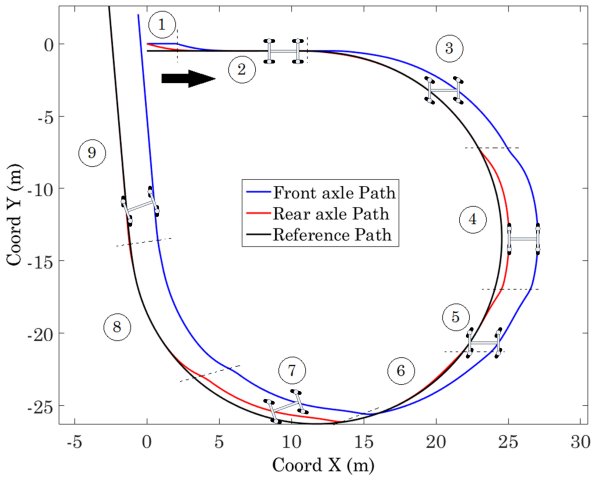


Fig. 7: Reference path (black), tracking (blue and red) and successive robot positions ( ).

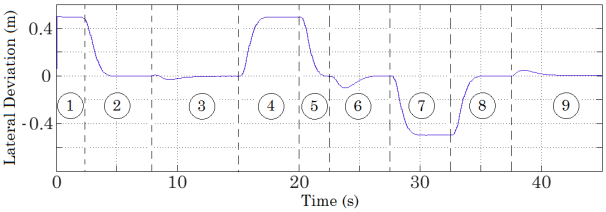


Fig. 8: Lateral deviation results.

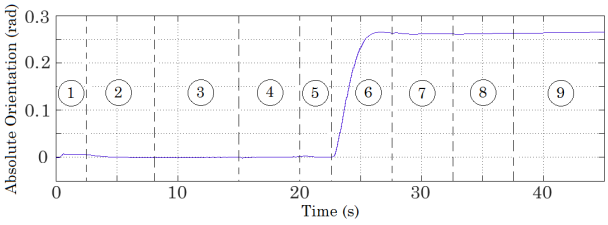


Fig. 9: Absolute orientation results.

As illustrated in Fig. 7, the vehicle is first located at an initial distance to the trajectory y_{init} equal to $0.5\,m$ (①). At the beginning ($t = 0\,s$), the robot is given a desired lateral deviation equal to zero ($y_d = 0\,m$) and a desired absolute orientation equal to zero ($\theta_d = 0$ degrees) (②). After a short straight line, the robot is asked to follow a curved trajectory (③). Then, while tracking this curve, the desired lateral deviation is firstly set to $0.5\,m$ at $t = 15\,s$ (④), then to $0\,m$ at $t = 20\,s$ (⑤). At $t = 22.5\,s$ the desired absolute orientation is set to 15 degrees ≈ 0.262 radians (⑥). Then, the desired lateral deviation is set to $-0.5\,m$ at $t = 27.5\,s$ (⑦) and finally to zero at $t = 32.5\,s$ (⑧). When the curve stops, the trajectory becomes again a straight line (⑨).

The results for the lateral deviation y and the absolute orientation θ of the robot are depicted in Figs. 8 and 9.

It is apparent in Fig. 8 that the proposed parallel steering strategy ensures the convergence of the robot towards different desired lateral deviations during curve tracking whatever the absolute orientation of the robot. Moreover, as depicted in Fig. 9 the proposed strategy ensures the convergence of the absolute orientation of the robot to different desired orientations while ensuring the convergence of the lateral deviation.

C. Manoeuvres applied to agricultural robotics

From previous simulations, one can see that the proposed parallel steering strategy is suitable for achieving accurate path tracking in off-road conditions. It is then possible to imagine several scenarios with this kind of control in different agricultural contexts where manoeuvres are tricky because of the lack of space. It is the case for instance when facing narrow parking places or doing U-turns between vine rows. We have chosen to illustrate these situations with the relevant simulations depicted in Figs. 10 and 11 that show the achievements and the accuracy of using the proposed parallel steering strategy in such situations.

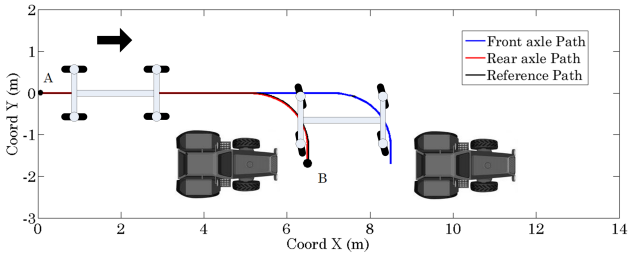


Fig. 10: Reference path (black), tracking (blue and red) and successive robot positions ( ) when parking.

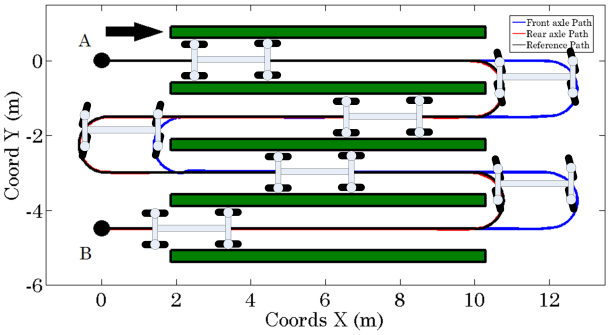


Fig. 11: Reference path (black), tracking (blue and red) and successive robot positions ( ) between vine rows (green).

VI. CONCLUSION

This paper proposes a parallel steering strategy dedicated to a four-wheel steering mobile robot in off-road contexts. Because of poor grip conditions and imposing an equal angle to the front and the rear axles, previous parallel steering approaches cannot ensure an accurate tracking because of the skid and the deviation from the desired path. Here, the grip conditions are considered by using an observer based on the extended kinematic model of the robot. This observer, derived from previous work, allows suitable estimations for the sideslip angles of a four-wheel steering mobile robot. These estimations then feed the proposed control laws based on a backstepping strategy ensuring both the convergence of the robot towards a desired lateral deviation and a desired absolute orientation. This new point of view allows to avoid possible skids while guaranteeing the accuracy of the tracking. As shown in the simulation section, the combination of the observer with the proposed control laws leads to high accurate path tracking and can be used for tricky manoeuvres in agricultural contexts for instance. Future work will be focused on the design and the implementation of an observer dedicated to each wheel of the robot to estimate more precisely sideslip angles to further increase the accuracy of the path tracking.

ACKNOWLEDGMENT

This work has received the support of the French National Research Agency under the grant number ANR-14-CE27-0004 attributed to AdAP2E project and has been sponsored by the IMobS3 Laboratory of Excellence under the grant number ANR-10-LABX-16-01.

REFERENCES

[1] G. Bayar, M. Bergerman and A. B Koku (2016). Improving the trajectory tracking performance of autonomous orchard vehicles using wheel slip compensation. Biosystems Engineering, vol. 146, pp. 149-164.

[2] M. Bergerman, J. Billingsley, J. Reid and E. van Hentenl (2016). Robotics in Agriculture and Forestry. In : Springer Handbook of Robotics. Springer International Publishing, p. 1463-1492.

[3] C. Cariou, R. Lenain, B. Thuilot and M. Berducat (2009). Automatic guidance of a four-wheel steering mobile robot for accurate field operations. Journal of Field Robotics, vol. 26, no 6-7, pp. 504-518.

[4] R. Fierro and F L. Lewis (1995). Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In : Decision and Control, Proceedings of the 34th IEEE Conference on. IEEE, p. 3805-3810.

[5] W. Grey Walter (1950). An imitation of life. Scientific American, vol. 182, no 5, pp. 42-45.

[6] L. Grimstad, C. Pham, H. Phan and P. From (2015). On the design of a low-cost, light-weight, and highly versatile agricultural robot. IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO), pp. 1-6.

[7] L. Haibo, D. Shuliang, L. Zunmin and Y. Chuijie (2015). Study and experiment on a wheat precision seeding robot. Journal of Robotics, pp. 12.

[8] K. Kozlowski and D. Pazderski (2004). Modeling and control of a 4-wheel skid-steering mobile robot. Int. J. Appl. Math. Comput. Sci, vol. 14, no 4, p. 477-496.

[9] R. Lenain, B. Thuilot, C. Cariou and P. Martinet (2006). High accuracy path tracking for vehicles in presence of sliding: Application to farm vehicle automatic guidance for agricultural tasks. Autonomous Robots, vol. 21, no. 1, pp. 79-97.

[10] R. Lenain, M. Deremetz, J-B. Braconnier, B. Thuilot and V. Rousseau (2017). Robust sideslip angles observer for accurate off-road path tracking control. Advanced Robotics, pp. 1-15.

[11] F. Marmoiton and M. Slade (2016). Toward Smart Autonomous Cars. Intelligent Transportation Systems: From Good Practices to Standards, pp. 84-109.

[12] P. Moubarak and P. Ben-Tzvi (2012). Modular and reconfigurable mobile robotics. Robotics and Autonomous Systems, vol. 60, no 12, p. 1648-1663.

[13] C. Samson (1995). Control of chained systems. Application to path following and time-varying point stabilization of mobile robots. IEEE Transactions on Automatic Control, vol. 40, no. 1, pp. 64-77.

[14] C. Samson, P. Morin and R. Lenain (2016). Modeling and control of wheeled mobile robots. In : Springer Handbook of Robotics. Springer International Publishing, p. 1235-1266.

[15] P. Sapaty (2015). Military robotics: Latest trends and spatial grasp solutions. International Journal of Advanced Research in Artificial Intelligence, vol. 4, no 4, p. 9-18.

[16] B. Thuilot, C. Cariou, P. Martinet and M. Berducat (2002). Automatic guidance of a farm tractor relying on a single CP-DGPS. Autonomous robots, vol. 13, no 1, pp. 53-71.

[17] D. Wang and F. Qi (2001). Trajectory planning for a four-wheel-steering vehicle. IEEE International Conference on Robotics and Automation (ICRA), vol 4, pp. 3320-3325.

Towards Autonomous Landing on a Moving Vessel through Fiducial Markers

Riccardo Polvara[1] Sanjay Sharma[1] Jian Wan[1] Andrew Manning[1] Robert Sutton[1]

***Abstract*— This paper propose an autonomous landing method for unmanned aerial vehicles (UAVs), aiming to address those situations in which the landing pad is the deck of a ship. Fiducial marker are used to obtain the six-degrees of freedom (DOF) relative-pose of the UAV to the landing pad. In order to compensate interruptions of the video stream, an extended Kalman filter (EKF) is used to estimate the ship's current position with reference to its last known one, just using the odometry and the inertial data. Due to the difficulty of testing the proposed algorithm in the real world, synthetic simulations have been performed on a robotic test-bed comprising the AR Drone 2.0 and the Husky A200. The results show the EKF performs well enough in providing accurate information to direct the UAV in proximity of the other vehicle such that the marker becomes visible again. Due to the use of inertial measurements only in the data fusion process, this solution can be adopted in indoor navigation scenarios, when a global positioning system is not available.**

I. INTRODUCTION

In the last few years, unmanned aerial vehicles (UAVs) attracted a lot of interest from the research and the military community, in particular those able of vertical take-off and landing (VTOL) [1].

The capability of autonomously landing, especially on the deck of a ship, is still an open research area. Given the marine conditions, characterised by adverse wind and sea currents, the estimation of the ship movements can be affected in such a way that landing is not always possible. Cameras have been identified as a solution in order to increase the estimation's accuracy; this can be further improved by the adoption of fiducial marker on the ship's deck (as depicted in Fig. 1). In the situation in which the marker tracking is interrupted, the 6-DOF ship's pose can be calculated with a state estimation filter. The choice of using an extended Kalman filter (EKF) to fuse only the odometry and inertial measurement unit (IMU) data of the moving vessel allows to successfully accomplish the task without relying on the global positioning system (GPS) signal. In this way, the algorithm can be used also in cluttered or GPS-denied environments.

In terms of the paper organisation, Section II presents the existing literature about autonomous landing for UAVs, while Section III formalise the solution proposed in this paper. In Section IV two kind of experiments, with a static landing platform and with a moving one, respectively, are presented and discussed. Finally, conclusions and future works are given in Section V.

[1]Riccardo Polvara and the other authors are with the Autonomous Marine System Research Group, School of Engineering, Plymouth University, Drake Circus, Plymouth PL4 8AA, United Kingdom riccardo.polvara@plymouth.ac.uk

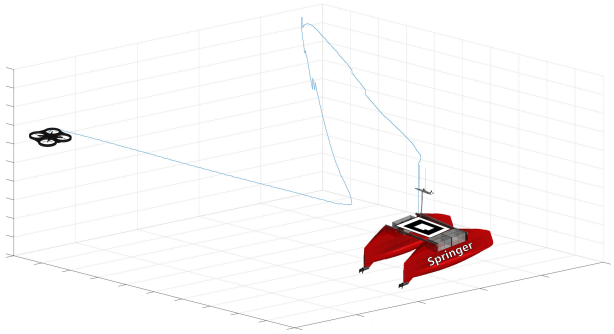


Fig. 1: An UAV landing on a fiducial marker located on the deck of an unmanned surface vehicle.

II. STATE OF THE ART

Autonomous landing is one of the most dangerous challenges for UAVs. Despite a historical large use of Inertial Navigation Systems (INS) and Global Navigation Satellite System (GNSS), vision-based solutions are becoming attractive because passive and not requiring any special equipment other than a camera and a processing unit [2], [3]. In [4] and [5] a IR-LED helipad is adopted for robust tracking and landing, while a T-shaped and an H-shaped helipad are respectively used in [6] and [7]. Here, the UAV's pose is calculated combining the projection of the pad with IMU measurements. In [8], the manoeuvre has been achieved after estimating the UAV's 3D relative position to a novel landing pad consisting of concentric circles, and assuming the vehicle is always parallel to the ground. Multiple circles of different sizes are also used in [9] with the scope of extending the detection range. In [12], multiple view geometry is used to hover after computing the relative position to a known target. In [13] authors, following the same approach, were also able to adjust the UAV's orientation. Other some bio-inspired works try to maintain a constant optic flow while descending [10], [11].

Fewer works address autonomous landing on a moving target. A first example is offered by [14], in which the authors landed on a slowly moving H-shaped marker (0.1 m/s). In [15] four light sources constitute a pattern identifiable up to 0.25m; unfortunately, this solution is not feasible when direct light interferes with the vision system. Differently, optical flow is used in [16] while an IR camera in [17].

III. METHODS

This section illustrates how autonomous landing has been achieved in this work.

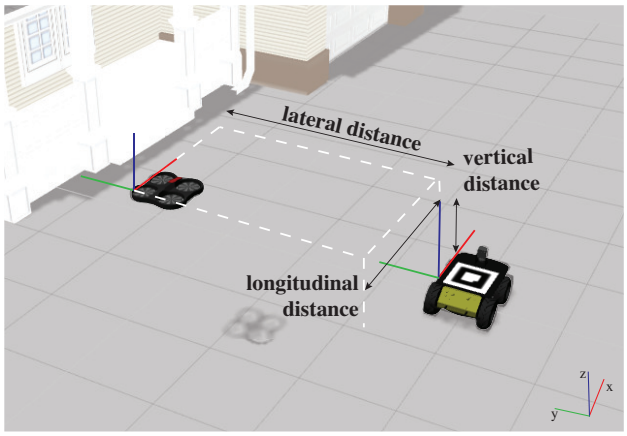


Fig. 2: The image processing algorithm estimates the 6-DOF UAV's relative-pose wrt the visual marker. Here, only the distances are reported for clarity.

A. Quad-copter model

To the scope of this paper, the Parrot AR Drone 2.0, widely used in research because affordable and integrating a complete sensor suite, has been chosen as testing platform. The Robot Operating System (ROS) [18] is used for interfacing the vehicle, using in particular the *ardrone-autonomy* and the the *tum-ardrone* packages [19]. The specification of the UAV are as follow:

- Dimensions: 53 cm x 52 cm (hull included);
- Weight: 420 g;
- IMU including gyroscope, accelerometer, magnetometer, altimeter and pressure sensor and producing data at 200 Hz;
- Front-camera with a High-definition (HD) resolution (1280x720), a field of view (FOV) of 73.5 x 58.5 and video streamed at 30 frame per second (fps);
- Bottom-camera with a Quarted Video Graphics Array (QVGA) resolution (320x240), a FOV of 47.5 x 36.5 and video streamed at 60 fps (mainly used for visual odometry);
- Central processing unit running an embedded version of Linux operating system;

The on-board controller (closed-source) is used to act on the roll Φ and pitch Θ, the yaw rotational speed Ψ and the vertical velocity of the platform $\dot{z}$. Control commands u = (Φ, Θ,Ψ, $\dot{z}$) $\in$ [-1,1] are sent to the quad-copter at a frequency of 100Hz.

B. Augmented Reality

In this work, high-contrast 2D augmented reality (AR) markers are adopted for identifying the landing platform. The identification and tracking have been realised with the *ar_pose* ROS package, a wrapper for the $ARToolkit$ human-computer interaction library[20].

The package subscribes to the camera's topic and the candidate marker is searched for within a database. Using the camera's calibration file and the actual size of the marker of interest, the 6-DOF relative-pose of the marker's with respect to the UAV is estimated at a frequency of 1 Hz and shown in Fig. 2. The time stamp and the transformation for the current and the last marker's observation are stored to actuate a compensatory behaviour when the marker is lost.

C. Controller

Within the $tum_ardrone$ package, a single PID controller is employed for three degrees of freedom (roll, pitch and yaw) and for the vertical velocity. Combined, they steer the quad-copter toward a desired goal pose $p = (\hat{x}, \hat{y}, \hat{z}, \hat{\psi}) \in \mathbb{R}^4$ in a global coordinate system sending commands at 100Hz.

In order to simplify the tuning process, the four PID controllers have been replaced by a single damped spring one. In the implementation used, two parameters, K_direct and K_rp, are responsible for modifying the spring strength of the directly controlled dimensions (yaw and z) and the leaning ones (x and y). An additional parameter, $xy_damping_factor$, takes account of external disturbances such as air resistance and the wind.

In this way, instead of controlling nine independent parameters (three for the yaw, three for the vertical speed and three for the pair roll-pitch) the control problem is reduced only to the three described above.

D. Pose estimation

An EKF has been adopted for estimating the position of the landing platform and offering more reliability when the marker is lost. For this purpose, data from the wheels and the IMU are fused in order to compensate the error affecting the odometry readings [21]. The filter predicts the actual position of the ship that is then forwarded to the controller. The EKF's estimate is used as follow:

- the platform's pose is estimated at 50Hz and saved in a hash table;
- when the tracking is interrupted, the table is accessed for retrieving the most recent estimation together with the last recorded observation;
- the deck's current relative-pose to the old known one is calculated;
- this information is forwarded to the controller and new commands are generated;

This procedure is iterated until the UAV is located above the marker and can newly perceive it.

E. Discussion

The quad-copter flies autonomously mainly using its fixed frontal camera, approaching the landing platform identified by a fiducial marker. Despite the EKF can compensate interruption in observation, it is required that the marker is perceived among all the landing manoeuvre. The ar_pose library computes the 6-DOF relative-pose between the UAV and the landing platform. This information is used by the controller to make the quad-copter approaching the marker with the right orientation.

Due to the hardware limitation of the UAV chosen, in particular the presence of two fixed non-tilting cameras, a

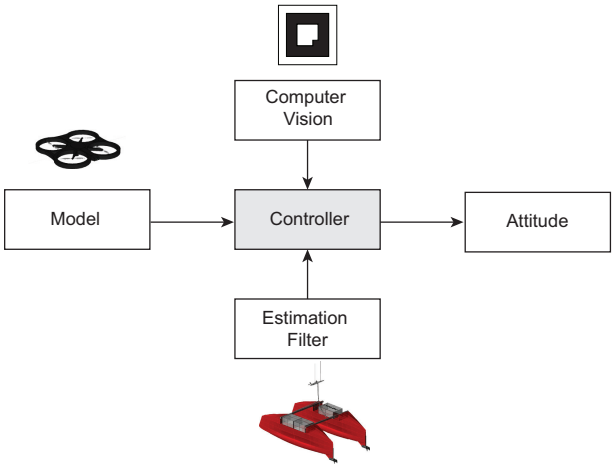


Fig. 3: Different components are integrated for achieving autonomous landing.

switching system has been adopted to guarantee maximum continuity while observing and tracking the landing pad. In fact, it happens that the UAV lose the tracking while descending and using the frontal camera. To overcome this problem, the video stream starts to be acquired from the downward-looking camera located at the bottom of the UAV. The quad-rotor can then try to land while centring the marker in the camera's FOV. Otherwise, the EKF estimates the actual position of the landing base and the drone is there redirected increasing its altitude. In this way, it is guaranteed to perceive the marker soon. When an user-specified distance from the marker is met, the drone shuts down its motors and land on it. A graphical representation of the overall system is depicted in Fig. 3.

It is important to remark that visual marker permits the estimation of the full 6-DOF relative-pose, meaning that pitch and roll of the deck are considered. Nevertheless, the algorithm does not directly take care of these movements since they can be easily addressed mounting a pan-tilt unit under the landing platform in order to stabilise it whatever weather conditions. For this reason, it is possible to claim that the method developed is theoretically applicable to every landing scenario, especially when involving UGVs not subject to significant rolling and pitching behaviours.

IV. Results

The proposed algorithm has been tested in simulation within a modified version of the $tum_simulator$ package, a 3D environment built on Gazebo 2.2.X and offering a 3D model of the AR Drone 2.0. Given the absence of maritime robot models and marine scenario, the Husky A200 has been chosen as landing base. Here, a square fiducial marker having a side length of 0.31 meter was placed. Due to the paper's length limitation, three experiments are now reported. In the first, the Husky remains in the same position for all the length of the flight. In the second experiment, the platform is moving in a straight line at constant speed. This is a common scenario when deploying manned/unmanned vessel

TABLE I: Controller parameters for the static landing platform experiment.

Parameter	Value	Parameter	Value
K_direct	5.0	K_rp	0.3
xy_damping_factor	0.65	max_gaz_drop (m/s)	0.1
max_yaw (rad/s)	1.0	–	–

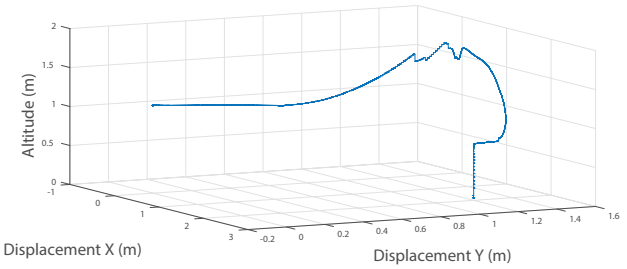


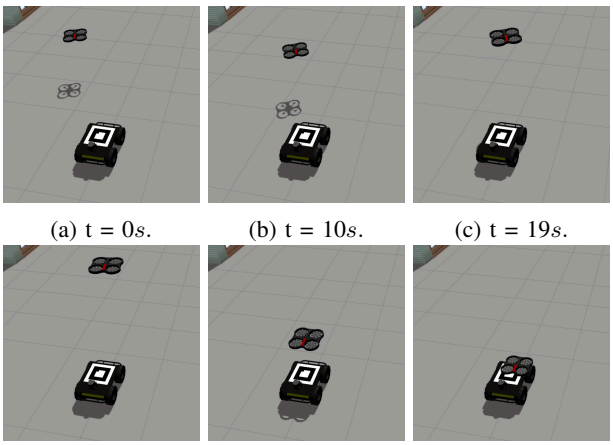
Fig. 4: Quad-copter trajectory in three-dimensional space during the experiment with a static platform.

that spends most of their time traversing keeping a fixed heading angle. As last, the UGV is also rotating.

A. Static Platform

The aim of this experiment is to test the alignment of the UAV with a visual marker. In Table I the controller setting is reported. The K_rp parameter, responsible for roll and pitch, is set to a small value to guarantee smooth movements while translating. In the same way, max_gaz_drop is reduced to a value of 0.1 for slowing down the descending manoeuvre. On the other hand, the max_yaw parameter, controlling the yaw rotational speed, is maximised because the alignment must be realised in the shortest amount of time possible.

Fig. 4 illustrates the UAV's trajectory for this experiment. The fiducial marker has been successfully recognised at around 2 meters in front of the UAV, and at 1.2 meter on its left. The displacement on the $z-$axis, used as the reference



(a) t = $0s$. (b) t = $10s$. (c) t = $19s$.

(d) t = $20s$. (e) t = $54s$. (f) t = $57s$.

Fig. 5: Landing maneuvre of a VTOL UAV on a static platform.

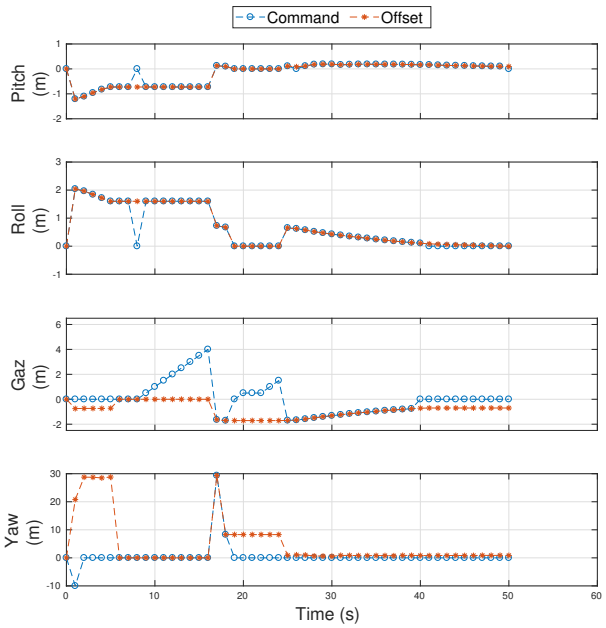


Fig. 6: Controller commands and visual offsets in the experiment with static landing platform.

for the altitude, was around 0.7 meter instead. The initial situation is illustrated in Fig. 5a. The UAV has been able to complete the landing in 50 seconds.

During the flight, the quad-copter approaches the pad keeping the marker at the centre of its camera's FOV. An interval of confidence of 10 degrees is defined and the UAV rotates accordingly when the marker is out of it. While descending, the UAV's low altitude prevents the frontal camera from perceiving the marker. This is basically what happens at $t = 10s$ and depicted in Fig. 5b. At this point, the video stream is switched to the downward-looking camera. The UAV is instructed to move towards the marker's last known position while increasing its altitude for increasing the area covered by the camera. When the UAV is located exactly above the marker (more precisely at $t = 19s$ as in Fig. 5c), it can complete the landing finding first the right orientation ($t = 20s$) and then descending keeping the marker centred, as shown in Fig. 5d. At $t = 54s$ the quad-copter reaches a user-defined altitude (set to 0.75m): it can now shut down its motors and land on the platform (Fig. 5f).

In Fig. 6 the controller commands are plotted against the marker's observation. For most of the time, the two curves overlap, meaning the marker's observations are directly forwarded to the controller. The only portion in which they do not is between $t = 8s$ and $t = 25s$, when the marker is lost twice. Therefore, the compensatory behaviour is adopted, sending the UAV to the marker's last known position while keeping its orientation constant. In this way, the pitch, roll and yaw commands do not change meanwhile the UAV's altitude increases. A dedicated analysis is reserved for the

TABLE II: Controller parameters for the moving landing platform experiment.

Parameter	Value	Parameter	Value
K_direct	5.0	K_rp	1.0
xy_damping_factor	0.65	max_gaz_drop (m/s)	-0.15
max_yaw (rad/s)	1.0	–	–

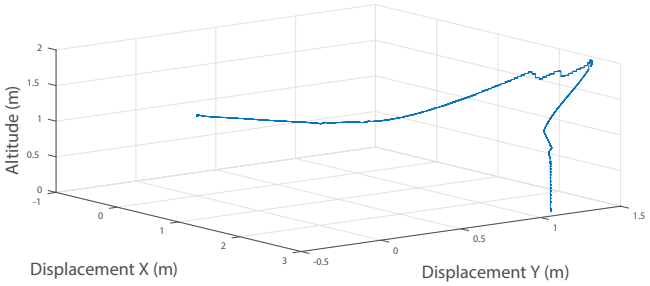


Fig. 7: Quad-copter trajectory in three-dimensional space and top view during the experiment with a moving platform proceeding in straight line.

altitude in the interval $t = [1, 5]s$ and $t = [40, 50]s$, and the yaw in $t = [1, 5]s$ and after $t = 18s$. Here, the offsets are below a user-defined threshold and a respective command equal to 0 is sent instead. The threshold is introduced because it was noticed that fixed-parameters controllers, like the PID or the one used, offer limited performances in a scenario like the one studied, in which high accuracy is required. An adaptive solution is planned as future work.

B. Moving Platform

In this subsection, the UAV is tested against landing on a moving platform. Due to the small size of the deck and the limitations posed by fixed non-tilting cameras, a stopping command is sent to the UGV once the quad-copter is located exactly above it. This decision was also forced by the trade-off posed by the dimension of the visual marker: a smaller one could be perceived from shortest distances because occupying a smaller portion in the camera's FOV; on the other hand, it would be more difficult to perceive it from far.

The controller setting is reported in Table II . Differently, from the previous experiment, the K_rp parameter assumes now a bigger value to make the UAV to translate faster. To allow the UAV descending at high speed while maintaining the tracking, the value of the max_gax_drop parameter is decreased. Lastly, max_yaw is kept at its maximum value to minimise the alignment time with the landing base. Regarding the UGV, a constant velocity command is sent to make it move in a straight line.

The flight trajectory is reported in Fig. 7. The quad-copter performed an autonomous landing in around 40 seconds.

The initial scenario is reported in Fig. 9a. At $t = 16s$, as seen in Fig. 9b, the UAV reaches an altitude such that it is impossible to continue to perceive the marker. The video stream is therefore acquired from the bottom camera and the UGV's estimated position is sent to the controller while

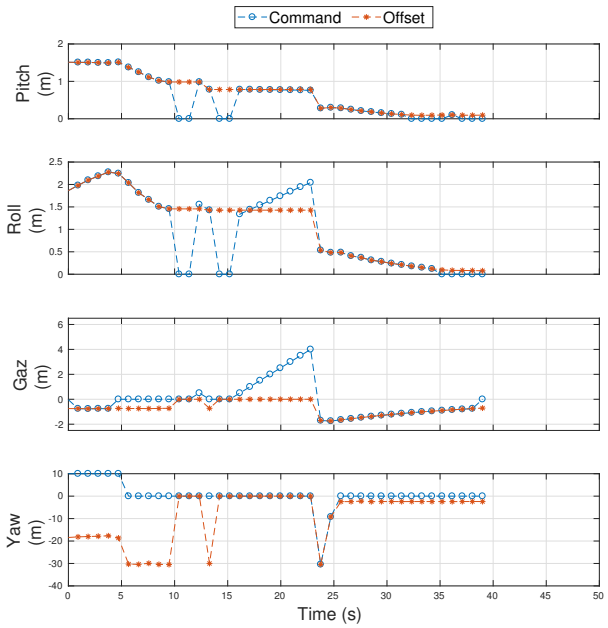


Fig. 8: Controller commands and visual offsets in the experiment with a mobile platform moving in straight line.

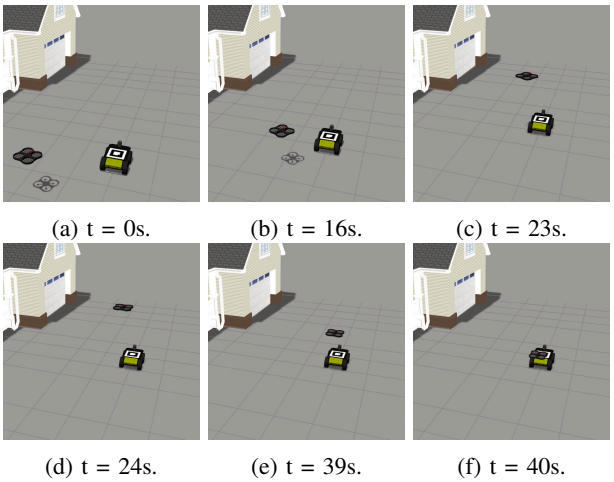


Fig. 9: Landing manoeuvre of a VTOL UAV on a mobile platform moving in straight line.

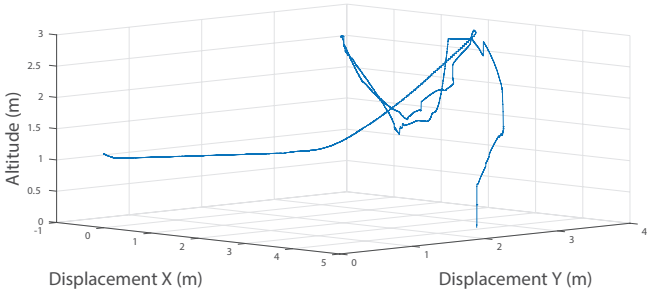


Fig. 10: Quad-copter trajectory in three-dimensional space and top view during the experiment with a moving platform that also rotate.

increasing the drone's altitude. Doing so, at $t = 23s$ the UAV is located exactly above the UGV and it rotates accordingly until the proper orientation is reached (Fig. 9d). Because the landing base is at the centre of the camera's FOV, a null velocity command is sent to stop the UGV. Fig. 9e and 9f show the UAV descending slowly on the marker and landing on it.

Further analysis can be done with the results reported in Fig. 8. As in the first experiment, the curve of the controller's commands and the offsets one overlap for most of the time. The difference with the previous case is given by the EKF's contribution. It is possible to have an example looking at the plot in $t = [10, 23]s$. Here, the marker has been lost twice and the curves differ: if, on one hand, the offsets are constant because no new observations have been done, on the other hand, the commands slightly change. In $t = [16, 23]s$, while the yaw commands assume a constant value of 0 because the UAV is properly oriented, the pitch does not change because the UGV is moving only in a straight line, not deviating from the lateral direction. For the same reason, the roll command is updated to account every instant of the new relative-pose (changing the longitudinal direction) of the UGV. In the last part of the graph, the difference between the two curves for the roll (in $t = [35, 40]s$), the altitude (in $t = [5, 10]s$) and the yaw (from $t = 25s$ till the end) is justified by the adoption of a threshold to speed up the completion of the landing manoeuvre. To conclude, the two yaw's curves deserve a deeper analysis. As already discussed, while using the frontal camera, the UAV keeps the marker at the centre of the FOV. If the UAV rotated to align, it would most probably lose the tracking. To avoid this situation, the rotation is performed only ten degrees per time and only if the marker is located at the edge of the camera's frame. Doing so, the alignment is slowed down but a continuous tracking is guaranteed.

A last experiment has been done with a moving platform that proceeds not only in straight line but rotates changing its heading angle. Note that the UGV is able only to move in straight line or rotate on the spot at the same time. The goal of this simulation is to show how the EKF can compensate the lack of the vision technique in predicting where the UAV has to move to perceive the visual marker again. The controller parameters used are the same of the previous experiment and they are shown in Table II. Results are reported in Fig. 10, where the UAV's trajectory is drawn. In Fig. 11 is shown the comparison between the offsets obtained through the vision algorithm and the commands sent to the controller. Here, it is possible to see that, as in the previous scenarios, the plot of the offsets and the one related to the commands overlap for most of the time. All the analysis made before still hold, but it is interesting to notice how the algorithm is able to react properly when the visual marker is lost, redirecting the UAV above it despite changes in position.

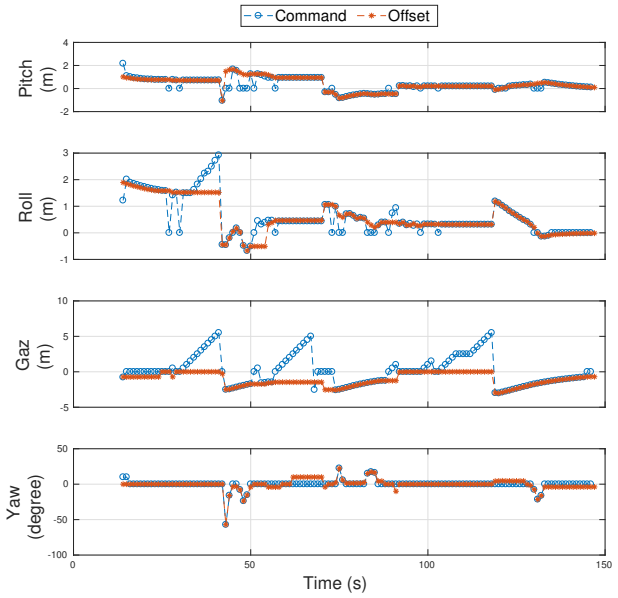


Fig. 11: Controller commands and visual offsets in the experiment with a mobile platform that also rotate.

V. Conclusion and Future Directions

In this paper, autonomous landing on a ship's deck is studied and tested in a 3D simulator. Due to unavailability of marine robot and scenarios model, simulations have been performed to a ground unmanned vehicle. The solution proposed resides only on the UAV's on-board sensors and on the adoption of a visual marker on the landing platform to easily calculate its 6-DOF pose. In this way, despite not tested yet, the algorithm is supposed to be robust also when adverse marine conditions affect the rolling and pitching dimension of the vehicle on which the UAV must land on. The adoption of EKF allows overcoming issues with fixed non-tilting cameras. Not involving GPS signals makes this solution feasible also in indoor scenarios or adverse weather conditions.

Three experiments, against a static and a moving base, were performed to validate the approach. In all the cases, successful results have been obtained. An adaptive controller, based on an intelligent solution such as artificial neural networks or fuzzy logic, is identified as future research to compensate the limits of a fixed-parameters controller such as the one used in this work.

References

[1] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[2] W. Kong, D. Zhou, D. Zhang, and J. Zhang, "Vision-based autonomous landing system for unmanned aerial vehicle: A survey," in *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*. IEEE, 2014, pp. 1–8.

[3] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[4] A. Masselli, S. Yang, K. E. Wenzel, and A. Zell, "A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 685–693.

[5] K. E. Wenzel, P. Rosset, and A. Zell, "Low-cost visual tracking of a landing place and hovering flight control with a microcontroller," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*. Springer, 2009, pp. 297–311.

[6] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi, "A vision-based guidance system for uav navigation and safe landing using natural landmarks," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*. Springer, 2009, pp. 233–257.

[7] S. Yang, S. A. Scherer, and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2013.

[8] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[9] T. Merz, S. Duranti, and G. Conte, *Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 343–352. [Online]. Available: http://dx.doi.org/10.1007/11552246_33

[10] B. Barber, T. McLain, and B. Edwards, "Vision-based landing of fixed-wing miniature air vehicles," *Journal of Aerospace computing, Information, and Communication*, vol. 6, no. 3, pp. 207–226, 2009.

[11] F. Ruffier and N. Franceschini, "Optic flow regulation in unsteady environments: a tethered mav achieves terrain following and targeted landing over a moving platform," *Journal of Intelligent & Robotic Systems*, vol. 79, no. 2, pp. 275–293, 2015.

[12] O. Shakernia, R. Vidal, C. S. Sharp, Y. Ma, and S. Sastry, "Multiple view motion estimation and control for landing an unmanned aerial vehicle," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 2793–2798.

[13] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, vol. 3. IEEE, 2002, pp. 2799–2804.

[14] ——, "Visually guided landing of an unmanned aerial vehicle," *IEEE transactions on robotics and automation*, vol. 19, no. 3, pp. 371–380, 2003.

[15] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of intelligent & robotic systems*, vol. 61, no. 1, pp. 221–238, 2011.

[16] B. Herissé, T. Hamel, R. Mahony, and F.-X. Russotto, "Landing a vtol unmanned aerial vehicle on a moving platform using optical flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, 2012.

[17] O. A. Yakimenko, I. I. Kaminer, W. J. Lentz, and P. Ghyzel, "Unmanned aircraft navigation for shipboard landing using infrared vision," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 4, pp. 1181–1200, 2002.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[19] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems (RAS)*, vol. 62, no. 11, pp. 1646–1656, 2014.

[20] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. IEEE, 1999, pp. 85–94.

[21] L. Jetto, S. Longhi, and G. Venturini, "Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219–229, 1999.

A Probabilistic Framework for Global Localization with Segmented Planes

Jan Wietrzykowski and Piotr Skrzypczyński

***Abstract*— This paper proposes a novel approach to global localization using high-level features. The new probabilistic framework enables to incorporate uncertain localization cues into a probability distribution that describes the likelihood of the current robot pose. We use multiple triplets of planes segmented from RGB-D data to generate this probability distribution and to find the robot pose with respect to a global map of planar segments. The algorithm can be used for global localization with a known map or for closing loops with RGB-D data. The approach is validated in experiments using the publicly available NYUv2 RGB-D dataset and our new dataset prepared for testing localization on plane-rich scenes.**

I. INTRODUCTION

Solutions to the Simultaneous Localization and Mapping (SLAM) problem in 3-D [1], [2], [3] usually assume incremental localization, relying on the robot/sensor pose prior for matching the current perception to the map. However, if a reliable prior is unavailable, the robot has to find the pose with respect to the already learned or *a priori* known map by means of a global localization method.

Consider exemplary indoor scene views depicted in Fig. 1a. Are those the same scenes observed from different viewpoints or are they just similar? A human can tell that easily (Fig. 1b) using the semantic context, but a robot has to rely on numerical computations on the basis of some scene representation. The abstraction level of scene representation is perhaps the most important problem in developing robust global localization methods. Point features are most common in 3-D SLAM [1], [3], but if we exploit higher-level features [4], the local environment geometry can resolve ambiguities stemming from an abundance of repetitive or similar visual patterns [5]. As recovering the geometry from passive vision data requires intensive computations we focus on active RGB-D sensors, which are cheap, compact, and provide a rich description of the scene.

Therefore, we propose a novel approach to the problem of global localization using higher-level features. Unlike many 3-D SLAM solutions that focus on accurate mapping of small areas, our localization system[1] works on larger indoor scenes, with loopy sensor trajectories of extended duration. We contribute: (i) a new probabilistic localization framework utilizing Gaussian kernel approximation; (ii) a localization solution using this framework and segmented planes as features; (iii) RGB-D dataset suitable for evaluation of the proposed method.

*This work was funded by the PUT DSMK/0172 and DSPB/0162 grants

Jan Wietrzykowski and Piotr Skrzypczyński are with the Institute of Control and Information Engineering, Poznań University of Technology, 60-965 Poznań, Poland {jan.wietrzykowski; piotr.skrzypczynski}@put.poznan.pl

[1]Available at https://github.com/LRMPUT/PlaneLoc

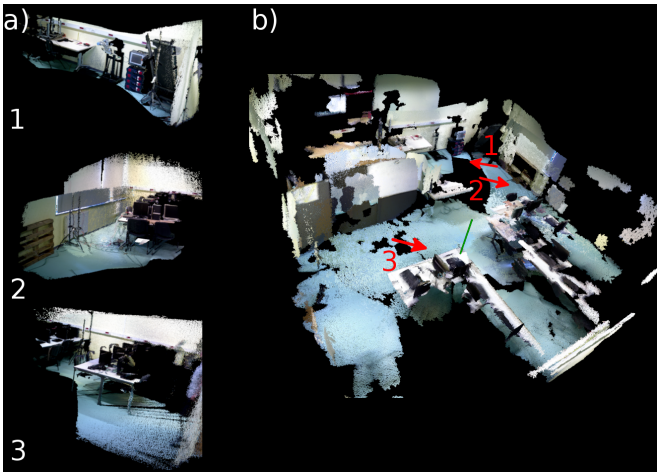


Fig. 1. Example of place recognition in a 3-D environment: perceived local views (a), and the global map view (b)

II. RELATED WORK

In keyframe-based 3-D SLAM systems, large loop closures are detected using appearance-based place recognition techniques [6]. One of the most widely used algorithms from this group is FAB-MAP [7]. In ORB-SLAM/ORB-SLAM2 [3] such a technique, based on the fast to compute ORB features is used also for relocalization [8]. Although Williams *et al.* [9] demonstrated that appearance-based methods scale better than map-based SLAM algorithms for large environments, these methods do not provide accurate estimates of the robot pose with respect to the map. SLAM systems that do explicit map reconstruction usually need to have a reasonable guess of the sensor pose before they attempt to match the local perception to the map [1]. The recent ElasticFusion system [10] that maintains a dense environment model applies an appearance-based approach for location recognition, but the database of locations contains predicted views of the dense map. If the global map is feature-based, synthesizing frame views becomes infeasible. Among the feature-based approaches, Heredia *et al.* [11] proposed a two-stage point-feature matching algorithm that facilitates global localization. Higher-level geometric features, that provide more localization constraints than points have been employed in a number of systems. An early attempt to use plane features was the 3-D EKF-SLAM by Weingarten and Siegwart [12]. More recently, an optimization-based approach to SLAM exploiting both, point and plane features was presented [13]. Optimization-based approaches to SLAM with infinite planes

as features are described in [14] and [4], whereas [2] explores plane segments in dense visual SLAM. Those approaches tackle, however, the incremental SLAM problem. Pathak *et al.* [15] proposed a fast method for registration of noisy planes. Although the Minimally Uncertain Maximal Consensus algorithm was demonstrated in [15] assuming limited translations and rotations between consecutive views, this method can solve unknown correspondences between planes, hence it has a potential for application in global localization. Similarly, Cupec *et al.* demonstrated in [16] that their earlier planar surface segments registration algorithm can be used for global localization employing a multi-hypothesis EKF to handle correspondence outliers. The solution of [17] is similar in spirit to our approach, but it addresses the place recognition problem by matching subgraphs representing the local topology of neighboring planar patches in a plane-based global map. Hence, although it uses a geometric rather than appearance-based approach, it is unable to produce an accurate estimate of the global robot pose.

III. PROBLEM STATEMENT AND SOLUTION

Consider two scenes visible in Fig. 2a, where the same place in a global map was shown from two different views using planes. These planes can be matched in a number of ways (Fig. 2b), but, if the views present the same place, only one association is valid. However, as examining all possible combinations is intractable, we build a probability distribution of the robot pose using cues from small subsets of planes.

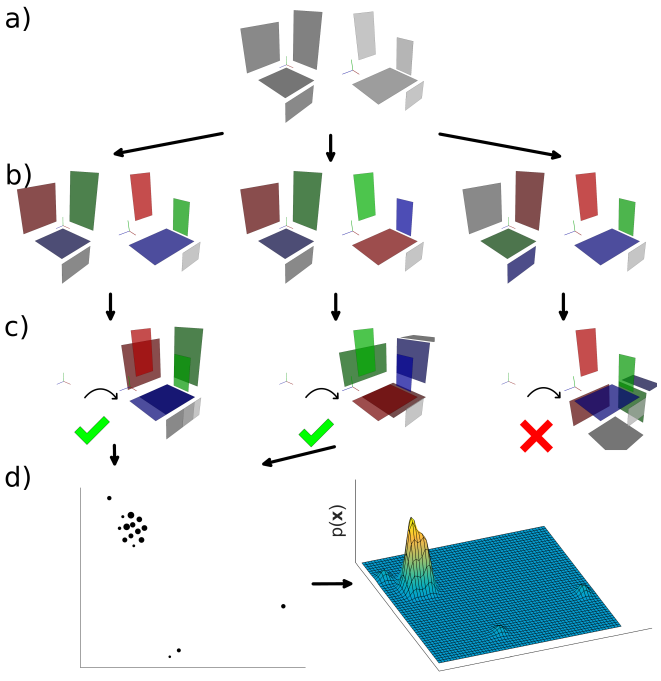


Fig. 2. Schematic illustration of generating the global robot pose PDF by matching sets of planes

At least three non-parallel pairs of matching planes are required to obtain an SE(3) transformation. Unfortunately, we don't know the associations between plane segments. We can try to discover them employing appearance (e.g. color), size and global location. However, using such criteria is insufficient, and there is a need to examine constraints imposed by the geometric transformations between the potentially matching features. Unfortunately, even using multiple above-mentioned criteria for matching we sometimes obtain wrong associations that act as strong outliers in typical estimation procedures, e.g. involving Kalman filtering [16]. A common solution to this problem is to embed the estimation into a RANSAC procedure, which however spawns other issues, such as extensive hypothesis evaluation and setting proper thresholds for outlier rejection. Therefore, our novel idea is to use multiple triplets of potentially corresponding planes to generate a kernel-based global probability density function (PDF) that describes the likelihood of the robot/sensor pose.

The triplets consist of three pairs of associated plane features (cf. Fig. 2b). Each triplet is evaluated if it induces a plausible transformation by projecting planes from the coordinate frame of the current sensor view into the coordinate frame of the global map (Fig. 2c). But even the plausible triplets should not contribute equally to the final pose hypothesis, as some triplets contain better matches than others. Therefore the contribution has to be weighted, which is illustrated in a simplified form in Fig. 2d. Inspired by [18], where a probability distribution was constructed from samples to find a feasible grasping sequence, we construct a PDF to find the transformation that best explains the observed triplets of segmented planes. A triplet supports the transformation by introducing a weighted Gaussian kernel that adds to the PDF. The kernels are placed in the location space, i.e. each point in that space corresponds to a possible transformation between the sensor view and the map. Hence, if many kernels are placed in some area, the probability density in that area is high. During localization, we seek the maximum of the PDF and finally test it for being the correct transformation.

IV. TRIPLETS OF PLANES

This section describes data processing steps used in generation and evaluation of triplets. The process begins with plane segmentation that isolates planar surfaces from a point cloud. The extracted planes are then matched to a set of planes in a global map and outcome pairs are used to form triplets of pairs representing possible transformations. Each transformation is evaluated to test if it is plausible and then passed to the probabilistic framework.

A. Extracting planes

We extract planar surfaces from a point cloud representing the observed scene using a simple method based on segmentation and segment merging by flood fill. The method is designed for the presented system because none of the off-the-shelf algorithms (e.g. [19]) satisfied our requirements. The point cloud is segmented by means of supervoxel clustering (Fig. 3a) and each segment with a low curvature is considered as a seed. The algorithm, using supervoxel adjacency list, recursively merges all segments connected to a seed that are sufficiently flat, their normals are approximately

parallel to the normals of the seed, and there are no steps between them. Merged segments are tested to have at least minimal size to avoid adding many small segments. The last operation is to compute plane equations using all points belonging to the generated segments (Fig. 3b).

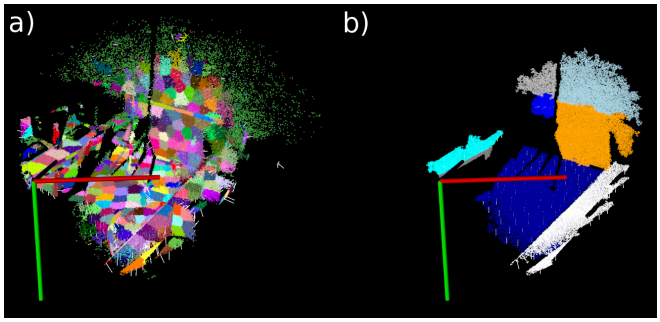


Fig. 3. Segmentation of planar surfaces with visualized normals (white lines): supervoxels (a), and merged segments (b)

B. Selecting triplets

Having the current view and the global map represented as planes, we pick pairs of planes, one plane from the current view, and another one from the map. Those pairs are potential matches, and each of them may be either correct if the two planes indeed represent the same planar surface, or incorrect if they don't. To limit the number of pairs only planes that are visually similar are considered. The appearance of each plane is represented as a histogram of the Hue and Saturation components of the HSV color model and is embedded into a vector $\mathbf{h}_i^s$ for the i-th plane from the current view, and into a vector $\mathbf{h}_j^m$ for the j-th plane from the map. Planes i and j are considered similar if the difference between their histograms doesn't exceed the predefined threshold:

$$h_{(i,j)} = |\mathbf{h}_i^s - \mathbf{h}_j^m| < \tau_h. \tag{1}$$

As three pairs allow to compute an SE(3) transformation, we form triplets of pairs that represent a valid transformation if all three matches are correct. Again, to limit the size of the search space, each triplet has to fulfill the following conditions:

- Each plane i and j has to appear in at most one of three pairs, as the same plane segment cannot be matched more than once.
- The map planes must not be further than τ_d each from the other. The map can be large in comparison to the current view, therefore if two planes are far from each other, they won't be visible in the same view.

C. Computing transformations

To evaluate correctness of the established triplets it is necessary to calculate an alleged SE(3) transformations between the frames of reference of the local view and the global map induced by those triplets. We use a general method that takes as input $n \geq 3$ pairs of planes and outputs a transformation given by the translation vector $\mathbf{t} = [t_x \quad t_y \quad t_z]^T$ and the rotation quaternion $\mathbf{r} = [r_x \quad r_y \quad r_z \quad r_w]^T$. The method consists of two steps. At first it calculates the rotation using normal vectors of the planes $\mathbf{n}_i^s$ and $\mathbf{n}_j^m$, then the translation is obtained using distances from origins d_i^s and d_j^m also. The normal vector and distance from the origin are parameters of the plane, and can be used to form an equation satisfied by every point $\mathbf{q}$ belonging to that plane: $\mathbf{n} \cdot \mathbf{q} - d = 0$. A derivation of the rotation calculation algorithm is based upon the method of Walker *et al.* [20]. The algorithm tries to minimize the differences between the views's plane normal vectors and the transformed map's plane normal vectors:

$$\begin{aligned} \mathbf{e}_{(i,j)} &= |\mathbf{W}(\mathbf{r})^T \mathbf{Q}(\mathbf{r}) \mathbf{n}_j^m - \mathbf{n}_i^s|^2 \\ &= 2\left[1 - \mathbf{r}^T \mathbf{Q}(\mathbf{n}_i^s)^T \mathbf{W}(\mathbf{n}_j^m)\mathbf{r}\right]. \end{aligned} \tag{2}$$

The total energy to minimize is given by equation:

$$E = \sum_{(i,j)} 2\left[1 - \mathbf{r}^T \mathbf{Q}(\mathbf{n}_i^s)^T \mathbf{W}(\mathbf{n}_j^m)\mathbf{r}\right] = \mathbf{r}^T \mathbf{C} \mathbf{r} + \text{const.}, \tag{3}$$

where: $\mathbf{C} = -2\sum_{(i,j)} \mathbf{Q}(\mathbf{n}_i^s)^T \mathbf{W}(\mathbf{n}_j^m)$. Taking a derivative of (3) with respect to $\mathbf{r}$ and using a Lagrangian multiplier we obtain equation:

$$\underbrace{-\frac{1}{2}(\mathbf{C} + \mathbf{C}^T)}_{\mathbf{D}} \mathbf{r} = \lambda \mathbf{r} \tag{4}$$

where the solution $\mathbf{r}^*$ is given by the eigenvector of matrix $\mathbf{D}$ that corresponds to the largest eigenvalue. The computed rotation is an unambiguous solution to (4) if the largest eigenvalue is unique as well. The planes are considered to be one-sided, and the normal vectors point in direction opposite to the direction which a plane was observed from. The one-side assumption is justified by the fact that we want to use planar surfaces of objects that have some volume.

Computing the translation between frames of references is done by minimizing square error between the sensor view's plane distance, and the transformed global map's plane distance [21]:

$$S = \sum_{(i,j)} (d_i^s - d_j^m - (\mathbf{n}_j^m)^T \mathbf{t})^2 = |\mathbf{A} - \mathbf{B}\mathbf{t}|^2, \tag{5}$$

where:

$$\mathbf{A} = \left[d_{i,1}^s - d_{j,1}^m, \ldots, d_{i,n}^s - d_{j,n}^m\right]^T, \tag{6}$$

$$\mathbf{B} = \left[\mathbf{n}_{j,1}^m, \ldots, \mathbf{n}_{j,n}^m\right]^T. \tag{7}$$

The solution is given by the pseudo-inverse method:

$$\mathbf{t}^* = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{A}. \tag{8}$$

D. Evaluating transformations

Unfortunately, as the transformation computation is only minimizing the errors, the solution may actually be implausible in case of planes mismatch. To verify this, we use plane parametrization represented as a unit quaternion [14]:

$$\mathbf{p} = \frac{1}{\sqrt{1+d^2}}[n_x, n_y, n_z, -d]^T, \tag{9}$$

and the transformation from map's frame of reference to the local view's frame of reference as a homogeneous matrix

$\mathbf{T}_{m,s}$. If the transformation is valid, for each pair of planes a representation of map's plane transformed to the view's frame should be approximately equal to the representation of the view's plane. The difference between those representations is computed using the logarithm map of quaternions:

$$\mathbf{f}_{(i,j)} = \log\left\{\left[\mathbf{T}_{m,s}^{T}\mathbf{p}_{j}^{m}\right]^{-1}\mathbf{p}_{i}^{s}\right\}. \tag{10}$$

Note that using a homogeneous matrix to transform quaternions is different than when transforming points:

$$\mathbf{p}^{s} = \mathbf{T}_{s,m}^{-T}\mathbf{p}^{m} = \mathbf{T}_{m,s}^{T}\mathbf{p}^{m}, \tag{11}$$

and the result has to be normalized afterwards. Finally, the criterion has a form $|\mathbf{f}_{(i,j)}| < \tau_f$, i.e. a norm of the logarithm map difference has to be below a certain threshold τ_f.

Planes are infinite and the computed transformation can be implausible, because the solution can point to a distant location, far away from the investigated environment. Therefore, it is also necessary to check if the transformation is justified by the available observations. In this work we assume the solution to be plausible when the convex hulls of the points belonging to the map planes, denoted by $\mathrm{chull}(\mathbf{P}_{j}^{m})$, after transformation to the current view frame of reference, overlap with the convex hulls of the points belonging to the current view planes, denoted by $\mathrm{chull}(\mathbf{P}_{i}^{s})$:

$$\mathbf{G}_{(i,j)} = \left[\mathbf{T}_{m,s}^{-1}\mathrm{chull}(\mathbf{P}_{j}^{m})\right] \cap \mathrm{chull}(\mathbf{P}_{i}^{s}). \tag{12}$$

In other words, we check if for each pair of planes the same segment of the plane is observed and represented in the global map and the current scene (Fig. 4). The area of the co-observed part has to have a certain size: $\mathrm{area}(\mathbf{G}_{(i,j)}) > \tau_g$.

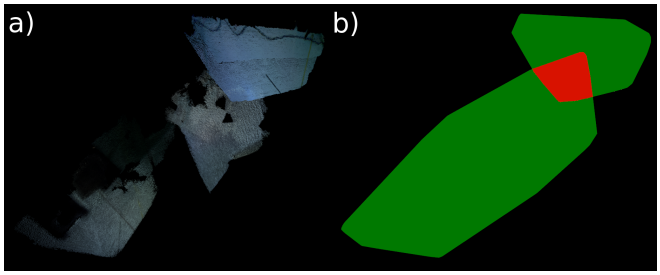


Fig. 4. Transformed point cloud of the map and point cloud of the current view (a), and intersection of convex hulls (b)

V. PROBABILISTIC FRAMEWORK

The set of triplets $\mathcal{T}$ generated in previous processing steps is next included in the probabilistic framework. Each triplet is scored and the computed scores are treated as weighting factors used to build a PDF. The final outcome of the method is the SE(3) transformation, which is most probable according to the supporting evidence.

A. Assigning weights

The triplet weight takes into consideration the appearance difference $h_{(i,j)}$ (1) and the area of the convex hulls intersection $\mathrm{area}(\mathbf{G}_{(i,j)})$ (12). Moreover, the weight depends on how frequently the respective plane segment was employed, as the same plane can be a part of multiple triplets, and therefore can introduce a bias to the PDF. We handle it by calculating an occurrence factor for each triplet a and each pair $(i,j) \in \mathcal{S}_a$ within the triplet. The more triplets including the same plane in a vicinity of induced transformation, the lesser the weight:

$$w_{a,(i,j)} = \left[\sum_{b\in\mathcal{T}}\sum_{(k,l)\in\mathcal{S}_b}\mathbb{I}_{i=k}\exp(-y_{(a,b)})\right]^{-1}, \tag{13}$$

where $\mathbb{I}_{i=k}$ is an indicator function equal to 1 whenever $i = k$ and 0 otherwise, and $y_{(a,b)}$ is a norm of a SE(3) logarithm map of a difference between transformations $\mathbf{v}$ for the triplets a and b:

$$y_{(a,b)} = \left|\log(\mathbf{v}_{a}^{-1}\mathbf{v}_{b})\right|. \tag{14}$$

Note that, in opposition to Section IV, transformations are parametrized as 6-D vectors $\mathbf{v}$ that contain 3 translation variables t_x, t_y, t_z and 3 rotation variables r_x, r_y, r_z (the r_w can be always restored assuming that is non-negative and the quaternion is a unit quaternion). The overall weight for the triplet a is expressed by the equation:

$$w_a = \sum_{(i,j)\in\mathcal{S}_a}\mathrm{area}(\mathbf{G}_{(i,j)})w_{a,(i,j)}\exp(-h_{(i,j)}). \tag{15}$$

B. Constructing localization distribution

The SE(3) transformations induced by triplets are represented as points in a 6-D space with 3 variables for the position, and 3 for the rotation. The strength of the contribution is controlled by a weight given by (15), and we convert this contribution to the probabilistic language by placing a weighted Gaussian kernel in each transformation point. The final PDF is therefore a sum of all kernels:

$$p(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x}) = \frac{1}{Z}\sum_{a\in\mathcal{T}}K_a(\mathbf{x}), \tag{16}$$

where Z is a normalizing constant, and the kernel is:

$$K_a(\mathbf{x}) = w_a\exp\left\{-\log(\mathbf{v}_{x}^{-1}\mathbf{v}_{a})^{T}\mathbf{I}_a\log(\mathbf{v}_{x}^{-1}\mathbf{v}_{a})\right\}. \tag{17}$$

The distance between kernel's center $\mathbf{v}_a$ and the transformation $\mathbf{v}_x$ represented by the point $\mathbf{x}$ is computed using logarithm map, and is embedded in the square form of multidimensional Gaussian distribution with the information matrix $\mathbf{I}_a$.

Having a probability distribution, we seek for the point with the highest probability. Inference in general distributions can be complicated and to avoid this, we exploit the fact that we already have a list of possible solutions. For each triplet, we evaluate the transformation induced by this candidate, and choose the one with the highest probability, denoted further as $\mathbf{x}_1$. Additionally, we search for the second-best solution $\mathbf{x}_2$ that is properly distant from the best one. To decide that the transformation $\mathbf{x}_1$ is correct, its probability has to exceed a certain threshold:

$$\tilde{p}(\mathbf{x}_1) > \tau_p \tag{18}$$

and has to be significantly greater than for the second-best maximum:

$$\tilde{p}(\mathbf{x}_1) - \tilde{p}(\mathbf{x}_2) > \tau_{pd}. \tag{19}$$

The last test is the fitness score test. It refers back to point clouds and assesses if the current view's point cloud transformed to the map's frame of reference is aligned with the map's point cloud. This test involves computation of the sum of squared distances from each point of the transformed views's point cloud $\mathbf{P}^s$ to the nearest neighbor in the map's point cloud $\mathbf{P}^m$, and has to be below the threshold:

$$\sum_{\mathbf{q}_l^s \in \mathbf{P}^s} (\mathbf{q}_l^s - \hat{\mathbf{q}}_l^m)^2 < \tau_{fs}, \tag{20}$$

where $\hat{\mathbf{q}}_l^m$ is the nearest neighbor in map's point cloud. It's worth noting that the fitness score test examines only points belonging to plane segments, and is much faster than ICP, as (20) is equivalent to a single iteration of ICP on a reduced size point cloud. If the transformation fulfills all three conditions, it is assumed to be the correct one.

VI. EXPERIMENTAL EVALUATION

We evaluated the proposed algorithm in the global localization task with a known map, as the software is not yet integrated within a SLAM system. For global localization, the algorithm requires a point cloud representing the current view (local scene), and a global map composed of plane segments. As we are not aware of any experimental RGB-D dataset for which a map of plane features is available, we built a "global" point cloud using the ElasticFusion software [10], with the dataset's ground truth trajectory used for registration to avoid drift. The fused point cloud was then segmented into plane features, as described in Section IV. Our approach needs also the local point cloud to extract planes in the current view. Using a single RGB-D frame for that purpose is not enough because a sufficient number of planes has to be detected. Hence, to widen the local view context, we used again ElasticFusion to fuse together point clouds from the last 100 RGB-D frames. Considering the Kinect frame rate this short sequence takes few seconds, and in most cases does not accumulate significant drift. For the local perception, the ElasticFusion's trajectory estimates are used in the presented experiments.

The main dataset used is the PUT RGB-D/Workshop (PUT RGB-D/W)[2], which consists of 10 sequences (*seq1* to *seq10*) acquired in a 8×8 metres robotic workshop. The ground truth data was captured using the OptiTrack motion capture system. Three non-overlapping sequences *seq5*, *seq6* and *seq7* were used to build the global map that contains 56 segments. Moreover, we used a sequence from the publicly available NYUv2 dataset [22] acquired in a typical household environment. Unfortunately, in NYUv2 no ground truth data for the sensor trajectory is provided. Thus, we had to use the poses computed by ElasticFusion as ground truth.

The localization performance was measured by counting the locations that were correctly recognized, and those that were recognized incorrectly (if any). For the performance tests, we applied the algorithm to every 10-th pose of the recorded sequence, attempting to localize the sensor with respect to the global map. We treated as correct the sensor poses that were distant at most 0.11 from the respective ground truth pose, using the metrics given by (14). The 0.11 value was chosen, because re-localization within this range usually enables to recover tracking in our RGB-D SLAM [1], and should be suitable for other similar SLAM systems. Additionally, mean Euclidean $\bar{d}$ and angular $\bar{\alpha}$ distances between the computed poses and the ground truth trajectory were computed in all tests. Results are gathered in Tab. I, where four sets of parameters were evaluated: optimal for the PUT RGB-D/W dataset, optimal for the NYUv2 dataset, with probability difference test switched off, and with point cloud alignment test switched off. Visualizations of the recognized places are presented in Fig. 5 and 6, for the PUT RGB-D/W and NYUv2 datasets, respectively. We used the following parameter values: $\tau_d = 5.0$, τ_h = 2.5 (1.3 for NYUv2), $\tau_f = 0.05$ and $\tau_g = 0.1$.

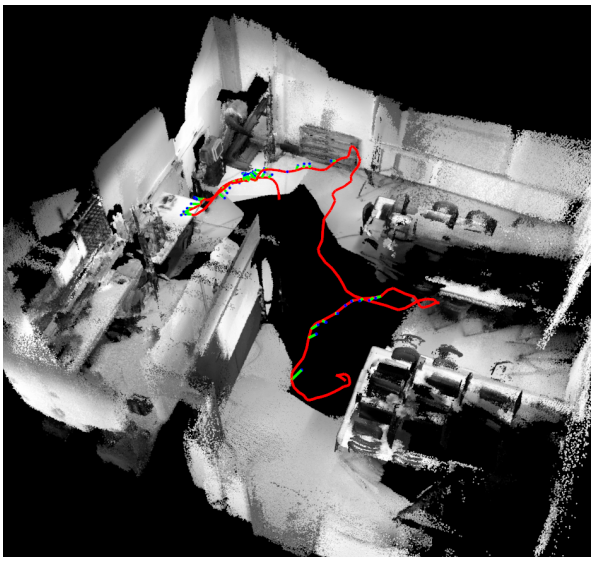


Fig. 5. Global localization results for the PUT RGB-D/W dataset *seq2* for $\tau_p = 1.1$, $\tau_{pd} = 0.2$, and $\tau_s = 0.07$. Test trajectories are marked with red lines, recognized places with blue dots, whereas green lines connect recognized locations to their respective ground truth poses

TABLE I

GLOBAL LOCALIZATION RESULTS FOR DIFFERENT PARAMETER SETS

parameters			PUT RGB-D/W					NYUv2				
τ_p	τ_{pd}	τ_{fs}	corr	incorr	unk	$\bar{d}$ [m]	$\bar{\alpha}$ [°]	corr	incorr	unk	$\bar{d}$ [m]	$\bar{\alpha}$ [°]
1.1	0.2	0.07	49	0	112	0.123	0.24	134	6	78	0.113	2.23
1.2	0.2	0.03	33	0	128	0.113	0.28	113	0	105	0.085	0.15
1.2	0.0	0.03	33	3	125	0.492	12.81	113	0	105	0.085	0.15
1.2	0.2	∞	49	13	99	0.559	35.82	147	7	64	0.122	2.07

The results obtained using two different datasets indicate that the proposed method is reliable, finding a large number of locations along the test trajectories. If the algorithm is correctly parametrized, it produces no false positives, which is of pivotal importance in localization task. The main cause for the number of places (local views) that remained unrecognized was the insufficient quality of depth data used to create the global maps. Maps produced from a single sequence (NYUv2) or few sequences of very limited overlapping had many areas that were empty or contained

[2]Available at `http://lrm.put.poznan.pl/rgbdw/`

point clouds of insufficient density to extract correct planes.



Fig. 6. Global localization results for the NYUv2 dataset for $\tau_p = 1.2$, $\tau_{pd} = 0.2$, and $\tau_s = 0.03$. Test trajectories are marked with red lines, recognized places with blue dots, whereas green lines connect recognized locations to their respective ground truth poses

The mean computing time for a single global localization act in the experiments was 21 s on a Core i5 2.6 GHz laptop. However, the most time-consuming step (14 s in average) was the computation of the fitness score (20). This step can be made much faster using approximate nearest neighbor search, taking less than 1 s, as shown by a preliminary implementation employing octree. Further optimization of the computation time is a matter of our current research.

VII. CONCLUSIONS

We tackled the problem of global localization applying a novel approach that creates a PDF describing the likelihood of sensor's pose using plane features. The experimental results suggest that the proposed method performs well in large-room-sized environments, yielding correct and accurate pose estimates whenever it is possible to provide a good quality of the *a priori* map and there are features available in the environment.

An important advantage of the new probabilistic framework is that not only paired planes can contribute to the PDF. The framework can handle localization cues coming from other feature types, or from other sensing modalities, e.g. an orientation sensor like AHRS. Applications of the presented algorithm are not limited to global localization and loop-closing. It can be easily adapted for matching plane features in graph-based SLAM utilizing planes [4].

ACKNOWLEDGMENT

The OptiTrack was funded under grant 6419/IA/SP/2015 and used courtesy of the Chair of Control and Systems Engineering. The authors also would like to thank M. Kiełczewski for the technical assistance with OptiTrack.

REFERENCES

[1] D. Belter, M. Nowicki, and P. Skrzypczyński, "Improving accuracy of feature-based RGB-D SLAM by modeling spatial uncertainty of point features," in *IEEE Int. Conf. on Robotics and Automation*, Stockholm, 2016, pp. 1279–1284.

[2] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," in *IEEE Int. Conf. on Robotics and Automation*, Stockholm, 2016, pp. 1285–1291.

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," in *arXiv preprint, arXiv:1610.06475v1*, 2016.

[4] J. Wietrzykowski, "On the representation of planes for efficient graph-based SLAM with high-level features," *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 10, no. 3, pp. 3–11, 2016.

[5] K. Ho and P. Newman, "Combining visual and spatial appearance for loop closure detection in SLAM," in *Proc. of European Conference on Mobile Robots (ECMR)*, Ancona, 2005, pp. 62–67.

[6] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2016.

[7] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[8] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based SLAM," in *IEEE Int. Conf. on Robotics and Automation*, Hong Kong, 2014, pp. 846–853.

[9] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[10] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "ElasticFusion: Dense SLAM without a pose graph," in *Robotics: Science and Systems (RSS)*, Rome, 2015.

[11] M. Heredia, F. Endres, W. Burgard, and R. Sanz, "Fast and robust feature matching for RGB-D based localization," in *arXiv, CoRR abs/1502.00500*, 2015.

[12] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, 2006, pp. 3062–3067.

[13] Y. Taguchi, Y. D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, 2013, pp. 5182–5189.

[14] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *IEEE Int. Conf. on Robotics and Automation*, Seattle, 2015, pp. 4605–4611.

[15] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3D mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[16] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov, and I. Petrović, "Global localization based on 3D planar surface segments detected by a 3D camera," in *Proc. of the Croatian Computer Vision Workshop*, Zagreb, 2013, pp. 31–36.

[17] E. Fernández-Moral, W. Mayol-Cuevas, V. Arévalo, and J. González-Jiménez, "Fast place recognition with plane-based maps," in *IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, 2013, pp. 2719–2724.

[18] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *Int. Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[19] T. T. Pham, M. Eich, I. Reid, and G. Wyeth, "Geometrically consistent plane extraction for dense indoor 3D maps segmentation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Daejeon, 2016, pp. 4199–4204.

[20] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, 1991.

[21] O. D. Faugeras and M. Hebert, "A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces," in *Proc. of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, 1983, pp. 996–1002.

[22] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proceedings of the 12th European Conference on Computer Vision*. Berlin: Springer, 2012, pp. 746–760.

Semantical Occupancy Grid Mapping Framework

Timo Korthals, Julian Exner, Thomas Schöpping, and Marc Hesse

Abstract— **In recent decades, mapping has been increasingly investigated and applied in unmanned terrain, aerial, sea, and underwater vehicles. While exploiting various mapping techniques to build an inner representation of the environment, one of the most famous remaining is occupancy grid mapping. It has been applied to all domains in a 2D/3D fashion for localization, mapping, navigation, and safe path traversal. Until now generally active range measuring sensors like LiDAR or SONAR are exploited to build those maps. With this work the authors want to overcome these barriers by presenting an occupancy mapping framework offering a generic sensor interface. The interface handles occupancy grids as inverse sensor models, which may represent knowledge on different semantical decision levels and therefore build up a semantic grid map stack. The framework offers buffered memory management for efficient storing and shifting and further services for accessing the 2D map stack in different cell-wise pre-fused and topometric ways. Within the framework, two novel techniques operating especially with occupancy grids are presented: First, a novel odds based interpolation filter is introduced, which scales grid maps in a Bayesian way. Second, a Supercell Extracted via Variance-Driven Sampling (SEVDS) algorithm is presented which, abstracts the semantical occupancy grid stack to a topometric map. While this work focuses on the framework's introduction, it is extended by the evaluation of SEVDS against state-of-the-art superpixel approaches to prove its applicability.**

I. Introduction

Mapping frameworks (MFs) are widely used within commercial products and build up the foundation in many services like navigation with Google Maps or process planning in farm management systems. Fig. 1 shows the principal concept of a MF, which consists mainly of the map storage. A map is a symbolic depiction which emphasizes relationships between elements of some space. These maps may have different representations depending on the application like topographic, metric, orthographic, etc. Data acquisition builds the map storage, whilst the preprocessing handles the transformation and embedding of incoming data. Services build up the link to other applications or clients, which invoke a processing on the maps due to their request. For the sake clarity, data acquisition and processing are explicitly depicted, but can be alternatively represented as a service as well. Within the authors' nomenclature of MFs, services are restricted to the derivation of alternative representations of stored maps which excludes navigation, task calculation, and others. Due to that definition, different mapping frameworks exist in robotics that model an inner representation of the world. Apparently, every known mapping algorithm is realized as a MF, which stores e.g. range readings into a data structure while the constructed map can be requested during navigation. The famous Simultaneous Localization and Mapping (SLAM) concept, for instance, can be interpreted as MF with feedback loop which updates a map hypothesis during data acquisition and loop closure.

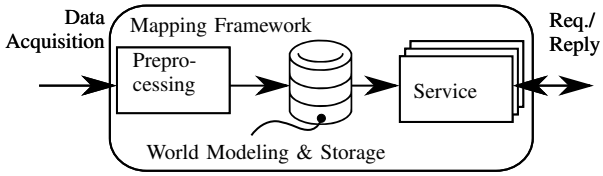


Fig. 1: Mapping framework concept

2D occupancy grids as statistical sufficient representation for an inner world model were introduced by Moravec and Elfes [1]. They have been widely applied and are still up to date for robotic mapping and SLAM applications due to their lean and efficient implementations. Semantical occupancy grid maps (SOGMs), also known as inference grids [2], [3], on the other hand got left behind since their first promotion. This could be of lacking applications or implementation complexity, because for each sensor detection algorithms, a mapping from raw sensor data to a 2D grid-based semantical interpretation of the environment has to be realized. On the other hand, nowadays enormous robotic sensor setups demand new approaches where SOGMs could be the next breakthrough.

With this definition of SOGMs and MFs in mind, the publication presents an Robot Operating System (ROS) implementation of a map centric, semantical occupancy grid mapping framework with generic interfaces to fuse inverse sensor models. First, related work is presented in Section II. In Section III the authors' framework is introduced where the occupancy mapping is depicted and generous sensor interfaces, interpolation, scan matching, services, and the implementation are defined. Section IV gives an evaluation for a supercell clustering service in a static table top scenario. Finally, a view on future applications in Section V is given where the proposed framework will be exploited.

II. Related Work

In the ongoing chapters the authors focus on robotic mapping applications for environment traversal, recapitulate common approaches and bring them into the given nomenclature.

Bielefeld University, Cluster of Excellence Cognitive Interaction Technologies, Cognitronics & Sensor Systems, Inspiration 1, 33619 Bielefeld, Germany, http://www.ks.cit-ec.uni-bielefeld.de/, {tkorthals, jexner, tschoepp, mhesse}@techfak.uni-bielefeld.de

A. Data Acquisition and Preprocessing

In the context of robotics achieving safe environment traversal, areas which are not occupied by an object are of interest. Active sensors concepts like LiDAR or SONAR, which are inherently able to detect the range to objects, are commonly used for this particular task [4]. Furthermore, passive and indirect measurements by cameras, in a stereo or inverse perspective mapping setup, became famous in automotive applications due to their inexpensiveness [5]. Range measurements are preprocessed, such that they are first transformed into the proper coordinate system of the map. After this, an inverse sensor model is applied, which is the deduction from measurements to causes [6].

B. World Modeling

The world modeling storage focuses the format in which the information is obtained. Nüchter et al. sorts maps used in robotics for representing environments in the following descending order from level of detail to compactness [7]: *Grid maps* spatialize the environment into an equidistant grid which can be stored in arrays, or more efficiently in sparse environments in quad- or octal-trees. Grid maps on one hand are limited in environmental representation by their resolution which influences memory consumption drastically, but their non-parametric property makes them ideal for mapping unstructured objects. *Feature maps* represent obstacles via geometric primitive descriptions, e.g. lines, planes, or cubes. Topometric maps are *hybrid maps* consisting of a topological structure with direct spatial relationship. The highest compactness and therefore greatest abstraction is achieved by *topological maps* where no more spatial but logical relations exist.

C. Services & Existing Mapping Frameworks

Services are fundamental extensions to the mapping approach. They offer any subsequent application the interface to access the mapped data in specific ways. For a navigation approach on unstructured environments commonly a metric grid map is necessary and therefore, a service which formats the map data in such way is recommended.

Applicable mapping frameworks (MF) exist and are generally implemented within the Robotic Operating System (ROS) middleware [8]. The most common approach for the map storage and for incorporating new scans is the occupancy grid map (OGM) algorithm based on grid maps in a 2D or 3D fashion [1]. Due to that, the authors focus on application which implemented that approach.

A popular ROS package that works with a grid-based occupancy map representation is the *costmap_2d* for incorporating range measurements and offering the result to the ROS 2D navigation stack [9]. *Karto SDK* is an commercial but educationally freely available MF which incorporates mapping and additionally localization, path planning, and exploration while interfaces for other robotic middlewares (MS Robotics Studio and Robot OS) exist [10]. Kohlbrecher et al. provide a comprehensive ROS package called *hector_slam* which is maintained by Team Hector [11]. It comprises the main application *hector_mapping* which realizes a SLAM algorithm. Various subpackages extend the mapping as services, e.g. map retrieval (*hector_map_server*) or trajectory creation (*hector_trajectory_server*), which are build upon that. *Gmapping* realizes a Rao-Blackwellized particle filter for SLAM and allows access of the map via a service [12]. All former approaches exploit 2D grid maps while they are mainly designed for autonomous ground (AGV) or surface (ASV) vehicles. 3D OGM is commonly applied in autonomous aerial vehicles (AAV). A famous 3D occupancy mapping framework is *OctoMap* by Hornung [13]. More recently Droeschel et al. [14] used occupancy grid mapping to fuse multiple sensor modalities into one representation to achieve robust mapping in AAVs. Multimapping approaches have been done by Morris et al. [15] handling one static and multiple temporal maps, and Frankhauser and Hutter handling multiple elevation maps [16].

Fundamentally, all named application share the same inner representation structure which is a two dimensional OGM approach for multiple reasons: First, grid maps can be stored, searched, and accessed very efficiently by computer systems. Second, the OGM algorithm introduced in III-A allows a robust and efficient online sensor fusion and map updating.

III. Semantical Occupancy Grid Mapping Framework

Within this publication, three challenges are faced: transfering the sensor detections into a map representation of the vehicle's environment, storing, and service advertisement. First, the semantical occupancy grid mapping approach as a map server is introduced in Section III-A. Second, the generic interface definition by inverse sensor models (ISM) is further explained in Section III-B. More deeply, Section III-C proposes a novel interpolation technique for OGMs/ISMs in Section III-C.1 and scan matching in Section III-C.2. Services are introduced for accessing the current map data in a raw, fused, or topometrical way in Section III-D.1 and III-D.2. Finally, Section III-E gives the functional description of the map server application in ROS.

A. Semantical Occupancy Grid Mapping

Two-dimensional occupancy grids were originally introduced by Elfes [17]. In this representation, the environment is subdivided into a regular array or a grid of quadratic cells. The resolution of the environment representation directly depends on the size of the cells. In addition to this discretization of space, a probabilistic measure of occupancy is associated with each cell. This measure takes on any real number in the interval $[0, 1]$ and describes one of the two possible cell states: unoccupied or occupied. An occupancy probability of 0 means definitely unoccupied space, and a probability of 1 means definitely occupied space. A value of 0.5 refers to an unknown state of occupancy.

An occupancy grid is an efficient approach for representing uncertainty, fusing multiple sensor measurements on the decision level, and to incorporate different sensor models [5]. To learn an occupancy grid M given sensor information

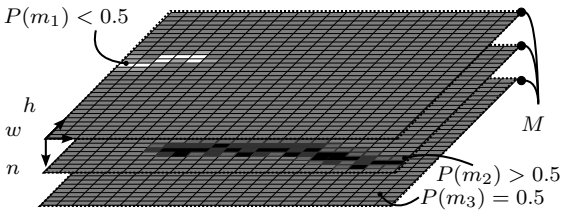


Fig. 2: SOGM with $N = 3$ layers

z, different update rules exist [18]. For the author's approach, Bayesian update rule is applied to every cell $m \in M$ at position (w, h) as follows: Given the positions x_t of a vehicle at time t, let $x_{1:t} = x_1, \ldots, x_t$ be the positions of the vehicle's individual steps until t, and $z_{1:t} = z_1, \ldots, z_t$ the environmental perceptions. Occupancy probability grids determine for each cell m of the grid the probability that this cell is occupied by an obstacle. Thus, occupancy probability grids seek to estimate

$$P\left(m|z_{1:t}, x_{1:t}\right) = \mathrm{Odd}^{-1}\left(\prod_{t=1}^{T} \underbrace{\frac{P\left(m|z_t, x_t\right)}{1 - P\left(m|z_t, x_t\right)}}_{\mathrm{Odd}(P(m|z_t, x_t))}\right) \quad (1)$$

This equation already describes the online capable, recursive update rule that populates the current measurement z_t to the grid, where $P\left(m|z_{1:t}, x_{1:t}\right)$ is the so called inverse sensor model (ISM). The ISM is used to update the OGM in a Bayesian framework, which deduces the occupancy probability of a cell, given the sensor information.

The extension to semantical occupancy grid maps or inference grids is straightforward and defined by an OGM M with W cells in width, H cells in height, and N semantical layers:

$$M : \{1, \ldots, W\} \times \{1, \ldots, H\} \rightarrow m = \{0, \ldots, 1\}^N \quad (2)$$

Compared a single layer OGM which allows the classification into three classes $\left\{\text{occupied}, \overline{\text{occupied}}, \text{unknown}\right\}$, the SOGM supports a maximum of $\left|\left\{\text{occupied}, \overline{\text{occupied}}, \text{unknown}\right\}\right|^N = 3^N$ different classes allowing much higher differentiability in environment and object recognition. The corresponding ISMs are fused via the occupancy grid algorithm in their nth associated semantical occupancy grid. The implementation of a semantical map stack as shown in Fig. 2 is further called *map server* (MS).

B. Inverse Sensor Model Handling

The ISM is the preprocessed sensor data originating from one or more sensors. It maps from causes to reasons so that information resides on a decision level (cf. [19]) as occupancy probability at the particular cell. It is commonly used for sensors with a planar sensor lobe, oriented parallel to the ground. In that case, a quite simplistic model can be applied, e.g. for a laser range finder. Each cell m that is covered by the beam of the observation z and whose distance to the sensor is shorter than the measured one, is supposed to be unoccupied. The cell in which the beam ends (the measurement point) is supposed to be occupied, and everything behind is unknown [20]. For generic implementations, however, sensors like cameras LiDAR and RADAR may be non-planary installed and thus their sensor lobes are tilted with respect to the map. Each sensor-algorithm combination requires its own ISM, converting from the algorithm's output to a 2D measurement grid representation. An ISM approach for monocular/stereo cameras, depth cameras, proximity sensors, RADAR, LiDAR and their corresponding algorithms has been published by Kragh et al. [21].

C. Preprocessing

Incorporating new sensor measurements on ISM level needs to be handled in two dependent ways: First, the resolution of the ISM is interpolated to meet the MS. Second, the ISM is matched and therefore transformed by a homography into the MS frame.

1) Interpolation: Kohlbrecher et al. states, that the discrete nature of occupancy grid maps limits the precision that can be achieved and also does not allow the direct computation of interpolated values or derivatives [11]. For this reason an interpolation scheme, allowing sub-grid cell accuracy, is necessary for estimating occupancy probabilities. Intuitively, the grid map cell values can be viewed as samples of an underlying continuous probability distribution which can be defined in various ways. A naive approach might be the nearest-neighbor interpolation. In [11], a bilinear filter is introduced for subsampling OGMs. Other approaches using higher polynomials should not be applied to OGM interpolation, due to their value range definition outside of the sampling points. This can lead to values outside of $(0, 1)$ which needs to be truncated or handled otherwise.

In this paper, a new interpolation method called *biodds interpolation* is proposed where it is assumed that not the occupancy values are linearized between the sampling points, but the amount of sensor readings. This way, the nature of the OGM algorithm in conjunction with the probability values, which are clearly non-linear as stated by Hähnel [18], is respected. To achieve this, an interpolation function is derived from a two-parametric odds model:

$$\widehat{\mathrm{Odd}}\left(R; \epsilon_0, \epsilon_1\right) = \mathrm{Odd}\left(\epsilon_0\right) \mathrm{Odd}\left(0.5 + \epsilon_1\right)^R \quad (3)$$

Defining $\epsilon_0 > 0$ and $1 - \epsilon_0$ as the lower and upper bounds and $\epsilon_1 > 0$ as the information increment, Equation 3 can be evaluated over $R \in [0, \log_{\mathrm{Odd}(0.5+\epsilon_1)}(\mathrm{Odd}(1 - \epsilon_0)/\,\mathrm{Odd}(\epsilon_0))]$ measurements. For the sake of clarity the value range R is mapped to $\hat{x} \in [0, 1]$. Fortunately, the given odds model can be approximated by function f with a vanishing error by a one-parametric hyperbolic function:

$$f(x; \alpha) = 0.5 \tanh((x - 0.5)\alpha) + 0.5 \quad (4)$$

Wrapping everything up, the interpolation m of point $\mathbf{P}$ can be calculated by applying the approximator f as depicted in

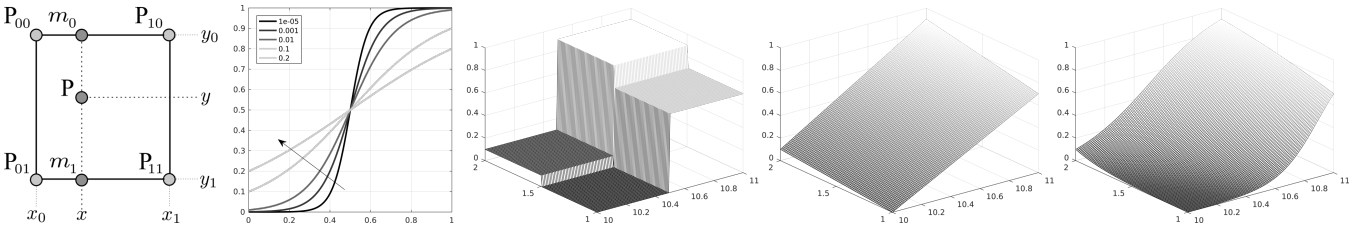


Fig. 3: From left to right: Interpolation point on a grid, odds interpolation for different ϵ_0, nearest neighbour, bilinear, and biodds interpolation

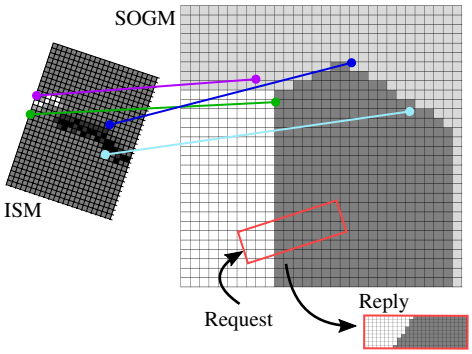


Fig. 4: Qualitative homography for a missaligned ISM into its corresponding SOGM and request of a sub map with different resolution.

Fig. 3 (left):

$$
\begin{aligned}
m_1 &= f\left(\frac{x-x_0}{x_1-x_0} f^{-1}(m(\mathrm{P}_{11})) + \frac{x_1-x}{x_1-x_0} f^{-1}(m(\mathrm{P}_{01}))\right) \\
m_0 &= f\left(\frac{x-x_0}{x_1-x_0} f^{-1}(m(\mathrm{P}_{10})) + \frac{x_1-x}{x_1-x_0} f^{-1}(m(\mathrm{P}_{00}))\right) \\
m(\mathrm{P}) &\approx f\left(\frac{y-y_0}{y_1-y_0} f^{-1}(m_1) + \frac{y_1-y}{y_1-y_0} f^{-1}(m_0)\right)
\end{aligned} \tag{5}
$$

Fig. 3 depicts the different interpolation models for a grid consisting of four cells. It is worth noticing that the two extrema $\epsilon_0 \to 0$ and $\epsilon_0 \to 0.5$ of the odds interpolation leads to either the nearest neighbor or the bilinear interpolation with respect to the boundaries (cf. Fig. 3).

2) Scan Matching: Scan matching is the process of aligning ISMs with each other or with an existing SOGM. This is necessary while ISMs can be produced in another coordinate frame than the one of the map, or the ISMs are misaligned due to poor odometry, registration, or ISM algorithm design. The first issue can be solved by knowing the registration between the map and the sensor frame and then applying an homography to the ISM. The latter one can be solved similar, but the missing transformation needs to be recovered. This is done by utilizing ORB feature extraction of the ISM and the corresponding SOGMs surrounding [22]. The features are matched via the distance of pairwise feature candidates which again leads to a homography that is depicted in Fig. 4.

D. Services

Services are defined, such that they process the requested type, pose, and resolution as shown in Fig. 4. Pose and resolution are features to facilitate requesting applications to outsource such post processing to the MS. Therefore, it is uniquely handled and benefits from the proposed interpolation technique from Section III-C.1.

1) Cell Wise Retrieval: Requesting preprocessed SOGMs is beneficial for applications only requiring one kind of information which can be derived from a set of SOGMs. Therefore, the authors implemented, besides the raw access of SOGMs, also the two following fusion techniques among layers: The first approach is based on a super Bayesian independent opinion pooling P_{B} [23]. It is applicable for the case when separate SOGMs with identical feature representations (e.g. set of maps for class "obstacle") are maintained. Second, a non-Bayesian fusion maximum pooling method P_{M} is applied to heterogeneous feature representations s(e.g. set of maps with varying classes). The fusion techniques are cell-wise and therefore do not introduce any clustering.

$$
P_{\mathrm{B}}(m) = \frac{1}{1 + \prod_N \frac{1-P(m_n)}{P(m_n)}}, \quad \tilde{P}_{\mathrm{M}}(m) = \max_n P(m_n) \tag{6}
$$

2) Superpixel Clustering: Unlike single layer OGM approaches, an SOGM incorporates multiple OGMs with varying classes residing in the map storage. For further applications, respecting every grid cell is not a feasible approach due to noise, sparse data, or offset between the layers. Even worse, high bandwidth would be necessary for requesting a raw SOGM. To transform the SOGM into a parametrized form by clustering e.g. via Gaussians is unfeasible as well, due to the risk of lacking objects. The authors' approach is therefore a superpixel-like clustering inspired by computer vision to find homogeneous regions and assigning a feature vector for these. This leads to a topometric map, which is derived from the centroids of the superpixels as shown in Fig. 6. Utilizing the Superpixels Extracted via Energy-Driven Sampling (SEEDS) algorithm from Van den Bergh [24] the authors revise the formulation

$$
H(s) = D(s) + \gamma G(s), \tag{7}
$$

to respect the nature, s.t. probability and locality of information of SOGMs more precisely. In Equation 7 s is the superpixel of interest, D is the color distribution term and G is the contour function which can be smoothed via the scalar

factor γ. As stated in the original paper a drawback of the color distribution term is the lack of respecting the variance of color values inside a superpixel. But especially for SOGM clustering applications, where data represents probabilities, this must be considered. For instance, superpixels consisting of contradicting values greater and less than 0.5 inside one layer should be avoided.

The authors introduce therefore the *Supercell Extracted via Variance-Driven Sampling* (SEVDS) algorithm, which substitutes the color distribution term of SEEDS by a variance driven formulation D'. The distribution term D' of a supercell c is defined as the sum of Eigenvalues e of the covariance matrix C of the probability histogram $h(c)$:

$$D'(c) = \sum_{n=1}^{N} e_n(\mathrm{var}(h(c))) \tag{8}$$

Each supercell c groups the cells $m \in c \subset M$ into a arbitrary but connected shape as depicted in Fig. 5. The probability range $[0, 1]$ is divided into K^N bins of size $1/K$ along every principal dimension. Thus, a bin $b_{k_1,k_2,\ldots,k_N} \in \mathbb{N}$ (with $k_n = 1, \ldots, K$) of a N dimensional histogram $h \in \mathbb{R}^{k \times k \ldots \times k}$ for one particular supercell c is calculated by

$$b_{k_1,\ldots,k_N} = \sum_{m \in c} \prod_{n=1}^{N} \Pi \left(\frac{m_n - \frac{1+2(k_n-1)}{2K}}{K} \right) \tag{9}$$

with m_n addressing a cell of the nth semantical layer and Π being the boxcar function. From that histogram, the covariance matrix $\mathrm{var}(h(c)) \in \mathbb{R}^{N \times N}$ is calculated. Further, from that covariance matrix, the N Eigenvalues e_n are calculated. Their sum is the quantifier of the distribution of occupancy probabilities in the corresponding supercell. This procedure becomes intractable with $\mathcal{O} = (K^2)^N$ for even a few semantical layers, but fortunately this derivation can be simplified twice so that the calculation becomes linear with $\mathcal{O} = KN$. First, the sum of Eigenvalues is equal to the trace and therefore no Eigenvalue decomposition needs to be performed. Second, while the covariances are no longer of interest due to the trace, the calculation of variances $\mathrm{var}(h(c_n)) \in \mathbb{R}$ of the marginal histogram is sufficient. The marginal histogram is nothing but the histogram of the a single semantic layer n in the supercell c_n (cf. Fig. 5). Finally, the distribution term can be simplified to

$$D'(c) = \sum_{n=1}^{N} e_n(\mathrm{var}(h(c))) = \sum_{n=1}^{N} \mathrm{var}(h(c_n)) \tag{10}$$

Using the variance of histograms as a distribution measurement becomes intuitive by taking the differences of probabilities in one supercell into account. Thus, the distribution of probabilities along the semantical layers is marginally respected. But with increasing contradictions along the spatial resolution the distribution term maximizes. Therefore, D' needs to be minimized to find the best supercell. This is implemented just like done by the SEEDS in a greedy fashion: For every supercell c an adjoining block c' with $\hat{c} = \{c_l, c'\}$ is taken into account for the distribution calculation.

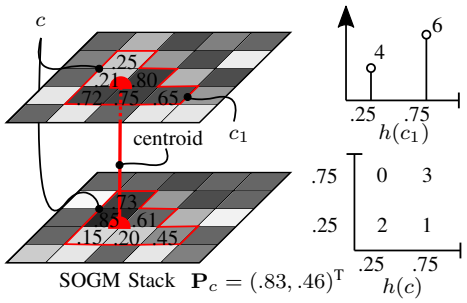


Fig. 5: Supercell with $N = 2$ layers and corresponding histograms with $K = 2$ bins.

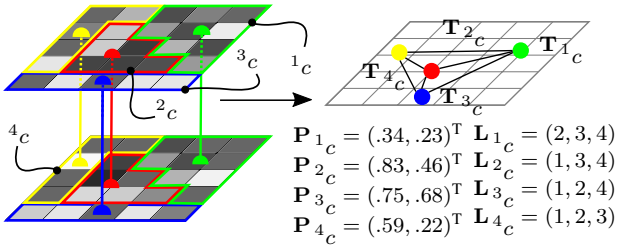


Fig. 6: Conversion of supercells to a graph of centroids labeled with feature vectors.

If $D(\hat{c}) < D(c)$ holds, the newly $\hat{c}$ will be used further. As depicted in Fig. 6, for every found supercell a tripel $\mathcal{C} = (\mathbf{T}_c, \mathbf{L}_c, \mathbf{P}_c)$ consisting of its centroid location $\mathbf{T}_c$, a list of adjunct supercell $\mathbf{L}_c$, and a feature vector $\mathbf{P}_c$ is calculated

$$\mathrm{Odd}\left(\mathbf{P}_c\right) = \left(\prod_{m \in c_1} \mathrm{Odd}(P(m)), \ldots, \prod_{m \in c_N} \mathrm{Odd}(P(m)) \right)^{\mathrm{T}} \tag{11}$$

Experiments revealed that supercells tend to grow uncoordinatedly. This was caused by neighbouring supercells comprising homogenous regions which leads to a total variance of zero. This can be explained by taking blocks of cells away from those zero-variance supercells their variance remains zero, while the other supercells' variance may decrease. This corner case can be circumvented by not allowing superpixels with a total variance of zero to shrink.

E. Mapping Framework Implementation in ROS

Fig. 7 depicts the composition of the mapping framework (MF) habitate which consists mainly of two components: The tile publisher keeps track of all necessary transforms of region of interest (ROI). The map server fuses incoming ISMs, offers services and debug messages.

An instance of the map server subscribes to all topics which needs to be mapped. For every topic a double buffer holding the current and last SOGM stack is allocated. To build a generic ISM interface, all messages have to have the ROS type `nav_msgs/OccupancyGrid`. It is worth mentioning that sensor fusion on the OGM level can directly be applied by letting different nodes publish their ISMs to the same topic. But further, one has to respect differing publishing frequencies and weightings which can distort a

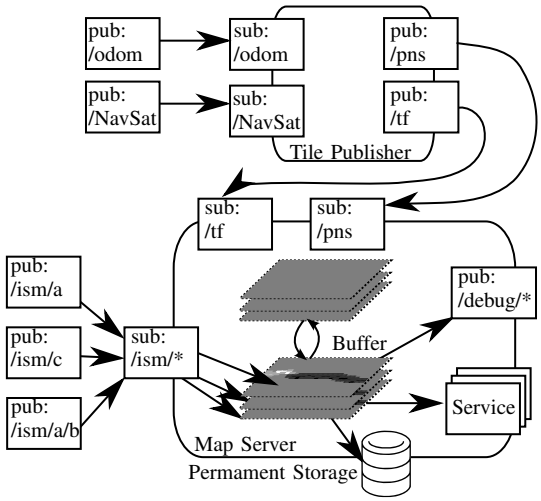


Fig. 7: Mapping framwork implementation

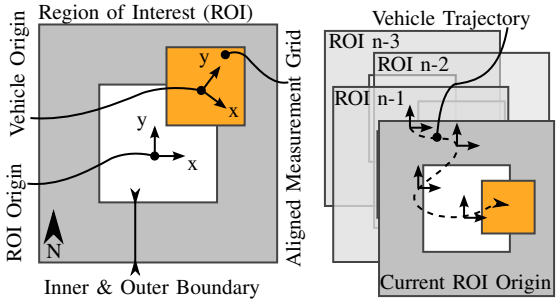


Fig. 8: 2D map centric mapping approach. The ROI comprises the horizontal and lateral ($H \times W$) dimesion of SOGM.

balanced fusion. While the amount of sensors measuring exteroceptive features is increasing and with them the derived ISMs, keeping track of all topics can be unfeasible. To overcome the unmanageable number of topics to which the map server has to subscribe to, a new scoping technique inspired by Wienke et al. [25] is introduced. It is build upon the common ROS pub./sub. model. Rather than subscribing to every single topic explicitly, the map server can be configured by a parent topic (e.g. `/ism`). All child topics (e.g. `/ism/a`, `/ism/a/b`, `/ism/c`) will be subscribed on demand after they have registered at the ROSMASTER.

In conjunction with the map server node, a tile publisher node keeps track of all map transformations depicted in Fig. 8. The tile publisher publishes transforms from the parent coordinate system to the origin of the current map tile. As the vehicle moves through the world, it will at some point exceed the current map dimension. Therefore, if an inner boundary dimension is passed, a new transform for a new map tile is published to ROS's tf tree. As proposed by *REP-105: Coordinate Frames for Mobile Platforms*, all sensor fusion has to be done in the local frame of a vehicle, because it is not affected by any discrete jumps caused by e.g. GPS. Therefore, all map tiles reside in the local frame, but are referenced in any global frame for later georeferencing or postprocessing. Just shifting the current map coordinate frame is not feasible, because discontinuities and jumps of frames are not well handled by ROS's tf API and therefore, new transforms with a new frame names are necessary. To inform the map server, first the transforms to the new map are published on the `/tf` scope. After a configurable delay a Pose/NavSat/String tuple (`/pns`) is published to inform about the current tile pose in the odometry frame, its NavSat datum for global localization, and a string containing the name of the current tile transform in ROS's tf tree. Finally, as the map server node receives a pns-tuple the double buffer of SOGMs is swapped so that the content can be independently stored in binary form, tagged with kind of content, resolution, and location to a permanent storage.

IV. Testcase and Evaluation

The proposed SEVDS is evaluated and compared against other superpixel algorithm implementations included in OpenCV 3.2.0. As metrics Intra-Cluster Variation, Explained Variation, Undersegmentation Error (UE) and Boundary Recall (BR) [24] are applied. Additionally, a pureness metric is defined as

$$g(s) = \frac{1}{|s|} \sum_{s_j} \frac{|\{m \in s_j | I(s) = \text{Label}(s_j)\}|}{|s_j|} \qquad (12)$$

with $\text{Label}(s_j) = \text{argmax}_i |\{s \in s_j | I(s) = i\}|$. The Pureness of a segmentation is defined by the average pureness of its superpixels. As proposed by Stutz et al., parameters were optimized such that $(1 - \text{BR}) + \text{UE}$ is minimized. Input data originates from simulating the Autonomous Mini Robot (AMiRo) via ROS and Gazebo in a table-top scenario inspired by RoboCup@Home competitions [26], [27] (cf. Fig. 9 top). The AMiRo maps its environment with known poses, a tilted laser rangefinder and a RGB camera using ISMs from [28]. The corresponding mapping for the enlarged test case scenario in Fig. 9 is depicted in Fig. 10. The composed SOGM is interpreted as multi-channel image and provides the input for all evaluated clustering algorithms presented in this work. A ground truth segmentation was provided by a human labeled segmentation (cf. Fig. 9 bottom). Fig. 12 to Fig. 16 compare the results of the algorithms using above-mentioned metrics for all four different scenarios (scenarios are depicted in different colors). In Fig. 11 the clustering is qualitatively depicted. Compared to SEEDS, the proposed SEVDS let the supercells not degenerate that much in object border regions and therefore provides more distinctive border segmentations. Other differences can barely be noticed. All algorithms achieve very similar Pureness. It should be noted that Pureness and Explained Variation are displayed with an offset to illustrate small differences in these metrics. SEVDS produces segmentations with noticeable, significant lower Intra-Cluster Variation (ICV), due to the fact that ICV's definition is very similar to the variance distribution term SEVDS aims to minimize. Explained Variation intents to measure boundary adherence independent of human annotated ground truth (higher is better). SEVDS's segmentation yields slightly lower Explained Variation. On average the Undersegmentation Error of SEVDS is 3.41

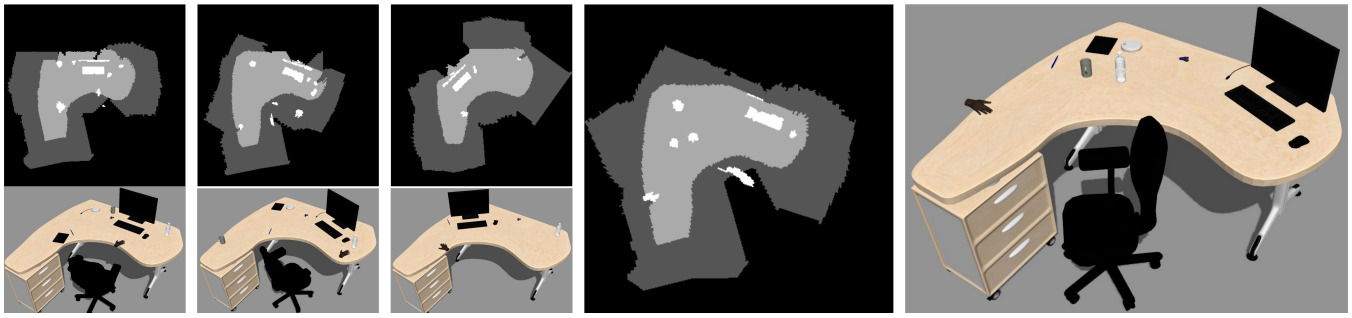


Fig. 9: Four randomized table-top scenarios used for recording maps and their corresponding ground truth map segmentation with four greyish shaded classes: unknown, floor, table, obstacle. Scenarios from left to right: first, second, third, forth.

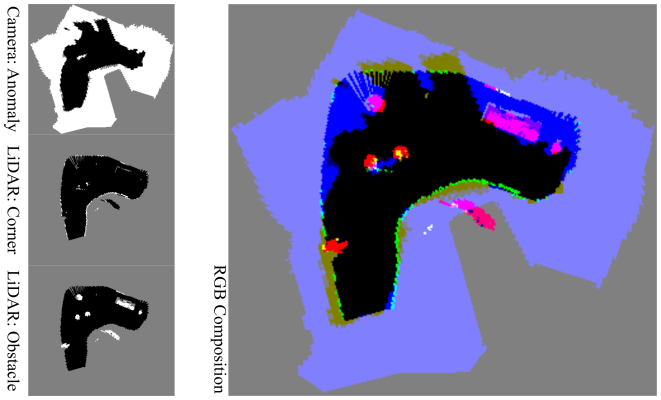


Fig. 10: Single SOGM layers produced by the "sensor: classification" set (left) for the fourth example in Fig. 9 and the composition in RGB color space that depicts the $|\{\text{occ.}, \overline{\text{occ.}}, \text{unknown}\}|^{\#\text{layer}} = 27$ possible classes.

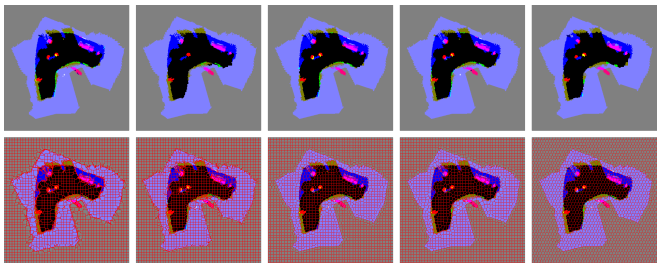


Fig. 11: Clustering of SOGM resulting from the fourth scenario (c.f. Fig. 10). From left to right: SEEDS, SEVDS, LSC, SLIC, SLICO. Top: Coloring of clusters with class majority. Bottom: Corresponding cluster segments in red.

times higher than SEEDS's. But it should be noted that the general power of all Undersegmentation Errors is pretty low (10^{-2}) and that this metric substantially depends on the provided ground truth. SEVDS achieves slightly lower Boundary Recall compared to SEEDS, but better results than the other algorithms. To conclude, SEVDS generates low Intra-Cluster Variation superpixels, with state-of-the-art Pureness and Explained Variation, but improvable boundary adherence. Anyway, this is an outstanding result which is beneficial for the author's definition on topometric features from Equation 11. Having low variation means that the derived feature vectors are highly separable. On the other hand, having low variation would result in resembling features among classes which would be hardly discriminable. A noteworthy implementation detail is, that these clustering algorithms now work on grids which have a spatial relation. Therefore, as a rule of thumb the minimal initial superpixel edge length should be of the size of smallest object to be found (s.t. $8\,\text{cm} \mathrel{\widehat{=}} 8\,\text{pixel}$ for the current evaluation).

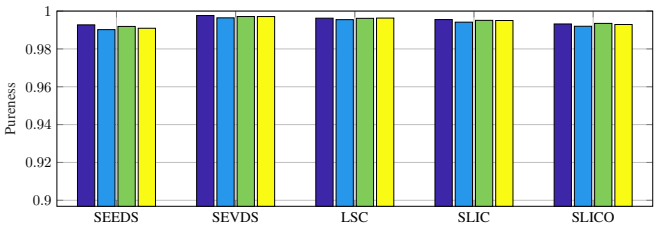


Fig. 12: Pureness

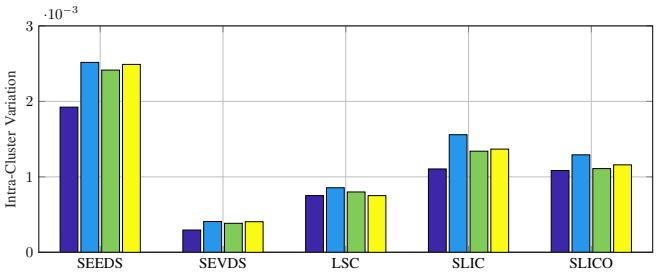


Fig. 13: Intra Cluster Variation

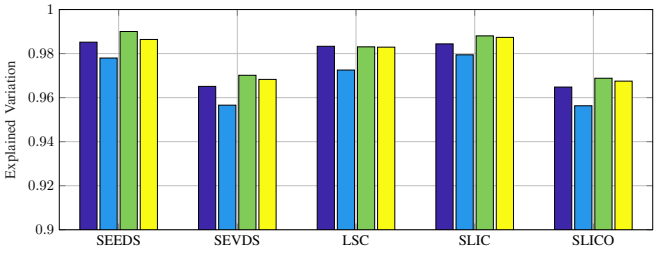


Fig. 14: Explained Variation

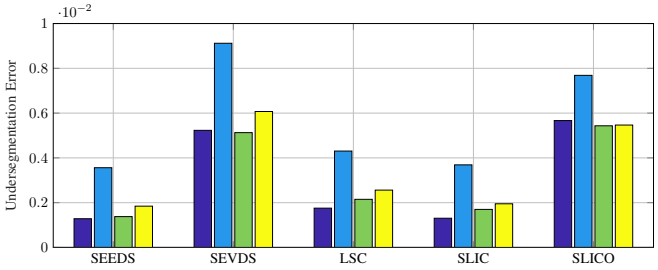


Fig. 15: Undersegmentation Error

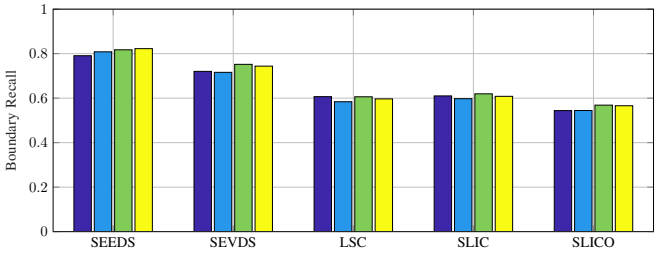


Fig. 16: Boundary Recall

V. Conclusion and Outlook

The authors present a complete semantical occupancy grid mapping framework (MF) which offers new possibilities of mapping sensor readings on a decision level in a generic way. Further, a new interpolation method has been proposed to incorporate sensor readings in a Bayesian way. Finally, services are introduced which refine multi layered, semantic maps to a single layer, or via the newly proposed SEVDS algorithm to a topometric map. Thus, this MF facilitates the possibility to handle SOGMs by standard navigation techniques. Ongoing investigations will exploit the proposed MF in dynamic agricultural and multi robotic applications to evaluate its mapping capability and quality. Further research will concentrate on applying optimized navigation techniques on SOGMs. Finally, it is intended to publish the MF as a ROS package on `https://opensource.cit-ec.de/`.

Acknowledgment

This research/work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG) and by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL) and managed by the Project Management Agency Karlsruhe (PTKA).

References

[1] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985.

[2] A. Elfes, "Dynamic control of robot perception using multi-property inference grids," 1992.

[3] ——, "Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework," *Lecture Notes in Computer Science*, vol. 1093, pp. 93–130, 1996.

[4] S. Thrun, "Robotic Mapping: A Survey," *Science*, vol. 298, no. February, pp. 1–35, 2002.

[5] H. Winner, *Handbuch Fahrerassistenzsysteme - Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Wiesbaden: Vieweg+Teubner Verlag, 2015.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Mass.: MIT Press, 2005.

[7] A. Nüchter, "Semantische dreidimensionale Karten für autonome mobile Roboter," Ph.D. dissertation, University of Bonn, 2006.

[8] A. Koubaa, *Robot Operating System (ROS): The Complete Reference*. Springer International Publishing, 2016, vol. 1, no. 1.

[9] E. Marder-Eppstein, D. V. Lu, and D. Hershberg, "costmap_2d." [Online]. Available: http://wiki.ros.org/costmap{_}2d

[10] KARTO, "KARTO," 2017. [Online]. Available: https://www.kartorobotics.com/

[11] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, pp. 155–160, 2011.

[12] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[13] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.

[14] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stöckler, and S. Behnke, "Multilayered Mapping and Navigation for Autonomous Micro Aerial Vehicles," *Journal of Field Robotics*, vol. 33, no. 4, 2016.

[15] T. Morris, F. Dayoub, P. Corke, G. Wyeth, and B. Upcroft, "Multiple map hypotheses for planning and navigating in non-stationary environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2765–2770, 2014.

[16] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," *Studies in Computational Intelligence*, vol. 625, pp. 99–120, 2016.

[17] A. Elfes, "Occupancy Grids: A Stochastical Spatial Representation for Active Robot Perception," in *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 1990.

[18] D. Hähnel, "Mapping with Mobile Robots," Ph.D. dissertation, University of Freiburg, 2004.

[19] M. E. Liggins, D. L. Hall, and D. Llinas, *Handbook of multisensor data fusion*, 2001.

[20] C. Stachniss, "Robot Mapping Features vs . Volumetric Maps Grid Maps Example."

[21] M. Kragh, P. Christiansen, T. Korthals, T. Jungeblut, H. Karstoft, and R. N. Jørgensen, "Multi-Modal Obstacle Detection and Evaluation of Occupancy Grid Mapping in Agriculture," in *International Conference on Agricultural Engineering*, Aarhus, 2016.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571.

[23] K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger, "3D Forward sensor modeling and application to occupancy grid based sensor fusion," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 2059–2064, 2007.

[24] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: An Evaluation of the State-of-the-Art," *CoRR*, vol. abs/1612.0, 2016.

[25] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," *2011 IEEE/SICE International Symposium on System Integration (SII)*, pp. 1183–1190, dec 2011.

[26] S. Herbrechtsmeier, T. Korthals, T. Schöpping, and U. Rückert, "AMiRo: A Modular & Customizable Open-Source Mini Robot Platform," in *ICSTCC*, 2016.

[27] S. Meyer zu Borgsen, T. Korthals, F. Lier, and S. Wachsmuth, "ToBI Team of Bielefeld: Enhancing Robot Behaviors and the Role of Multi-Robotics in RoboCup@Home," 2016.

[28] T. Korthals, T. Krause, and U. Rückert, "Evidence Grid Based Information Fusion for Semantic Classifiers in Dynamic Sensor Networks," *Machine Learning for Cyber Physical Systems*, vol. 1, no. 1, p. 6, 2015.

Consistency of Feature-based Random-set Monte-Carlo Localization

Manuel Stübler, Stephan Reuter and Klaus Dietmayer

Abstract— **Self-localization is one of the most critical parts in robotics and automated driving. Thus, it is quite essential to have some kind of self-assessment for the respective pose estimate. Therefore, this paper introduces a new online approach to check the consistency of feature-based random-set Monte-Carlo Localization (MCL). The basic idea is to detect inconsistencies of the assumed measurement process in a stochastic manner in order to infer validity of the localization result. This concept is closely linked to the Normalized Innovation Squared (NIS) in Kalman filtering techniques. The problem of checking the consistency online, in absence of ground-truth data, is formulated using confidence intervals for the estimated measurement model parameters. In contrast to the single-object Kalman filter, multi-object filters not only consider the spatial uncertainty of sensor measurements, but also the clutter rate and missed detections. Thus, all those measurement model parameters need to be observed and checked for consistency. The proposed concept is applied to a random-set formulation of MCL that is formally derived in the present contribution. The evaluation is done using real-world data from a test vehicle in a scenario that covers public urban and rural roads.**

I. Introduction

Robust localization is a key prerequisite for automated cars and robots. Besides Global Navigation Satellite System (GNSS), map based localization methods have evolved in the last decades and lots of them are formulated in a Bayesian manner. For example, Sequential Monte-Carlo (SMC) methods represent the Probability Density Function (PDF) of a robot's pose by a set of weighted particles, conditioned on a known map of the environment. This method is also known as Monte-Carlo Localization (MCL) and the respective map may incorporate a dense representation of the environment, e.g. grid-based methods [1], or a set of features [2]. In the case of GNSS, the term integrity is often used to express the trust that can be put into the result of a system [3]. A common framework in the GNSS domain is called Receiver Autonomous Integrity Monitoring (RAIM) [4] which is based on statistical tests and redundant information sources. Likewise, the present contribution proposes a reliable trust value for feature-based random-set MCL. There are other techniques that address the integrity of map matching algorithms, e.g. empirically derived [5] or statistically motivated methods [6]. However, they are either formulated in a heuristic manner or do not consider the full multi-object likelihood. In general, statistical derivations typically incorporate the NIS which is used especially for Kalman filters [7]. An extension of the NIS for the multi-object case is presented in [8], [9]. In the present contribution, this idea is adapted to feature-based random-set MCL. This is done by regarding not only the spatial uncertainty of a sensor, but also the clutter rate and detection probability in a stochastic manner.

Manuel Stübler, Stephan Reuter and Klaus Dietmayer are with driveU / Institute of Measurement, Control and Microtechnology, Ulm University, 89081 Ulm, Germany. E-mail: {firstname.lastname}@uni-ulm.de

In contrast to Simultaneous Localization and Mapping (SLAM) algorithms, MCL is based on a known a priori map. However, maps may degrade over time and contain inaccuracies which lead to inconsistent localization results. This contribution thus introduces a method to detect such inconsistencies by estimating the measurement model parameters online and stochastically validate them. This validation is performed by detecting a deviation from the expected parameters. With the assumption that those parameters do not vary over time, they are determined by the sensor and scenario. Both should be known in advance. Scenario, in this case, includes all additional influencing factors on the measurement model parameters, e.g., weather condition or road type. If the estimated parameters diverge from the assumed ones, the localization result and especially the corresponding uncertainty cannot be trusted anymore. This is due to the fact that the stochastic prerequisites are no longer valid. Note that the opposite is not true: it is possible that the localization result is perfectly consistent while all measurement model parameters deviate from the assumptions.

A few examples on the cause of a deviation between estimated and assumed parameters: assuming that the map is outdated, the estimated clutter rate will raise since landmarks are detected by the sensor that are actually not present in the map. Those detections are then falsely regarded as clutter measurements. The same, however, happens if the sensor itself is faulty and produces more clutter measurements than it typically does. Another example is the presence of landmarks in the map that actually disappeared in the real world. This results in a lower estimate for the detection probability than assumed. Again, this can also be caused by a faulty sensor. Further, if the localization result has a slight offset, this results in a bias of the corresponding estimated spatial uncertainty. This happens because landmarks from the global a priori map are transformed into the local sensor coordinate system of the respective particle based on its pose hypothesis. But just like in the cases before, the sensor can be the cause for such a bias, too. In general, thus, it is not possible to determine the source for an inconsistency since there may be a lot of different reasons that can cause it.

The subsequent parts of this contribution are organized as follows: the mathematical foundation is given in Section II and the formulation of the consistency check for the multi-

object measurement model is presented in Section III. In Section IV an evaluation with real-world data is shown and finally Section V provides a conclusion.

II. Basics

This section provides a brief introduction into the relevant part of the multi-object calculus. First the landmark map and measurement representation as Random Finite Sets (RFSs) is introduced. Further, the random-set formulation of MCL and most important the corresponding measurement model is described.

A. Landmark Representation

By modeling the map of landmarks and also the measurements as RFSs following [10], an artificial ordering of landmarks is avoided. This results in an absence of explicit data association and further facilitates the modeling of missed detections and clutter. The map of landmarks is thus given by

$$\mathrm{M} = \{\underline{m}^{(1)}, \ldots, \underline{m}^{(\nu)}\} \tag{1}$$

where the state vector $\underline{m}$ represents the landmark position and ν is the total number of landmarks in the map. The set of measurements with cardinality μ_k at time step k is

$$\mathrm{Z}_k = \{\underline{z}_k^{(1)}, \ldots, \underline{z}_k^{(\mu_k)}\} \tag{2}$$

and all measurements up to time step k are denoted by Z^k.

B. Monte-Carlo Localization

With Bayes theorem and the multi-object calculus, the PDF of the current pose $\underline{x}_k$ conditioned on the map and measurements is written as [11]

$$p(\underline{x}_k|\mathrm{Z}^k, \mathrm{M}) = \frac{\int g(\mathrm{Z}_k|\underline{x}_k, \mathrm{M}) f(\underline{x}_k|\underline{x}_{k-1}) p(\underline{x}_{k-1}|\mathrm{Z}^{k-1})\, d\underline{x}_{k-1}}{\iint g(\mathrm{Z}_k|\underline{x}_k, \mathrm{M}) f(\underline{x}_k|\underline{x}_{k-1}) p(\underline{x}_{k-1}|\mathrm{Z}^{k-1})\, d\underline{x}_{k-1} d\underline{x}_k}. \tag{3}$$

The denominator is a normalization constant, $g(\cdot|\cdot)$ is the multi-object likelihood and $f(\cdot|\cdot)$ is the transition kernel based on the process model. It is also possible to add an external control input u_k to the above equation. This can, e.g., be used for a dead reckoning process model by additionally incorporating velocity and yaw rate measurements. Equation (3) can be solved using SMC methods analogous to [12], [13] where each sample - or particle - represents a hypothetical robot pose. Using the Markov property, a set of particles is propagated in time recursively which is known as *Bootstrap Filtering* or *Sequential Importance (Re-) Sampling* [14], [15].

C. Multi-object Likelihood

In order to apply SMC algorithms for the RFS formulation, the set-based measurement likelihood $g(\mathrm{Z}_k|\underline{x}_k, \mathrm{M})$ must be evaluated. This is done for a set of measurements Z_k conditioned on the current pose $\underline{x}_k$ of the respective particle and the map M. In RFS theory, it is typically assumed that a sensor detects landmarks with a probability $p_D(\cdot)$ and additionally generates Poisson distributed clutter with intensity $\kappa(\cdot)$ and mean λ_C (c.f. [16]). The expected measurement of a global landmark in the local sensor coordinate system is given by

$$\widehat{\underline{m}}_k^{(i)} \triangleq h(\underline{m}_k^{(i)}, \underline{x}_k) \tag{4}$$

with the measurement function $h(\cdot)$ and the global landmark position $\underline{m}_k^{(i)}$. The hypothetical pose $\underline{x}_k$ of a particle is necessary to transform the global landmark position into the respective local sensor coordinate system. The measurement likelihood is then evaluated in the local sensor coordinate system. The multi-object measurement likelihood itself is defined by [16, Ch. 12]

$$g(\mathrm{Z}_k|\underline{x}_k, \mathrm{M}) = \pi_C(\mathrm{Z}_k)\pi(\emptyset|\mathrm{M}) \times \sum_{\theta} \prod_{i:\theta(i)>0} \frac{p_D(\widehat{\underline{m}}_k^{(i)}) g(\underline{z}_k^{(\theta(i))}|\widehat{\underline{m}}_k^{(i)})}{(1 - p_D(\widehat{\underline{m}}_k^{(i)}))\kappa(\underline{z}_k^{(\theta(i))})}. \tag{5}$$

The probability that all measurements are clutter is

$$\pi_C(\mathrm{Z}_k) = e^{-\lambda_C} \prod_{\underline{z}_k \in \mathrm{Z}_k} \kappa(\underline{z}_k) \tag{6}$$

and the probability that all landmarks are not detected is

$$\pi(\emptyset|\mathrm{M}) = \prod_{i=1}^{\nu} (1 - p_D(\widehat{\underline{m}}_k^{(i)})). \tag{7}$$

The single-object measurement likelihood is typically assumed to follow a Gaussian distribution with

$$g(\underline{z}_k^{(j)}|\widehat{\underline{m}}_k^{(i)}) = \mathcal{N}(\underline{z}_k^{(j)}; \widehat{\underline{m}}_k^{(i)}, \underline{\Sigma}_k^{(i,j)}). \tag{8}$$

The covariance matrix $\underline{\Sigma}_k^{(i,j)}$ follows the respective sensor properties and optionally incorporates the uncertainty of a map landmark. Finally, the association functions θ describes an assignment of measurements to landmarks. It incorporates missed detections (i.e. $\theta(i) = 0$), clutter measurements (i.e. $\theta^{-1}(j)$ is undefined) and furthermore ensures that a measurement cannot be assigned to more than one landmark (i.e. $\theta(i_1) = \theta(i_2) > 0 \Rightarrow i_1 = i_2$). Equation (5) states that the sum is taken over all possible associations θ. This of course is impractical and a reasonably approximation is to evaluate only the most probable associations and abort when a summand adds no more significant value to the likelihood. This is achieved, e.g., by using a ranked assignment algorithm [17] or Gibbs sampling methods [18].

III. Consistency Check

When implementing random-set MCL algorithms, the parameters of the multi-object likelihood must be known in advance. Even though clutter rate and detection probability can be estimated alongside the multi-object state following [19], [20], in the present contribution it is assumed that those parameters do not vary over time. The measurement model is assumed to depend only on the sensor properties, feature extraction process and scenario which are all assumed to be fixed and known in advance. Of course one could always model time-varying parameters and adapt them online, but then there is no guaranty that the parameters which were

learned are correct. Hence, the idea here is to use this redundant information (assumed parameters versus estimated ones) exactly for the validation process. Therefore, a deviation of the estimated measurement model parameters is assumed to be the result of a divergent a priori map, a defective sensor, a biased pose estimate or a different scenario than expected. This inevitably leads to a pose estimate that cannot be trusted anymore since the stochastic prerequisites are violated.

The following subsections derive confidence intervals for all three measurement model parameters of the multi-object likelihood: the spatial uncertainty of landmark measurements, the state-independent detection probability and the clutter rate. These assumptions are then tested individually for consistency. The derivation of the consistency tests is done by assuming that both the true association θ_k and the true pose $\underline{x}_k$ at time step k are known. The knowledge of any ground-truth data, however, is not a prerequisite for the final algorithm which is explained in detail later.

A. Spatial Uncertainty

As described earlier, the spatial uncertainty of landmark measurements is typically assumed to follow a Gaussian distribution with known covariance matrix $\underline{\Sigma}^{(i,j)}$. If the true pose $\underline{x}_k$ and true association θ_k between measurements and landmarks are known and if the measurement noise is uncorrelated, then

$$\xi_k^S \triangleq \sum_{i:\theta_k(i)>0} \left(\left(\underline{\gamma}_k^{(i)}\right)^{\mathrm{T}} \cdot \left(\underline{\Sigma}^{(i,\theta_k(i))}\right)^{-1} \cdot \left(\underline{\gamma}_k^{(i)}\right) \right) \tag{9}$$

with the residual

$$\underline{\gamma}_k^{(i)} \triangleq \left(\underline{z}_k^{(\theta_k(i))} - h(\underline{m}_k^{(i)}, \underline{x}_k) \right) \tag{10}$$

follows a χ^2 distribution with $d \cdot \nu_k^D$ degrees of freedom (c.f. also [9]). Here, d is the dimension of the measurement space and ν_k^D is the number of detected landmarks. The two-sided confidence interval with a significance level of α for the respective χ^2 distribution is then given by

$$P\left\{\xi_k^S \in [q_k^-, q_k^+]\right\} = 1 - \alpha \tag{11}$$

with lower and upper quantiles

$$q_k^- = \left\{ x : \int_0^x \chi^2(t; d \cdot \nu_k^D)\, dt = \frac{\alpha}{2} \right\}, \tag{12}$$

$$q_k^+ = \left\{ x : \int_0^x \chi^2(t; d \cdot \nu_k^D)\, dt = 1 - \frac{\alpha}{2} \right\}. \tag{13}$$

In contrast to the NIS provided in [7], the innovation in Equation (9) is calculated for every landmark with an associated measurement and accordingly summed up.

B. Detection Probability

Assuming that the detection probability p_D is state-independent, it follows a Bernoulli distribution with expectation p_D and variance $p_D(1-p_D)$. Thus, using the *central limit theorem* it is

$$\xi_k^D \triangleq \sqrt{\nu_k} \frac{\frac{\nu_k^D}{\nu_k} - p_D}{\sqrt{p_D(1-p_D)}} \to \mathcal{N}(0,1) \tag{14}$$



Fig. 1. Overview of the test track in Ulm with laser scanner landmarks. Both evaluation recordings, which are independent of the mapping datasets, start in the lower left corner and were driven clock-wise.[1]

for $\nu_k \to \infty$ where ν_k denotes the number of all landmarks in the sensor's Field of View (FOV). This, however, only works reliably for a large number of visible landmarks in each time step k. An asymptotic and symmetric two-sided confidence interval is then constructed for the corresponding Gaussian distribution as follows:

$$\lim_{k\to\infty} P\left\{\xi_k^D \in [-q, +q]\right\} = 1 - \alpha, \tag{15}$$

$$q = \left\{ x : \int_{-\infty}^{x} \mathcal{N}(t; 0, 1)\, dt = 1 - \frac{\alpha}{2} \right\}. \tag{16}$$

Since the parameters are assumed to be time invariant, the estimation can also be done using w consecutive time steps. This procedure is especially helpful for a small number of visible landmarks in each time step. With $l = k - w + 1 \geq 0$ it is:

$$\xi_{l:k}^D \triangleq \sqrt{\nu_{l:k}} \frac{\frac{\nu_{l:k}^D}{\nu_{l:k}} - p_D}{\sqrt{p_D(1-p_D)}} \to \mathcal{N}(0,1) \tag{17}$$

for $\nu_{l:k} \to \infty$, with $\nu_{l:k} \triangleq \sum_{i=l}^{k} \nu_i$ and $\nu_{l:k}^D \triangleq \sum_{i=l}^{k} \nu_i^D$. The same trick can be done analogously for the spatial uncertainty in the previous section to incorporate several time steps.

It is also possible to calculate confidence intervals for the Bernoulli distribution without the need for an approximation using the *central limit theorem*, e.g. based on the Clopper-Pearson interval [21]. However, a preliminary evaluation showed no significant difference between both approaches.

C. Clutter Rate

The multi-object likelihood is often formulated with the assumption that clutter is distributed uniformly within the FOV. The clutter intensity $\kappa(\underline{z}_k)$ is thus defined by the Poisson distributed clutter rate λ_C and the area (or volume) of the sensor's FOV:

$$\kappa(\underline{z}_k) = \begin{cases} \frac{\lambda_C}{\mathrm{Vol(FOV)}} & \text{if } z_k \in \text{FOV}, \\ 0 & \text{else.} \end{cases} \tag{18}$$

[1] Aerial photography: © Stadt Ulm, Abteilung Vermessung, 2010.

For a known association, the clutter rate at time step k is denoted by $\lambda_k = \mu_k - \nu_k^D$, where μ_k is the number of measurements at time step k and again ν_k^D is the number of landmarks with an associated measurement.

Using the *central limit theorem* and the properties of the Poisson distribution, i.e. mean and variance are both λ_C, then for a time window of size $w \to \infty$ it is:

$$\xi_{l:k}^C \triangleq \sqrt{w}\frac{\bar{\lambda}_{l:k} - \lambda_C}{\sqrt{\lambda_C}} \to \mathcal{N}(0,1) \tag{19}$$

with $\bar{\lambda}_{l:k} = \frac{1}{w}\sum_{i=l}^{k}\lambda_k$. Similar to Section III-B and with q stated in Equation (16), an asymptotic confidence interval is constructed as follows:

$$\lim_{k\to\infty} P\left\{\xi_{l:k}^C \in [-q, +q]\right\} = 1 - \alpha. \tag{20}$$

In contrast to the spatial uncertainty and the detection probability, the clutter rate provides only a single estimate λ_k per time step k. Therefore, it necessarily needs to be regarded over several time steps to be adequately approximated by a Gaussian.

D. Online Estimation

In order to check the consistency of all three measurement model parameters as described previously, they first have to be estimated following Equations (9), (14) and (19). The estimation of those parameters, however, was achieved by using the true pose $\underline{x}_k$ and true association θ_k. In a real-world scenario both are unknown and therefore must be estimated themselves.

1) Pose: With the assumption that the prediction error of the transition between two time-steps is negligible, the true pose is approximated by the mean of the predicted PDF. When using a particle filter, this is done by calculating the weighted mean of all particles. Note that an offset in the prediction step that is not negligible compared to the spatial uncertainty of landmark measurements will result in a bias of landmark measurements, c.f. Equation (4). This bias will probably cause the consistency test for the spatial uncertainty to fail. Thus, an offset in the prediction step that is comparably large, e.g. a drift of the odometry, is detected by the proposed method just like an offset regarding the spatial landmark measurements of the sensor itself.

2) Association: Estimating the unknown association can be achieved by several approaches: if there is one dominating association in Equation (5), this association also contributes the most to the multi-object likelihood and thus to the localization result. In this case, all other associations are negligible which, e.g., happens for an unambiguous landmark-to-measurement constellation. Another approach is to separate the estimation of the association function θ_k from the multi-object likelihood. The association is then estimated by solving the equation

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{\nu} d_c(\underline{m}^{(i)}, \underline{z}_k^{(\theta(i))}) \tag{21}$$

where $d_c(\underline{m}, \underline{z}) = \min\{c, d(\underline{m}, \underline{z})\}$ is a distance measure with cut-off parameter c.

This approach is similar to the optimal assignment problem stated in [22]. Equation (21) determines the assignment with the minimum overall distance between measurements and landmarks in the sensor's FOV. The cut-off parameter c works as follows: if there is no measurement within the radius c of a landmark, then it is regarded as a missed detection. Further, if a measurement is not within the radius of any landmark it is considered a clutter measurement.

The estimation of θ as stated in Equation (21) provides some logical separation from the filtering process itself. It is especially separated from the RFS formulation in Equation (5) which considers every possibility and thus avoids the idea of having a single, true landmark-to-measurement association. Further, this approach is particularly useful when a measurement model is created for a completely new sensor or scenario and if no a priori information is available so far. It makes sense, if landmarks are quite distinct and the consistency checking is carried out over several consecutive time steps. This reduces the impact of a false association in a single time step. Besides, a false association is typically more likely to violate at least one of the consistency tests than to support all of them. Hence, the estimation of θ following Equation (21) is also used for the evaluation provided in Section IV based on the Munkres assignment algorithm [23].

IV. Evaluation

A. Setup

The evaluation is done based on real-world data from a Mercedes-Benz E-Class (S212) equipped with a front-facing laser scanner and a Real-time Kinematic (RTK) system. Additionally, the on-board wheel and yaw rate sensors are used to predict the ego vehicle in time based on a dead reckoning transition model. The route is shown in Figure 1. It is approximately 3 km long and divided into an urban and a rural part. Landmarks are extracted by clustering the raw points of a laser scan using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [24] with a distance threshold of 1.5 m and a minimum of 3 and maximum of 8 points per cluster. Next, the corresponding 2D mean and covariance matrix is calculated for each cluster. The centroids of clusters are then used as a landmark measurement if the smaller eigenvalue of the covariance matrix is below 0.05 and the larger eigenvalue is below 0.1. This procedure discards large and connected clusters but preserves those that are quite compact, distinct and free-standing. The choice of parameters was done completely empirically. Even though this extraction process is quite simple, it works very well due to the robust probabilistic formulation that correctly captures detection probability, clutter rate and spatial uncertainty of measurements. The a priori map that is used for the MCL consists of 310 landmarks and was built from several recordings in summer using the RTK system. The evaluation itself is done with two separate datasets that were recorded in winter time. The sample windows size was set to incorporate $w = 12$ consecutive time steps and the sensor has a frequency of $f = 12.5$ Hz. This results in a time window of about one second. If the time window is too small, the approximation

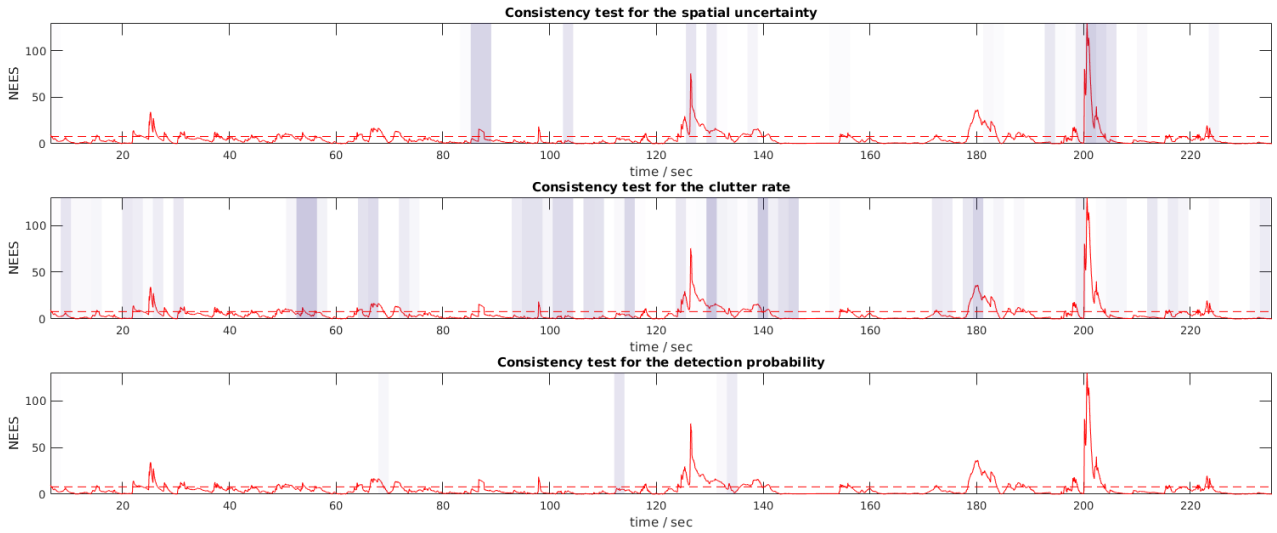


Fig. 2. Evaluation showing an exemplary run. All three parts of the plot show the NEES of the position and orientation error. The dashed horizontal line represents the respective 95% confidence interval. The gray background in a plot indicates that the consistency test of the corresponding parameter lies outside of its respective confidence interval. A darker background means that this happened more often in the corresponding time window.

using the *central limit theorem* is not reasonable. However, if the time interval is too long, the reaction time of the consistency tests is unnecessarily protracted.

B. Results

The proposed method is evaluated by using ground-truth position data from the RTK reference system. The random-set MCL algorithm and also the corresponding measurement model consistency tests themselves do not incorporate any ground-truth information and thus are performed online.

Figure 2 shows an exemplary run that was processed with one of the two evaluation recordings. The three plots each show the Normalized Estimation Error Squared (NEES) in red which actually corresponds to the squared Mahalanobis distance and is calculated as follows:

$$\varepsilon_k = (\underline{\hat{x}}_k - \underline{\bar{x}}_k)^{\mathrm{T}} (\underline{\mathrm{P}}_k + \underline{\mathrm{R}})^{-1} (\underline{\hat{x}}_k - \underline{\bar{x}}_k), \qquad (22)$$

where $\underline{\hat{x}}_k$ is the weighted mean of the particle set, $\underline{\bar{x}}_k$ is the reference position from the RTK system, $\underline{\mathrm{P}}_k$ is the weighted covariance matrix representing the particle set uncertainty and $\underline{\mathrm{R}}$ is the time-invariant covariance matrix of the RTK system. In the best case, which is assumed for this evaluation, the standard deviation of the reference system is 2 cm and 0.1 deg respectively. Note that the NEES is calculated by approximating the particle set with a Gaussian distribution $\mathcal{N}(\underline{\hat{x}}_k, \underline{\mathrm{P}}_k)$ comprising the three states x_k, y_k and ϕ_k. Thus, if the localization result regarding its mean and uncertainty estimate is consistent with the ground-truth data, Equation (22) follows a χ^2 distribution with three degrees of freedom (c.f. [7]). This is similar to the spatial uncertainty described in Section III-A. The NEES is not available in an online scenario since it needs the true position $\underline{\bar{x}}_k$ from a reference system. It is thus only used to evaluate the proposed method which is performed online by using only the predicted mean of the particle filter and the association estimate as described in Equation (21). A dark background in Figure 2 means that the consistency test of the respective measurement model parameter lies outside of its confidence interval. The darker the background the more often this happened in the corresponding time window. However, only the upper boundary of the spatial uncertainty and the clutter rate as well as the lower boundary of the detection probability is considered here (c.f. Table I). The reason is that only those parts are of interest where the assumed measurement process is worse than expected because we also only consider the one-sided confidence interval of the NEES.

TABLE I

RESULTS OF TWO INDEPENDENT RECORDINGS WITH FIVE MONTE-CARLO RUNS EACH, SHOWING THE PERCENTAGE OF OUTLIERS (o) AND THE AVERAGE NORMALIZED ESTIMATION ERROR SQUARED ($\bar{\varepsilon}$)

	Rec1	Rec2
B : Baseline	$o = 20\% \mid \bar{\varepsilon} = 5.31$	$o = 16\% \mid \bar{\varepsilon} = 4.65$
$S : \xi^S_{l:k} < q^+_{l:k}$	$o = 18\% \mid \bar{\varepsilon} = 4.61$	$o = 14\% \mid \bar{\varepsilon} = 3.89$
$C : \xi^C_{l:k} < q$	$o = 18\% \mid \bar{\varepsilon} = 5.00$	$o = 15\% \mid \bar{\varepsilon} = 4.41$
$D : \xi^D_{l:k} > -q$	$o = 20\% \mid \bar{\varepsilon} = 5.30$	$o = 15\% \mid \bar{\varepsilon} = 4.61$
S & C & D	$o = 15\% \mid \bar{\varepsilon} = 4.16$	$o = 12\% \mid \bar{\varepsilon} = 3.58$

In the following, a confidence level of 5% is assumed for both, the NEES test which is performed offline with ground-truth data as a reference and the measurement model parameter tests which are performed online without ground-truth data. The evaluation itself is shown in Table I. The percentage of outliers is the relative amount of NEES values that lie beyond the one-sided 95% confidence interval which is denoted by the red dashed line in Figure 2. It is expected to be exactly $o = 5\%$ for a perfectly consistent Gaussian pose estimate of the particle filter. The average NEES in this scenario should be $\bar{\varepsilon} = \frac{1}{n}\sum_{k=1}^{n} \varepsilon_k = 3$ due to the 3D state vector. Besides the fact that the particle distribution will

TABLE II
ROOT MEAN SQUARE ERROR (RMSE) IN VEHICLE COORDINATES (VC) WITH FIVE MONTE-CARLO RUNS PER RECORDING

Rec1	Rec2
$\text{RMSE}(x_{\text{vc}}) = 0.10\,\text{m}$	$\text{RMSE}(x_{\text{vc}}) = 0.11\,\text{m}$
$\text{RMSE}(y_{\text{vc}}) = 0.14\,\text{m}$	$\text{RMSE}(y_{\text{vc}}) = 0.14\,\text{m}$
$\text{RMSE}(\phi_{\text{vc}}) = 0.38\,\text{deg}$	$\text{RMSE}(\phi_{\text{vc}}) = 0.39\,\text{deg}$

never perfectly match a Gaussian distribution, the uncertainty that is assumed for the reference system also has a huge impact on the NEES. Since the best case for the reference system was assumed, it is clear that the optimal absolute values $o = 5\%$ and $\bar{\varepsilon} = 3$ are unlikely to be reached. However, the interesting part here is the relative behavior between the baseline and the respective subsets as shown in Table I. The baseline B is calculated by incorporating all time steps whether they violate one of the consistency tests or not. If one would randomly choose a subset from the complete sequence, the expected number of outliers and the average NEES would correspond to those baseline values in B. However, using only those subsets where one of the tests is within the defined boundary and discarding all the other values - since it is already known that they violate the assumptions - significantly reduces the number of outliers and similarly the average NEES. The combination of all three tests provides a further improvement which shows that the three stochastic tests reliably detect inconsistencies at different time steps. Because the detection probability $p_D = 0.2$ is quite small, the lower boundary of the respective Bernoulli consistency test is seldom reached and thus this test provides the smallest benefit. This is due to the fact that occlusion of landmarks was not modeled so far. Since Table I shows the numbers for two separate recordings with 5 Monte-Carlo runs each, it is very improbable to have chosen a subset with such tremendous decrease by chance. For the sake of completeness, Table II depicts the corresponding Root Mean Square Error (RMSE) of the localization result compared to the RTK reference system.

V. CONCLUSION

In this contribution, a new online approach to estimate and validate the measurement model parameters of feature-based random-set MCL algorithms was introduced. The idea is to stochastically detect an inconsistent measurement process by estimating the relevant parameters online and creating corresponding confidence intervals. The evaluation showed that the concept can be successfully applied to challenging real-world scenarios. The presented approach reliably detects situations where one cannot trust the localization result anymore. Besides an online self-assessment of feature-based random-set MCL algorithms, the proposed method can also be used to estimate all relevant parameters of the multi-object measurement model of a yet unknown sensor and scenario combination.

REFERENCES

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 1999, pp. 1322–1328.

[2] P. Jensfelt, D. Austin, O. Wijk, and M. Andersson, "Feature based CONDENSATION for mobile robot localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2000, pp. 2531–2537.

[3] W. Y. Ochieng, K. Sauer, D. Walsh, G. Brodin, S. Griffin, and M. Denney, "GPS integrity and potential impact on aviation safety," *Journal of Navigation*, vol. 56, no. 1, pp. 51–65, 01 2003.

[4] R. G. Brown, "A baseline RAIM scheme and a note on the equivalence of three RAIM methods," *Navigation: Journal of The Institute of Navigation*, vol. 39 No. 3, no. Fall 1992, pp. 301–316, 1992.

[5] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Integrity of map-matching algorithms," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 4, pp. 283–302, 2006.

[6] M. Jabbour, P. Bonnifait, and V. Cherfaoui, "Map-matching integrity using multihypothesis road-tracking," *Journal of Intelligent Transportation Systems*, vol. 12, no. 4, pp. 189–201, 2008.

[7] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*. Academic Press, Inc., 1988.

[8] R. Mahler, "Divergence detectors for multitarget tracking algorithms," in *Proceedings SPIE 8745, Signal Processing, Sensor Fusion, and Target Recognition XXII*, 2013.

[9] S. Reuter, B.-T. Vo, B. Wilking, D. Meissner, and K. Dietmayer, "Divergence detectors for the δ-generalized labeled multi-Bernoulli filter," in *Proceedings of the 8th Workshop Sensor Data Fusion: Trends, Solutions, and Applications*, Bonn, Germany, 2013.

[10] J. Mullane, B.-N. Vo, M. Adams, and W. Wijesoma, "A random set formulation for Bayesian SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2008, pp. 1043–1049.

[11] J. Mullane, B.-N. Vo, M. Adams, and B.-T. Vo, *Random Finite Sets for Robot Mapping and SLAM - New Concepts in Autonomous Robotic Map Representations*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 72.

[12] J. E. Handschin and D. Q. Mayne, "Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering," *International Journal of Control*, vol. 9, no. 5, pp. 547–559, 1969.

[13] P. Jensfelt, "Approaches to mobile robot localization in indoor environments," Ph.D. dissertation, Signal, Sensors and Systems (S3), 2001.

[14] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[15] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[16] R. Mahler, *Statistical Multisource-Multitarget Information Fusion*. Artech House Inc., Norwood, 2007.

[17] B.-T. Vo and B.-N. Vo, "Labeled random finite sets and multi-object conjugate priors," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3460–3475, 2013.

[18] B. N. Vo, B. T. Vo, and H. G. Hoang, "An efficient implementation of the generalized labeled multi-bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1987, April 2017.

[19] R. Mahler, B.-T. Vo, and B.-N. Vo, "CPHD filtering with unknown clutter rate and detection profile," *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3497–3513, 8 2011.

[20] B.-T. Vo, B.-N. Vo, R. Hoseinnezhad, and R. Mahler, "Robust multi-Bernoulli filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 399–409, 2013.

[21] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, p. 404, 1934.

[22] D. Schuhmacher, B.-T. Vo, and B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, 8 2008.

[23] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 3 1957.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *International Conference on Knowledge Discovery and Data Mining*, pp. 22–231, 1996.

Factor descent optimization for sparsification in graph SLAM

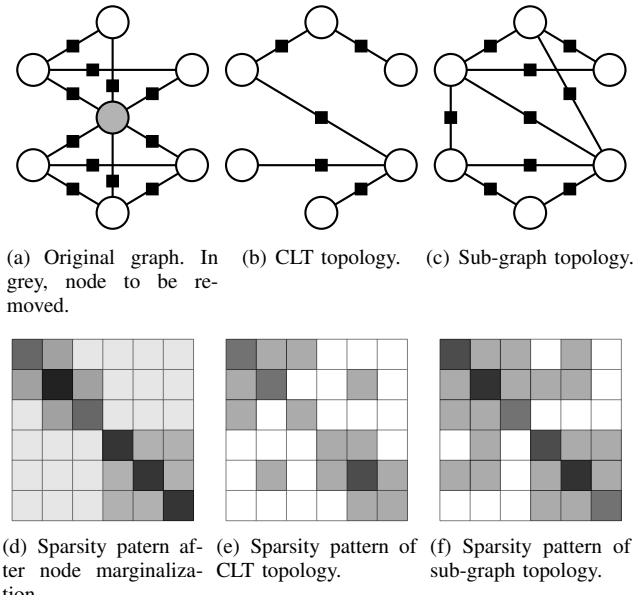
Joan Vallvé, Joan Solà, Juan Andrade-Cetto

***Abstract*— In the context of graph-based simultaneous localization and mapping, node pruning consists in removing a subset of nodes from the graph, while keeping the graph's information content as close as possible to the original. One often tackles this problem locally by isolating the Markov blanket sub-graph of a node, marginalizing this node and sparsifying the dense result. It means computing an approximation with a new set of factors. For a given approximation topology, the factors' mean and covariance that best approximate the original distribution can be obtained through minimization of the Kullback-Liebler divergence. For simple topologies such as Chow-Liu trees, there is a closed form for the optimal solution. However, a tree is oftentimes too sparse to explain some graphs. More complex topologies require nonlinear iterative optimization. In the present paper we propose Factor Descent, a new iterative optimization method to sparsify the dense result of node marginalization, which works by iterating factor by factor. We also provide a thorough comparison of our approach with state-of-the-art methods in real world datasets with regards to the obtained solution and convergence rates.**

I. Introduction

Simultaneous localization and mapping (SLAM) is the problem of building a representation of the environment while getting localized in it. Without any strategy to face it, the longer the experiment, the larger the problem to solve. Efforts to reduce resource demands have been focused mainly in two directions: by facing the computational complexity of the algorithms, or by tackling the problem size. Even though several improvements have been made in the first direction [1, 2], the later is still of concern, as the solution is always linked to the length of the experiment. This claims for suboptimal strategies that reduce problem size but still keep as much information as possible.

Several SLAM methods include mechanisms to limit problem size growth. One of the simplest approaches consists in considering a temporal or spatial window and discarding the landmarks and/or poses that lie outside this window. This implies giving up loop closures, such as in visual and visual-inertial odometry methods [3, 4]. In Pose SLAM [5], new observations and robot poses are only added to the problem if their entropy-based information content is significant. In contrast, in [6] a hierarchical graph structure is devised. The graphs higher up in the hierarchy represent marginalized sub-graphs and if the error is low enough, only part of the problem is solved. Johannsson et al. [7] propose a reduced pose graph that only grows with the size of the environment being mapped by marginalizing poses with regards to distance. Chouldhary et al. [8] on the other hand propose an information-based reduced landmark SLAM system. The method proposes a trade off between memory footprint and accuracy using an entropy-based cost function to decide which landmarks should be discarded.



(a) Original graph. In grey, node to be removed. (b) CLT topology. (c) Sub-graph topology.

(d) Sparsity patern after node marginalization. (e) Sparsity pattern of CLT topology. (f) Sparsity pattern of sub-graph topology.

Fig. 1. Tree topology can be too sparse to accurately approximate the dense distribution resulting of a node marginalization.

One important characteristic of SLAM is sparsity: the network of geometrical constraints corresponding to sensor measurements between robot trajectory and/or environment representation is only a small subset of all the possibilities. Graph-SLAM methods take profit of this sparsity to speed up the computation of the optimal solution.

Moreover, SLAM is a non-linear problem that is faced by linearizing it. The capability of relinearization greatly improves the accuracy of the solution.

Normally, the only way of reducing the problem size without loss of information is marginalization. However, marginalization induces fill-in, increasing computational cost, and does not allow for relinearization, deriving in accuracy loss.

Several works have been published with methods for finding the best sparse and relinearizable approximation of the dense and not-relinearizable result of a marginalization. This is known as sparsification, and is the focus of the present paper. Kretzschmar and Stachniss [9] present an information-theoretic compression method for pose graph SLAM that selects the nodes containing the most informative laser scans. They find the subset of measurements that maximize the mutual information of the map for that subset. More recently, [10]–[12] approach the problem by finding the sparse approximation that minimizes the Kullback-Liebler divergence (KLD) with the dense distribution resulting from the node marginalization. While there is a closed form for the simplest topology, *e.g.* the Chow-Liu tree (CLT), an iterative optimization is needed for richer topologies.

The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona, Spain. {jvallve,jsola,cetto}@iri.upc.edu.

This work has been supported by the Spanish Ministry of Economy and Competitiveness under Project ROBINSTRUCT (TIN2014-58178-R), by the EU H2020 Project LOGIMATIC (H2020-Galileo-2015-1-687534) and by the María de Maeztu Seal of Excellence to IRI MDM-2016-0656.

In the majority of cases, and as we reveal in the experimental section, a tree topology is too simple to accurately approximate the dense result of a node marginalization (see Fig.1). Hence, in this paper we focus on iterative optimization for sparsification. We introduce Factor Descent optimization for sparsification. Given a non-dense factor topology, we iteratively optimize each of the factors leaving fixed the rest. For each factor, we compute its parameters (mean and information matrix) that minimize the KLD given the rest of topology factors' parameters.

The paper is organized as follows. The next section includes the problem formulation and existing methods. Section III presents our novel factor descent method. Section IV presents the results, and conclusions and future work are exposed in the last section.

II. Node removal and sparsification in graph SLAM

Graph-based SLAM methods represent the problem as a set of variables (nodes) and a set of geometrical constraints (factors). The state $\mathbf{x}$ includes nodes representing poses of the vehicle along its trajectory and/or some map representation. Each factor expresses the discrepancy or error $\mathbf{e}$ between a measurement $\mathbf{z}$ and its expectation,

$$\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) - \mathbf{z} + \mathbf{v}, \qquad \mathbf{v} \sim \mathcal{N}(0, \mathbf{\Omega}^{-1}) \tag{1}$$

being $h(\mathbf{x})$ the sensor's measurement model and $\mathbf{\Omega}$ the information matrix of the measurement Gaussian noise $\mathbf{v}$.

The problem is solved iteratively by minimizing the Mahalanobis squared norm of all linearized errors

$$\Delta\mathbf{x}^* = \arg\min_{\Delta\mathbf{x}} \sum_k \|h_k(\mathbf{x}) - \mathbf{z}_k + \mathbf{J}_k\Delta\mathbf{x}\|^2_{\mathbf{\Omega}_k^{-1}} \tag{2}$$

being $\mathbf{x}$ the state estimate at the current iteration, and $\mathbf{J}_k$ the Jacobian of the k-th measurement. [1] Imposing null derivative of the cost in (2) w.r.t $\Delta\mathbf{x}$, the optimal step $\Delta\mathbf{x}^*$ is found and used to update the estimate. Current methods for solving for $\Delta\mathbf{x}^*$ use Cholesky [2, 14, 15] or QR [1, 16, 17] matrix factorizations. Important speed-ups are obtained with incremental methods [1, 2, 15, 17], which update the problem directly on the factorized matrix.

Reducing the problem size in graph SLAM is usually approached in two steps: node marginalization and sparsification (see Fig. 2). These two stages do not necessarily have to be immediately consecutive, and the second one can be postponed depending on computational availability [11].

Having selected a node to prune (Fig. 2.a), the process is faced locally. A local problem around the node (Fig. 2.b) is defined by cropping the node's Markov blanket (all nodes at distance 1) and all its intra-factors (the factors involving only nodes in the Markov blanket). Optionally, this cropped problem can be solved. Then, all factors can be relinearized using the new solution before proceeding, yielding slightly better results especially in on-line cases [12].

After that, marginalization of the selected node is performed via Schur complement. This marginalization can be understood as adding a dense factor (Fig. 2.c) that substitutes all intra-factors that involve the removed node. This new dense factor has no measurement model associated to; hence, its error cannot be re-evaluated, and re-linearization is not possible.

The goal of the sparsification process is to approximate the dense distribution $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, resulting from node marginalization, with a sparse distribution $q(\mathbf{x}) \sim \mathcal{N}(\breve{\boldsymbol{\mu}}, \breve{\mathbf{\Sigma}})$ defined by a new set of (relinearizable) factors (Fig. 2.d). This is usually split in two phases: building a topology (i.e. define a set of factors with their measurement model) and computing their mean and information that best approximate the original distribution.

A. Topology

The topology defines the arrangement between the Markov blanket nodes and the new set of factors, each factor with a measurement model. Typically, the factors are made up of relative measurements between pairs of nodes. The simplest topology using relative measurements is a spanning tree. The Chow-Liu tree (CLT) defines a tree topology with factors between the most correlated nodes (i.e. the ones with most mutual information).

However, even taking the maximum mutual informative factors, a tree topology is usually too sparse to approximate the original distribution. For this reason, the so-called subgraph topology departs from the CLT and adds (a few) more factors, also based on the nodes' mutual information [12]. Alternatively, the cliquey topology [12] takes the CLT and converts pairs of independent factors into one single factor by correlating them.

Differently to the CLT-based methods, a ℓ_1-regularized KLD minimization can be used to compute the topology that will encode the most information [11].

B. Sparsification through KLD minimization

Given the topology, we want to compute its factors' means $\breve{\mathbf{z}}_k$ and information $\breve{\mathbf{\Omega}}_k$ that minimize the KLD between the dense $p(\mathbf{x})$ and sparse $q(\mathbf{x})$ distributions. This can be posed as

$$D_{KL} = \frac{1}{2}\Big(\langle\breve{\mathbf{\Lambda}}, \mathbf{\Sigma}\rangle - \ln|\breve{\mathbf{\Lambda}}\mathbf{\Sigma}| + \|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}^{-1}} - d\Big), \tag{3}$$

where $\langle\cdot,\cdot\rangle$ denotes the matrix inner product and $\breve{\mathbf{\Lambda}} = \breve{\mathbf{\Sigma}}^{-1}$ is the information matrix of $q(\mathbf{x})$.

This expression can be minimized as follows. The dimension d of both distributions and $\mathbf{\Sigma}$ are constant w.r.t the information of each measurement $\breve{\mathbf{\Omega}}_k$. The squared norm term $\|\breve{\boldsymbol{\mu}} - \boldsymbol{\mu}\|^2_{\breve{\mathbf{\Lambda}}}$ is null if the means of all measurements are set using the dense distribution mean $\breve{\mathbf{z}}_k = h_k(\boldsymbol{\mu})$. Then, introducing the block diagonal matrix $\breve{\mathbf{\Omega}} = diag(\breve{\mathbf{\Omega}}_1 \dots \breve{\mathbf{\Omega}}_k \dots)$ containing all new factors' information matrices, and the Jacobian $\breve{\mathbf{J}} = [\breve{\mathbf{J}}_1^\top \dots \breve{\mathbf{J}}_k^\top \dots]^\top$ stacking all new factors' Jacobians, the sparse information matrix of the approximate distribution is $\breve{\mathbf{\Lambda}} = \breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}}$. Considering the above, the factors' information that minimize the KLD in (3), can be written as the constrained problem

$$\begin{aligned}\breve{\mathbf{\Omega}}^* = \arg\min_{\breve{\mathbf{\Omega}}} & \langle\breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}}, \mathbf{\Sigma}\rangle - \ln|\breve{\mathbf{J}}^\top\breve{\mathbf{\Omega}}\breve{\mathbf{J}}| \\ \text{s.t.} \quad & \breve{\mathbf{\Omega}} \in \mathcal{D}, \breve{\mathbf{\Omega}} \succ 0\end{aligned} \tag{4}$$

where $\mathcal{D}$ refers to the set of block-diagonal matrices.

[1] In case of manifolds, (1) and the squared Mahalanobis norm in (2) become $\mathbf{e}(\mathbf{x}) = h(\mathbf{x}) \ominus \mathbf{z} \oplus \mathbf{v}$ and $\|h_k(\mathbf{x}) \ominus \mathbf{z}_k + \mathbf{J}_k\Delta\mathbf{x}\|^2_{\mathbf{\Omega}_k^{-1}}$, respectively, with $\mathbf{J}_k = \partial(h_k(\mathbf{x}) \ominus \mathbf{z}_k)/\partial\Delta\mathbf{x}$. The $\oplus$ and $\ominus$ are the addition and subtraction operators on the manifold, as described in [13].

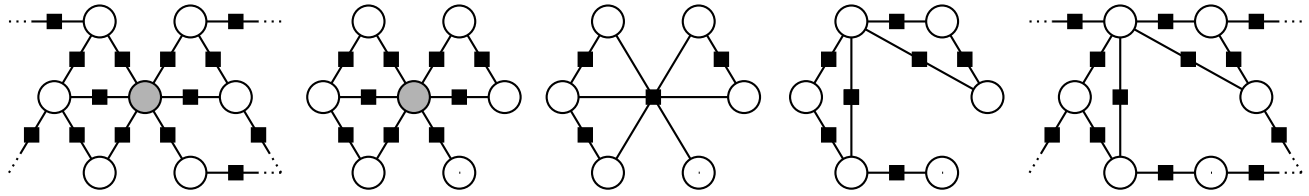


Fig. 2. Example of a pruning sequence of actions. The removed node is depicted in grey. (a) Initial graph. (b) Markov blanket and intra-factors are kept. (c) Node marginalization produces a dense factor (the central factor). (d) Sparsification computes an approximation with a set of new factors. (e) Substitution of the sparse approximation into the initial graph.

In some cases such as dense problems with only relative measurements, the dense problem has a rank-deficient information matrix $\mathbf{\Lambda}$, and the covariance matrix $\mathbf{\Sigma}$ is not defined. In that case, we can apply a projection $\mathbf{\Lambda} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ such that $\mathbf{D}$ is invertible. Then, all formulation derived from (4) holds by substituting

$$\begin{aligned} \breve{\mathbf{J}} &\mapsto \breve{\mathbf{J}}\mathbf{U} \\ \mathbf{\Sigma} &\mapsto \mathbf{D}^{-1}. \end{aligned} \tag{5}$$

This projection can be obtained by re-parametrizing the problem to relative poses w.r.t an arbitrarily chosen node [10, 11] or using a rank-revealing eigen decomposition [12].

C. Sparsification in closed form

Certain topologies admit a closed form solution to (4). When $\breve{\mathbf{J}}$ is invertible, imposing null derivative of (4) w.r.t. all factor information matrices yields

$$\breve{\mathbf{\Omega}}_k = (\breve{\mathbf{J}}_k \mathbf{\Sigma} \breve{\mathbf{J}}_k^\top)^{-1}. \tag{6}$$

This is the case of the tree topology in SLAM of relative measurements, using a projection as (5). However, and as has been said, this topology can be too sparse to accurately approximate the exact dense distribution. Also, (6) holds for the cliquey topology. However, it can carry convergence issues to the SLAM problem solution [12]. Moreover, the cliquey forms break the homogeneity of factors, which is valuable in many cases.

D. Sparsification via iterative optimization

Other topologies with non-invertible Jacobian $\breve{\mathbf{J}}$ do not admit a closed form solution and the problem (4) has to be solved using iterative optimization. The state-of-the-art literature proposes two different optimization algorithms: Interior Point (IP) and Limited-memory Projected Quasi-Newton (PQN) [18]. IP includes the constraint of positive definiteness of the solution in the cost function

$$\langle \breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}, \mathbf{\Sigma} \rangle - \ln |\breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}| - \rho \ln |\breve{\mathbf{\Omega}}|. \tag{7}$$

The log barrier parameter ρ is iteratively decreased towards 0. It requires an initial guess for $\breve{\mathbf{\Omega}}$ that strictly accomplishes the positive definite restriction, such as the identity matrix [12]. IP has quadratic convergence, but requires the (costly) computation of the Hessian and gradient of (7). A stricter constraint can be applied in (7) instead of the log barrier term to guarantee the conservativeness: $\rho \ln |\mathbf{\Lambda} - \breve{\mathbf{J}}^\top \breve{\mathbf{\Omega}} \breve{\mathbf{J}}|$ [11].

On the contrary, PQN does not require the computation of the Hessian (it still needs the gradient), nor a feasible initial guess. The positive definiteness constraint is accomplished through the projection $\mathcal{P}(\breve{\mathbf{\Omega}})$ onto the positive semi-definite subspace, by setting all negative eigenvalues to zero,

$$\mathcal{P}(\breve{\mathbf{\Omega}}) = \mathbf{V} \operatorname{diag}(\max\{0, \lambda_i\}) \mathbf{V}^\top, \tag{8}$$

being $\breve{\mathbf{\Omega}} = \mathbf{V} \operatorname{diag}(\lambda_i) \mathbf{V}^\top$ the eigen decomposition. PQN has a slower convergence than IP, but it can be initialized closer to the optimal solution using an initial guess based on the off-diagonal blocks of the dense information matrix, as proposed in [19],

$$\breve{\mathbf{\Omega}}_k = \mathbf{J}_{k_1}^{-\top} \mathbf{\Lambda}_{k_1,k_2} \mathbf{J}_{k_2}^{-1} \tag{9}$$

being k_1, k_2 the two nodes involved in the factor k (so $\mathbf{J}_{k_1}, \mathbf{J}_{k_2}$ are the non-zero blocks of $\mathbf{J}_k$) and being $\mathbf{\Lambda}_{k_1,k_2}$ the off-diagonal block corresponding to the involved nodes. Such initial guess is normally not symmetric nor positive semi-definite, and one usually takes its closest symmetric positive semi-definite approximation [20]. Since this may result in a semi-definite positive guess (or close), it cannot be used in the IP method.

III. Factor descent optimization for sparsification

We propose Factor Descent sparsification (FD), a novel optimization approach for solving (4) that takes inspiration in coordinate descent optimization. FD is a cyclic block-coordinate descent method; each step of the cycle consists in solving for a (small) block of variables (those defining one factor's information matrix $\breve{\mathbf{\Omega}}_i$) while fixing the rest.

Consider a given topology and an initial guess $\breve{\mathbf{\Omega}}$. The derivative of (4) w.r.t the i-th factor's information matrix $\breve{\mathbf{\Omega}}_i$ is

$$\frac{\partial D_{KL}}{\partial \breve{\mathbf{\Omega}}_i} = \breve{\mathbf{J}}_i \mathbf{\Sigma} \breve{\mathbf{J}}_i^\top - \breve{\mathbf{J}}_i (\breve{\mathbf{\Upsilon}}_i + \breve{\mathbf{J}}_i^\top \breve{\mathbf{\Omega}}_i \breve{\mathbf{J}}_i)^{-1} \breve{\mathbf{J}}_i^\top \ , \tag{10}$$

where $\breve{\mathbf{\Upsilon}}_i$ is the information matrix of the problem considering only the rest of factors,

$$\breve{\mathbf{\Upsilon}}_i = \sum_{j \neq i} \breve{\mathbf{J}}_j^\top \breve{\mathbf{\Omega}}_j \breve{\mathbf{J}}_j \ . \tag{11}$$

Imposing null derivative and applying the Woodbury matrix identity twice,[2] we get the closed form,

$$\breve{\mathbf{\Omega}}_i = \underbrace{(\breve{\mathbf{J}}_i \mathbf{\Sigma} \breve{\mathbf{J}}_i^\top)^{-1}}_{\mathbf{\Phi}_i} - (\breve{\mathbf{J}}_i \breve{\mathbf{\Upsilon}}_i^{-1} \breve{\mathbf{J}}_i^\top)^{-1}. \tag{12}$$

This is the optimal i-th factor's information matrix in terms of KLD if the rest of factors are fixed.

Descent of the full KLD cost of (4) is achieved factor by factor, and hence the Factor Descent name. This can be

[2] applying the Woodbury matrix identity forwards and backwards from
$$\begin{aligned} D(A+BCD)^{-1}B &= D(A^{-1} - A^{-1}B(C^{-1}+DA^{-1}C)^{-1}DA^{-1})B \\ &= DA^{-1}B - DA^{-1}B(C^{-1}+DA^{-1}C)^{-1}DA^{-1}B \\ &= ((DA^{-1}B)^{-1} + C)^{-1} \end{aligned}$$

Algorithm 1 Factor descent sparsification

Input: Dense mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, topology $\mathcal{Z}$

```
// Precompute constant variables
for z_i ∈ Z do
    J_i ← evaluateJacobian(z_i, μ)
    Φ_i ← (J̆_i Σ J̆_i^⊤)^{-1}
end for
i := 1
while not endConditions() do
    // Compute the information of the rest of factors
    Ῠ_i ← Σ_{j≠i} J̆_j^⊤ Ω̆_j J̆_j
    // i-th factor descent
    Ω̆_i ← Φ_i − (J̆_i Ῠ_i^{-1} J̆_i^⊤)^{-1}
    // Ensure positive semi-definite solution
    if Ω̆_i ≺ 0 then
        V, λ ← eigenDecomposition(Ω̆_i)
        Ω̆_i ← V diag(max(0, λ)) V^⊤
    end if
    // Cycle for all factors
    i++
    if i > N then
        i := 1
    end if
end while
```

iterated as many times as desired. While the second term of (12) should be computed at each iteration, the first one $\boldsymbol{\Phi}_i$ is constant for each factor and should be computed only once. The method is described in algorithm 1. The first term $\boldsymbol{\Phi}_i$ can be interpreted as the information of the dense exact distribution projected in the measurement space of the i-th factor. Analogously, the second term is the projection of the information of the rest of the factors onto the measurement space of the i-th factor.

We want to emphasize that (12) is a generalization of (6). The conditions in which (6) is applicable are the same in which the second term in (12) is null. For example, in the tree topology, the projection of the information of the rest of the factors to each factor's measurement space is null.

As in other methods, in case of rank-deficient $\boldsymbol{\Lambda}$, the method holds using the projection (5). Note that the rest of new factors must be projected too, with $\breve{\boldsymbol{\Upsilon}} \mapsto \mathbf{U}^\top \breve{\boldsymbol{\Upsilon}} \mathbf{U}$.

Since the optimal solution in the factor's subspace is computed in closed form (12), iterations in FD refer to the fact that we iterate on the factors, not on finding the optimal for each factor through repeated linearizations. In other words, a clear benefit of FD is that there is no fitting to any linear or quadratic function. The convergence rate mainly depends on how much the direction to the optimal solution is aligned with the sub-spaces corresponding to each factor (see Fig. 3 for an illustrative example).

A. Positive-definiteness

It follows from (12) that $\breve{\boldsymbol{\Omega}}_i$ is positive definite only if

$$(\breve{\mathbf{J}}_n \boldsymbol{\Sigma} \breve{\mathbf{J}}_n^\top)^{-1} \succ (\breve{\mathbf{J}}_n \breve{\boldsymbol{\Upsilon}}^{-1} \breve{\mathbf{J}}_n^\top)^{-1}. \tag{13}$$

This happens when the projection of the information of the dense distribution to the measurement space is 'bigger' than that of the rest of the factors. A zero eigenvalue of $\breve{\boldsymbol{\Omega}}_i$ implies that the rest of new factors already explain completely the original distribution in some direction. Further, a negative eigenvalue implies that the approximation is not

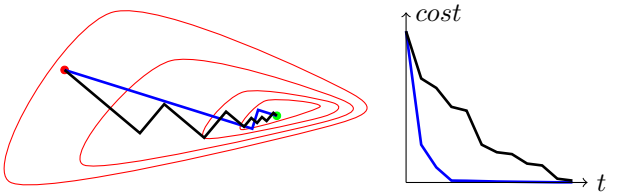


Fig. 3. Examples of convergence of the cyclic coordinate descent optimization for two different alignments of the coordinates w.r.t. the direction to the optimal solution (green dot).

conservative (without considering the i-th factor) and the optimal factor would subtract this excess of information. After each iteration, we impose positive semi-definite result applying (8).

B. Initial guess

Since the positive semi-definite constraint is imposed after solving, the initial guess is not required to be in a strictly feasible point. Thus, we can use the off-diagonal blocks based initialization (9).

Alternatively, FD can be used to get an initial guess by just applying the first cycle of (12). During this first cycle, only the previously computed factors must be considered in $\breve{\boldsymbol{\Upsilon}}_i$, and therefore it can be computed incrementally,

$$\breve{\boldsymbol{\Upsilon}}_i = \breve{\boldsymbol{\Upsilon}}_{i-1} + \breve{\mathbf{J}}_{i-1}^\top \breve{\boldsymbol{\Omega}}_{i-1} \breve{\mathbf{J}}_{i-1}. \tag{14}$$

IV. Results

In order to test the performance of the FD sparsification method just presented and to compare it with the state-of-art IP and PQN methods, we implemented all methods in Matlab. To prevent linearization errors to be confused with sparsification inaccuracy, we relinearize the whole SLAM problem and solve it at each new trajectory pose using our own implementation of $\sqrt{SAM}$ [16]. We implemented the IP method using the gradient and Hessian in [12]. For PQN, we used the authors' Matlab implementation [21]. Finally, we build the sub-graph topology as explained in section II-A with a number of factors doubling that of the tree topology.

A. Initial guess and convergence rate

We want to test the different combinations of optimization method and initial guess on several sparsification problems. The methods PQN and FD are combined with three different initial guesses: the identity matrix (Id) as proposed in [12], the one based on the off-diagonal blocks (ODB) as proposed in [19], and our First FD cycle (FFD) of Sec. III-B. The method IP is only combined with Id, as it diverges otherwise. This gives a total of seven combinations.

We executed a SLAM for the Manhattan M3500 dataset [22] with 80% of node removal. To guarantee equal conditions for all the methods under test, all sparsification problems after node marginalization are stored. Then, we compare all methods by solving the stored problems. In all cases, we solve and relinearize the cropped problems before marginalization.

First, we analyze the relation between the type of initial guess used and problem size. Fig. 4 depicts the mean and variance of the KLD between each initial guess and the dense exact distribution, as a function of the sparsification problem size, i.e. the number of nodes in the Markov blanket. As expected, in all cases the smaller the size of the Markov blanket, the better. ODB initialization performs

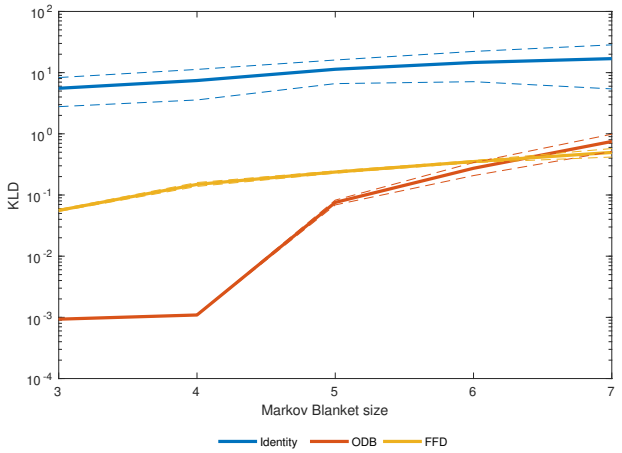


Fig. 4. Mean and variance (dashed line) of all three initial guess KLD vs. the Markov blanket size of for sparsification processes made in the Manhattan experiment with 80% node removal.

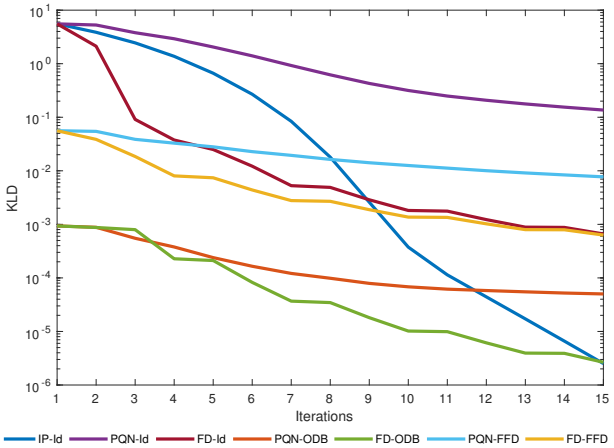


Fig. 5. Mean KLD evolution of all sparsification combinations (methods and initial guesses), for problems with Markov blanket size = 3 in the Manhattan experiment with 80% of node removal.

better for small Markov blankets whereas the proposed FFD works better for larger problems. The identity initial guess (Id) is significantly inferior than the other two initialization methods (notice the vertical log axis).

We analyze now the convergence rate for all seven combinations of initial guess and methods. In Matlab, the most computationally expensive operations are optimized, and therefore CPU time measures are not reliable. Also, since the size of the sparsification problems is small and independent of the full problem size, the complexity of each method w.r.t. the Markov blanket size is not necessarily representative. Then, the evaluation of the convergence rates is based on the number of optimization iterations.

Fig. 5 depicts the evolution of the mean KLD for all method-initial guess combinations, for Markov blankets of size 3. Although IP converges faster, PQN and FD take profit of a (much) better initial guess at the initial iterations, becoming better alternatives for implementations with low computational resources that do not allow for many iterations. If the number of iterations is not a constraint, IP reaches the best results.

As well as the initial guess, the convergence of the methods depends also on the size of the sparsification problem. We show in Fig. 6 the mean KLD evolution for all problem sizes. While in small problems FD-ODB is the best combination up to 15 iterations, for larger Markov blanket sizes IP becomes the best choice from a smaller amount of iterations.

B. Application

We tested all methods on three different datasets [22] to evaluate their performance. The typology of the chosen datasets is very different. The Manhattan M3500 sequence is a large problem and, since it is highly connected, it has large Markov blankets. On the contrary, the MIT Killian Court sequence has few loop closures and small Markov blankets. The Intel Research Lab sequence is somewhere in between.

In all cases, the same 80% volume of nodes were marginalized. Since node selection is out of the scope of this paper, we applied the simple strategy of keeping one node every 5.

The following combinations of optimization method and initial guess were tested: IP-Id, PQN-ODB, FD-ODB, FD-FFD. We ran four independent SLAM solutions for each

TABLE I

FINAL KLD AFTER 80% OF NODE REMOVAL

	Method	Iterations			
		0	5	10	15
Manhattan	IP-Id	1285	110.6	**4.967**	**4.786**
	PQN-ODB	**7.36**	6.424	5.587	5.399
	FD-ODB	**7.36**	**5.469**	5.291	4.852
	FD-FFD	81.09	11.46	7.617	5.82
	CLT	25.84	-	-	-
Intel	IP-Id	314.8	85.07	8.083	7.293
	PQN-ODB	**18.19**	16.33	9.35	8.423
	FD-ODB	**18.19**	**7.962**	**7.247**	7.027
	FD-FFD	29.26	10.47	7.574	**6.902**
	CLT	27.68	-	-	-
Killian	IP-Id	429.4	210.1	76.73	**77.29**
	PQN-ODB	78.59	78.07	78.28	78.39
	FD-ODB	78.59	**77.21**	**77.4**	77.37
	FD-FFD	**70.11**	78.37	77.49	77.59
	CLT	82.92	-	-	-

combination fixing different number of optimization iterations: 0, 5, 10 and 15. For greater completeness, we also ran a CLT using the closed form solution.

The baseline for evaluation is the batch optimization of the original SLAM graph without removing any node. Following [12], factors involving previously removed nodes were redirected to the closest existing node. This was also done for the baseline graph in order to evaluate only the sparsification performance.

To evaluate the performance of the different approaches, we computed its KLD with the baseline using (3), but this time evaluating for the whole SLAM problem instead of just the Markov blanket.

Table IV-B shows the final KLDs of all 17 experiments for the three datasets. In the table, KLD for CLT is reported only for zero iterations since it has a closed form solution. The comparison between CLT and all methods evidences the limitation of the tree topology for accurately approximating the dense distribution. For highly connected problems (Manhattan and Intel), the use of a sub-graph topology produces a much smaller KLD. And for sparser cases such as the Killian dataset, the sub-graph topology slightly outperforms CLT. This is not surprising, since the average Markov blanket sizes are small in this case.

Note that, even without any optimization iteration, using

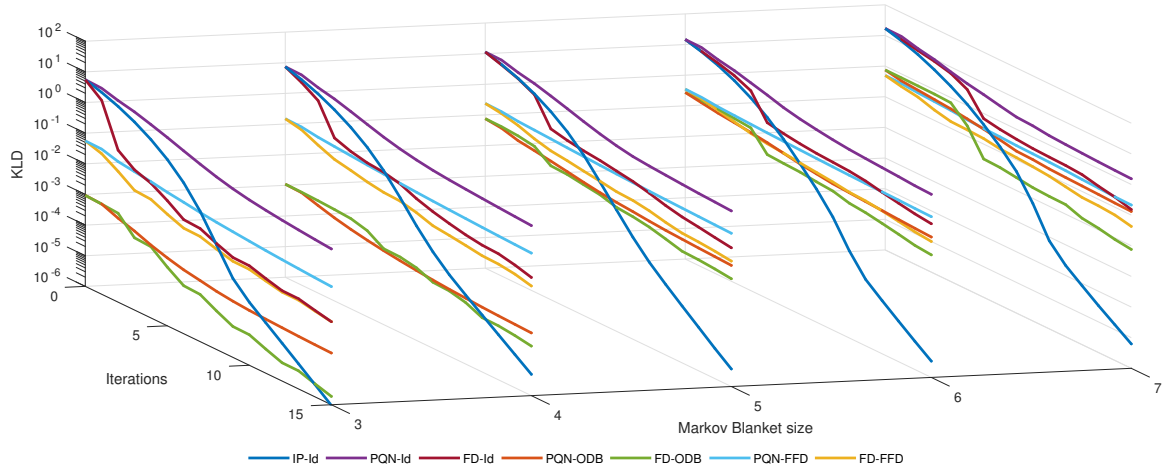


Fig. 6. Mean KLD evolution of all sparsification combinations (methods and initial guesses) for different Markov blanket sizes corresponding to all problems in the Manhattan experiment with 80% of node removal.

the ODB initialization always yields a better approximation than CLT.

The presented results validate the initial hypothesis: the tree topology is too sparse to accurately approximate the dense distribution. A more complex topology can approximate the original graph better without computational burden.

V. Conclusions and Future Work

Sparsification is a useful mechanism to maintain the SLAM problem bounded. The topology chosen determines the existence of a closed form solution and strongly affects the accuracy of the approximation.

Tree topologies admit a closed form solution but are usually too simple to approximate the original graph. On the contrary, more populated topologies lose the applicability of closed form solution but can encode more information of the original graph.

We presented the novel Factor Descent optimization method for sparsification that provides more accurate approximations with less iterations than other state-of-the-art sparsification methods. When combined with the off-diagonal block initialization, Factor Descent gives the best results in a larger number of situations.

In addition, we observed that the existence of a closed form solution is not a sufficient argument for choosing a tree topology instead a more complex one. Better approximations in terms of KLD can be reached using a more complex topology with few or no optimization iterations.

In future work we consider the implementation in C++ of all analyzed sparsification methods to be able to compare them also with regards to computational time. Also, a non cyclic version of Factor Descent can be explored to further improve convergence.

References

[1] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *Int. J. Robotics Res.*, vol. 31, no. 2, pp. 216–235, 2011.

[2] V. Ila, L. Polok, M. Solony, and P. Svoboda, "SLAM++-A highly efficient and temporally scalable incremental SLAM framework," *Int. J. Robotics Res.*, vol. 36, no. 2, pp. 210–230, 2017.

[3] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *Int. J. Robotics Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[4] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robotics Res.*, vol. 32, no. 6, pp. 690–711, 2013.

[5] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact Pose SLAM," *IEEE Trans. Robotics*, vol. 26, no. 1, pp. 78–93, Feb. 2010.

[6] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proc. IEEE Int. Conf. Robotics Autom.*, Anchorage, May 2010, pp. 273–287.

[7] H. Johannsson, M. Kaess, M. Fallon, and J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *Proc. IEEE Int. Conf. Robotics Autom.*, Karlsruhe, May 2013, pp. 54–61.

[8] S. Choudhary, V. Indelman, H. Christensen, and F. Dellaert, "Information-based reduced landmark SLAM," in *Proc. IEEE Int. Conf. Robotics Autom.*, Seattle, May 2015, pp. 4620–4627.

[9] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based SLAM," *Int. J. Robotics Res.*, vol. 31, no. 11, pp. 1219–1230, 2012.

[10] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Trans. Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[11] K. Eckenhoff, L. Paull, and G. Huang, "Decoupled, consistent node removal and edge sparsification for graph-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, Oct. 2016, pp. 3275–3282.

[12] M. Mazuran, W. Burgard, and G. D. Tipaldi, "Nonlinear factor recovery for long-term SLAM," *Int. J. Robotics Res.*, vol. 35, no. 1-3, pp. 50–72, 2016.

[13] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, 1990, pp. 167–193.

[14] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, May 2011, pp. 3607–3613.

[15] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, "Incremental block Cholesky factorization for nonlinear least squares in robotics," in *Robotics: Science and Systems*, Berlin, Jun. 2013.

[16] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robotics Res.*, vol. 25, no. 12, pp. 1181–1204, 2006.

[17] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[18] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, "Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm," in *Artificial Intelligence and Statistics*, 2009, pp. 456–463.

[19] M. Mazuran, G. D. Tipaldi, L. Spinello, and W. Burgard, "Nonlinear graph sparsification for SLAM," in *Robotics: Science and Systems*, Berkeley, Jul. 2014, pp. 1–8.

[20] N. J. Higham, "Computing a nearest symmetric positive semidefinite matrix," *Linear algebra and its applications*, vol. 103, pp. 103–118, 1988.

[21] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, "Matlab PQN toolbox," https://www.cs.ubc.ca/~schmidtm/Software/PQN.html.

[22] L. Carlone, http://www.lucacarlone.com/index.php/resources/datasets.

Collaborative Object Picking and Delivery with a Team of Micro Aerial Vehicles at MBZIRC

Matthias Nieuwenhuisen, Marius Beul, Radu Alexandru Rosu, Jan Quenzel,
Dmytro Pavlichenko, Sebastian Houben, and Sven Behnke

***Abstract*— Picking and transporting objects in an outdoor environment with multiple lightweight MAVs is a demanding task. The main challenges are sudden changes of flight dynamics due to altered center of mass and weight, varying lighting conditions for visual perception, and coordination of the MAVs over unreliable wireless connections.**

At the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) teams competed in a Treasure Hunt where three MAVs had to collaboratively pick colored disks and drop them into a designated box. Only little preparation and test time on-site required robust algorithms and easily maintainable systems to successfully achieve the challenge objectives.

We describe our multi-robot system employed at MBZIRC, including a lightweight gripper, a vision system robust against illumination and color changes, and a control architecture allowing to operate multiple robots safely. With our system, we—as part of the larger team NimbRo of ground and flying robots—won the Grand Challenge and achieved a third place in the Treasure Hunt.

I. Introduction

Aerial manipulation—especially picking, transporting, and delivering objects—became an area of much interest in recent years. Micro aerial vehicles (MAV) are well suited to quickly deliver small, but valuable objects, e.g., spare parts or medical substances. A particular advantage of employing aerial vehicles to detect and pick objects is that—in contrast to ground vehicles—they can reach otherwise hard to access or even dangerous areas.

To facilitate the development in this field, one of the tasks at the Mohamed Bin Zayed International Robotics Challenge 2017 (MBZIRC) was collaborative picking with MAVs. The Treasure Hunt task was to find and pick colored, ferromagnetic discs from the ground of an outdoor arena and to deliver them to a designated box in a predefined drop zone. Fig. 1 shows one of our MAVs while picking a moving object.

Teams were provided with rough specifications of the objects, i.e., diameter, height above ground, maximum weight of 500 g, and the possible colors, in advance. The drop box was specified by its approximate dimensions. Nevertheless, the exact arena setup—including colored markings on the ground making color-based perception challenging—was not known in advance and teams had to develop robust and flexible systems.

This work has been supported by a grant of the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) and grants BE 2556/7-2 and BE 2556/8-2 of the German Research Foundation (DFG).

The authors are with the Autonomous Intelligent Systems Group, Computer Science VI, University of Bonn, Germany nieuwenh@ais.uni-bonn.de

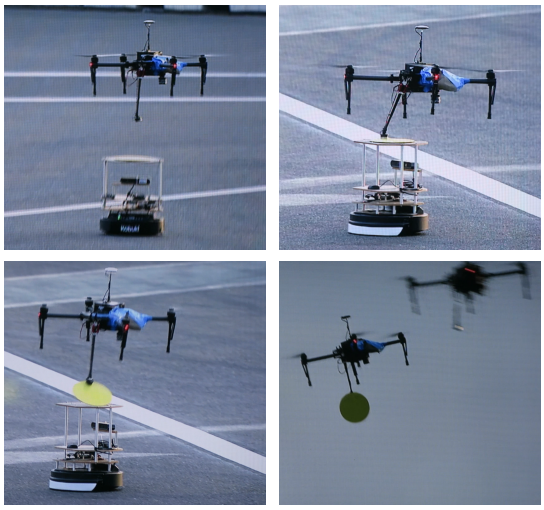


Fig. 1. Picking a dynamic object. Our MAV follows the yellow disc with visual servoing. The telescopic rod and the ball joint of our electromagnetic gripper allow compliant picking without disturbing attitude control of the MAV. The picked objects were delivered to a drop box up to 75 m away.

In contrast to lab experiments or controlled field tests, the particular challenge of the competition was the much reduced testing time. The arena was only accessible for teams at assigned time slots. In total, individual teams had two rehearsal slots of 35 minutes and four competition trials—two for the Treasure Hunt and two for the Grand Challenge. The systems had to be set up in the arena in only five minutes in each run. Consequently, complex algorithms that need extensive fine-tuning or are prone to failing in some cases are not an option for a competition system. Thus, we focused on simple but robust approaches and tried to identify and cover as many issues in advance as possible. Experiences gained during trials had to be incorporated into the system without additional testing before the next trial. Hence, the system complexity needed to be as low as possible to eliminate error sources. Furthermore, quickly changing overcast and light sandstorms changed illumination and navigation conditions significantly from trial to trial.

Due to these challenging conditions, only very few teams managed to score in this task autonomously. Our main contributions are

- robust detection of objects with only roughly specified color under varying lighting conditions,
- relative navigation while picking and dropping,
- lightweight and flexible picking hardware,
- coordination of multiple MAV with and without WiFi connections, and

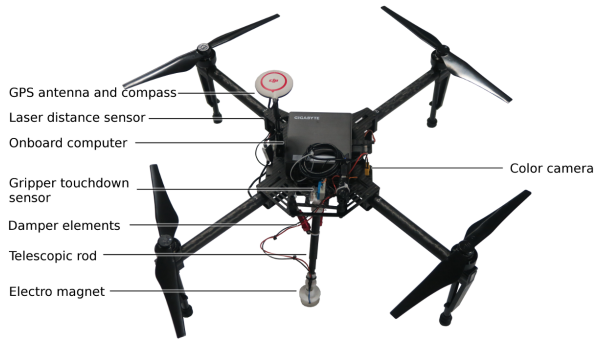


Fig. 2. Closeup of our MAV. The Matrice 100 is equipped with a down-pointing color camera for object and drop box detection. Objects are picked with an electromagnetic gripper on a telescopic rod. A small lidar sensor measures the distance to the ground. All calculations are performed on a powerful onboard PC.

- evaluation of our integrated multi MAV system in challenging conditions at MBZIRC.

II. Related Work

Aerial manipulation has been investigated by multiple research groups.

Morton et al. [1], for example, developed an MAV with manipulation capabilities for outdoor use. The MAV is equipped with a 3-DoF arm which is operated in hover mode without object perception or autonomous flight. An MAV with a 2-DoF robot arm that can lift relatively heavy weights is presented by the authors [2]. The controller explicitly models the changes in the vehicle dynamics by attaching a heavy object. The object positions are known beforehand. We employ a trajectory generator that uses a very simple dynamics model with frequent replanning on top of a model-free attitude controller to achieve robustness against changes in flight dynamics.

Ghadiok et al. [3] built a lightweight quadrotor for grasping objects in indoor environment. Similar to our work, they have a lightweight and compliant gripper to cope with uncertainties during grasping. Target objects are equipped with infrared beacons; we detect objects based on coarse color specifications. The ETH Zürich MBZIRC team Electronic Treasure Hunters describe a preliminary state of their approach to solving the challenge in [4]. They employ an electro-permanent magnetic gripper and color blob detection for visual servoing.

Some work exists on cooperative transport of objects [5], [6]. However, this work assumes a permanently connected object, i.e., no picking and placing.

III. System Setup

Our picking MAV, depicted in Fig. 2, is based on the DJI Matrice 100 quadcopter platform. This platform is designed for research and development—and consequently offers an easy ROS integration. We equipped the basic platform with a small, but powerful Gigabyte GB-BSi7T-6500 onboard PC with an Intel Core i7-6500U CPU running at $2.5/3.1$ GHz and 16 GB of RAM. For object and drop box detection, we employ a downward facing Point Grey BFLY-U3-23S6C-C color camera with a wide-angle lens.

For allocentric localization, we use the filter result from the MAV flight control unit (FCU) fusing GNSS, barometer, and IMU data. A small Garmin Lidar Lite measures the distance to the ground to allow for exact, drift-free navigation close to the ground. To avoid electromagnetic interference between components—in particular USB3 and GPS—the core of our MAV is wrapped in electromagnetic shielding material. This increases the system stability significantly.

Our gripper is an electromagnet with a telescopic rod. The rod is passively extended to its full length by gravity and can be shortened up to 31 cm when in contact with an object. A switch detects shortening the rod. Two dampers avoid fast oscillations while still allowing the rod to align with the gravity vector. The gripper weighs 220 g including mounting and electronics.

All three MAVs are similar in hard- and software—also most of the configuration is derived from a single copter ID—to simplify the handling of multiple robots in stressful competition situations.

To make all components easily transferable between the test area at our lab and also different arenas on site, we defined all coordinates (x, y, z, θ_{yaw}) in a field-centric coordinate system. The center and orientation of the current field were broadcasted by a base station PC to all active MAVs—and the ground robot in the Grand Challenge. Furthermore, the base station PC, an Intel NUC equipped with a DJI N3 module, constantly measures its GNSS position and broadcasts position correction offsets to eliminate larger position deviations caused by atmospheric effects.

The communication among the MAVs and the base station is conducted over a stationary WiFi infrastructure. For robustness, we employ a UDP protocol that we developed for connections with low-bandwith and high-latency [7].

IV. Visual Perception

The mission plan demands for robust perception in two phases: when sweeping the field at a very high speed, the copter has to detect and track the pickable objects with low latency; after arrival in the drop zone the box has to be reliably detected and tracked during approach.

A. Object Detection

The challenge rules define two kinds of pickable objects: thin ferromagnetic disks with a diameter of 20 cm and thin rectangular objects with a size of 200×20 cm. The former were colored in red, green, blue, and yellow while the latter were exclusively orange. Since little detail was given beforehand about the competition arena and, thus, possible distracting objects, the detection algorithm was based on both color and shape information where the specific color was supposed to be trained quickly on-site when the actual objects were available. The learned color distribution is also able to model the effect of different lighting conditions and reflective object surfaces.

Briefly, the camera image is scaled down depending on the copter altitude and a bird's-eye perspective transform is applied in order to account for its attitude (see Fig. 3

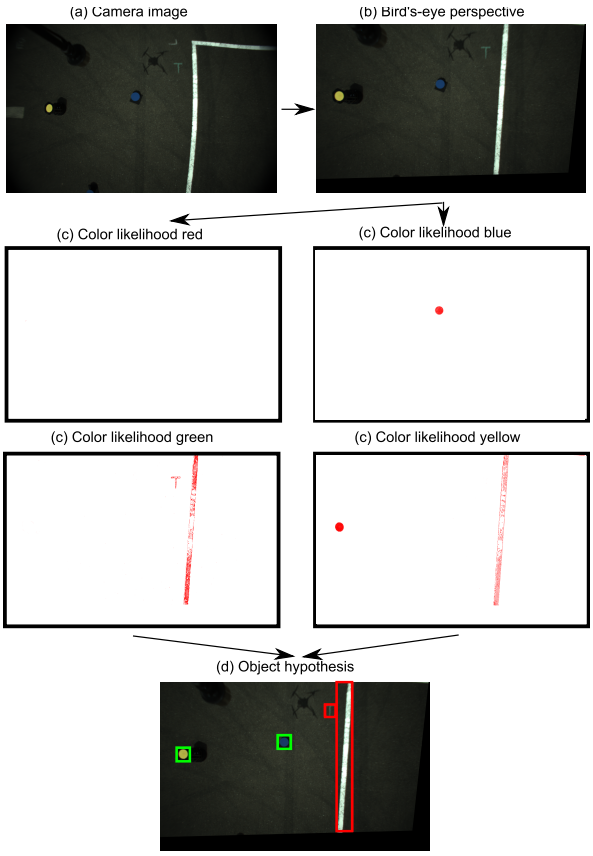


Fig. 3. Overview of the object detection pipeline: (a) original camera image, (b) undistorted bird's-eye representation, (c) color likelihood images, (d) detection hypothesis in green (accepted) and red (discarded).

(b)). Please note, that during maneuvering the camera can be significantly tilted. A pixel-wise transform is computed assigning the likelihood of belonging to one of the relevant colors which results in one likelihood image per color (see Fig. 3 (c)). A blob detection method identifies the connected regions which are then filtered by several shape (aspect ratio, convexity, size) and color (average likelihood, contrast to background) criteria (see Fig. 3 (d)).

The initial image transform serves to simplify the detection problem but also to limit the computational burden. Let therefor be r the magnitude of the shorter side of a detectable object in meters and h the MAV altitude obtained by relative barometric and laser range measurements. The image is scaled by

$$s := max\left(min\left(30 \cdot \tfrac{h}{rf}, 1.\right), 0.1\right),$$

where f is the camera focal length. For later convenience, s is rounded to the first position after the decimal point, hence, s may only obtain one of ten possible values.

For defining the bird's-eye transformation, let $\hat{r}_z$ be the IMU gravitational vector in camera coordinates. The rotation matrix

$$\begin{aligned} \hat{R} &:= (\hat{r}_x, \hat{r}_y, \hat{r}_z) \\ \text{with } \hat{r}_x &:= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T \times \hat{r}_z, \\ \hat{r}_y &:= \hat{r}_z \times \hat{r}_x, \end{aligned}$$

describes the rotation from the camera frame into a camera frame where the image plane is aligned with the ground plane, i.e., the matrix

$$M := sK_c\hat{R}K_c^{-1}$$

yields a pixel coordinate transform into the bird's-eye representation via homogeneous coordinates. Finally, taking the lens distortion into account, we arrive at

$$(u, v) \mapsto M \begin{pmatrix} d(u, v) \\ 1 \end{pmatrix} \tag{1}$$

with an invertible radial-tangential lens undistortion function d operating on the image coordinates (u, v). K_c is given by the camera intrinsics. In order to efficiently execute this transform, please note that d is static such that a pixel-wise lookup table for each of the ten possible rounded scale factors s can be precomputed. The second part of the mapping in equation (1) is linear-projective and can be computed very efficiently.

For detection processing, the color image is transformed into HSV space. As pixel-wise color likelihood we use a max-mixture of Gaussians model:

$$\max_i \left(exp\left\{-(x - c_i)^T \, diag\left(\sigma_h^2, \sigma_s^2, \sigma_v^2\right)(x - c_i)\right\}\right) \tag{2}$$

where $x = (x_h, x_s, x_v)^T$ denotes the three channel pixel value, $c_i = (c_{i,h}, c_{i,s}, c_{i,v})^T$ are a number of trained prototype pixel values, and $\sigma_h, \sigma_s, \sigma_v$ three hyperparameters. During training, the channel-wise mean of all pixels from a manually labeled object detection is computed and stored in a single prototype pixel value c_i. In order to efficiently calculate (2), a lookup-table is set up where the HSV color space is sampled with a grid of $20 \times 20 \times 20$ points.

The resulting images contain a point-wise likelihood for each of the detectable colors. As blob detection we use the implementation by Nistér and Stewénius [8] of the maximally-stable extremal regions (MSER) algorithm which yields a number of initial hypotheses. In order to select the final detections, we regard

- the size: the number of pixels of the region,
- the aspect ratio of its oriented bounding box,
- the convexity: the ratio of the number of pixels over the area of their convex hull,
- the color: the average likelihood in the region, and
- the background: the average discrepancy between the likelihood of pixels inside the region and pixels sampled from a surrounding circle.

A classifier can be trained on these quantities, but for the scope of this venture, the selection criteria were manually and individually tuned which allowed us to better follow the behavior of the detector and quickly adapt it in case of failure.

Upon a significantly confident detection, the algorithm switches to a tracking mode where only a window around the last known object position is searched for only the identified color. This enables a much faster detection rate, in particular during the picking maneuver. In close proximity

to the ground when the object is expected to be only partly visible in the image, the shape criteria are ignored when filtering the hypothesis.

It is further possible to use this algorithm to detect whether an object is attached to the gripper by filtering for very large detections in the specific color.

B. Drop Box Detection

In contrast to the pickable objects, the drop box was not specified by the challenge rules. Hence, we deployed a very general approach, only assuming that the box was rectangular and would provide some contrast to the surrounding ground. It is explicitly not assumed that the box would be uniformly colored[1]. Nevertheless, the dimensions of the box are parametrized. As for the object detection, the camera image is transformed to a bird's-eye perspective to account for the copter attitude. A Hough transform of the resulting gradient image yields line segments that are combined in a RANSAC-like procedure. In order to combine only promising pairs of line segments, a hash table with the line orientation as key is set up and only approximately perpendicular line segments are sampled. Testing the rectangularity, aspect ratio, and size of all RANSAC hypotheses provides the detection.

V. State Estimation

At the onboard PC, we use state estimation filters for maintaining a height offset between the measured height over ground and the barometer, and for estimating the position and velocity of (faster) moving objects during picking. Our generic filter design does not make any model assumptions and all dimensions are treated independently. Thus, we can employ the same filter with different dimensionality for both use cases. We modified our state estimation filter from [9] by replacing the acceleration-based prediction step with a constant velocity assumption.

A. Laser Height Correction

The operation close to the ground during picking makes a good height estimate over ground obligatory. The Matrice 100 provides absolute GNSS altitude and a barometric height to a starting position. While the first is usually not very accurate, especially close to the ground, the second is prone to drift over time. Hence, we employ a laser distance sensor in order to correct the drift. The laser measurements close to the ground are very noisy, at greater heights they are assumed to be not reliable due to bright sunlight, but if available their measurement is correct. In contrast, the barometer is very reliable and locally consistent. Thus, we maintain an offset between laser height and barometric measurements and use this offset to correct the barometer drift. To acquire the correct heights, we first transform the laser measurements into an attitude-corrected frame. If the resulting measurement is between $0.1\,\mathrm{m}$ to $6\,\mathrm{m}$, we use this value to correct the height offset. The advantage of this approach is that even without laser measurements over longer periods of time, the MAV can safely navigate in higher altitudes, e.g., to explore or deliver objects, but the filter still converges quickly to the correct height over ground when picking.

[1]As a matter of fact, it was uniformly colored.

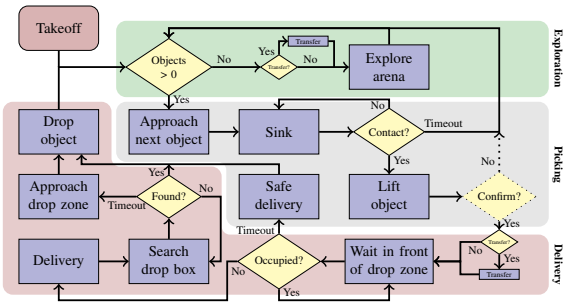


Fig. 4. Overview of our state machine. To avoid any false negatives, the dotted part was shortcut during competition.

B. Object Tracking

To track dynamic objects, we filter their positions and velocities in an allocentric frame to omit the orientation dimension. Furthermore, in contrast to our MAV state estimation filter, we only incorporate position measurements, letting the filter predict velocities without explicit correction. Outputs of the filter are allocentric 2D position and velocity estimates, used to intercept the target objects. For the very slow moving objects used in the actual MBZIRC challenges, estimating the object velocities was not necessary such that we omitted their estimation in favor of filter stability.

VI. Navigation and Control

Whereas the competition arena is of rectangular shape without larger obstacles and with good GNSS coverage, picking small objects from the ground and the coordination of a team of multiple collaborating robots pose challenges for navigation and control.

The top-level coordination is achieved by is a state machine running at $50\,\mathrm{Hz}$, depicted in Fig. 4. The state machine selects navigation targets and configures the perception and navigation modules and the hardware. After takeoff, and when the list of detected objects is empty, the system starts to explore the arena in a spiral pattern at a height of $4\,\mathrm{m}$. The maximum exploration speed is $6\,\mathrm{m/s}$. Immediately after object detection, we approach the closest object for picking[2]. When reaching a position above the detected object, we confirm its color and reconfigure the object perception to use its fast tracking mode with only one color. With visual servoing, the MAV descents within a cone around the object center until a) contact of the gripper with the object is detected, b) the measured distance to the ground from the laser falls below a safety threshold, or c) the object is not perceivable any more. If the object is no longer perceived, the MAV enters the exploration mode, in the other cases it ascents and starts visual confirmation that an object is attached. Please note that during the MBZIRC we disabled the visual object confirmation and assumed that every picking attempt was successful as false positives were far less problematic than false negatives regarding the scoring scheme. Furthermore, the gripper was much more reliable than expected.

[2]Due to the high exploration velocity, it is possible to observe multiple objects before switching to the approach state.

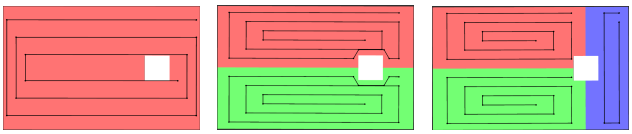


Fig. 5. Sectors for safe operation of multiple MAVs. We divide the arena into one to three sectors, depending on the number of active MAVs, all with access to the drop zone (white rectangle). The black lines depict the exploration patterns.

To drop objects, the MAV enters the drop zone at a height of 8 m and starts a local exploration flight to detect the drop box. If the drop box is detected, the MAV descents to 1 m height and drops the object. After a timeout, the MAV drops the object at the predefined center of the drop zone. As long as the drop zone is occupied, the MAV waits outside. It enters the drop zone close to its border after a timeout to drop the object safely for partial points.

The allocentric navigation is based on GNSS positions—bias corrected with help of the base station—in a field-centric coordinate system. Positions in the arena, e.g., endpoints of exploration trajectories, starting points of picking attempts, and the drop zone, are directly approached by means of our time-optimal trajectory generator [10] with maximum velocity. Except for exploration this is 8.33 m/s. We use a generic motion model in our trajectory generator combined with frequent replanning. Our controller generates attitude setpoints which are executed by the Matrice 100 onboard controller. This approach is independent from the accurate weight and other parameters that change when picking or dropping objects and, thus, robust but still efficient.

To ensure safe operation of multiple MAVs flying at high speed, we divide the arena into sectors (see Fig. 5). Sectors are derived from the number of active robots and their IDs. Within their sector, the MAVs are allowed to freely navigate below a maximum altitude. Outside their sectors, the MAVs transfer at assigned higher altitudes on straight lines.

We assume that wireless connections between agents are unreliable. Consequently, we designed our system in a way to stay operative without communication, but to employ knowledge about the other agents to operate more efficiently, if available. Our system has no central control instance or explicit negotiation between agents. The MAVs broadcast selected parts of their knowledge, namely a) allocentric 3D position, b) current navigation target, c) detected objects outside of own operation sector, d) if the MAV is flying or landed. The received information is integrated into the individual world models. If the team communication is reliable, the agents can enter the drop zone immediately if unoccupied and follow dynamic objects into neighboring sectors while picking. In case of disturbed communication, fallback strategies are in place, e.g., more conservative navigation in other sectors and time slots for entering the drop zone.

VII. Evaluation

The competition objective was to pick metal discs with a radius of 10 cm on a 20 cm stand in the colors red, green, blue, and moving discs in yellow, and deliver them to a designated drop zone. The elevated position of the disks made it necessary to pick the objects during flight without the possibility to land nearby. A white box with ground plane of one square meter—the preferred place to drop objects—was placed in the drop zone of size 10×10 m. Dropping objects next to the box but into the drop zone was awarded with half of the points. Overall, 16 discs were randomly distributed over the 100×60 m arena. Furthermore, three larger elongated objects of orange color were placed in the arena, which we were able to detect, but did not attempt to pick. The maximum challenge duration was 20 minutes. Fig. 6 shows the arena setup in the Grand Challenge. Videos of our evaluation can be found on our website[3].

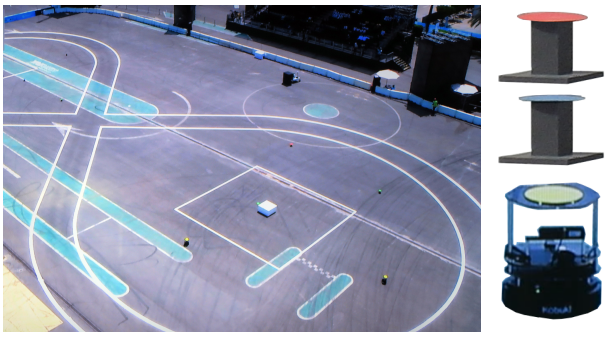


Fig. 6. Arena at MBZIRC. The colored discs randomly distributed over the arena had to be detected, picked, and dropped into the white box or surrounding drop zone. Many white lines and colored markings on the ground posed a challenge for object detection. Right: Closeup of some objects.

In the first attempt of the first trial, we began to explore the arena with three MAVs simultaneously. The trial was canceled because of very strong winds with a speed of up to 9 m/s. Qualitatively, all MAVs followed their assigned exploration trajectories until then.

In the second attempt of the first trial, we explored the arena with three MAVs and successfully picked two discs on moving bases. One of the discs could be delivered into the drop box. Before the second disc could be delivered, the referees called a reset and the MAV landed with the disc still attached. Due to conservative safety distances to the ground, we could not pick that disc after the reset. Furthermore, two MAVs arrived at the drop zone at the same time and were kept in a deadlock situation. Modifications to the system during the competition were not allowed, so we could only address these issues between trials. This was the fourth-best result of all 36 trials—18 teams with two trials per team where the better trial counted for the final score—in the Treasure Hunt and worth a third place.

The second trial took place with very strong wind. Objects were detected reliably and the descent of the MAVs was stable despite the wind, but the MAVs always had an offset of a few centimeters into the wind direction when picking.

In the first trial of the Grand Challenge, we started with three MAVs, one failed directly at takeoff due to a hardware defect. The other two explored the arena and started picking and delivering objects. As the field was not covered in full

[3] www.ais.uni-bonn.de/videos/ECMR_2017_Picking

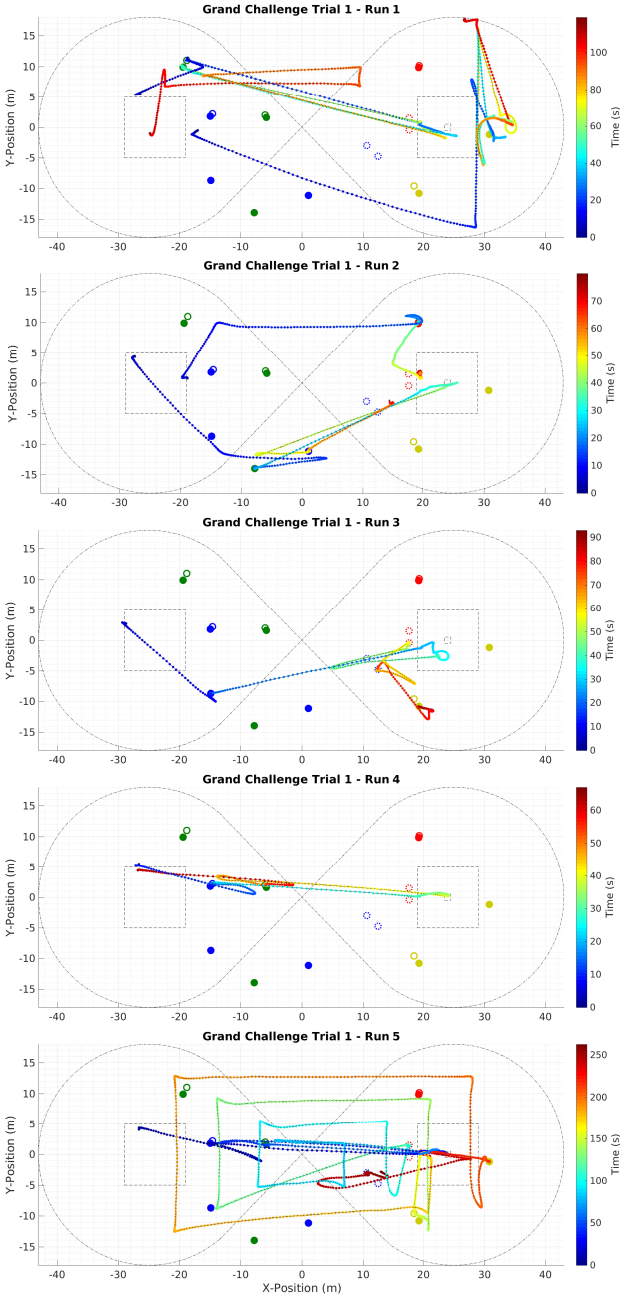


Fig. 7. Treasure Hunt in the Grand Challenge. Each image shows the trajectories of the active MAVs during the Grand Challenge (separated by 4 resets). Solid disks represent successful and rings show missed picks. The dotted disks indicate disks lying on the ground. The left rectangle is the starting zone, the right one the drop zone including the drop box. In Run 1 and Run 2, two MAVs were active. In Runs 3-5, only one MAV was active since the other one worked erroneous. It flew way to high so we had to call a reset. The following colored disks were picked (p) and missed (m) during the Grand Challenge: Run 1: m-p; Run 2: p-p-p-p (the blue and the red disk had to be put on the ground, because we called a reset. Each disk is attempted to be picked twice later); Run 3: p-m-m-p (the yellow disk was picked during a reset and had to be put back on the cart); Run 4: m-m; Run 5: p-p-m-m-p-m. Total time airborne is 624 s.

due to one missing MAV, we reconfigured the system to use only two MAVs in a reset. After a second reset due to another hardware problem, the remaining MAV operated on the whole arena. We successfully picked nine discs and were able to deliver seven of them—six into the drop zone and one into the drop box. Two discs were still attached to MAVs during a reset and, thus, were lying on the ground after resuming the trial. Overall, we scored 10.5 points and reached a second place in this Grand Challenge subtask. Fig. 7 details our trial. We canceled the second Grand Challenge trial due to severe hardware issues without a score.

VIII. Conclusion

Operating complex robotic systems without manual adaptation to the current situation and with virtually no testing time is very challenging. Whereas many highly sophisticated state-of-the-art algorithms to all subproblems of the challenge exist, simpler and failsafe solutions are often key to success. The complexity of the task is represented in the final results: From 18 teams participating in the Treasure Hunt only four were able to autonomously achieve partial task fulfillment. Five more teams were able to deliver at least one object with manual control. We came in third in the Treasure Hunt after the second competition day, and, while winning the Grand Challenge overall—in collaboration with an MAV landing on a moving target and a ground robot operating a valve—we achieved the second highest score in the Treasure Hunt sub-challenge out of 14 participating teams.

We addressed many possible issues in advance; still, unforeseen challenges occur during actual competitions, e.g., the unexpected strong wind and deadlock situations. The system could be robustified by adding more elements of randomness to the algorithms to prevent repetitive failing.

References

[1] K. Morton and L. F. Gonzalez, "Development of a robust framework for an outdoor mobile manipulation UAV," in *Proc. of IEEE Aerospace Conference (AEROCONF)*, 2016.

[2] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two dof robotic arm," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[3] V. Ghadiok, J. Goldin, and W. Ren, "Autonomous indoor aerial gripping using a quadrotor," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[4] A. Gawel, M. Kamel, T. Novkovic, J. Widauer, D. Schindler, B. Pfyffer von Altishofen, R. Siegwart, and J. Nieto, "Aerial picking and delivery of magnetic objects with MAVs," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[5] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[6] A. Tagliabue, M. Kamel, S. Verling, R. Siegwart, and J. Nieto, "Collaborative transportation using MAVs via passive force control," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[7] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, S. Schüller, D. Schwarz, A. Topalidou-Kyniazopoulou, and S. Behnke, "NimbRo Rescue: Solving disaster-response tasks through mobile manipulation robot Momaro," *Journal of Field Robotics*, vol. 34, no. 2, pp. 400–425, 2017.

[8] D. Nistér and H. Stewénius, "Linear time maximally stable extremal regions," in *Proc. of European Conf. on Computer Vision (ECCV)*, 2008.

[9] M. Nieuwenhuisen, J. Quenzel, M. Beul, D. Droeschel, S. Houben, and S. Behnke, "ChimneySpector: Autonomous MAV-based indoor chimney inspection employing 3D laser localization and textured surface reconstruction," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2017.

[10] M. Beul and S. Behnke, "Fast full state trajectory generation for multirotors," in *Proc. of Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, 2017.

A Predictive Online Path Planning and Optimization Approach for Cooperative Mobile Service Robot Navigation in Industrial Applications

Felipe Garcia Lopez[1], Jannik Abbenseth[1], Christian Henkel[1] and Stefan Dörr[1]

***Abstract*— In this paper we address the problem of online trajectory optimization and cooperative collision avoidance when multiple mobile service robots are operating in close proximity to each other. Using cooperative trajectory optimization to obtain smooth transitions in multi-agent path crossing scenarios applies to the demand for more flexibility and efficiency in industrial autonomous guided vehicle (AGV) systems.**

We introduce a general approach for online trajectory optimization in dynamic environments. It involves an elastic-band based method for time-dependent obstacle avoidance combined with a model predictive trajectory planner that takes into account the robot's kinematic and kinodynamic constraints. We augment that planning approach to be able to cope with shared trajectories of other agents and perform an potential field based cooperative trajectory optimization.

Performance and practical feasibility of the proposed approach are demonstrated in simulation as well as in real world experiments carried out on a representative set of path crossing scenarios with two industrial mobile service robots.

I. Introduction

Imagine two autonomously navigating mobile robots operating in close proximity to each other. If we command them to swap places, during execution they will perceive each other as obstacles. Independently, both robots will start to perform obstacle avoiding routines which in most cases result in sub-optimal or even infeasible path planning solutions for the overall situation.

Current trends in industrial manufacturing demand autonomous guided vehicle (AGV) systems with a high degree of autonomy in order to fulfill the requirements of flexible production systems. When it comes to path planning, these multi-agent AGV systems usually rely on cooperative route planning on a global scale that either try to completely avoid conflicts, which often results in expensive detours, or when path crossing is inevitable, e.g. at intersections, these systems implement simple right of way regulations where one or several agents need to halt to let another agent pass. However, enabling smooth transitions in multi-agent path crossing scenarios by sharing and cooperatively optimizing the agents trajectories at runtime would reduce the overall costs.

In mobile robot applications it has become a common strategy to use a decoupled approach for the motion planning problem [19]. Thereby, the problem is separated into a) *global path planning*, for offline computing collision-free paths in the robot's configuration space from its current pose to a desired goal, and b) *local motion planning*, for online trajectory planning and reactive trajectory optimization in the global path's homotopy. Local motion planning is executed periodically on a reduced spatial and temporal horizon as part of the robot's control loop. It is designed to immediately react to changes in the environment and to consider the robot's kinematic and kinodynamic constraints.

[1]Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Department for Robot and Assistive Systems, Stuttgart, Germany. Email to {flg; jba; cch; srd}@ipa.fraunhofer.de

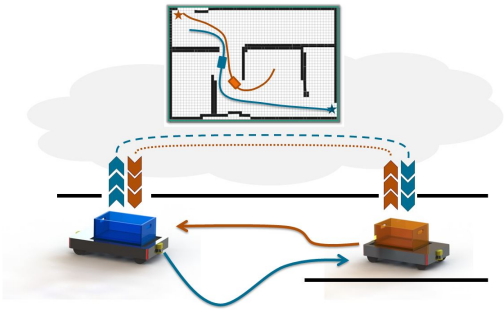


Fig. 1. Schematic illustration of two agents sharing each others planned paths using a cloud-based infrastructure and performing cooperative trajectory optimization for collision avoidance.

In previous work [3], we introduced a cloud-based navigation architecture for AGV systems in industrial environments that moves parts of the navigation tasks, particularly global path planning, off the mobile robots to a cloud server enabling cooperative global planning solutions. In that setting, however, local motion planning still needs to run detached on the local agents due to its real-time requirements.

Using the same cloud-based infrastructure, we present in this paper a solution that also enables online cooperative trajectory optimization even though the local motion planning instances run on distributed systems. For trajectory planning on the single agents we use a model predictive approach to predict the agents' trajectories over horizons of up to 20 seconds. In path crossing scenarios the predicted trajectories are shared with other agents in the shape of obstacle trajectories using the cloud infrastructure. In case of predicted collisions between two agents, the local planners perform a potential field based cooperative trajectory optimization to resolve the collisions. Figure 1 illustrates that setting.

A. Related Work

An established example for local motion planning is the dynamic window approach (DWA) [7]. A sampling-based method that continuously simulates a set of potential trajectories from a restricted velocity state space. A minimum cost trajectory is selected and applied to control the robot.

In terms of reactive obstacle avoidance our approach builds on the classic elastic-band method (EBM) [20], an efficient

potential field approach (PFA) that uses artificial forces to deform a path at runtime. The original EBM has two major drawbacks: a) As it only optimizes in a fixed configuration space homotopy, the method gets stuck in local minima in scenarios with path crossing dynamic obstacles. b) The underlying PFA does not allow to incorporate arbitrary robot-specific kinodynamic constraints on the path.

To tackle these issues, we enhance the concept of the original EBM from configuration space to configuration-time (CT) space to be able to incorporate predicted trajectories of dynamic obstacles. We combine that method with a separate model predictive control (MPC) based trajectory planner that considers the robot's kinematic and kinodynamic constraints.

In terms of incorporating predicted trajectories of dynamic obstacles, in [14] a time-dependent planning approach for social robot navigation is presented. Predicted movements of pedestrians are incorporated by using time-layered occupancy grid maps. A search-based path planning is introduced using A* with directly executable motion primitives. In [8] a trajectory deformation scheme is presented that already uses elastic-band logic in configuration-time space. The approach directly optimizes the robot's trajectory using artificial forces that are designed to ensure feasibility regarding the robot's dynamic model. In contrast, we use the EBM in CT space only for reactive obstacle avoidance in order to provide a time-dependent free space corridor around the robot's current trajectory. Actual trajectory optimization is performed inside that corridor by the underlying MPC trajectory planner.

Using corridor planning to reduce and unify search spaces for trajectory planners is a common strategy, see e.g. [23]. The original EBM already provides a concept of bubbles that model a free space corridor around the robot's path. We enhance that concept to generate a set of cylindrical bubbles in CT space to cover the robot's trajectory, see Fig. 3. A similar concept is already presented in [6] using a space-time exploration guided heuristic search technique for kinodynamic motion planning. Thereby, time-dependent cylindrical tubes are computed to provide a free space corridor. Our approach mainly differs from this w.r.t. the use of a reactive elastic-band technique.

In terms of combining EBMs with kinodynamic planning and optimal control, in [11] a decoupled space-time trajectory planning structure is introduced. It combines an EBM based path planner for obstacle avoidance with sampling-based kinematically-feasible trajectory generation. Another EBM enhancement in that regard is the TEB approach [21]. Thereby, the system model, the kinodynamic constraints and all obstacle information are transformed into a weighted regulation problem which is efficiently solved by a tailored Levenberg-Marquardt algorithm. In order to cope with path crossing dynamic obstacles, the TEB method simultaneously optimizes multiple trajectories in distinct topologies. In contrast, our approach directly incorporates predicted trajectories of dynamic obstacles and decouples reactive obstacle avoidance from actual trajectory planning. For solving resulting optimal control problems (OCPs) in our MPC trajectory planner, we use Bock's direct multiple shooting method [4].

For multi-agent systems using cooperative approaches combined with cloud computing is an arising field in mobile robot navigation. With regard to cooperative global path planning, several approaches have been introduced, e.g. [15](PRM), [5](RRT) and [3](CBS). For cooperative local motion planning, there are several approaches that focus on the special case of formation navigation, e.g. in [18] a priority based approach is presented using a tailored RRT algorithm to generate kinematically-feasible paths. During runtime an online priority strategy is introduced to avoid the mutual collisions of the robots. In [17] a multi-robot online path-planning method is introduced using an artificial bee colony algorithm. The approach is evaluated only in simulation but shows valid results in several path crossing scenarios. In [10] an optimal cooperative motion planning for vehicle coordination at intersections is presented. The approach assumes the agents' paths as fixed and focuses on optimal priority coordination and determination of suitable velocity profiles for the agents. For cooperative collision avoidance in automotive applications, a study is introduced in [9] that compares several real-time approaches in simulated path crossing scenarios. The study compares i.a. tree search, mixed-inter programming and elastic-band techniques. The presented EBM proposes additional artificial forces to resolve collisions between two vehicles. We use a similar approach in our solution by introducing additional cooperative forces to our configuration-time EBM. Thereby, the cooperative optimization is influenced by agents-wise priority scores that set the proportional intensity of the agents' cooperative forces.

For the realization of our cooperative planning architecture we use a cloud-based navigation setup that the authors presented in previous work [3]. The communication between agents and server is realized utilizing the Manufacturing Service Bus (MSB) [22], a communication middle-ware for industrial requirements.

B. Contribution

The main contribution of this work is the introduction of a novel online trajectory planning and optimization approach for cooperative multi-robot navigation. We present a general local planning approach for dynamic environments that involves an elastic-band based method for time-dependent obstacle avoidance combined with an optimal control based model predictive trajectory planner. In a practical implementation, we use a cloud-based navigation infrastructure for industrial AGV systems to online share predicted trajectories with other agents in path crossing scenarios. Predicted collisions between agents are cooperatively resolved using a priority based potential field optimization adding additional artificial forces to the EBM. We demonstrate feasibility and evaluate our cooperative planning approach in real-world experiments carried out on a set of path crossing scenarios with two industrial service robots. We also validate the approach in simulation in an intersection scenario with four mobile robots.

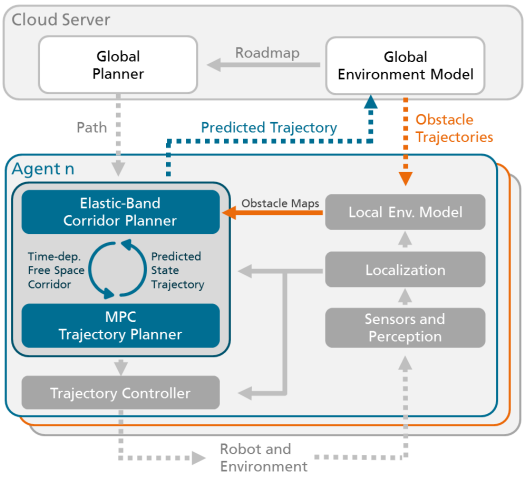


Fig. 2. Our motion planning system realized in a cloud-based navigation architecture showing the server level, agent level, the planning modules and their interconnections.

II. Motion Planning Architecture

In this section we describe the motion planning architecture that enables our online cooperative trajectory optimization approach for industrial AGV systems. Thereby, we build on the cloud-based navigation infrastructure from [3] that features two types of planning components, a cloud server and the mobile agents. The proposed architecture is depicted in Fig. 2.

The cloud server manages all parts of the navigation that are not real-time critical. On a central environment model, cooperative global path planning solutions are computed and then provided to the mobile agents via wireless communication.

On the agent level, local motion planning is periodically performed to control the robot in the global path's homotopy. We consider two main local planning modules: a) an *elastic-band based corridor planner* and b) a *model predictive trajectory planner*.

The elastic-band based corridor planner performs reactive obstacle avoidance in configuration-time space and provides a time-dependent free space corridor to the underlying MPC trajectory planner. In that way we reduce and unify the search space for trajectory planning, as we only consider a single free space constraint per time instance, no matter how many distinct obstacles are present.

The MPC trajectory planner solves finite-horizon OCPs on a cost-minimizing strategy. It considers the robot's kinematic and kinodynamic constraints, time-dependent path constraints as well as user-defined soft optimality criteria. Resulting state trajectories are fed back to the corridor planner in the shape of internal contracting forces to generate optimal free space information around the robot's current trajectory. Corresponding control trajectories are provided by the planner to a trajectory controller. The trajectory controller implements an MPC algorithm as presented in [13]. The corridor planner and the trajectory planner run as a unified module, in our implementation with a frequency of 30 Hz. The trajectory controller runs with 100 Hz.

In order to enable cooperative optimization, predicted state trajectories are provided to the central environment model on the cloud server. In path crossing scenarios, these trajectories are distributed to other agents in the shape of time-dependent obstacle information. To resolve predicted collisions between agents, we introduce additional cooperative forces to the EBM in the corridor planner.

III. Elastic-Band Corridor Planning in Configuration-Time Space

Let us consider a simple pose model for a robot moving in a plane and denote by $C = \{(x,y,\theta) : x,y \in \mathbb{R}, \theta \in [0,2\pi)\}$ the robot's configuration space, the set of all possible positions (x,y) and orientations θ of the robot. Given the robot's footprint, we decompose C into C_{free}, the collision-free configurations of the robot, and C_{obs}, the set of configurations that result in collisions with obstacles.

Let $T = [t_0,t_h] \subset \mathbb{R}$ be a time interval. We then define the configuration-time space as the direct product $C \times T = \{(q,t) : q \in C, t \in T\}$.

A. A Concept of Free Space Bubbles in $C \times T$

Operating in C, the original EBM [20] uses a concept of reactive free space bubbles $\mathbf{B}(q) \subset C_{free}$ for describing local free space around given configurations $q \in C$. It shows that every collision-free path in C can be covered with a finite number of such bubbles. We augment that concept to work in $C \times T$.

Let $\rho(q,t)$ denote the potential field function providing the distance to the closest obstacle to a robot's configuration $q \in C$ at time $t \in T$. We relax that function to provide the minimal distance over a whole time interval Δt:

$$\rho_{\Delta t}(q,t) = \min\{\rho(q,\tau) : |\tau - t| < \Delta t\}. \tag{1}$$

All configurations in that time interval which are closer to q than the value $\rho_{\Delta t}(q,t)$ are collision-free by definition. Hence, in the sense of the original EBM, we can define cylindrical bubbles of free space in $C \times T$ by

$$\mathbf{B}_{\Delta t}(q,t) = \{(\acute{q},\tau) : \|\acute{q} - q\|_C < \rho_{\Delta t}(q,t), |\tau - t| < \Delta t\}. \tag{2}$$

Let $\varphi : T \longrightarrow C$ be a collision-free trajectory. Then there is a finite equidistant time discretization $t_0,\ldots,t_n$ with $\Delta t = t_{i+1} - t_i$ and configurations $q_i \in C$ such that the trajectory is covered by corresponding free space bubbles:

$$\{(\varphi(t),t) : t \in T\} \subset \bigcup_{i=0}^{n} \mathbf{B}_{\Delta t}(q_i,t_i). \tag{3}$$

This arrangement is schematically illustrated in Fig. 3.

B. Modifying Free Space Bubbles

To optimize the shape of the corridor at runtime, we apply a set of artificial forces on the bubbles $\mathbf{B}_{\Delta t}(q_i,t_i)$. Such a force vector $F(t_i) \in C$ acts on the center pose (q_i,t_i) to move the bubble in its time layer $C \times \{t_i\}$:

$$\mathbf{B}^{\text{opt}}_{\Delta t} = \mathbf{B}_{\Delta t}(q_i + F(t_i),t_i). \tag{4}$$

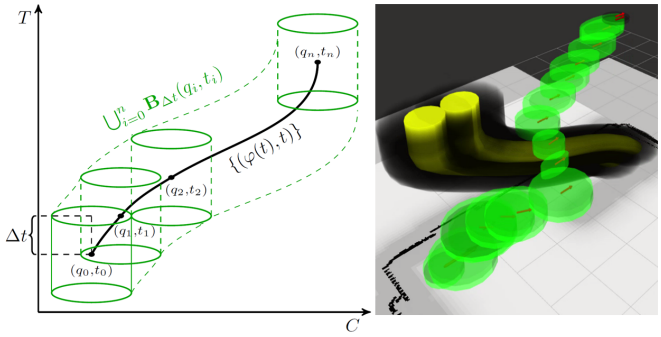


Fig. 3. Left: Schematic illustration of a trajectory $\varphi(t)$ in $C \times T$ space covered by cylindrical bubbles that represent a local free space corridor. Right: ROS-visualization of that concept in a dynamic obstacle avoidance scenario (predicted obstacle in yellow) with time as vertical dimension.

1) Internal Contracting Forces: Let $\varphi(t)$ be the robot's currently planned trajectory. To provide optimal local free space around that trajectory, we introduce internal contracting forces pushing the bubbles towards the trajectory:

$$F_{\text{int}}(t_i) = k_{\text{int}} \cdot (\varphi(t_i) - q_i), \tag{5}$$

where $k_{\text{int}} \in [0,1]$ is a configurable gain.

2) External Forces: To avoid collisions, the corridor is deformed by external forces that push the bubbles away from (dynamic) obstacles, $k_{\text{ext}} \in [0,1]$:

$$F_{\text{ext}}(t_i) = k_{\text{ext}} \cdot (\rho_{\max} - \rho_{\Delta t}(q_i,t_i)) \frac{\frac{\partial \rho_{\Delta t}}{\partial q}(q_i,t_i)}{\left\| \frac{\partial \rho_{\Delta t}}{\partial q}(q_i,t_i) \right\|_C} \tag{6}$$

with $\rho_{\max}$ being the maximal distance up to which obstacles are taken into account. As in the original EBM [20], to prevent undesired effects, only the perpendicular part of external forces $F_{\text{ext}}^{\perp}(t_i)$ must be applied. In our case, we compute that part such that $\left\langle F_{\text{ext}}^{\perp}(t_i), \dot{\varphi}(t_i) \right\rangle = 0$ holds.

3) Cooperative Forces: Let $O : T \longrightarrow \mathcal{P}(C)$ denote the trajectory of a dynamic obstacle. To resolve predicted collisions of the corridor with dynamic obstacles, say $q_i \in O(t_i)$, we replace our regular external forces with forces $F_{\text{dyn}}(t_i)$ that depend on the obstacle's direction of motion $\frac{\partial O}{\partial t}(t_i)$. We use the angle γ between vectors $\frac{\partial O}{\partial t}(t_i)$ and $(q_{i-1} - q_i)$ to obtain

$$\begin{aligned} F_{\text{dyn}}(t_i) = (1 - |\cos(\gamma)|) \cdot (\rho_{\max} - \rho_{\Delta t}(q_i,t_i))(q_{i-1} - q_i) \\ + |\cos(\gamma)| \cdot F_{\text{ext}}^{\perp}(t_i) \end{aligned} \tag{7}$$

The first term of the right-hand side of that equation pushes the bubble backward in direction of its predecessor by using the vector $(q_{i-1} - q_i)$. That means the robot is forced to slow down in that part of its path to let the dynamic obstacle pass. Proportionally, that behavior is suppressed when the obstacle moves rather tangentially to the robot's free space corridor.

For our cooperative collision avoidance, we have to take into account that all involved agents will simultaneously apply $F_{\text{dyn}}(t_i)$ to resolve predicted collisions. As full optimization steps on all agents won't be necessary, we use a priority strategy to influence the strength of $F_{\text{dyn}}(t_i)$ on the respective agents. This allows us to influence at runtime which of the two agents executes the major part of the collision avoidance routine w.r.t. the involved agents' footprints, kinematics, loads and overall agility. The cloud server provides the agents with weighting factors $k_{\text{coop}}^{(\cdot)} \in [0,1]$ such that collisions can be resolved pairwise between two agents A and B with respective forces

$$F_{\text{coop}}^{(\cdot)}(t_i) = k_{\text{coop}}^{(\cdot)} \cdot F_{\text{dyn}}^{(\cdot)}(t_i) \tag{8}$$

and $k_{\text{coop}}^{(A)} + k_{\text{coop}}^{(B)} = 1$.

4) Preserving Connectivity of the Corridor: Applying artificial forces can harm connectivity of the corridor. As in the original EBM [20], we reconnect consecutive bubbles $\mathbf{B}_{\Delta t}(q_i,t_i)$ and $\mathbf{B}_{\Delta t}(q_{i+1},t_{i+1})$ not having enough overlap by inserting an additional bubble $\mathbf{B}_{\Delta t}(q_i + \frac{1}{2}(q_{i+1} - q_i), t_{i+1})$ at time t_{i+1} and by shifting all subsequent bubbles Δt ahead in time. As time is also optimized in the MPC trajectory planner, internal contracting forces will push redundant bubbles towards the goal pose, where they can be deleted.

Algorithm 1 Optimize Free Space Corridor

Require: Current corridor $\mathbf{B}_{\Delta t}(q_i,t_i)$ for $i = 0,\ldots,n$. The robot's currently planned trajectory $\varphi(t)$. Dynamic obstacles $O_j(t)$ for $j = 0,\ldots,m$.

for $i = 0$ to n **do**
  $q_i \leftarrow (q_i + F_{\text{int}}(t_i))$ // apply internal contracting forces
  if $q_i \in O_j(t_i)$ **then** // check for collisions with dynamic obstacles
    $q_i \leftarrow (q_i + F_{\text{coop}}(t_i))$ //apply cooperative force
  else
    $q_i \leftarrow (q_i + F_{\text{ext}}^{\perp}(t_i))$ //apply regular external force
  end if
  while Not enough overlap to $\mathbf{B}_{\Delta t}(q_{i+1},t_{i+1})$ **do**
    $n \leftarrow n+1$
    for $k = i+1$ to n **do**
      $q_{k+1} \leftarrow q_k$
    end for
    $q_{i+1} \leftarrow (q_i + \frac{1}{2}(q_{i+1} - q_i))$ //insert new bubble at time t_{i+1}
  end while
end for
while $\|q_n - q_{n-1}\| < \epsilon$ **do** //delete redundant bubbles at the end of the corridor
  $(q_{n-1},t_{n-1}) \leftarrow (q_n,t_n)$
  $n \leftarrow n-1$
end while
return Optimized corridor $\mathbf{B}_{\Delta t}^{\text{opt}}(q_i,t_i)$ for $i = 0,\ldots,n$.

IV. Model Predictive Trajectory Planning

To model the system dynamics of a mobile robot, we generally consider augmented state spaces $X \hookleftarrow C$ not only representing the robot's localization (x,y,θ) but also its movement in terms of linear and angular velocities (v_x,v_y,v_θ), accelerations (a_x,a_y,a_θ), or any other quantity of interest: $\mathbf{x} = (x,y,\theta,v_x,v_y,v_\theta,\ldots) \in X$.

Let $U \subset \mathbb{R}^m$ be a set of feasible control inputs $\mathbf{u} \in U$ that modify the robot's state as a function of time t. We denote by f the right-hand side function of an ordinary differential equation (ODE) modeling the robot's dynamics: $\dot{\mathbf{x}}(t) = f(t,\mathbf{x}(t),\mathbf{u}(t))$.

A. OCP Formulation

To transfer the time-dependent path constraints provided by the corridor planner to our OCP, we seek to obtain analytical representations of the free space bubbles $\mathbf{B}_{\Delta t}(q_i,t_i)$. For $q_i = (q_{i,x},q_{i,y},q_{i,\theta})$ we extract an euclidean sphere $S(q_i,t_i) = \left\{(x,y) \in \mathbb{R}^2 : (x - q_{i,x})^2 + (y - q_{i,y})^2 \leq \rho_{\Delta t}(q_i,t_i)\right\}$ at time t_i that can be formulated as inequality constraint in our OCP.

Let $\mathbf{x}_0$ denote the robot's current state and let $\mathbf{x}_f$ be its desired goal state. The trajectory planner then iteratively computes optimal trajectories $\mathbf{x}: T \longrightarrow X$ and $\mathbf{u}: T \longrightarrow U$ by solving the following OCP on a receding horizon of finite length:

$$\min_{t_f, \mathbf{x}(t), \mathbf{u}(t)} \int_{t_0}^{t_f} L(t, \mathbf{x}(t), \mathbf{u}(t)) dt + \Phi(t_f) \tag{9a}$$

subject to

$$\dot{\mathbf{x}}(t) = f(t, \mathbf{x}(t), \mathbf{u}(t)) \tag{9b}$$

$$(\mathbf{x}(t_0), \mathbf{x}(t_f)) = (\mathbf{x}_0, \mathbf{x}_f) \tag{9c}$$

$$\forall_{t \in [t_i, t_{i+1})} \quad (x(t), y(t)) \in S(q_i, t_i) \tag{9d}$$

$$t_f \in [t_n - \Delta t, t_n] \tag{9e}$$

$$\mathbf{v}(t) = (v_x(t), v_y(t), v_\theta(t)) \in [\mathbf{v}_{\min}, \mathbf{v}_{\max}] \tag{9f}$$

$$\mathbf{u}(t) = (a_x(t), a_y(t), a_\theta(t)) \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}] \tag{9g}$$

In our implementation, the cost functional of the above OCP consists of $\Phi(t_f) = k_{t_f} \cdot t_f$ for weighted time optimization and a Lagrange term $L(\cdot) = \|\mathbf{x}(t)\|_Q^2 + \|\mathbf{u}(t)\|_R^2$. The weighting matrices R and Q allow us to configure different optimality criteria on the system's state and control trajectories.

B. Numerical Solution Method

Resulting OCPs are solved by using Bock's direct multiple shooting method [4], [16]. The infinite dimensional problem from Eq. 9 is transformed into a finite dimensional NLP, which is numerically solved by a tailored sequential quadratic programming (SQP) method, in our case using exact Hessian approximations. Among the advantages of that solver is that it allows warm starts for subsequently solving similar problems and that it supports nonlinear inequality path constraints as they arise from the corridor bounds.

V. Evaluation

We evaluated our approach in real-world experiments using two mobile service robots of type rob@work 3 [2] that are interconnected via MSB [22] to the cloud-based navigation infrastructure from [3].

A. Implementation and Communication

All navigation components, on server and agent level, run ROS implementations. We use occupancy grid maps for static sensor-based environment modeling. For dynamic obstacles we use feature maps with time-dependent multi-disc approximations of the obstacles' footprints, see right-hand side of Fig. 3. The implementation of our corridor planner builds on the ROS navigation stack [19]. For numerically solving OCPs, the trajectory planner implementation makes use of libraries provided by the ACADO Toolkit [12].

For communication purposes, a layer 3 connection is present between agents and cloud server using wireless LAN. As described in [3], we utilize the MSB as communication middle-ware. For ROS-based systems it provides a web-socket interface that allows to send any message in a *rostopic* to any other component. When navigating in path-crossing scenarios, agents share their predicted trajectories with a frequency of 2 Hz.

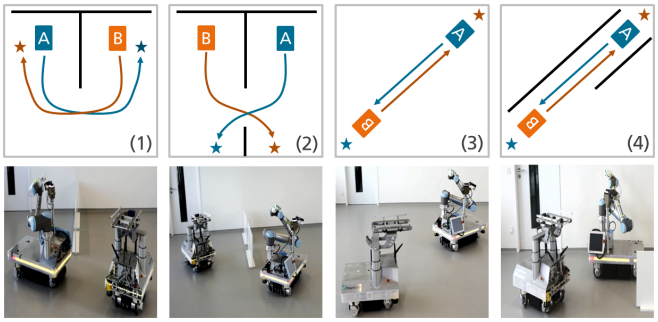


Fig. 4. Top row: Illustration of the conducted path crossing scenarios (1-4). Bottom row: Snapshots of the scenarios with two mobile robots of type rob@work 3. For videos of the experiments, visit `https://youtu.be/zaIW76A7ips`.

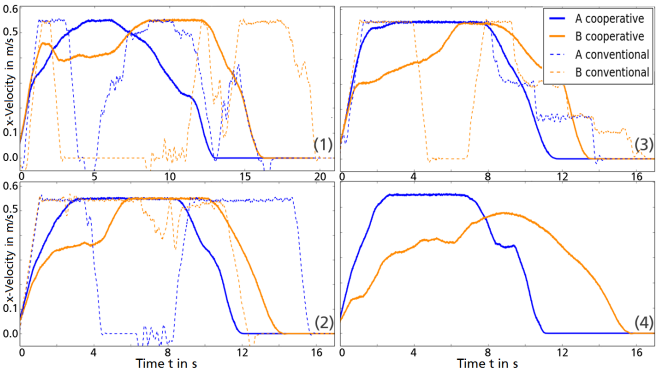


Fig. 5. Velocity profiles of the two robots for scenarios (1-4), v_x in $\frac{m}{s}$. Cooperative approach in solid lines. Conventional approach in dashed lines.

B. Experimental Setup

We use the two mobile robots on a set of basic path crossing scenarios as illustrated in Fig. 4. The robots have footprints of $1.03 \times 0.6\,m$ and operate in a $7 \times 7\,m$ workspace. The robots receive their global paths in configuration space. For cooperative optimization, dynamic obstacle trajectories are only shared at runtime, so the local motion planners need to adapt online and are challenged to sustain smooth path and velocity profiles. In comparison, we run the same scenarios with conventional single-robot path planning using a DWA implementation from the ROS navigation stack [1]. In terms of planning constraints, we consider both robots having differential drive kinematics with a state vector $\mathbf{x} = (x, y, \theta, v_x, v_\theta)$ and a control vector $\mathbf{u} = (a_x, a_\theta)$.

Thereby, velocities are bound to $|v_x| \leq 0.55\,\frac{m}{s}$ and to $|v_\theta| \leq 1.5\,\frac{rad}{s}$. For the control vector we have acceleration constraints $|a_x| \leq 1.5\,\frac{m}{s^2}$ and $|a_\theta| \leq 2.5\,\frac{rad}{s^2}$.

The elastic-band corridor planner uses $k_{\text{int}} = 0.7$, $k_{\text{ext}} = 1.0$, a time discretization of $\Delta t = 1.0\,s$ and a maximal corridor width of $\rho_{\max} = 1.0\,m$. For weighting the cost function of the trajectory planner's OCP solver, we use $k_{t_f} = 10$, $R = \text{diag}(100, 100)$ and $Q = \mathbf{0}$. We use a maximal number of 10 SQP iterations per trajectory planning instance. In terms of weighting the cooperation forces in the scenarios from Fig. 4, we use $k_{\text{coop}}^{(A)} = 0.3$ and $k_{\text{coop}}^{(B)} = 0.7$, as we consider robot B to be more agile than robot A.

In addition to these practical experiments, we performed further validation in simulation on path crossing scenarios

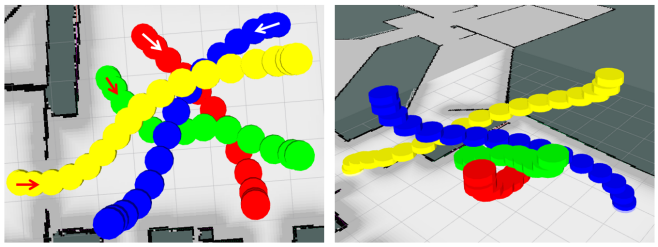


Fig. 6. Intersection scenario in simulation with paths of four circular mobile robots. Left: Top view. Right: Orbit view with time as vertical dimension.

with up to four robots, see one of the sets in Fig. 6.

C. Results

In all experimental path crossing scenarios (1-4) our approach successfully obtained smooth transitions. Thereby, robot B performed the major part of evasive movements. None of the robots had to stop to realize (1-4). Compared to conventional local planning, the cooperative approach results in smoother velocity profiles and shorter arrival times for the overall system, see Fig. 5. For all scenarios (1-4), the usage of conventional single-robot navigation resulted in collisions. We implemented a slightly larger safety stop sensor field on robot A, to force it to stop to let robot B pass when they get too close to each other. In that way feasible solutions could be obtained for scenarios (1-3) with conventional planning. In comparison to the best conventional results, the cooperative approach reduced arrival times for the overall system by 25% for scenario (1), by 13% for (2) and by 22% for (3). Video footage of the conducted experiments is available online: `https://youtu.be/zaIW76A7ips`.

All computations were performed on standard PCs featuring an Intel i7 CPU at 2.4GHz running ROS indigo under Ubuntu 14.04. In terms of planning times, we considered horizons for trajectory prediction of up to 20 seconds. In that range the average computing time for optimizing the local free space corridor is stable at $2\,ms$. The optimization times of the model predictive trajectory planner are depicted in the following table:

Trajectory Planning Times			
horizon length h in sec.	min.	mean	max.
$0 < h \leq 5s$	6 ms	8 ms	10 ms
$5 < h \leq 10s$	8 ms	11 ms	14 ms
$10 < h \leq 15s$	10 ms	13 ms	16 ms
$15 < h \leq 20s$	15 ms	17 ms	21 ms

VI. Conclusion and Outlook

This work proposes an approach for cooperative online trajectory optimization using a cloud-based architecture for mobile robot navigation. The feasibility of the approach was shown in simulation and in real world experiments on a set of path-crossing scenarios.

Even though our corridor planner copes with dynamic obstacles, to resolve predicted collisions it is still up to configuration and selection of heuristics for the robot to slow-down or to speed up. To optimally change homotopy classes in $C \times T$, we will enhance our corridor planner with an approach similar to the distinct topologies optimization described in [21].

References

[1] DWA base local planner for ROS. `http://wiki.ros.org/dwa_local_planner`. Accessed: 2017-04-25.

[2] rob@work 3, an industrial mobile service robot. `http://www.care-o-bot.de/de/rob-work.html`. Accessed: 2017-04-25.

[3] J. Abbenseth, F. Garcia Lopez, C. Henkel, and S. Dörr. Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments. In *Proceedings of the 2017 ACM Symposium on Applied Computing*, April 2017.

[4] H. Bock and K. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press.

[5] M. Čáp, P. Novák, J. Vokrínek, and M. Pechoucek. Multi-agent RRT: sampling-based cooperative pathfinding. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1–2, 2013.

[6] C. Chen, M. Rickert, and A. Knoll. Kinodynamic motion planning with space-time exploration guided heuristic search for car-like robots in dynamic environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pages 2666–2671.

[7] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics & Automation Magazine*, 1997.

[8] T. Fraichard and V. Delsart. Navigating Dynamic Environments with Trajectory Deformation. *Journal of Computing and Information Technology*, 17(1):27 – 36, 2009.

[9] C. Frese and J. Beyerer. A comparison of motion planning algorithms for cooperative collision avoidance of multiple cognitive automobiles. In *IEEE Intelligent Vehicles Symposium (IV), 2011*, pages 1156–1162.

[10] J. Gregoire, S. Bonnabel, and A. de La Fortelle. Optimal cooperative motion planning for vehicles at intersections. *CoRR*, abs/1310.7729, 2013.

[11] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J. Lee. Tunable and stable real-time trajectory planning for urban autonomous driving. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pages 250–256.

[12] B. Houska, H. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[13] B. Houska, H. J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[14] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard. Time dependent planning on a layered social cost map for human-aware robot navigation. In *2015 European Conference on Mobile Robots, ECMR 2015*.

[15] S. Kumar and S. Chakravorty. Multi-agent generalized probabilistic RoadMaps: MAGPRM. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3747–3753, 2012.

[16] D. Leineweber, I. Bauer, A. Schäfer, H. Bock, and J. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II). *Computers & Chemical Engineering*, 27:157–174, 2003.

[17] J.-H. Liang and C.-H. Lee. Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. *Adv. Eng. Softw.*, 79(C):47–56, 2015.

[18] S. Liu, D. Sun, and C. Zhu. A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming. *Robot. Comput.-Integr. Manuf.*, 30(6):589–596, Dec. 2014.

[19] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The Office Marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307. IEEE, may 2010.

[20] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807, 1993.

[21] C. Rösmann, F. Hoffmann, and T. Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153, 2017.

[22] D. Schel, C. Henkel, D. Stock, O. Meyer, G. Rauhoeft, P. Einberger, M. Stöhr, M. A. Daxer, and J. Seidelmann. Manufacturing Service Bus: an Implementation. In *11th CIRP Conf. Intell. Comput. Manuf. Eng.*, 2017.

[23] Z. Zhu, E. Schmerling, and M. Pavone. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In *54th IEEE Conference on Decision and Control, CDC 2015*, pages 835–842.

”Look At This One” Detection sharing between modality-independent classifiers for robotic discovery of people

Joris Guerry[1], Bertrand Le Saux[1] and David Filliat[2]

***Abstract*— With the advent of low-cost RGBD sensors, many solutions have been proposed for extraction and fusion of colour and depth information. In this paper, we propose new different fusion approaches of these multimodal sources for people detection. We are especially concerned by a scenario where a robot evolves in a changing environment. We extend the use of the Faster RCNN framework proposed by Girshick *et al.* [1] to this use case (i), we significantly improve performances on people detection on the InOutDoor RGBD People dataset [2] and the RGBD people dataset [3] (ii), we show these fusion handle efficiently sensor defect like complete lost of a modality (iii). Furthermore we propose a new dataset for people detection in difficult conditions: ONERA.ROOM (iv).**

I. INTRODUCTION

When exploring an unknown place, a robot can find itself in very different situations. Because of these uncertain conditions, different sources of information may be used to ensure greater detection robustness. In the context of computer vision, a colour camera (Red-Green-Blue) can be coupled to a so-called ”depth map” camera providing additional spatial information. Kinect or Xtion cameras are examples of RGBD sensors providing all of this information to the user. The low cost of these devices and their plug’n’play design have raised a recent interest for the scientific community. The nature of the information provided by each built-in sensor is different and can be complementary. For this reason, it is important to cleverly merge this information to improve classification performance and to be robust to specific failure conditions for each modality. In particular, for a robot equipped with such sensor (Figure 1) in an exploration scenario, situations such as a dark room can suppress the RGB modality, while sunny areas can suppress the depth modality.

In this paper, we focus on people detection because of its importance in many scenarios for domestic or search-and-rescue robotics. Many methods have been proposed for this detection task on RGB images, such as Histogram of Oriented Gradients for people detection [4] and Deformable Part Models [5] to name but a few. The use of a second modality like depth permits to be less dependant to colour and texture features and instead to focus on global geometric shape or object instance separation. Using such a multimodal RGBD approach, Gupta *et al.* [6] proposed a method based on the Girshick *et al.* Region-CNN [7].

[1]Joris Guerry and Bertrand Le Saux are with ONERA The French Aerospace Lab, F-91761 Palaiseau, France firstname.name@onera.fr

[2]David Filliat is with the ENSTA ParisTech, U2IS, Inria FLOWERS team, Université Paris-Saclay F-91762 Palaiseau, France david.filliat@ensta-paristech.fr

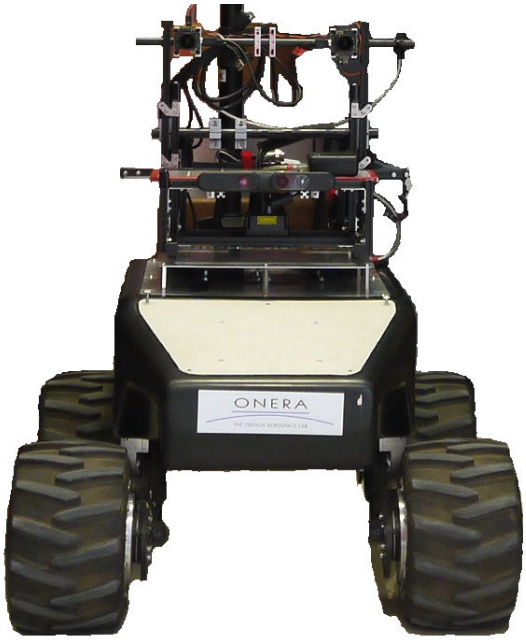


Fig. 1. Our mobile robot used for ONERA.ROOM acquisitions

They use an independent Region Of Interest (ROI) proposal module and then extract features of the region through a convolutional neural network (CNN). These features are subsequently classified by a support vector machine (SVM). The major contribution of their work is an efficient encoding of the depth information: HHA (Horizontal disparity, Angle of normal vector to gravity). However, the ROIs proposition phase as well as the HHA processing are time-consuming and the SVM makes the training process complex. Aware of this complexity problem of HHA encoding, Eitel *et al.* proposed in [8] to make a simple and fast rendering of the depth map, transforming the spatial distance into a colour information with the Jet colormap. The authors proposed to merge three CNNs, an RGB expert, a depth expert and an optical flow expert, replacing the last layer of each expert by a common fully-connected fusion layer and appending a final global softmax layer. The evolution of this work led to the use of a CNN module that weights the outputs of each expert on the fly, rather than learning the combination statically. This module, called Gating network [2], is based on features extracted at the last levels of each expert. Thus, with both class probability vector, a final vector is obtained as a weighted sum of the coefficients proposed by the Gating Network. Therefore, an expert can overwrite the information

of others if the Gating Network weights it sufficiently. Since the neural network used is non-linear, this kind of decision can lead to the complete loss of information of one of the experts.

Another multimodal fusion strategy was proposed by Hazirbas *et al.* with FuseNet [9], where the intermediate tensors of the depth expert neural network are concatenated in the RGB expert network. The convolution kernels of the depth expert remain independent of the colour, but the convolution kernels of the RGB expert must now process the depth information. If the resulting activation of the convolutional filters is not sufficient for at least a single modality, it is possible that the total activation is not sufficient for the macro network, locally blocking the information. For example, if the RGB image is too dark, only the depth expert should be able to express itself, which is not possible here. Badrinarayanan *et al.* [10] applied a new encoder-decoder CNN structure for semantic segmentation on a RGBD dataset showing better results with only RGB modality than former RGBD based methods. But this dataset was not presenting hard luminosity condition.

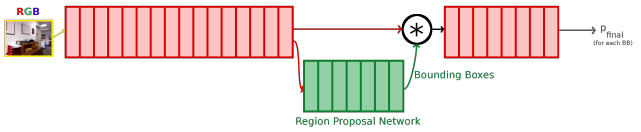


Fig. 2. Fr RCNN architecture [1]

However, some efficient methods have been also proposed for single modality. For range images [11] computes feature on the point cloud which is processed from the depth acquisitions. For RGB-only data, object detection is a very active domain. The R-CNN adapted by Gupta for RGBD has already been outperformed by Fast RCNN [12]. This is a deep network object classification method, also used in [2], which uses an ROIs proposal module independent of the classification network. In following works, the Region Proposal Network (RPN) has been added to a new version of the method: Faster RCNN [1] (abbreviated to Fr RCNN). The RPN provides regions of interest directly within the CNN architecture. The CNN ROIs provided by the RPN are then extracted from the 5th convolution layer output to be classified by the second part of the network (see Figure 2). In this paper, we study how multimodal RGBD information can be exploited in the frame of the Fr CNN approach. Indeed, more than a particular neural network, Fr RCNN is a concept that can be applied to many structures of CNN. Yet, the Fr RCNN is just made for a single source of information and, to our knowledge, has not been structurally adapted to the use of multimodal sources. We will show that these approaches significantly improve performances on people detection on the InOutDoor RGBD People dataset [2] and the RGBD people dataset [3] and that these fusion handle efficiently sensor defect like complete lost of one of the modalities. We also propose a new dataset, ONERA.ROOM, containing more challenging detection conditions acquired in a mobile robotics exploration scenario.

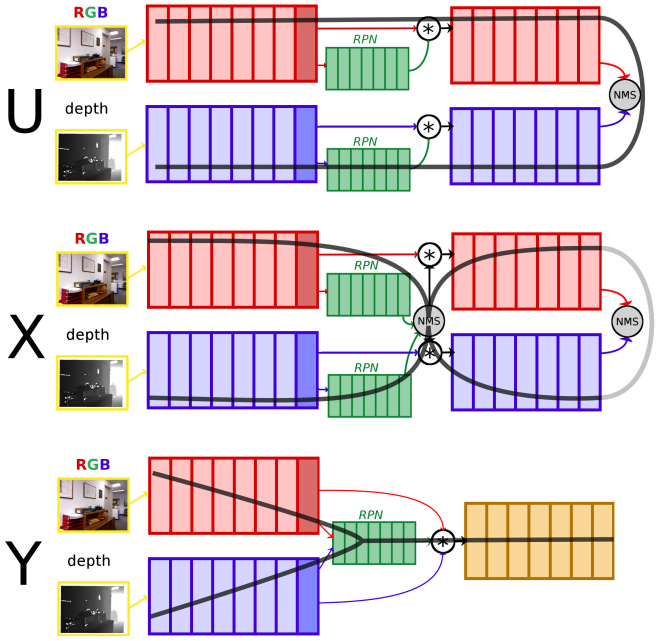


Fig. 3. Different fusion architectures, from top to bottom: U-fusion (NMS with the expert classifier outputs), X-fusion (NMS on the RPN outputs and NMS on the classifier outputs), Y-fusion (working on the concatenation of the tensors of each semi-expert).

II. JOINT-MODE APPROACH FOR OBJECT DETECTION

We propose multiple approaches for fusion of information from both modes: RGB and depth. First, we build single mode experts based on the effective region proposal module of Fr RCNN [1]. We show that this architecture can also be successfully applied to depth data. Second, the objective of fusion is to be able to adapt to realistic data in which the conditions of luminosity can completely degrade the performances of the RGB expert (in dark environment, blurry, smoky, ...) or of the depth expert (outdoor, long distance, ...). This is why the approaches we propose try to keep the experts independent (i.e avoid hybrid architectures) while allowing them to help each other.

In this paper, NMS is referring to Non-Maximum Suppression, recently called GreedyNMS in [13] as opposed to learnt NMS. This post-processing sorts and selects the best object detections among proposals in order to only keep one detection per object.

From now on, our different architectures are named RGBD RCNN (see Fig. 3) with the following variants.

- The **U** architecture is a naive version with two parallel networks. Thus, the two streams of information are only merged at the end of the detection. The fusion step is performed by NMS. This simple approach allows to have a better recall than single experts since one network can make up for the failing one but can lead to worse precision. This strategy can be applied to every computer vision technique that proposes bounding boxes.

- The **X** architecture is more subtle. Placing a common NMS after the RPN makes it possible to share the ROIs before the classification process of each expert. This pooling of detections allows an expert to share its detections with the other expert : "Look at this one !". Thus, class-blind object detections from both RGB and depth can be classified by the two experts meanwhile the redundant proposed regions are handled by the final NMS.
- The **Y** architecture aims to use only one RPN, taking as input the 5th convolution outputs of both experts concatenated in a single tensor. The underlying assumption is that a RPN which gets both RGB and depth inputs will be able to predict better ROIs. However, the RPN feature space is now twice as big as before and thus training is more complex. So, longer time is required for optimisation. A second benefit of such a fusion is that it results in a lighter architecture with a single classifier, so less parmeters to optimize and faster prediction times.

The U and X architectures are structurally independent: RGB and depth experts are trained separately and it is only at test time that ROIs are shared. They are thus incremental approaches: a new sensor modality could be added without retraining the existing ones. On the contrary, the Y architecture would require a new training with all sources. The advantage is that it would also be able to learn cross-modality features.

In this paper, our new method uses Fr RCNNs based on the VGG16 [14] network[1]. Each training is done with stochastic gradient descent for 10,000 iterations with a constant learning rate of 0.001 through Pytorch framework. Parameters are initialized with pre-trained weights on the ImageNet [16] dataset. At test time, each expert runs at 3 frames per second (fps) and the X-fusion runs at 5 fps.

III. RESULTS



Fig. 4. Examples of predictions with models trained and tested on RGBD People dataset [17], from left to right : RGB expert, depth expert, X-fusion. The depth expert was able to find all the people in the image but the X-fusion propose better bounding box alignments with the ground truth.

We compare our approach with state-of-the-art methods on two public RGBD datasets for people detection: RGBD People in part III-A and the more recent InOutDoor dataset with a moving camera in part III-B. We also run experiments of people detection with our robotic platform in even more challenging set-ups, which yields the new dataset we deliver: ONERA.ROOM (in part III-C). For performance evaluation we use standard metrics of object detection used in similar contexts [2]: Average Precision (AP), Equal Error Rate (EER), Intersection-over-Union (IoU) and the harmonic mean of the precision/recall pair (F1).

[1]Fr-RCNN was also implemented with Residual Networks [15], leading to improvement over VGG networks on many datasets but at a large computational cost

A. RGBD PEOPLE DATASET

We first test our method on the RGBD People [3] dataset. This set of over 3000 *RGBD* images was acquired with 3 static cameras in a large hall with stable lighting conditions. By reproducing the experiment in [2], we produced 5 random sets (70 % training / 30 % test) and give the averaged results in Table I, ignoring cases of occlusions . We consider a correct detection if $IoU > 0.6$. The method Fr RCNN allows a significant gain on the method proposed in [2] (+9.1 points) and the X-fusion slightly increases this result (+0.2 points). An illustration of these results is shown in Figure 4.

TABLE I
EER ON RGBD PEOPLE [17] DATASET FOR SEVERAL DETECTORS. HOD IS SHORT FOR HISTOGRAM OF ORIENTED DEPTHS AND HGE FOR HIERARCHICAL GAUSSIAN PROCESS MIXTURES OF EXPERTS.

Method	Source	EER
HOD [18]	*D*	56.3
HGE [18]	*RGBD*	87.4
Gating Net. [2]	*RGBD-Optical flow*	89.3
Fr RCNN [1]	*D*	98.3
Fr RCNN [1]	RGB	98.4
RGBD RCNN U	*RGBD*	98.4
RGBD RCNN Y	*RGBD*	98.3
RGBD RCNN X	*RGBD*	**98.6**

B. INOUTDOOR RGBD PEOPLE DATASET (IOD)

TABLE II
PERFORMANCE ON THE IOD DATASET (REFERENCE SETS [2]) FOR DIFFERENT DETECTORS.

Method	Source	Precision/Recall	AP	EER	IoU	F1
Gating Net. [2]	*RGBD*	–/81.1	80.4	–	–	–
Fr RCNN [1]	RGB	**73.2**/94.2	91.9	90.1	**80.3**	**82.4**
Fr RCNN [1]	*D*	46.9/85.9	84.0	84.8	79.4	60.7
U	*RGBD*	45.6/96.4	**94.4**	92.1	**80.3**	61.9
X	*RGBD*	43.5/**96.5**	94.3	**92.4**	79.9	60.0
Y	*RGBD*	59.4/93.3	90.2	90.1	80.2	72.6
U *	~~*RGB*~~*-D*	46.9/85.9	84.0	84.8	79.4	60.7
X *	~~*RGB*~~*-D*	45.9/85.9	84.1	84.8	79.4	59.9
Y *	~~*RGB*~~*-D*	n.a/0	n.a	n.a	0	n.a

The more challenging IOD [2] dataset is obtained by an RGBD camera embedded on a mobile robot which evolves among people in motion. It has sharp changes in brightness (indoor / outdoor) and consists in 4 sequences (8305 images) captured at different times of the day with variations in ambient light. We consider a detection is correct if $IoU > 0.6$. The Fr RCNN [1] alone, (RGB or D) already achieves a

better score than the previous state of the art [2], with a gain of 11.5 points on the AP. This important gain can be partially explained by a better recall (detection rate) thanks to the RPN. Fusion strategies further improve the score by 2.5 points, raising AP to 94.4%. A classification example on IOD is available on the left column of Figure 5.

We notice that X and U strategies have better performance than Y. The training is more delicate for the Y strategy. Indeed, the task is more complicated because the feature space is bigger for the single RPN/classifier. Moreover, this type of hybrid network can not handle the loss of one of the sources as shown by the total absence of detections for Y * in the experiments with ablation of the RGB modality (see Table II). On the contrary, U * and X * react well by equalizing the performances of the depth expert alone.



Fig. 5. Examples of predictions with models trained on IOD [2] and tested on IOD and ONERA.ROOM datasets. It is interesting to notice that the X-fusion (left column) has a new ROI. This is allowed by the intermediate NMS, post-RPN: the depth expert found an object and was not able to classify it, however, the RGB expert did not initially find this object but was finally able to classify it. The U-fusion (right column) does not allow this pre-classification detection exchange but allows to reorder, by score, the ROIs during the final NMS. In this example, the ROI from the depth expert is dismissed in favor of the RGB expert central ROI, which allows the depth expert to provide a previously ignored ROI (in yellow).

C. ONERA.ROOM

We now propose a new and more challenging dataset in a robotic exploration scenario.

Robot framework: The experimental set-up consists in a four-wheeled Robotnik Summit XL (cf. Fig. 1) equipped with a RGBD camera (Asus Xtion in its last version, but formerly Kinect v1 and Realsense camera). The sensors are linked to an embedded computer and wi-fi transmitter for remote data processing.

Dataset: ONERA.ROOM is a new data set with 27 sequences acquired by various RGBD sensors (Kinect v1, RealSense and mainly Xtion), embedded on a remote-controlled mobile robot. 23 sequences containing people have been labelled, representing 27201 ROIs of people distributed in 35379 images. Oriented to robotic scenarios for search and rescue, the ONERA.ROOM dataset includes sequences acquired in the dark, sequences blurred by the movement of the robot and cases of unconscious people on the ground. It has three main sets of increasing difficulty level: "Easy", "Average" and "Hard", and is made publicly available for research purposes on our website[2].

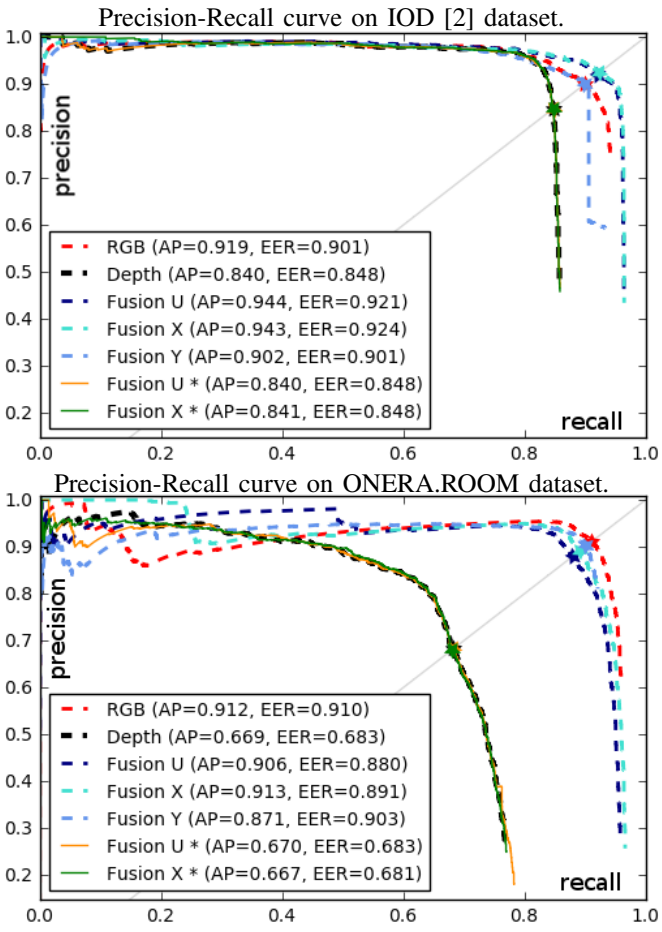


Fig. 6. The asterisk means that the source RGB was unavailable (black image). U and X-fusions behave well against this defect (see fusion curve U * and fusion X *) which fall back to the performance level of the depth expert. The X-fusion has a better AP than the RGB expert. Although the EER of the Y-fusion is good, this strategy is unable to make any detection in the event of loss of the RGB source.

Experiments: To impose a statistical independence between the training data and the test data, we used the models trained on the IOD *train* set and apply them to ONERA.ROOM. All the quantitative results on ONERA.ROOM refer to the *Easy* set. We consider a detection is correct if $IoU > 0.5$. The trends are similar to the experiments on IOD (see Figure 6 and Table III). The U and X fusions are better than the RGB expert and as good as the depth expert in the absence of light. A U-fusion classification sample is

[2]http://jorisguerry.fr/ONERA.ROOM

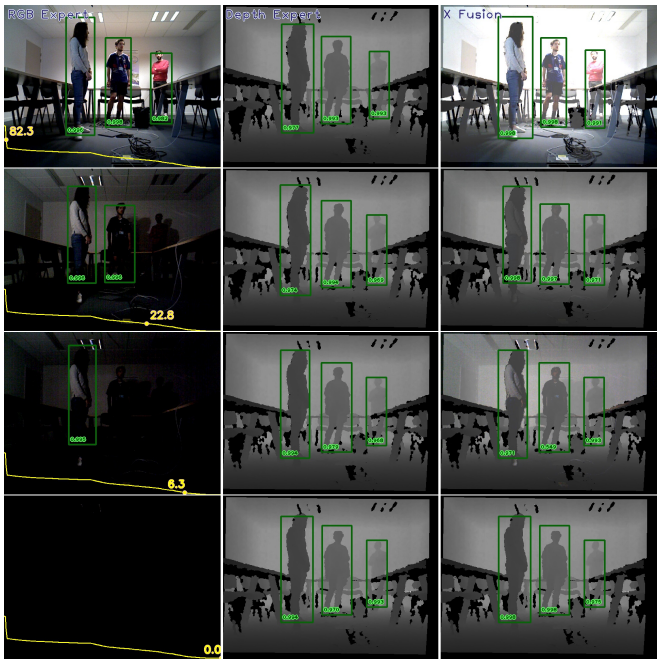


Fig. 7. Influence of decreasing luminosity. ONERA.ROOM propose a static sequence where the only changing factor is ambient light. The yellow curve indicate the mean pixel intensity. First column shows true positive detection of RGB expert, second column concerns depth expert and third column are the X-fusion detections. The last column images are made from the RGB and depth mean image for illustration purpose only.

TABLE III
PERFORMANCES ON THE *ONERA.ROOM* DATASET, "EASY" SET, FOR VARIOUS DETECTORS TRAINED ON IOD [2].

Method	Source	Precision/Recall	AP	EER	IoU	F1
Fr RCNN [1]	RGB	61.0/96.1	91.2	**91.0**	**72.8**	74.6
Fr RCNN [1]	depth	25.6/76.9	66.9	68.3	65.3	38.5
U	*RGBD*	26.7/95.8	90.6	88.0	71.1	41.8
X	*RGBD*	25.7/**96.6**	**91.3**	89.1	71.7	40.7
Y	*RGBD*	**81.1**/92.1	87.1	90.3	71.7	**86.3**
U *	~~*RGB*~~-*D*	18.2/78.3	67.0	68.3	65.3	29.5
X *	~~*RGB*~~-*D*	25.0/77.0	66.7	68.1	65.3	37.8
Y *	~~*RGB*~~-*D*	n.a/0	n.a	n.a	0	n.a

available in the right column of Figure 5. In the case of a rescue mission such an approach will be more robust to unpredictable, degraded conditions. A video illustrating several conditions is available on the website of the ONERA.ROOM dataset. Figure 7 illustrates RGB expert, depth expert and X-fusion behaviours facing luminosity reduction: when the environment is too dark for the standard RGB expert, the depth one is able to compensate and the X-fusion detects and localise the right silhouettes. Other challenging situations present in ONERA.ROOM are shown in Figures 8 (people in bright, sun-illuminated environments), Figure 9 (people crouching or lying on the ground) and Figure 10 (multiple people occluding each other). These situations are examples of the X-fusion strength versus single experts, explaining the gain of performance shown in Table III.

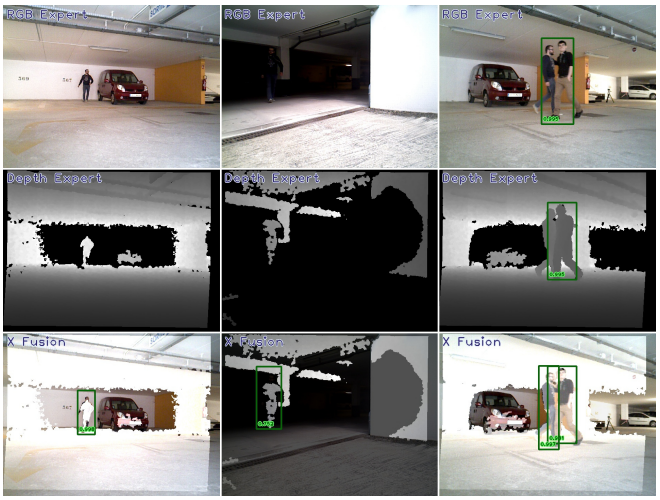


Fig. 8. X-fusion on ONERA.ROOM allows to make a detection where both RGB/D experts were impotent (left and middle column) and can differentiate two very close people mingled by both RGB/D experts (right column).

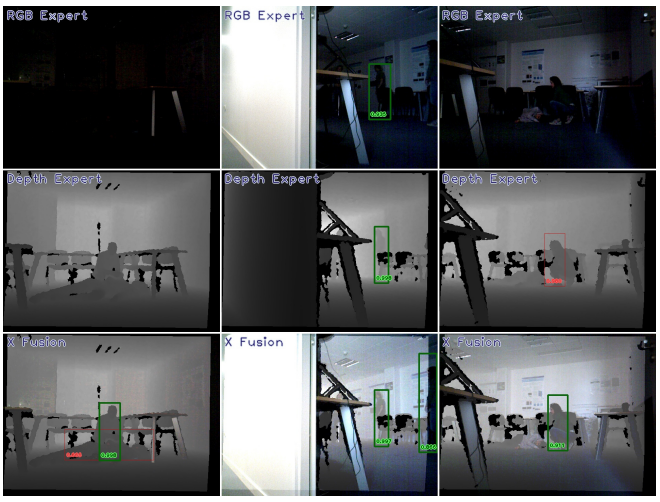


Fig. 9. X-fusion on ONERA.ROOM was close to detect the unconscious person on the floor (left column) but lacks of precision in the ROI. Still, it is the only one to detect a crouching person (left column), a person at the edge of the depth rectified image (middle column), and propose a better ROI in the last column than the depth expert.

IV. CONCLUSION

We have presented several strategies for merging the predictions of CNNs experts on different modalities. The multimodal object detection architecture based on *Faster RCNN* [1] enhances robustness in the case of heterogeneous conditions. The results on the InOutDoor RGBD People [2], RGBD People [3] and ONERA.ROOM datasets show that these strategies result in an average precision gain under normal conditions and remain robust under extreme conditions. In addition, we set new references in the state of the art on these datasets. Our best proposal, the X RGBD RCNN, gets more than 90% of AP on all these datasets and is able to withstand the failure of one of the two sensors. Lastly, we made ONERA.ROOM publicly available in the hope that it will encourage and facilitate the work on challenging RGBD

data.

V. PERSPECTIVE AND FUTURE WORK

Our future work will aim at incorporating temporal information to enable re-identification of previous detections and to implement temporal filtering of class probability vectors. A detection tracker could be seen as a third expert proposing ROIs previously revealed: "Look at this one, again!". As mentioned in [13], the NMS here is still a hand crafted processing who can be improved by deep learning. This is particularly interesting considering that both experts here are equally weighted whereas the RGB expert alone is better than the depth expert. Thus, a trained NMS could benefit from this kind of *a-priori* knowledge.

VI. ACKNOWLEDGEMENTS

We would like to thank all of the ONERA.ROOM participants and the ONERA "ATEXPA TEAM", especially Martial Sanfourche, Anthelme Bernard-Brunel, Aurélien Plyer and Hélène Roggeman for their contribution to ONERA.ROOM.



Fig. 10. X-fusion on ONERA.ROOM allows to improve ROIs proposition (left and middle columns) and is even able to detect the unconscious person (right column). However, as for the U-fusion, these strategies suffer from false positive (left column) because they can not remove a detection.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, 2015.

[2] O. Mees, A. Eitel, and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016.

[3] L. Spinello and K. O. Arras, "People detection in rgb-d data," in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886–893, IEEE, 2005.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2010.

[6] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in European Conference on Computer Vision, 2014.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014.

[8] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.

[9] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based CNN architecture," in Proc. ACCV, vol. 2, 2016.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," arXiv preprint arXiv:1511.00561, 2015.

[11] B. Steder, G. Grisetti, M. Van Loock, and W. Burgard, "Robust on-line model-based object detection from range images," in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 4739–4744, IEEE, 2009.

[12] R. Girshick, "Fast R-CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2015.

[13] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," arXiv preprint arXiv:1705.02950, 2017.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv:1512.03385, 2015.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 248–255, IEEE, 2009.

[17] M. Luber, L. Spinello, and K. O. Arras, "People tracking in RGBD data with on-line boosted target models," in Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 3844–3849, IEEE, 2011.

[18] L. Spinello and K. O. Arras, "Leveraging RGBD data: Adaptive fusion and domain adaptation for object detection," in Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 4469–4474, IEEE, 2012.

A bio-inspired celestial compass applied to an ant-inspired robot for autonomous navigation*

Julien Dupeyroux[1], Julien Diperi[1], Marc Boyron[1], Stéphane Viollet[1] and Julien Serres[1]

Abstract— **Common compass sensors used in outdoor environments are highly disturbed by unpredictable magnetic fields. This paper proposes to get inspiration from the insect navigational strategies to design a celestial compass based on the linear polarization of ultraviolet (UV) skylight. This bio-inspired compass uses only two pixels to determine the solar meridian direction angle. It consists of two UV-light photosensors topped with linear polarizers arranged orthogonally to each other as it was observed in insects' Dorsal Rim Area. The compass is embedded on our ant-inspired hexapod walking robot called Hexabot. The performances of the celestial compass under various weather and UV conditions have been investigated. Once embedded onto the robot, the sensor was first used to compensate for yaw random disturbances. We then used the compass to maintain Hexabot's heading direction constant in a straight-forward walking task over a flat terrain while being perturbated in yaw by its walking behaviour. Experiments under various meteorological conditions provided steady state heading direction errors from 0.3° (clear sky) to 1.9° (overcast sky). These results suggest interesting precision and reliability to make this new optical compass suitable for autonomous field robotics navigation tasks.**

I. INTRODUCTION

Autonomous navigation systems mostly make use of Global Positioning System (GPS) and Inertial Measurement Units (IMU). Unfortunately, the precision of GPS methods is in the range of one meter, which makes them not suitable for all robotic applications. Besides, local magnetic fields produced by ferrous materials in urban infrastructures can cause incorrect magnetometer measurements. These magnetic disturbances are classically compensated applying Kalman filtering and require data fusion from gyroscopes and accelerometers [1]. The solutions currently proposed remain sensitive to multiple sources of disturbances since gyroscopes and accelerometers are also prone to drifts. Biomimetic approach has led to interesting models for optimization, control and automation in robotics. In complex outdoor environments where magnetic interferences are increasingly present and often unpredictable, bio-inspired solutions would be interesting to abstract from magnetic pollution and GPS precision.

We propose here to get inspiration from the insects' skylight polarization compass [2] to provide a reliable measurement of the heading direction for outdoor robotic tasks. The scattering of sunlight within the Earth's atmosphere produces a polarization pattern across the sky. Solar radiations remain unpolarized until their entry into the atmosphere where scattering interactions with atmospheric constituents induce the partially linear polarization of the skylight [3]. The direction of the linear polarization of skylight at the zenith point is always perpendicular to the solar meridian[1]. The angle direction of the solar meridian is not constant and highly depends on the position of the Earth about the sun and the location of the observer.

Biological studies showed the existence of ommatidia sensitive to the polarization of skylight in the Dorsal Rim Area (DRA) of the insect's compound eye ([4], [5]). Each ommatidium is sensitive to a unique direction of polarization and its orthogonal direction, and the spectral sensitivity is generally in the ultraviolet (UV) range, except for a few species whose maximal sensitivity is in the blue range or in the green range. Many hypotheses have been proposed to explain why ommatidia are sensitive to UV light instead of other spectral ranges, but the most reasonable assumption is that the skylight polarization remains strong in the UV under canopies and clouds ([6], [7]). In the locust brain, the information perceived through the DRA is first integrated by polarization neurons, called POL-neurons, in the optic lobe, which shows a high synaptic activity for three distinct orientations (10°, 60° and 130°), while in the central complex, POL-neurons show a rather uniform synaptic activity for all polarization angles [8]. DRA-based neural models mostly provide an estimated heading direction by computing the logarithmic difference between the response of the ommatidium to a single polarization orientation, and the response of the same ommatidium to the corresponding orthogonal polarization orientation [5]. The study of the DRA in desert ants and honeybees showed that insects refer to a unique global polarization angle [9] to get their bearings. For instance, during a foraging trip in an unknown place, desert ants Cataglyphis integrate their heading direction through their celestial compass. Although their foraging trip consists of a slightly random exploration trajectory, their homing trajectory tends to be direct and straight to the nest [9].

The first autonomous wheeled robot to use a celestial compass, Sahabot 1, was created by Lambrinos et al. in the late 1990s [10]. Getting inspiration from the DRA of the cricket, the spectral sensitivity ranged from $400nm$ to $520nm$. The project sought to test three models of heading direction measurement: (a) the scanning model uses only

*This work was supported by the French Direction Générale de l'Armement (DGA), CNRS, Aix-Marseille Université, the Provence-Alpes-Côte d'Azur region and the French National Research Agency for Research (ANR) with the Equipex/Robotex project.

[1]Aix-Marseille University, CNRS, ISM, Inst Movement Sci, Marseille, France. julien.serres@univ-amu.fr

[1]The solar meridian is the circle that crosses the zenith and the sun relative to an observer on the Earth.

one polarization sensor, consisting of a pair of photodiodes mounted below orthogonally set linear polarizer, and makes the robot rotate to find angle that provides the highest sensor response. However, the rotating phase often induces 2D displacements and therefore increase the final position error in a navigational task; (b) the extended scanning model uses the same procedure as the scanning model but with three polarization sensors set at different orientations (0°, 60° and 120°) as in the optic lobe of insects. The solar meridian direction is then computed simply by subtracting the sensor signals. This method provides more reliable results since peaks detected at the corresponding linear polarization angle are sharper than in the scanning model, but 2D drift remains an issue; (c) the simultaneous model uses three polarization sensors without rotating the robot. Logarithmic differences are computed between each sub-unit of POL-sensors so that the heading direction can be correctly estimated. Tests were performed in the early morning and the average angular error was of 0.66° using the simultaneous model, and 1.73° using the simple scanning model. The simultaneous model was then applied to Sahabot 2 in order to implement ant-inspired path integration models [11]. It is still unclear how insects distinguish solar and anti-solar angles, but some suggest that insects use a circadian clock to both dispel the heading direction ambiguity and compensate the sun path [12]. Sahabot simply integrated the position of the sun to avoid any ambiguity ([10], [11]).

Chu et al. developed a celestial compass based on the one integrated in the Sahabot projects, using the simultaneous model ([13], [14]). The optical compass was embedded onto a wheeled robot using a fuzzy logic controller to follow a preprogrammed trajectory. Tests were performed at the end of the day to prevent any sensors saturation. A miniaturized version of the celestial compass has also been proposed [15] but no implementation onto a mobile robot has been recorded yet. Another implementation of the celestial compass has been embedded on a small Unmanned Aerial Vehicle (UAV) [16]. Three polarization sensors, including effective directions and their corresponding orthogonal, were integrated in an ocelli based autopilot designed to control the UAV roll and pitch over ten seconds of flight [16].

In this paper, we propose to merge both scanning and simultaneous models proposed by Lambrinos et al. into a UV-polarized light scanning model providing highly accurate measurement of the heading direction of our walking robot under various meteorological conditions and a low UV-index[2]. Section II presents the UV-polarized light compass and analyzes its performance under highly different weather and UV conditions. Section III describes the hexapod walking robot. Section IV examines two practical experiments of heading direction recovery using our celestial compass in real outdoor conditions.

[2]Experiments conducted with Sahabot were done in desert conditions whith high UV range (UV-index of 11 in Maharès, Tunisia, in August 1996). Most of the time, the sky was clear. Source: http://www.temis.nl/uvradiation/UVindex.html

II. THE UV-POLARIZED LIGHT COMPASS

A. The 3D-printed UV-polarized light sensor

The celestial compass uses two UV-light sensors mounted below rotating UV linear sheet polarizers held by two 70-teeth gears (see figure 1.B) which are driven by a third one composed of 10 teeth and actuated by a stepper motor (AM0820-A-0,225-7, Faulhaber). Due to its symmetric properties, the two UV sheet polarizers holder gears turn in the same direction. The entire prototype was printed using PLA filament (polyactic acid). The angular resolution of the compass can be modified by changing the micro-step settings of the stepper motor. The UV-light sensor is SG01D-18 (SgLux) which active area is equal to $0.5mm^2$ and spectral sensitivity is between $200nm$ and $375nm$ with a maximum spectral response at $280nm$. Each POL-unit has an angular field of view of $\pm 50°$, and a refresh rate of $33.3Hz$. The UV linear sheet polarizer has a local maximum single (resp. parallel) UV-light transmission of 52% (resp. 27%) for wavelengths between $270nm$ and $400nm$. The peak transmission is located near $330nm$.

We call POL-sensor any sub-unit of the compass composed of a UV-light photo-receptor topped with a UV sheet polarizer. The left (resp. right) POL-sensor is called UV_0 (resp. UV_1). Let x be the rotation angle of the UV sheet polarizer holder gears, and ψ the solar meridian direction angle. Therefore, the response of each POL-sensor unit is:

$$\begin{cases} UV_0(x) = A_0 + B_0 \cdot cos(2(x+\psi)) \\ UV_1(x) = A_1 + B_1 \cdot cos(2(x+\psi+\frac{\pi}{2})) \end{cases} \quad (1)$$

where $x \in [0; 2\pi]$ is the angle of rotation of the polarizers and $\psi \in [0; \pi]$ is the solar meridian direction angle, A_0 and A_1 are offsets determined by the average UV-light radiance and inner bias of each photo-sensor, B_0 and B_1 are constants determined by the degree of polarization and inner gain of each photo-sensor, and UV_0 and UV_1 are π-periodic sinusoidal functions. POL-sensors measurements are normalized so that the minimum value is set at 0 and the maximum value is set at 1. In case of bad weather conditions, B_0 and B_1 values are significantly reduced due to the weakening of the degree of polarization, implying heavy noise disturbances in POL-sensors measurements. To prevent incorrect computation of ψ, we propose to restrict the signal to its first harmonic. Let UV_0^{nc} (resp. UV_1^{nc}) be the normalized and corrected UV_0 (resp. UV_1) POL-sensor measurement:

Algorithm 1 First harmonic restriction

1: **for** $i \in [0:1]$ **do**
2: $\quad \widehat{UV_i} = \boldsymbol{fft}(UV_i)$
3: $\quad \widehat{UV_i}[2 : length(\widehat{UV_i}) - 3] = 0$
4: $\quad UV_i^n = \boldsymbol{abs}(\boldsymbol{ifft}(\widehat{UV_i}))$
5: $\quad UV_i^{nc} = UV_i^n - min(UV_i^n) + \varepsilon$
6: $\quad UV_i^{nc} = UV_i^{nc} / max(UV_i^{nc})$

where ***fft*** and ***ifft*** are respectively the direct and reverse fast Fourier transforms, and $\varepsilon = 0.0001$ is set to prevent

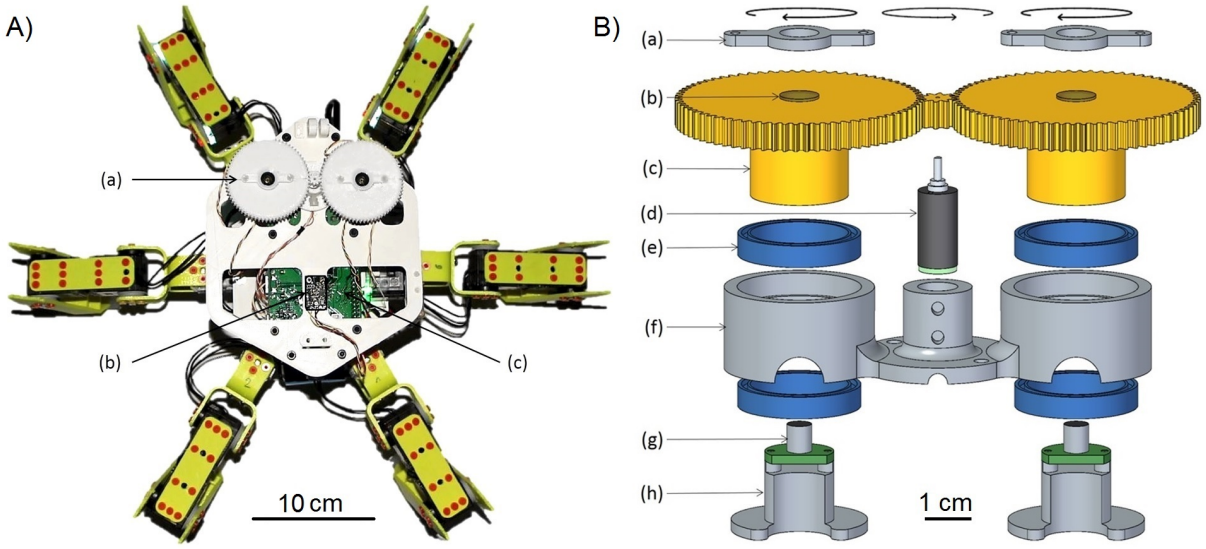


Fig. 1. **A**. Top view of the Hexabot robot equipped with the UV-polarized light compass. (a) Celestial compass; (b) MinImu-9 v.3 gyro, accelerometer & compass (Pololu) used for outdoor ground truth measurement; (c) Raspberry Pi 2B board. **B**. An exploded view of the compass. (a) 3D-printed fixation (PLA, polyactic acid) for the UV sheet polarizer; (b) UV linear sheet polarizer (HNP'B replacement, UV grade $275-750nm$); (c) 3D-printed gears (PLA); (d) stepper motor AM0820-A-0,225-7 (Faulhaber); (e) ball bearing; (f) 3D-printed support (PLA); (g) UV-light sensor SG01D-18 (SgLux); (h) 3D-printed support (PLA) for UV-light sensor.

from logarithm calculation failure. The POL-unit response is defined as follows :

$$p(x) = log_{10}\left(\frac{UV_1^{nc}(x)}{UV_0^{nc}(x)}\right). \quad (2)$$

We then compute the solar meridian direction ψ by locating the two local minimum values of the p function, the first one being in $[0;\pi]$ and the second one in $[\pi;2\pi]$:

$$\psi = \frac{1}{2}\left(\underset{x\in[0;\pi]}{\arg\min}\, p(x) + \underset{x\in[\pi;2\pi]}{\arg\min}\, p(x) - \pi\right). \quad (3)$$

due to the symmetry of the polarization pattern around the zenith point, ψ is only known within $[0;\pi]$. Classical methods to eliminate the ambiguity between ψ_{Solar} and $\psi_{Anti-Solar}$ use the ambient radiance distribution. As none of the tasks asked from the robot imply a turn back movement, there was no reason for ψ to change for $\psi+\pi$. Therefore, we assume that $\psi \in [0;\pi]$. Using the average value of the two minima of function p provides more accuracy in determining the solar meridian direction angle ψ relative to the robot.

B. Performances of the celestial compass under various weather conditions

The celestial compass was tested under various weather conditions in order to quantify its reliability and performances in determining the solar meridian angular direction. Four sets of data obtained in February and April 2017 under both clear and covered sky conditions are shown in figure 2. The UV-index was equal to 1 in February, and 7 in April according to the French meteorological services.

The table I provides the average peak-to-peak magnitudes $\overline{UV_{0,P-P}}$ (resp. $\overline{UV_{1,P-P}}$) of raw signals $UV_{0,k}$ (resp. $UV_{1,k}$), and the corresponding coefficient of variation Cv_p, where $k \in [\![1..n]\!]$ stands for the k-th test, n is the number of tests conducted, and $p \in \{0,1\}$ corresponds to the left and right POL-units. Letters standing for conditions are used with respect to the identifiers of graphs in figure 2.

TABLE I
PEAK-TO-PEAK MAGNITUDE OF RAW SIGNALS

Conditions	$\overline{UV_{0,P-P}}$	Cv_0	$\overline{UV_{1,P-P}}$	Cv_1	n
(a,b)	333.19	6%	396.00	6%	21
(c,d)	79.47	22%	124.93	22%	15
(e,f)	959.06	5%	1137.11	5%	36
(g,h)	176.11	18%	111.22	21%	36

The estimated cosine waves UV_0^{nc} and UV_1^{nc} were compared to the normalized raw signals by applying the Mean Squared Error (MSE) method. Therefore, the error $\varepsilon_{p,k}$ is defined as

$$\varepsilon_{p,k} = MSE(UV_{p,k}^{nc}) = \frac{1}{N}\sum_{i=1}^{N}\left(UV_{p,k}^{nc}(i) - UV_{p,k}^{n}(i)\right)^2 \quad (4)$$

where N is the total length of the signal. Due to some improvements made in the stepper-motor control, N got risen from 280 in February to 373 in April. Finally, in table II we compute the average error $\overline{\varepsilon_p}$ as the mean value of all $\varepsilon_{p,k}$ for $k \in [\![1..n]\!]$. The coefficients of variation $Cv[\varepsilon_p]$ were also calculated.

The overall results show that it is clearly impossible to distinguish whether the estimation of the angular direction ψ was done in winter or not, and under clear sky or not. We notice that both average errors and coefficients of variation are highly similar between February and April at a given weather condition. Besides, though the variability of signals is heavily increased under covered sky, the celestial compass

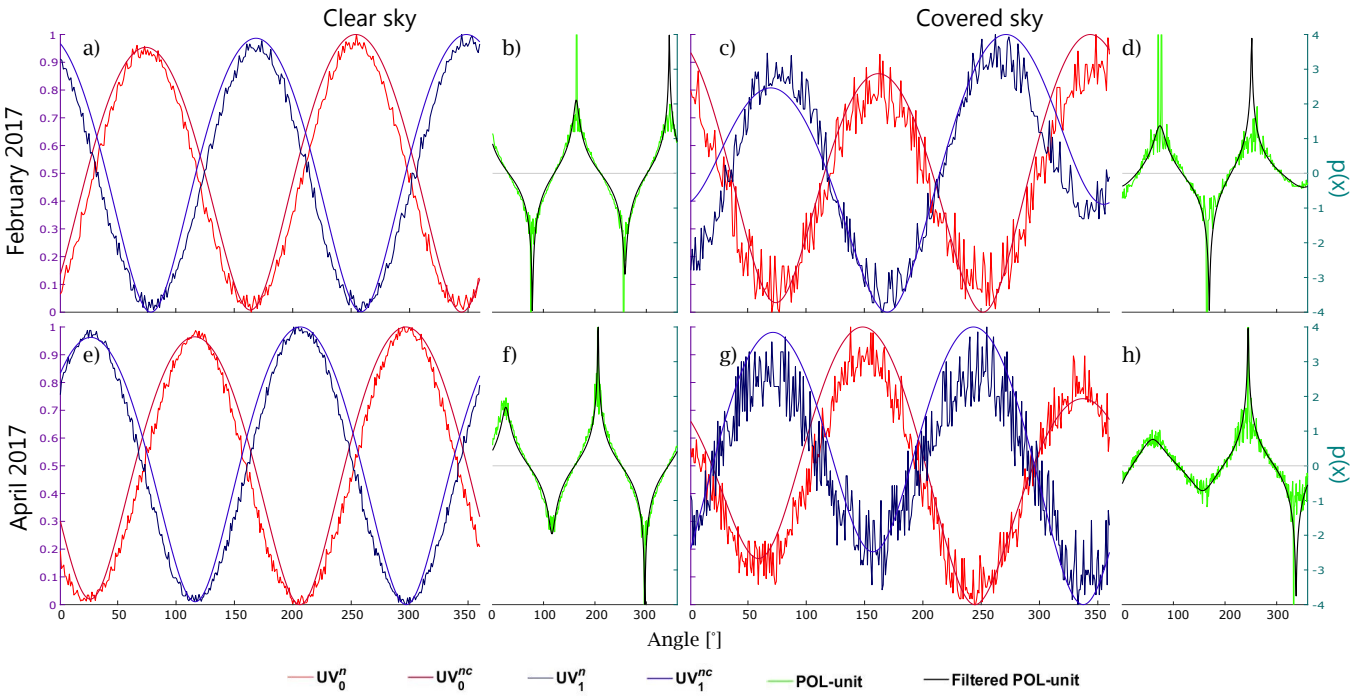


Fig. 2. Examples of signals obtained during several acquisitions from the UV-polarized light compass. Each pairs of graphs (a,b), (c,d), (e,f) and (g,h) show normalized raw and filtered outputs for UV_0 and UV_1 photo-sensors (left), and the POL-unit response for both raw and filtered data(right). Pair (a,b) was obtained in February 2017 under clear sky conditions with a UV index equal to 1, pair (c,d) at the same period but under covered sky. Pairs (e,f) and (g,h) were obtained in April 2017 with a UV index of 7 - the first one under clear sky, the second one under covered sky.

TABLE II
STEADY STATE ERROR BETWEEN NORMALIZED AND FILTERED DATA

Conditions	$\overline{\varepsilon_0}$	$Cv[\varepsilon_0]$	$\overline{\varepsilon_1}$	$Cv[\varepsilon_1]$	n
(a,b)	4.28e-03	6%	4.83e-03	4%	21
(c,d)	9.02e-03	36%	7.31e-03	32%	15
(e,f)	3.99e-03	10%	4.14e-03	5%	36
(g,h)	6.14e-03	27%	8.36e-03	19%	36

remains fully able to accurately estimate ψ. The restriction of the signal to its first harmonic is consequently an interesting approach to provide a good estimate of the solar meridian angle direction ψ without introducing delay.

III. HEXABOT : THE ANT-INSPIRED ROBOT

A. The robot platform

A fully open source, 3D-printed, six-legged walking robot called Hexabot[3] (figure 3) has been developped to mimic the desert ant in several navigational tasks such as homing in unknown environments. The overall weight of Hexabot, including batteries, is $925g$, and the maximum length is $360mm$ with a maximum height of the center of mass is $145mm$. Its battery endurance, depending on the capacity, ranges from half an hour to one hour. The robot has three DYNAMIXEL XL-320 actuators per leg which allows to reach high walking speed (approximately $35cm/s$ in optimal conditions) and execute complex motion when crossing over an uneven terrain. Besides, six-legged robots show more stable walking motion than the four-legged ones since they can operate static gait (*i.e.* three to five legs remain on the ground at any time).

[3]Based on Metabot, a quadruped walking robot. See http://metabot.cc/



Fig. 3. Hexabot robot equiped with a pair of UV-polarized light sensors forming a celestial compass.

B. Robot electronic architecture

An OpenCM9.04C micro-controller (32-bit ARM Cortex-M3) controls the robot. This first board is connected to a second one, a Raspberry Pi 2B board (32-bit quad-core

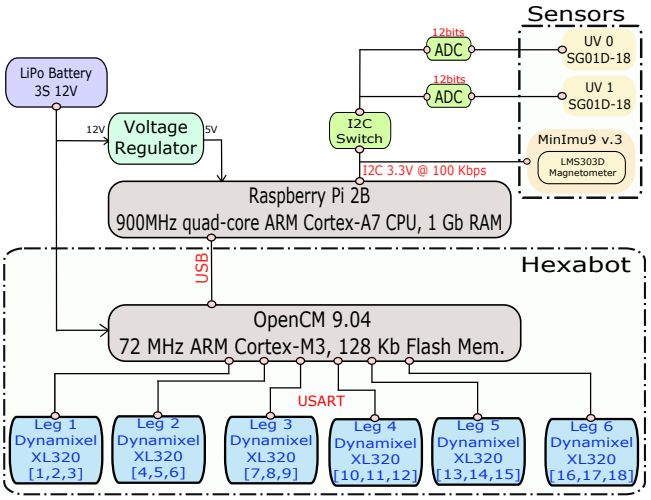


Fig. 4. Robot electronic architecture. The dashed line marks out the robot controller and actuators. The magnetometer and UV-polarized light sensors (in yellow) are connected to the Raspberry Pi 2B board using I2C communication protocol.

ARM Cortex-A7) which achieves sensor data acquisition and processing, and send high level orders the robot controller. The celestial compass is embedded on the dorsal part of the robot (see figure 1.A). Communication with the Raspberry Pi board was implemented by means of I2C protocol (figure 4).

IV. EXPERIMENTS

Hexabot was set to tripod gait for all navigational tasks as it provides an optimal compromise between high walking speed and moderate attitude disturbances. However, interactions between legs and the ground tend to cause a large drift of the trajectory. We propose here to use our UV-polarized light compass to contain the drift occurring after each stride. All experiments were done between 02/02/2017 and 02/20/2017 in outdoor conditions, at any time of the day, and were located in an open-air car park in the Luminy campus ($43°14'01.6''N$; $5°26'39.2''E$) of Aix-Marseille Université, Marseille, France. The angular precision of the UV-light polarized compass was arbitrarily set at $1.29°$ for all experiments. The acquisition time was consequently of $42s$.

Drift measurements were made for five outdoor straight-forward walking tasks on a flat but rough terrain. Tests were conducted over six seconds at maximum walking speed. Results show an average heading direction disturbance of $28°$ (magnetometer measurements). In those conditions, Hexabot drifts from initial walking axis by an average length of one meter.

A. Recovery of orientation under various weather conditions

The ability to reorientate Hexabot after a random yaw disturbance was first tested. The reorientation tasks consists of : (1) the robot acquires its initial heading direction with the celestial compass; (2) the robot turns by a random angle and then acquires its new heading direction; (3) the robot computes the difference between the initial and the new heading direction and uses it as an order to turn back to its initial orientation. Finally, we compare the ground truth measurements (magnetometer[4]) before the disturbance and after the angular correction. To prevent any yaw ambiguity, disturbances were set between $-70°$ and $+70°$. Such a restriction makes sense as angular drifts over a straight-forward walking task are systematically small (less than $30°$ over a 3-meter walk) but randomly oriented in the both directions due to interactions with the ground.

Figure 5 shows heading errors under three different weather conditions. The variability of performances in orientation recovery can be explained by the non-regularity of Hexabot's stride length while turning due to interactions with the ground. In view of this, the presented results exhibit great performance, especially under clear sky conditions. The decrease noticed under cloudy sky and overcast sky conditions stems from the low degree of polarization of the skylight [6] which implies the overpowering of the Rayleigh scattering, thus disturbing the polarization pattern of the skylight.

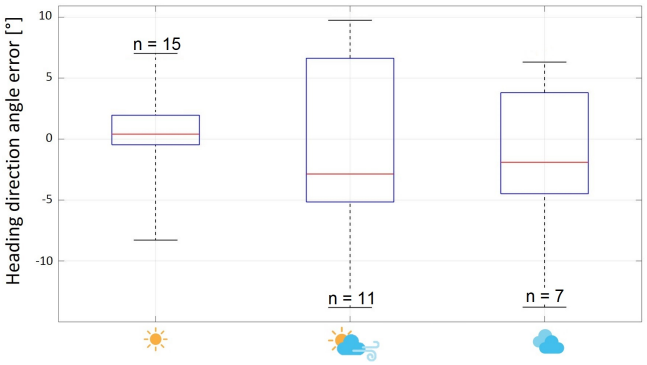


Fig. 5. Heading direction angle errors in degrees in function of weather conditions. From left to right, the median heading direction angle error measured is equal to $0.4°$, $-2.9°$, and $-1.9°$. UV-index from 1 to 2 (source: French Meteorological services).

Considering the results obtained with Sahabot robot in [10], our UV-polarized light compass provides similar and even slightly better results under clear sky, and promising results under bad meteorological conditions such as clouds in the sky and a much lower level of UV radiance due to the period and the location of experiments.

B. Heading-lock over a straight-forward walking task

As mentioned above, Hexabot exhibits important drifts in yaw orientation and the celestial compass successfully corrected yaw disturbances at fix point. We now test this ability to maintain the robot within a straight-forward trajectory applying yaw correction after each walking step. First, the initial angle is acquired. Then Hexabot executes a series of strides during two seconds and measures its new yaw orientation, the value of which is compared to the initial one to compute the yaw angle correction to be applied. Finally, Hexabot executes the corresponding turning

[4]Experiments were conducted with a calibrated magnetometer and away from any magnetic fields interference.

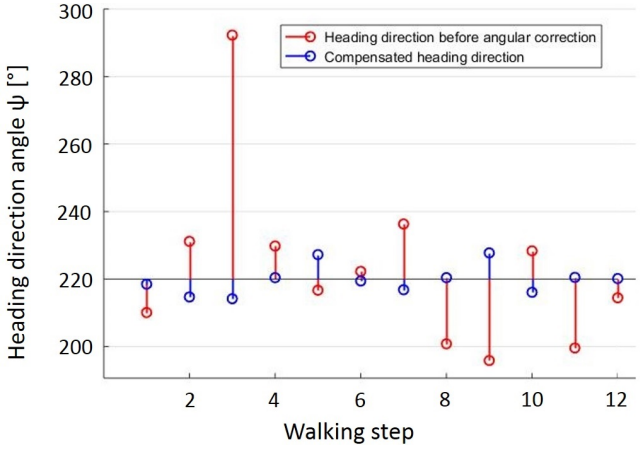


Fig. 6. Evolution of the measured heading direction angle ψ before (in red) and after (in blue) angular correction using only the UV-polarized light compass during a straight-forward walking task. Walking steps measurements from 1 to 6 were acquired on 02/18/2017 while the next six measurements were acquired on 02/20/2017.

movement before moving to the next series of strides. Due to power supply limits and to avoid the impact of polarization shifts induced by the sun movements, data were acquired over two distinct days (02/18/2017 and 02/20/2017), but experiments were all performed at the same time (2:00 pm) under perfectly clear sky conditions with a UV index of 2.

Results for all experiments are shown in figure 6. The average heading angle error measured is of $-0.3°$ which is consistent with performances exhibited previously under clear sky conditions. The peak error measured is of $7.7°$, occurring during the ninth walking step. Since there were no clouds in the sky, the polarization pattern remained rather constant all over the experiments. As a consequence, the heading direction error is mainly caused by interactions between legs and the ground.

V. CONCLUSION

In this paper, a novel insect-inspired celestial compass was introduced. Performances analysis during winter and spring showed this compass can be used in all weather conditions, including high and low UV-index, clear and covered sky. The sensor was then embedded onto an ant-inspired walking robot to maintain the robot's heading direction constant while walking.

The heading recovery experiments performed both at fix point and during a walking task showed highly precise and reliable results under clear-sky conditions, with an average steady state error as small as $0.3°$. Results under cloudy-sky conditions also exhibits good performances, from $0.8°$ under variable weather, to $1.9°$ under overcast sky, but slightly less reliable due to the high variability of meteorological conditions.

Future work will focus on the impact of the turning uncertainty of the robot on the heading direction. Residual heading-lock errors can be reduced by changing to a closed-loop system showing the suitability of this new optical compass sensor for autonomous robotic tasks such as homing.

ACKNOWLEDGMENT

The authors would like to thank Grégoire Passault and Olivier Ly for their investment to the maintenance of the hexapod robot.

REFERENCES

[1] E. Bergamini, G. Ligorio, A. Summa, G. Vannozzi, A. Cappozzo, and A. Sabatini. Estimating orientation using magnetic and inertial sensors and different sensor fusion approaches: accuracy assessment in manual and locomotion tasks, Sensors, vol. 14, no 10, pp. 18625-18649, 2014.

[2] R. Wehner, B. Michel, and P. Antonsen, Visual navigation in insects: coupling of egocentric and geocentric information, Journal of Experimental Biology, vol. 199, pp. 129-140, 1996.

[3] K.L. Coulson, Polarization and Intensity of Light in the Atmosphere, A Deepak Pub, 1988.

[4] T. Labhart, and E.P. Meyer, Detectors for polarized skylight in insects: a survey of ommatidial specializations in the dorsal rim area of the compound eye, Microscopy research and technique, vol. 47, no 6, pp. 368-379, 1999.

[5] T. Labhart, Polarization-opponent interneurons in the insect visual system, Nature, vol. 331, no 6155, pp. 435-437, 1988.

[6] M.L. Brines, and J.L. Gould, Skylight polarization patterns and animal orientation, J. exp. Biol, vol. 96, pp. 69-91, 1982.

[7] A. Barta, and G. Horváth, Why is it advantageous for animals to detect celestial polarization in the ultraviolet? Skylight polarization under clouds and canopies is strongest in the UV, Journal of Theoretical Biology, vol. 226, no 4, pp. 429-437, 2004.

[8] S. Heinze, and U. Homberg, Linking the input to the output: new sets of neurons complement the polarization vision network in the locust central complex, Journal of Neuroscience, vol. 29, no 15, pp. 4911-4921, 2009.

[9] R. Wehner, Desert ant navigation: how miniature brains solve complex tasks, Journal of Comparative Physiology A, vol. 189, no 8, pp. 579-588, 2003.

[10] D. Lambrinos, H. Kobayashi, R. Pfeifer, M. Maris, T. Labhart and R. Wehner, An autonomous agent navigating with a polarized light compass, Adaptive Behavior, vol. 6, pp. 131-161, 1997.

[11] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner, A mobile robot employing insect strategies for navigation, Robotics and Autonomous systems, vol. 30, pp. 39-64, 2000.

[12] R. Wehner, and S. Wehner, Insect navigation: use of maps or Ariadne's thread?, Ethology Ecology & Evolution, vol. 2, no 1, pp. 27-48, 1990.

[13] J. Chu, K. Zhao, Q. Zhang, and T. Wang, Construction and performance test of a novel polarization sensor for navigation, Sensors and Actuators A: Physical, vol. 148, no 1, pp. 75-82, 2008.

[14] J. Chu, H. Wang, W. Chen, and R. Li, Application of a novel polarization sensor to mobile robot navigation, Mechatronics and Automation, 2009, ICMA 2009, International Conference on. IEEE, pp. 3763-3768, 2009.

[15] J. Chu, Z. Wang, L. Guan, Z. Liu, Y. Wang, and R. Zhang, Integrated polarization dependent photodetector and its application for polarization navigation, IEEE Photonics Technol. Lett, vol. 26, no 5, p. 469-472, 2014.

[16] J. Chahl, and A. Mizutani, Biomimetic attitude and orientation sensors, IEEE Sensors Journal, vol. 12, no 2, pp. 289-297, 2012.

[17] G. Passault, Q. Rouxel, F. Petit, and O. Ly, Metabot: a low-cost legged robotics platform for education, In : Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on. IEEE, pp. 283-287, 2016.

[18] J. Dupeyroux, G. Passault, F. Ruffier, S. Viollet, and J. Serres, Hexabot: a small 3D-printed six-legged walking robot designed for desert ant-like navigation tasks, 20th World Congress of the International Federation of Automatic Control, Toulouse, France, in press.

A Fully Automatic Hand-Eye Calibration System

Morris Antonello, Andrea Gobbi, Stefano Michieletto, Stefano Ghidoni and Emanuele Menegatti

***Abstract*— To retrieve the 3D coordinates of an object in the robot workspace is a fundamental capability for industrial and service applications. This can be achieved by means of a camera mounted on the robot end-effector only if the hand-eye transformation is known. The standard calibration process requires to view a calibration pattern, e.g. a checkerboard, from several different perspectives. This work extends the standard approach performing calibration pattern localization and hand-eye calibration in a fully automatic way. A two phase procedure has been developed and tested in both simulated and real scenarios, demonstrating that the automatic calibration reaches the same performance level of a standard procedure, while avoiding any human intervention. As a final contribution, the source code for an automatic and robust calibration is released.**

I. INTRODUCTION

The hand-eye calibration problem consists in estimating the rotation and translation of the end effector or gripper of a robot, i.e. the hand, with respect to the camera mounted on it, i.e. the eye. As reviewed in [1], a number of typical robot tasks like visual servoing [2], grasping and stereo vision requires the knowledge of this transformation or, at least, could highly benefit from it. This way, the robotic arm can move the camera with respect to a fixed reference system, e.g. the robot base or the workcell world reference. Unfortunately, in most applications the hand-eye transformation needs to be calculated several times during the life cycle of a system, thus requiring many manual interventions.

A well-known approach was proposed by Tsai et al. [1] in 1988. Such approach requires the user to acquire a large set of images (more than 50) framing a calibration pattern (usually a checkerboard) from different views and the corresponding set of robot poses. Its implementation is already available in the Robot Operating System (ROS) [3], a framework widely used in robotics, and in the Visual Servoing Platform (ViSP) [4], a library for developing visual servoing systems. This task can be partially automatized by fixing the pattern position inside the robotic work-cell and saving a pre-determined robot motion. Even if this can be easily done with recent robotic arms, there are some drawbacks: it might not be possible to fix the pattern position, and this solution would not work after any change in the work-cell or robot-camera configuration. An automatic approach automatizing the robot motion has been proposed by Tsai et al. [5] but, given that it requires the pattern calibration, i.e. the knowledge of the pattern position in the world, it shares the same drawbacks. Indeed, the pattern localization can be cumbersome. It is usually achieved by means of two standard procedures described in the robot vendor manuals: the calibration of a tip mounted on the end effector and the calibration of the work object, in this case the calibration pattern. The tip allows to accurately touch three points on the calibration pattern so as to define a Cartesian reference system on it.

In this paper, such standard approach [1], [5] is extended so as to perform both the calibration pattern localization and hand-eye calibration in a fully automatic way. This is a chicken-and-egg problem: to localize the pattern by means of the camera, the hand-eye calibration is needed; however, in order to automatize the hand-eye calibration, the pattern localization is also needed. Here, this is overcome by means of two phases, in each of which an hand-eye calibration is performed as shown in the overview in Figure 1. In particular, our main contributions are:

- A fully automatic hand-eye calibration procedure, which does not require the knowledge of the calibration pattern 3D location;
- An open-source module[1] based on the framework ROS and the libraries ViSP and MoveIT! [6], which can be easily exploited on other robot platforms.

These results could speed up the research of the wide community working with robotic manipulators on complex and precise manufacturing tasks. In the challenge EuRoC[2][7], the hand-eye transformation was necessary for visual servoing in tasks like car door assembly [8] and coil winding [9], [10]. In the challenge MBZIRC[3], many mobile manipulators were equipped with a camera on the robotic arm. The hand-eye transformation was necessary for grasping a wrench and rotating a valve stem. Furthermore, from our experience, the hand-eye transformation turns out to be useful also in other specific scenarios. In the context of the EU project FibreMap[4][11], it allowed to inspect a carbon fiber preform in order to analyze its surface with a high accuracy and map carbon fibres on its model [12]. In ThermoBot[5], it allowed to subtract the background from thermographic images, thus improving the speed and reducing false positives when detecting defects in carbon fiber parts [13]. Different examples of robot hand-eye configurations are reported in Figure 2. In our real experiments, the configuration in Figure 2(a) has been adopted.

The authors are with the Intelligent Autonomous Systems Laboratory (IAS-Lab), Department of Information Engineering (DEI), University of Padova, Via Ognissanti 72, 35129, Padova, Italy. morris.antonello, michieletto, ghidoni, emg@dei.unipd.it

[1] http://robotics.dei.unipd.it/129-auto-hand-eye
[2] http://www.euroc-project.eu/
[3] http://www.mbzirc.com/
[4] http://fibremap.eu/
[5] http://thermobot.eu/

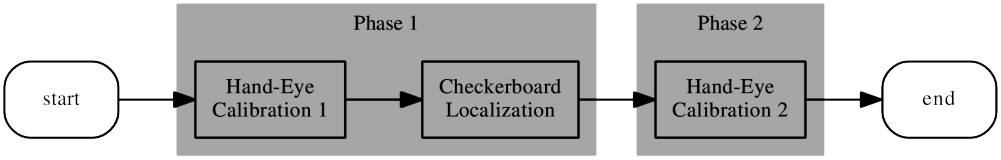
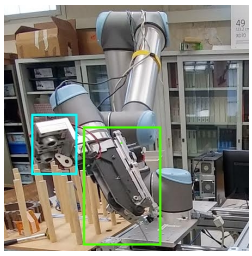
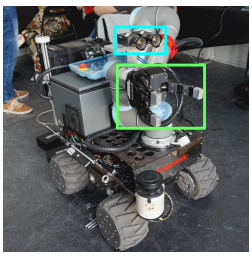


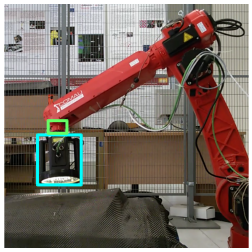
Fig. 1. The proposed two phase procedure. In the first phase, a raw hand-eye calibration is calculated with images grabbed while the robot is moving in the robot base or world reference system. Hence, the checkerboard can be localized. In the second phase, a refined hand-eye calibration is calculated using images grabbed with the robot moving in the checkerboard reference system.



(a)



(b)



(c)

Fig. 2. The robot setup exploited in this paper is similar to that of the project EuRoC (a). Other two hand-eye configurations, respectively in the challenge MBZIRC (Desert Lion Team) (b) and the project FibreMap (c), are reported. The hand is in the green box, the eye is in the cyan box.

The remainder of the paper is organized as follows. Section II reviews the work related to the hand-eye calibration problem. Section III describes our novel approach, first giving a picture of the entire workflow, then focusing on the two steps with an eye on the framework adaptability to other robots. In Section IV, our methods are thoroughly evaluated in a simulated and a real environment. Finally, in Section V, conclusions are drawn and future directions of research identified.

II. RELATED WORK

In the past, many methods have been proposed to perform the hand-eye calibration. As reviewed in [1], they can be divided into two categories: approaches coupling hand-eye calibration with conventional robot kinematic model calibration like [14] and approaches decoupling hand-eye calibration from conventional robot kinematic model calibration like [15]. In [1], a new approach belonging to the second category is also proposed, which is faster and more accurate. This is now a common approach implemented in ROS and ViSP.

In [5], Tsai et al. discussed what are the main sources of error. In order to keep the errors low, they suggested to move the robot in different positions, the so-called stations, for each of which the robot is stopped and an image is acquired. They are chosen with a star-drawing technique giving a systematic way of generating an arbitrary number of view points with varying camera distances and angles. In our work, we used their calibration function and ensured the acquisition of images with varying distances and angles, but we did not implement their star-drawing technique. As previously discussed, even if the work by Tsai et al. has been a clear step towards the complete task automation, their procedure requires the calibration pattern position to be known, in turn requiring the human intervention. This is overcome by our two phase approach.

More recently, in [16], [17], two extended hand-eye calibration approaches without the requirement of a calibration pattern have been presented. Indeed, in medical applications, an unsterile calibration pattern cannot be used. In [16], feature tracking and a structure-from-motion approach are exploited. This proved to be useful to simplify the calibration process, which has to be done before every surgical procedure. Nevertheless, this approach is not as accurate as the common one. In [17], the approach is claimed to be accurate. Anyway, it uses the surgery instruments with known CAD models as calibration objects. In our scenario, an object with a known CAD model is less practical than a checkerboard.

III. METHODS

This section focuses on the description of our automatic procedure. For ease of understanding, the main reference systems and transformations are depicted in Figure 3. The symbols B, H, C and P are the initials of Base, Hand, Camera and Pattern, the names of the main reference systems. The symbols BH, HE, EP and BP stay for the respective rototranslations from the Base to the Hand, the Hand to the Eye, the Eye to the Pattern and the Base to the Pattern. Our goal is the estimation of BP and HE (in red).

The automatic procedure can start after moving the robot to a starting position from which the calibration pattern is visible. Our calibration pattern is the asymmetrical circle pattern, currently supported in OpenCV [18], but could be also the classical black-white checkerboard with equally spaced squares. This procedure is characterized by two phases, displayed from left to right in Figure 1. In the first phase, a raw hand-eye transformation (HE) is estimated

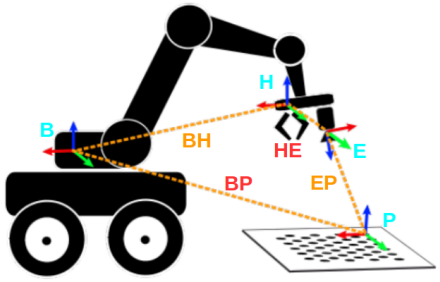
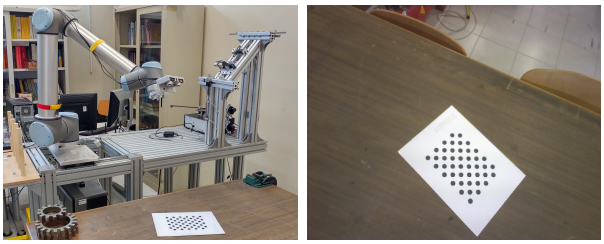


Fig. 3. Representation of the main reference systems (colored triads) and transformations (dot lines). Our goal is the estimation of the Base-Pattern (BP) and the Hand-Eye transformation (HE).

from images grabbed while the arm is moving the hand (H) in the robot base or world reference system (B) with a stop-and-go motion. Indeed, the links BP and HE are both unknown so, instead of moving the camera (E) with respect to the the calibration pattern (P), we can only move the hand (H) with respect to the world/robot base (B). Thus, the actual path of the camera with respect to the checkerboard depends on the starting position. This means that, for instance, the actual number of acquired images framing the calibration pattern and the number of angulations cannot be controlled, impacting the quality of the estimated hand-eye transformation, which, as a result, is also influenced by the starting position. Nevertheless, thanks to this first hand-eye transformation, the checkerboard (P) can be roughly localized. Thus, in the second phase, the robot camera (E) can be moved by the arm with respect to this reference system (P), finally providing a second hand-eye transformation (HE). This second phase gives the advantage of starting the procedure from a known point and moving the arm along a more controlled and repeatable path.

A. Pattern Localization

As previously mentioned, this automatic procedure starts from a position from which the calibration pattern falls in the camera Field of View (FoV). An example is reported in Figure 4. From here, the robotic arm moves through several



(a) (b)

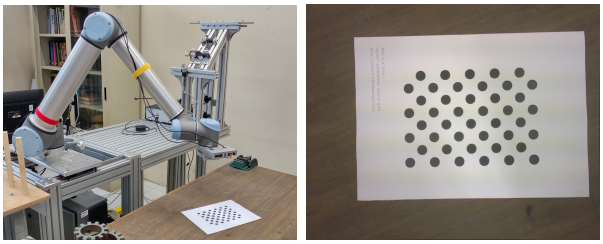
Fig. 4. An example of starting position of the pattern localization: (a) the robotic arm UR10 with a stereo-camera rigidly mounted on the gripper (EuRoC project setup) (b) the checkerboard. The checkerboard is in the camera FoV.

stations with a stop-and-go motion so as to avoid issues when synchronizing the robot-pose with the image. These stations are not chosen a priori. The robotic arm moves its gripper along each axis until the pattern stops being detected. In particular, at fixed steps, a check is performed to verify that the pattern is still detected, hence visible. If so, the acquired image can be used to calibrate and locate the checkerboard, and the robot can keep moving along that axis. Additionally, more stations can be added by rotating the gripper around the gripper axis. Collision avoidance and target reachability is guaranteed by the MoveIt! library.

At the end of this phase, the pattern localization is performed from a set of rectified images $\{I_i\}$ framing the calibration pattern and a set of Base to Hand transformations $\{BH_i\}$ with $i \in \{0, \dots, N-1\}$. For each image, the transformation Eye to Pattern EP can be estimated from the 3D-2D point correspondences, e.g. by means of the iterative method based on the Levenberg-Marquardt optimization available in the OpenCV library. From $\{BH_i\}$ and $\{EP_i\}$, the hand-eye calibration method by Tsai et al. [1] leads to the estimation of HE. Thus, the pattern calibration location can be easily estimated as $BP = BH_{N-1} \times HE \times EP_{N-1}$, where BH_{N-1} is a valid robot pose and EP_{N-1} the respective valid Eye to Pattern transformation.

B. Automatic Acquisition

Once a first hand-eye HE is estimated and the checkerboard P is located in the work-cell, the camera can be moved with respect to the checkerboard independently from its location in the workspace along a controlled and repeatable path. As shown in Figure 5, the acquisition can always start from the same view of the checkerboard.

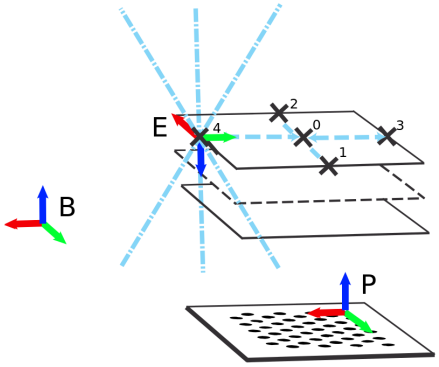


(a) (b)

Fig. 5. An example of starting position of the automatic acquisition: (a) the robotic arm UR10 with a stereo-camera rigidly mounted on the gripper (EuRoC project setup) (b) the checkerboard. The checkerboard is in the camera FoV and clearly visible.

In this phase, the robot moves mimicking a typical acquisition performed by a human and, as suggested by Tsai et al. in [5], varying distances and angles are considered. The robot moves again in a stop-and-go fashion stopping with a fixed stride to check that the calibration pattern keeps being visible. A scheme representing the camera movements is reported in Figure 6. In particular:

- the camera E scans a number of planes *num_p*, which are parallel to the checkerboard but outdistanced of a fixed step *d_p* starting from a minimum distance *d_min_p*. For instance, *num_p* can be equal to 3, *d_p* to $0.05\,\text{m}$ and *d_min_p* to $0.40\,\text{m}$;



(a)

Fig. 6. The camera can be moved with respect to the checkerboard independently from its location in the workspace along a controlled and repeatable path.

- the plane scan is performed from a position 0, almost in the center. Then, the camera is moved along the checkerboard axis x and y, between the position 1 and 2 or 3 and 4. The terminal positions are not fixed. Indeed, as in the previous phase, the camera is moved along an axis until the checkerboard stops being in the FoV;
- each plane can be scanned with varying configurable angles. This is shown in Figure 6, in which the camera E at the position 4 is rotated with three different angles around the axis y. Rotations around other axes can be set too. For instance, one plane can be scanned several times, first with all the angles set to 0, then setting a rotation angle of $10°$ around the axis x, then $10°$ around the axis y and so on.

Again, collision avoidance and target reachability is guaranteed by the MoveIt! library. If any of the stations cannot be reached, the robot continues moving towards the subsequent one. From the new sets $\{BH_i\}$ and $\{EP_i\}$, the hand-eye calibration method by Tsai et al. [1] leads to the estimation of HE.

C. System Adaptability

This approach has been implemented by means of the libraries ViSP, ROS, MoveIt!, OpenCV and PCL, which are widely adopted by the robotic community. Currently, the MoveIt! library supports 65 robots. Thus, even if this implementation has been tested on the Universal Robot UR10 only, it can be potentially used with many others. We selected an approach based on well-known libraries and frameworks for being able to work with different robots and cameras with almost no need for code and system modifications. In fact, our approach obtains the hand-eye calibration starting solely by camera and robot. Standard procedures integrated in the ROS framework provide robot and camera information to our algorithm. A configuration file stores parameters related to different characteristics of the system.

Camera intrinsic parameters are collected by using a *topic*[6]. In the majority of the sensors available in ROS, the *topic* is published directly by the camera driver, and it can be updated by using a camera calibration procedure already available in ROS. Anyway, if no information about intrinsic parameters is exposed by the camera driver, the message firm is publicly available on the ROS website.

Similarly, the system looks for a robot model to learn its physical structure, know about kinematic chains, and understand dynamics. The physical structure is represented by means of the Unified Robot Description Format (URDF)[7]. In the URDF, the robot is composed of joints and links: part dimensions, motion limits, weights, visual and collision shapes complete the robot description. Kinematic chains are defined inside the Semantic Robot Description Format (SRDF)[8]. Moreover, SRDF is usually accompanied by a series of files providing information about collision avoidance, inverse kinematics, robot controllers. MoveIT! is able to exploit such information simply by referring to a specific kinematic group. In our system, it is possible to select the correct kinematic group by changing a parameter in the configuration file. Dynamics is fundamental for a correct simulation in the Gazebo environment. This information can be integrated directly in the URDF file and it is the basis for obtaining realistic tests with physics engines.

IV. EXPERIMENTAL RESULTS

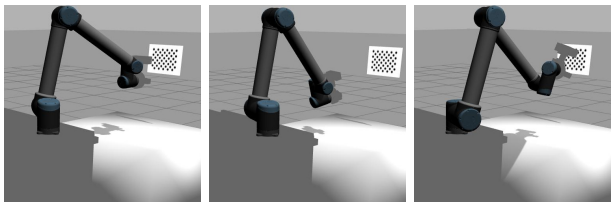
To test the automatic procedure, a scenario similar to the one of the EuRoC project has been selected. The robotic arm is the Universal Robot UR10, collaborative and with 6 degrees of freedom. A stereo-camera composed of two PC webcams is rigidly mounted on it. One of them is taken as a reference, and the hand-eye calibrations are computed with respect to it. In the following, the results of experiments performed in both simulated and real test-beds are discussed.

A. Simulation Experiments

Thanks to the simulation environment, the ground-truth hand-eye transformation and checkerboard can be easily retrieved. To assess the generality of the system, it was tested with the robot starting from 3 different positions (Test A) and with 3 different checkerboard positions in the work-cell (Test B). The 6 different configurations are reported in Figure 7 and 8, respectively. In total, 12 hand-eye transformations and 6 checkerboard locations have been estimated.

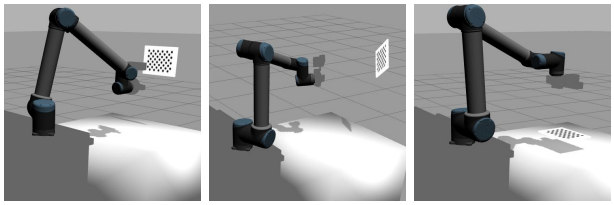
With regard to Test A, the main results are summed up in Table I. The hand-eye transformations have been compared in both translation and angle, in particular Δt is the Euclidean distance in meters while Δq the difference vector between two quaternions in degrees. In addition to the ground truth, the HE transformations have been compared with a baseline consisting in the HE calculated with a manually determined motion defined by 10 waypoints at varying heights and angulations. The baseline, the hand-eye calculated in the first

[6] http://wiki.ros.org/Topics
[7] http://wiki.ros.org/urdf
[8] http://wiki.ros.org/srdf



(a) (b) (c)

Fig. 7. Different starting positions with varying inclinations of the camera. The calibration pattern stays on a vertical plane.



(a) (b) (c)

Fig. 8. Different checkerboard positions: on a vertical, horizontal and inclined plane.

phase and the hand-eye calculated in the second phase are almost equivalent with a standard deviation of only $0.0017\,\mathrm{m}$ and $0.1016°$. Interestingly, in this ideal simulated scenario, the second hand-eye calibration leads to worse performance even if in the second phase the camera is closer to the calibration pattern. This implies that the image quality is higher but also that the number of acquired images is minor since the camera is staying closer to the checkerboard and moving with a fixed step from one position to the next until the checkerboard is visible. In this ideal scenario, the second effect is clearly prominent. Finally, as shown in Table II, the differences in the checkerboard localization are also minimal. The worst entry deviates from the ground-truth of about $0.0159\,\mathrm{m}$ (Position 2) and $1.1412°$ (Position 1). As expected, these deviations are higher since the error on the hand-eye is accumulated with the errors in the robot positioning (BH_{N-1} transformation) and, mainly, the error in the checkerboard positioning with respect to the camera (EP_{N-1} transformation).

With regard to Test B, the main results are summed up in Table III. Even if the robot starts from 3 different positions, both phases of the calibration procedure succeed giving results similar to those already presented.

B. Real Experiments

The entire procedure has been executed from 5 different positions, which have been randomly chosen around the checkerboard at a distance of about 1.00 m. In the second phase, the robot has scanned 3 planes at the distances of 0.40 m, 0.45 m and 0.50 m, from which the checkerboard is better focused. One of these trials has been already shown in Figure 4 and 5. In total, 10 hand-eye transformations, 5 for each of the two phases, and 5 checkerboards have been estimated. In Table IV, the main results are recapped. Given the lack of the ground-truth, no direct comparison can be performed. Anyway, it can be seen that, as expected, the standard deviations of the hand-eye translations and rotations obtained after the first phase (respectively $0.0113\,\mathrm{m}$ and $1.8703°$) are much higher than the standard deviations in the second phase (respectively $0.0054\,\mathrm{m}$ and $0.5324°$). This is due to the fact that the second phase moves the robot along a known path in the checkerboard reference system.

V. CONCLUSIONS

This paper presented an automatic system for localizing the calibration pattern and performing the hand-eye calibration taking a step forward in comparison to existing methods. Indeed, previous works focused on optimizing the hand-eye calibration instead of trying to fully automatize the procedure from the beginning to the end. The proposed approach is based on two phases, in each of which an hand-eye calibration is performed. The second phase allows acquiring images in a controlled setup by moving the camera with respect to the calibration pattern. Both phases were tested in simulated and real scenarios showing that stable results can be obtained after the second phase. As a final contribution to the research community, the toolbox is released online. We plan to use this tool in the upcoming projects and keep on improving it. We would like to test and compare different tool and camera paths in the two phases in order to investigate on the relation between path and overall accuracy.

REFERENCES

[1] R. Y. Tsai and R. K. Lenz, "Real time versatile robotics hand/eye calibration using 3d machine vision," in *Robotics and Automation, 1988 IEEE International Conference on*, pp. 554–561, IEEE, 1988.

[2] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009.

[4] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, pp. 40–52, December 2005.

[5] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[6] I. A. Sucan and S. Chitta, "Moveit!," *moveit. ros. org*, 2016.

[7] B. Siciliano, F. Caccavale, E. Zwicker, M. Achtelik, N. Mansard, C. Borst, M. Achtelik, N. O. Jepsen, R. Awad, and R. Bischoff, "Euroc-the challenge initiative for european robotics," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pp. 1–7, VDE, 2014.

[8] S. Michieletto, F. Stival, F. Castelli, and E. Pagello, "Teaching door assembly tasks in uncertain environment," in *ISR 2016: 47st International Symposium on Robotics; Proceedings of*, pp. 1–7, VDE, 2016.

[9] F. Stival, S. Michieletto, and E. Pagello, "How to deploy a wire with a robotic platform: Learning from human visual demonstrations," in *27th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2017); Proceedings of*, ELSEVIER, 2017.

[10] S. Michieletto, F. Stival, F. Castelli, M. Khosravi, A. Landini, S. Ellero, R. Landó, N. Boscolo, S. Tonello, B. Varaticeanu, C. Nicolescu, and E. Pagello, "Flexicoil: Flexible robotized coils winding for electric machines manufacturing industry," in *ICRA 2017 Workshop on Industry of the future: Collaborative, Connected, Cognitive*, 2017.

TABLE I

HAND-EYE TRANSFORMATIONS IN THE SIMULATION EXPERIMENT TEST A: TRANSLATIONS AND ROTATIONS IN QUATERNIONS. THE ROBOT STARTS FROM THREE DIFFERENT POSITIONS. THE CHECKERBOARD IS VERTICAL.

HE	x	y	z	qx	qy	qz	qw	Δt	Δq
Ground truth	0.0842	0.0959	0.1139	0.5879	-0.3928	0.5879	-0.3928	0.0000	0.0000
Baseline	0.0887	0.0936	0.1087	0.586417	-0.390973	0.589555	-0.39445	0.0072	1.5200
Phase 1 - Position 1	0.0879	0.0914	0.1133	0.5880	-0.3928	0.5878	-0.3929	0.0058	1.4806
Phase 1 - Position 2	0.0866	0.0940	0.1105	0.5888	-0.3929	0.5886	-0.3904	0.0045	1.5720
Phase 1 - Position 3	0.0884	0.0916	0.1119	0.5877	-0.3943	0.5868	-0.3935	0.0063	1.2554
Phase 2 - Position 1	0.0927	0.0945	0.1128	0.5885	-0.3931	0.5875	-0.3923	0.0086	1.4433
Phase 2 - Position 2	0.0924	0.0942	0.1130	0.5886	-0.3930	0.5876	-0.3922	0.0084	1.4893
Phase 2 - Position 3	0.0931	0.0942	0.1126	0.5885	-0.3931	0.5875	-0.3923	0.0091	1.5119

TABLE II

BASE-CHECKERBOARD TRANSFORMATIONS IN THE SIMULATION EXPERIMENT TEST A: TRANSLATIONS AND ROTATIONS IN QUATERNIONS. THE ROBOT STARTS FROM THREE DIFFERENT POSITIONS. THE CHECKERBOARD IS VERTICAL.

BP	x	y	z	qx	qy	qz	qw	Δt	Δq
Ground truth	0.8000	1.5000	1.4000	-0.7071	0.0000	0.0000	0.7071	0.0000	0.0000
Position 1	0.7997	1.4965	1.3977	-0.7066	0.0002	-0.0002	0.7075	0.0042	1.1412
Position 2	0.7864	1.4928	1.3957	-0.7082	-0.0011	-0.0015	0.7059	0.0159	1.1248
Position 3	0.8012	1.4929	1.3964	-0.7063	0.0011	0.0016	0.7078	0.0080	1.1411

TABLE III

HAND-EYE TRANSFORMATIONS IN THE SIMULATION EXPERIMENT TEST B: TRANSLATIONS AND ROTATIONS IN QUATERNIONS. THE CHECKERBOARD IS POSITIONED IN 3 DIFFERENT PLACES: VERTICAL, HORIZONTAL AND ON AN INCLINED PLANE.

HE	x	y	z	qx	qy	qz	qw	Δt	Δq
Ground truth	0.0842	0.0959	0.1139	0.5879	-0.3928	0.5879	-0.3928	0.0000	0.0000
checkerboard 1	0.0927	0.0945	0.1128	0.5885	-0.3931	0.5875	-0.3923	0.0086	1.4433
checkerboard 2	0.0928	0.0943	0.1129	0.5885	-0.3930	0.5875	-0.3922	0.0088	0.9978
checkerboard 3	0.0914	0.0943	0.1130	0.5885	-0.3932	0.5875	-0.3924	0.0074	1.3638

TABLE IV

HAND-EYE TRANSFORMATIONS IN THE REAL EXPERIMENTS AND THEIR STANDARD DEVIATIONS IN THE TWO PHASES.

HE	x	y	z	qx	qy	qz	qw	std(t)	std(q)
Phase 1 - Position 1	-0.2394	-0.0280	-0.0961	0.5141	0.4865	0.5205	0.4776		
Phase 1 - Position 2	-0.2527	-0.0341	-0.0880	0.4856	0.5081	0.5162	0.4894		
Phase 1 - Position 3	-0.2394	-0.0237	-0.1095	0.5067	0.4917	0.5152	0.4859	0.0113	1.8703
Phase 1 - Position 4	-0.2452	-0.0293	-0.0934	0.4993	0.5001	0.5183	0.4816		
Phase 1 - Position 5	-0.2372	-0.0368	-0.0965	0.5004	-0.4933	-0.5223	-0.4831		
Phase 2 - Position 1	-0.2521	-0.0215	-0.1075	0.4974	0.5055	0.4783	0.5180		
Phase 2 - Position 2	-0.2500	-0.0191	-0.1091	0.4973	0.5031	0.4775	0.5211		
Phase 2 - Position 3	-0.2450	-0.0289	-0.1117	0.4950	0.5093	0.4807	0.5143	0.0054	0.5324
Phase 2 - Position 4	-0.2517	-0.0181	-0.1096	0.4960	0.5059	0.4745	0.5225		
Phase 2 - Position 5	-0.2505	-0.0194	-0.1096	0.4974	0.5054	0.4763	0.5199		

[11] M. Munaro, M. Antonello, M. Moro, C. Ferrari, G. Clemente, E. Pagello, and E. Menegatti, "Fibremap: Automatic mapping of fibre orientation for draping of carbon fibre parts," in *IAS-13 Workshop Proceedings: Workshop on ROS-Industrial in European Research Projects*, pp. 272–275, 2014.

[12] M. Antonello, M. Munaro, and E. Menegatti, "Efficient measurement of fibre orientation for mapping carbon fibre parts with a robotic system," *IAS-14, Shanghai, China*, 2016.

[13] M. Antonello, S. Ghidoni, and E. Menegatti, "Autonomous robotic system for thermographic detection of defects in upper layers of carbon fiber reinforced polymers," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pp. 634–639, IEEE, 2015.

[14] G. Puskorius and L. Feldkamp, "Calibration of robot vision," in *Proc. IEEE Int. Conf. on Robotic and Automation*, 1987.

[15] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb," *ieee Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 16–29, 1989.

[16] J. Schmidt, F. Vogt, and H. Niemann, "Calibration–free hand–eye calibration: a structure–from–motion approach," in *Joint Pattern Recognition Symposium*, pp. 67–74, Springer, 2005.

[17] K. Pachtrachai, M. Allan, V. Pawar, S. Hailes, and D. Stoyanov, "Hand-eye calibration for robotic assisted minimally invasive surgery without a calibration object," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 2485–2491, IEEE, 2016.

[18] G. Bradski, "The opencv library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

Lidar-based Urban Road Detection by Histograms of Normalized Inverse Depths and Line Scanning

Shuo Gu, Yigong Zhang, Jian Yang and Hui Kong

***Abstract*— In this paper, we propose to fuse the geometric information of a 3D Lidar and a monocular camera to detect the urban road region ahead of an autonomous vehicle. Our method takes advantage of both the high definition of 3D Lidar data and the continuity of road in image representation. First, we obtain an efficient representation of Lidar data, an organized 2D inverse depth map, by projecting the spatially unorganized 3D Lidar points onto the camera's image plane. The approximate road regions can be quickly estimated by extracting vertical and horizontal histograms of the normalized inverse depths. To accurately find the road area, a row and column scanning strategy is applied in the approximate road region. We have carried out experiments on the public KITTI-Road benchmark, and achieve one of the best performance among the Lidar-based road detection methods without learning procedure.**

I. INTRODUCTION

The problem of traversable road detection has been studied for decades in the context of autonomous driving vehicles. Many successful strategies for road detection using images are proposed in the literature. However, the image based detection methods can become unreliable when strong changes in illumination occur. In some challenging scenes, 3D range data, obtained from stereo cameras, radars or Lidars, are necessary for road detection task. In recent years, 3D Lidar scanners have been widely used in autonomous driving. The Lidar based road detection methods do not suffer from the external illumination and can capture geometry in a very high resolution. The high definition of 3D Lidar data makes it possible to detect most obstacles and the traversable area. However, one major drawback of the Lidar based methods is that the point cloud in each Lidar frame contains large number of spatially discrete and unorganized 3D points as shown in Fig.1 (a). Searching or indexing among these Lidar points is time-consuming. Therefore, it is necessary to design a method to transform the spatially discrete and unorganized Lidar point cloud into a continuous and organized form.

Sensor fusion is a solution to solve this problem as the points in image representation are spatially continuous and organized. In this paper, we propose to fuse the geometric information of a 3D Lidar and a monocular image to detect the road region. Through the sensor fusion, we can take advantage of both the high definition of 3D Lidar data and the continuity of the road in image representation. First, the 3D Lidar and the monocular camera are cross calibrated. Then, the discrete and unorganized 3D Lidar points are projected onto the image plane as shown in Fig.1 (b). We can obtain a new representation of Lidar data, a spatially continuous and organized 2D inverse depth map by applying a linear interpolation (Fig.1 (c)). The road plane detection problem in discrete and unorganized Lidar data is converted to a simple linear classification task in 2D space. Based on this new representation, we can acquire the intermediate representations of road scene by extracting vertical and horizontal histograms of the normalized inverse depths as shown in Fig.1 (d). In the horizontal histogram map, the road plane is projected as a straight line segment. In the vertical histogram map, the pixels that correspond to road area have small values. The approximate road regions can be quickly estimated with both histograms.

The inverse depth histograms are very accurate in detecting road area when the road can be well approximated by a plane. However, the road region is not a perfect plane in general. The road scene can include both road and sidewalk regions, and can be slanted or tilted. Usually, a larger threshold is set to cover most possible road scenarios and in this case, the road area cannot be accurately detected.

To solve this problem, we present a row and column scanning strategy to improve the road detection performance. Height difference relative to a reference point, which can be reliably located in the approximately detected road region, is the only criterion in our scanning strategy to classify between road and non-road points.

To summarize, our paper makes the following major contributions: (i) we propose a continuous and organized representation of the spatially discrete and unorganized Lidar data. (ii) we propose maps of histograms of normalized inverse depth to quickly estimate the approximate road region. (iii) we propose a row and column scanning strategy to improve the road detection performance of (ii), and show promising results on a variety of road scenes. The online evaluation result of our algorithm (HID-LS) on the KITTI-Road benchmark is available at http://www.cvlibs.net/datasets/kitti/eval_road.php. Some video demos of road detection results on the KITTI (visual odometry) road sequences are available at www.youtube.com/channel/UC9BMwRuISFVoepoUibQANdA.

II. RELATED WORKS

Road detection has been an active research area for many years. A variety of approaches based on different sensors have been proposed to detect road regions in some kinds of scenes. In [1], a survey of the recent progress in road detection is presented. There are two main trends in road

The authors are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing city, Jiangsu, China. Email: gs110817@gmail.com, {zhangyigong,csjyang,konghui}@njust.edu.cn

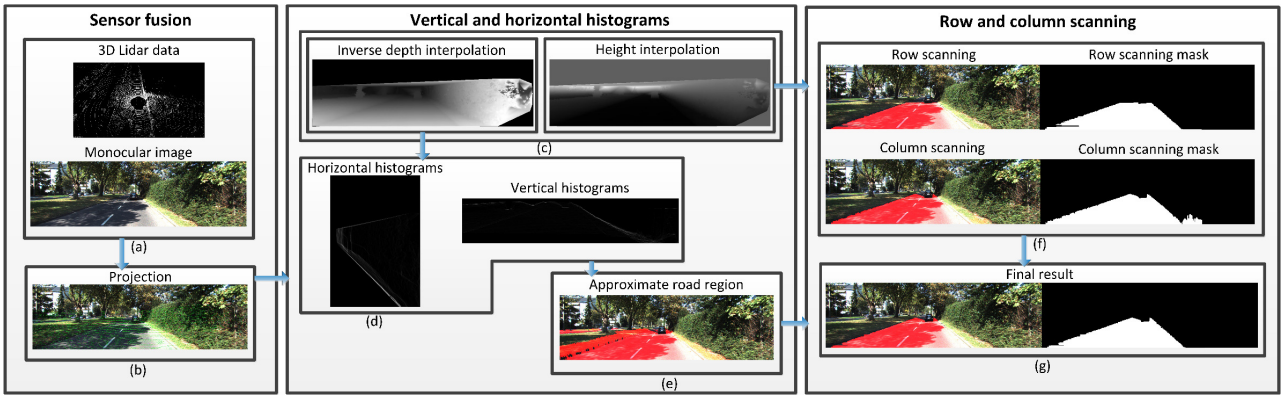


Fig. 1. The flowchart of our algorithm. (a) the input includes 3D Lidar data and a monocular image; (b) projection of Lidar points shown in green; (c) the dense interpolated inverse depth map and height map; (d) the vertical and horizontal histogram extracted from (c); (e) the approximate road region estimated with vertical and horizontal histograms; (f) the row scanning and column scanning based on (e); (g) the road detection result of the proposed method.

detection: (i) the approaches based on single sensors (monocular camera, or stereo vision, or Lidar) and (ii) multiple sensors.

Among the single-camera based road detection methods ([2][3]), most of them are based on learning classifiers. They often take the assumption of lower part of the image being road in order to learn a classifier online or make use of manually labelled datasets to train the classifier offline [4]. The color ([5][6]) and texture ([7][8]) information are often employed as the feature to train a classifier. However, these approaches can easily fail if the evaluation scenarios are different from the training data.

Among the stereo-vision based approaches, most make use of the disparity map acquired through stereo matching. The U-V-disparity [9] and local descriptors [10] obtained from disparity map can be used to analyze the road. Some digital elevation mapping techniques [11] are used to detect the road curb. However, most of these methods need dense stereo matching, which is usually time consuming and the error of disparity value increases with the distance.

Among the 3D Lidar based algorithms, the flat plane assumption [12] is often applied. Most of these methods make use of the high definition of Lidar data and take the flat area as the road. Some approaches model the road with various road boundary detection methods ([13] [14]). Robust regression method named least trimmed squares (LTS) [15] is introduced to fit the road curb. The min-max elevation map method in [16] has been widely used in road detection. In [17], a Gaussian differential filter is used to segment the Lidar scan into road and non-road. In [18], a Lidar-histogram method is proposed to detect road on the Lidar-imagery. These methods can obtain good results when the height difference between road and non-road area is significant. If the height difference is not salient, Lidar based approaches may fail.

In order to overcome the drawbacks of single sensors, multiple sensor fusion has been used to incorporate strengths of single sensors while reduce the individual weakness. In [19], the Lidar points are projected into the image plane to build a graph and the local spatial relationship between points is used to estimate the road region. In [20], after projecting 3D Lidar points to a 2D reference plane, a sliding window technique is applied in the upsampled dense height maps to estimate the road region. In [21], the Lidar points are used to estimate the road plane and then, the road points are projected into the image plane to learn a gaussian model. In [22], the Lidar data and monocular images are fused in the framework of conditional random field to detect road robustly in different scenes. In [18], a weighted fusion strategy is applied to combine color and position information learning from training set and the geometry information obtained through Lidar-histogram to estimate the traversable region. Our method is similar to [19][20] in the way of sensor fusion, we only use the geometric properties of the 3D Lidar and the monocular camera, no photometric properties are used. Photometric information like color and texture from images can be added to further improve the detection performance.

III. OUR APPROACH

Our method includes the procedure of (i) the sensor fusion, (ii) the vertical and horizontal histograms of normalized inverse depths and (iii) the row and column scanning.

A. Sensor Fusion

As presented in [23], Lidar points are stored as $(x, y, z)^T$ (x=forward, y=left, z=up). In this step, we transform the spatially discrete and unorganized 3D Lidar points to continuous and organized 2D image points. This consists of two parts. First, we transform the 3D point $P_v = (x_v, y_v, z_v, 1)^T$ in Lidar coordinate system to 3D point $P_c = (x_c, y_c, z_c, 1)^T$ in camera coordinate system. Second, we project the 3D point P_c in camera coordinate system to 2D point $p = (u, v, 1)^T$ in image coordinate system. The transformation matrices are defined as:

$$P_c = T_{velo}^{cam} \cdot P_v, \quad p = T_{proj} \cdot P_c \tag{1}$$

where T_{velo}^{cam} is the transformation between the Lidar and camera coordinate systems. T_{proj} is the projection matrix

from camera to image coordinate systems. Only 2D points in the field of view of the camera ($0 < u <= width$, $0 < v <= height$) are used. Then, a spatially continuous and organized 2D inverse depth map as shown in Fig.1 (c) can be obtained by applying linear interpolation to Fig.1 (b).

B. Vertical and Horizontal Histograms for Rough Road Detection

After the fusion step, the road detection problem in spatially discrete and unorganized 3D Lidar data is converted to a simple linear classification task in 2D space. Inspired by the works related to the u-v disparity [9], we present a histogram-based method to estimate the road plane. As the inverse depth map in Fig.1 (c) is equivalent to the disparity map in stereo vision up to a scale, we convert the inverse depth to a equivalent disparity as follows:

$$d(u,v) = s \cdot \frac{1}{x} \tag{2}$$

where $d(u,v)$ is the equivalent disparity of point $p = (u,v)^T$ and s is a scale factor used in the inverse depth normalization. The horizontal and vertical histograms of the normalized inverse depths are defined as follows:

$$\mathcal{H}_n^t = \sum_{n=0}^{cols} \delta_{m,n}, \quad \mathcal{V}_m^t = \sum_{m=0}^{rows} \delta_{m,n}$$
$$\delta_{m,n} = \begin{cases} 1 & \text{if } d(m,n) = t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{H}_n^t$ and $\mathcal{V}_m^t$ represent the horizontal and vertical histogram maps, respectively, which accumulate the number of pixels whose equivalent disparity value equals t row-wisely and column-wisely, respectively, in the normalized inverse depth map. In the vertical histogram map, the height is equivalent to the maximum value of the normalized inverse depth, and the width is equal to that of the input image. The small values in the vertical histogram map correspond to the pixels in road plane, i.e.,

$$\mathrm{I}(m, \phi(m)_n) \in \begin{cases} \text{road}, & \text{if } \mathcal{V}_{m,n} < \delta_{\mathcal{V}} \\ \text{non-road}, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathrm{I}(m, \phi(m)_n)$ is the pixel at the location of $(m, \phi(m)_n)$ of the input image. The $\mathcal{V}_{m,n}$ is the pixel value at (m,n) of the vertical histogram map $\mathcal{V}$. The $\phi(m)_n$ is the row-coordinate of a pixel in the mth column of the color image whose inverse depth value is n. The $\delta_{\mathcal{V}}$ is a threshold that is used to decide whether $\mathrm{I}(m, \phi(m)_n)$ is a road pixel or belongs to non-road region.

In the horizontal histogram map, the height is equal to that of the input image, and the width is equivalent to the maximum value of the normalized inverse depth. The road plane corresponds to the straight line in the horizontal histogram map. The road profile line can be fitted by the RANSAC (Random Sample Consensus) [24] algorithm and the road region can be quickly estimated. We assume that the line equation is $v = ku + b$, where (u,v) is the coordinate of a point on the line. Theoretically, this line can classify the point cloud into a road plane, positive and negative obstacles. In reality, road is not a perfect plane in 3D space. Therefore, we specify a tolerance margin to contain the road points, written as follows:

$$\begin{cases} v = ku + \alpha b, & \alpha > 1, \quad \text{upper bound} \\ v = ku + \beta b, & \beta < 1, \quad \text{lower bound} \end{cases} \tag{5}$$

We can roughly obtain the classification of road and obstacles. For a certain scanning row v, the bottom and top margins of its disparity can be denoted by $B(v)$, $T(v)$. Assuming that $D(v)$ is the disparity of a point, if $D(u,v) < B(v)$, this point is lower than the road plane, belonging to negative obstacle. If $D(u,v) > T(v)$, this point is higher than the road plane, belonging to positive obstacle. If $B(v) <= D(u,v) <= T(v)$, this point is nearly on the the road plane, belonging to the road.

The major problem of this kind of histogram based approach is that the applied tolerance margin has a great influence on the road detection result. It is hard to achieve reliable and accurate detection performance in varieties of road scenarios if based on only one set of threshold values. To ensure that we can detect the road surface for most of the road scenes, we usually set a large value to $\delta_{\mathcal{V}}$ in (4), and a large value to α and a small value to β in (5), respectively. But this usually results in a false detection region which could include both road and sidewalk simultaneously.

An example of approximate road detection based on horizontal histograms with large tolerance margin in line fitting is shown in Fig.2 (c) and (d). The results show that the large tolerance margin may lead to misclassification of obstacles which will affect the reliability of the following reference point selection. Fig.2 (e) and (f) show examples of the approximate road detection based on vertical histograms. Likewise, it also results in false detection where the detected region contains area above the horizon. To reduce the false detection in the approximate road region as much as possible, we combine both the horizontal- and vertical-hisogram based strategies. Specifically, we first obtain an approximate road region by applying the horizontal-histogram based scheme (5), and then apply the vertical-histogram based scheme (4) to the region obtained by the horizontal-histogram based scheme. The approximate road region obtained by combining both histogram based schemes is shown in Fig.2 (a) and (b).

C. Row and Column Line Scanning

We propose a row and column scanning strategy to accurately find the road within the approximate road region.

1) Height Difference:

To make use of the high definition of 3D Lidar data, we classify road and non-road pixels based on the height difference relative to a reference point. A point is classified as *non-road* if the function below returns zero and otherwise *road*.

$$road(p) = \begin{cases} 0, & \text{if } \|h_r - h_p\| > t \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

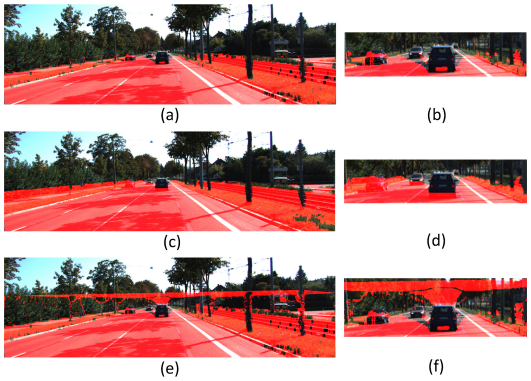


Fig. 2. Approximate road region estimated by different histogram strategies. (a) and (b) vertical and horizontal histogram method; (c) and (d) horizontal histogram method; (e) and (f) vertical histograms approach.

where h_r is the height value of a reference point p_r. h_p is the height value of a point p. t is the threshold of the height difference.

2) Reference Point Selection:

Height difference is the only criterion applied in our scanning method which is essentially finding points of the same type (road or non-road) as the reference points. Thus, the selection of reference points has large influence on the road detection result. In this paper, we apply a dense reference point selection strategy within the approximate road region. Instead of selecting only one in a row or in a column, we use more reference points. A new reference point will be selected if the distance relative to the current reference point is beyond a certain threshold. Our dense selection strategy can take advantage of the local flatness property of the road region and can be applied in non-flat scene.

3) Row and Column Scanning:

Row Scanning: The row scanning process starts from middle (the first reference point) to left and right, respectively, and ends when meets the boundary, i.e., consecutive *non-road* points. The position of the first reference point of each row is defined as the average position of the detected road boundaries in previous row. If it is not in the approximate road region, we will use the nearest point in the same row of the approximate road area as the actual first reference point.

As shown in Fig.1 (b), the Lidar sensor lacks sensing information in distant regions, e.g. near the end of road. The interpolated height data around this area are not reliable and we will stop the scanning process if the number of road points in a row is less than a certain threshold to avoid the inaccurate interpolation.

Column Scanning: Similar to the row scanning process, the column scanning starts from bottom to top and ends when meets consecutive *non-road* points. The difference lies in that the row scanning results are used to determine whether the points on the last row of the image are road or not, then, all the first reference points are selected from these road points. This makes column scanning dependent on the scanning results of last few rows.

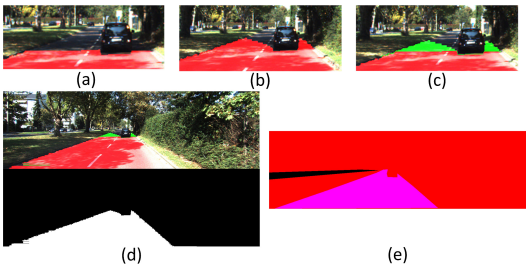


Fig. 3. Road detection results of row scanning and column scanning (Perspective view). (a) row scanning approach; (b) column scanning approach; (c) and (d) row and column scanning approach; Red denotes detection result of row scanning and green represents supplementary result from column scanning; (e) ground-truth of road area.

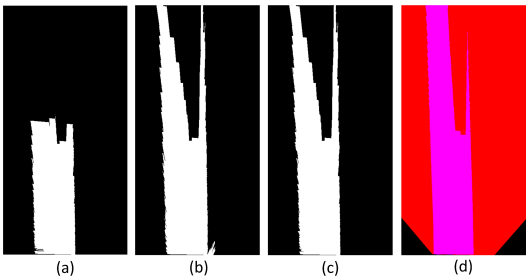


Fig. 4. Road detection results of row scanning and column scanning (BEV). (a) row scanning approach; (b) column scanning approach; (c) row and column scanning; (d) ground-truth of road area.

Row and Column Scanning: The stop mechanism in row scanning usually leads to early stop when cars are present on road, which results in missing road detection as shown in Fig.3 (a). The column scanning does not have this problem, but it can be more prone to false detection in the nearby regions due to dependence on row scanning results.

To solve the problems mentioned above, we propose to combine the row and column scanning. We use the row scanning result as a base and the column scanning as a supplement to fill the missing part in the distance. Experiments show that the combination improves the detection performance.

IV. EXPERIMENTS

To evaluate the performance of the proposed method, we test it on the public KITTI benchmark [25]. All the experiments below are tested on a standard PC with a 16GB of RAM and a quad-core *Intel Core i5-4460* CPU clocked at 3.2GHz. The algorithm is implemented with C++ under Ubuntu 14.04 and the average consuming time is about 0.5 second.

A. KITTI-Road dataset

The KITTI-Road dataset [23] includes synchronized images and Lidar data with calibration parameters, ground-truth, and scripts for evaluation. The KITTI-Road dataset consists of two parts, the training dataset of 289 images with ground-truth images and the testing dataset of 290 images which only allows evaluation by submission of result

TABLE I

EVALUATIONS ON KITTI-ROAD TRAINING BENCHMARK(BEV)

UM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
Row	89.05 %	80.43 %	**91.31 %**	86.90 %	**3.80 %**	13.10 %
Column	87.99 %	79.39 %	84.18 %	92.16 %	7.96 %	7.84 %
Ours	**91.35 %**	**83.58 %**	88.79 %	**94.06 %**	5.46 %	**5.94 %**

UMM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
Row	92.45 %	87.51 %	**95.37 %**	89.70 %	**4.75 %**	10.30 %
Column	92.66 %	88.20 %	91.80 %	93.53 %	9.11 %	6.47 %
Ours	**94.49 %**	**90.63 %**	94.47 %	**94.50 %**	6.03 %	**5.50 %**

UU_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
Row	88.30 %	77.33 %	**88.69 %**	89.92 %	**4.08 %**	10.08 %
Column	83.06 %	72.19 %	82.40 %	83.73 %	6.36 %	16.27 %
Ours	**90.17 %**	**81.51 %**	87.03 %	**93.55 %**	4.96 %	**6.45 %**

images to the website. The two datasets have three different categories of road scenes, namely Urban Marked (UM), Urban Multiple Marked (UMM), and Urban Unmarked (UU). Scripts provided by [23] can generate a specific score for each one. The dataset offers pixel-based evaluation in perspective view and bird's eye view (BEV). The metrics include maximum F1-measure (MaxF), average precision (AP), precision (PRE), recall (REC), false positive rate (FPR) and false negative rate (FNR), and are computed after transforming from the image domain to BEV space. The KITTI-Road dataset uses the MaxF as the primary metric value for comparison between different methods. Note that our method is training-free, we test it on both training and testing datasets.

B. Experiments Setting

Row Scanning: The distance threshold of the reference point selection is set as 0.2m in y direction and the threshold of the height difference relative to a reference point is set as 2cm.

Column Scanning: The distance threshold of the reference point selection is set as 1m in x direction and the threshold of the height difference relative to a reference point is set as 5cm.

C. Comparison of Different Methods

To show the effectiveness of the combination of row and column scanning, we evaluate the scanning methods quantitatively in KITTI-Road training dataset. Table I shows the comparison results in BEV. It shows that the combination of row and column scanning does improve the performance.

In this part, we also compare the line scanning method with and without approximate road region estimation step in KITTI-Road training dataset quantitatively. Fig.5 shows an example of the comparison. Fig.5 (a) and (d) are the line scanning results with the constraints that the reference points are selected in the approximate road region. Fig.5 (b) and (e) are results of line scanning operated on the whole image. It is obvious that result based on the whole image may cover the area above horizon. As the BEV representation only covers from -10m to 10m in lateral (y) direction and from 6m to 46m in longitudinal (x) direction, the misclassifications above horizon have little influence on the BEV results as shown in Fig.5 (e). Thus, we compare the road detection results in perspective view. Table II shows that the approximate road region can help improve the detection performance in perspective view.

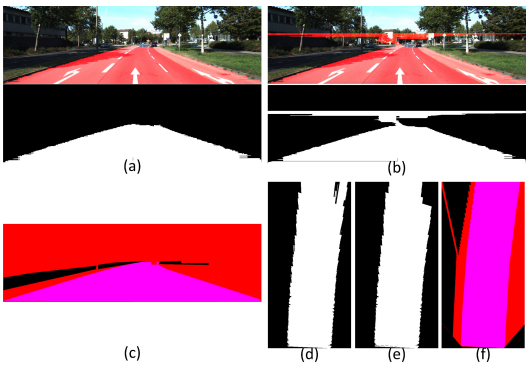


Fig. 5. Comparison of different approximate road regions. (a) and (d) approximate road area estimated by vertical and horizontal histograms; (b) and (e) the whole image; (c) and (f) ground-truth of road area.

TABLE II

EVALUATIONS ON KITTI-ROAD TRAINING DATASET(PERSPECTIVE)

UM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
OnlyScanning	92.98 %	82.18 %	88.75 %	**97.64 %**	2.44 %	**2.36 %**
Ours	**93.72 %**	**83.76 %**	**90.49 %**	97.19 %	**2.01 %**	2.81 %

UMM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
OnlyScanning	94.55 %	86.44 %	92.70 %	**96.48 %**	2.38 %	**3.52 %**
Ours	**95.28 %**	**88.25 %**	**94.69 %**	95.87 %	**1.68 %**	4.13 %

UU_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
OnlyScanning	89.77 %	79.13 %	85.68 %	**94.26 %**	2.49 %	**5.74 %**
Ours	**91.52 %**	**82.67 %**	**89.57 %**	93.55 %	**1.72 %**	6.45 %

KITTI-Road: In this section, we compare our approach with some other Lidar based methods which can be referred ([18][19][26]) in the KITTI-Road benchmark. Fig.6 illustrates some visual results of our method in BEV and perspective view. The quantitative results in the BEV space are shown in Table III. The performance of our method ranks the first among these methods. The result is very promising considering that we only use the geometric information of Lidar and monocular images. The detection performance can be further improved by combing photometric information like color and texture from image.

V. CONCLUSIONS

In this paper, a road detection method based on the fusion of the geometric information of the 3D Lidar and the monocular camera is proposed. Through the spatially continuous and organized representation of Lidar data, a histogram based method can be applied to quickly estimate the approximate road regions. We also propose a row and column scanning strategy to improve the approximate road detection results. Experimental tests on KITTI-Road benchmark show that our method can successfully detect road in multiple scenes. As

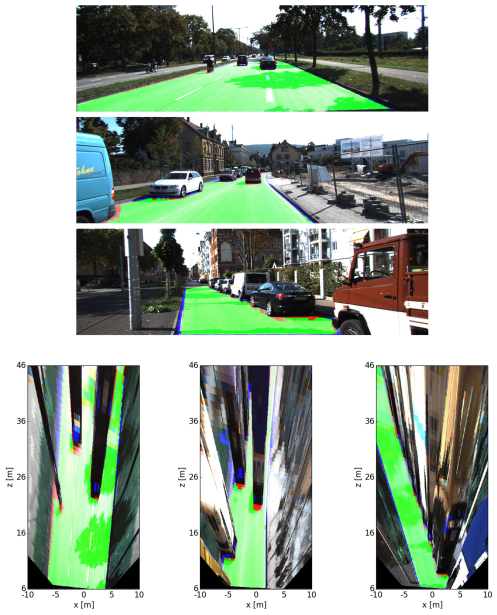


Fig. 6. Road detection in perspective view and BEV. Here, red denotes false negatives, blue denotes false positives and green represents true positives.

TABLE III

EVALUATIONS ON KITTI-ROAD TESTING BENCHMARK(BEV)

UM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
RES3D-Velo	83.81 %	73.95 %	78.56 %	89.80 %	11.16 %	10.20 %
LidarHisto	89.87 %	83.03 %	**91.28 %**	88.49 %	**3.85 %**	11.51 %
HybridCRF	90.99 %	**85.26 %**	90.65 %	91.33 %	4.29 %	8.67 %
Ours	**92.03 %**	83.73 %	88.97 %	**95.31 %**	5.38 %	**4.69 %**

UMM_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
RES3D-Velo	90.60 %	85.38 %	85.96 %	**95.78 %**	17.20 %	**4.22 %**
LidarHisto	93.32 %	**93.19 %**	**95.39 %**	91.34 %	**4.85 %**	8.66 %
HybridCRF	91.95 %	86.44 %	94.01 %	89.98 %	6.30 %	10.02 %
Ours	**94.36 %**	91.01 %	94.88 %	93.84 %	5.57 %	6.16 %

UU_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
RES3D-Velo	83.63 %	72.58 %	77.38 %	90.97 %	8.67 %	9.03 %
LidarHisto	86.55 %	**81.13 %**	**90.71 %**	82.75 %	**2.76 %**	17.25 %
HybridCRF	88.53 %	80.79 %	86.41 %	90.76 %	4.65 %	9.24 %
Ours	**89.10 %**	80.53 %	86.13 %	**92.29 %**	4.84 %	**7.71 %**

URBAN_ROAD

Algorithm	MaxF	AP	PRE	REC	FPR	FNR
RES3D-Velo	86.58 %	78.34 %	82.63 %	90.92 %	10.53 %	9.08 %
LidarHisto	90.67 %	84.79 %	**93.06 %**	88.41 %	**3.63 %**	11.59 %
HybridCRF	90.81 %	**86.01 %**	91.05 %	90.57 %	4.90 %	9.43 %
Ours	**92.36 %**	85.83 %	90.86 %	**93.90 %**	5.21 %	**6.10 %**

future work, we intend to integrate visual information like color or texture to further improve the detection accuracy.

REFERENCES

[1] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, pp. 1–19, 2014.

[2] J. Alvarez, T. Gevers, Y. LeCun, and A. Lopez, "Road scene segmentation from a single image," *Computer Vision–ECCV 2012*, pp. 376–389, 2012.

[3] C. Tan, T. Hong, T. Chang, and M. Shneier, "Color model-based real-time learning for road following," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 2006, pp. 939–944.

[4] P. Y. Shinzato, V. Grassi, F. S. Osorio, and D. F. Wolf, "Fast visual road recognition and horizon detection using multiple artificial neural networks," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 1090–1095.

[5] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 309–318, 2004.

[6] B. Wang, V. Frémont, and S. A. Rodríguez, "Color-based road detection and its evaluation on the kitti road benchmark," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 31–36.

[7] H. Kong, J. Y. Audibert, and J. Ponce, "Vanishing point detection for road detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 96–103.

[8] H. Kong, J. Y. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2211–2220, 2010.

[9] Z. Hu and K. Uchimura, "Uv-disparity: an efficient algorithm for stereovision based scene analysis," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 48–54.

[10] K. Wang, L. Qu, L. Chen, Y. Gu, and X. Zhang, "Non-flat road detection based on a local descriptor," *arXiv preprint arXiv:1609.08436*, 2016.

[11] M. Kellner, M. E. Bouzouraa, and U. Hofmann, "Road curb detection based on different elevation mapping techniques," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 1217–1224.

[12] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3d range data," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1146–1152.

[13] W. S. Wijesoma, K. S. Kodagoda, and A. P. Balasuriya, "Road-boundary detection and tracking using ladar sensing," *IEEE Transactions on robotics and automation*, vol. 20, no. 3, pp. 456–464, 2004.

[14] H. Cramer and G. Wanielik, "Road border detection and tracking in non cooperative areas with a laser radar system," in *Proceedings of German Radar Symposium*. Bonn, Germany, 2002, pp. 24–29.

[15] A. Y. Hata, F. S. Osorio, and D. F. Wolf, "Robust curb detection and vehicle localization in urban environments," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 1257–1262.

[16] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[17] W. Zhang, "Lidar-based road and road-edge detection," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 845–848.

[18] L. Chen, J. Yang, and H. Kong, "Lidar-histogram for fast road and obstacle detection," in *Proc. International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[19] P. Y. Shinzato, D. F. Wolf, and C. Stiller, "Road terrain detection: Avoiding common obstacle detection assumptions using sensor fusion," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 687–692.

[20] R. Fernandes, C. Premebida, P. Peixoto, D. Wolf, and U. Nunes, "Road detection using high resolution lidar," in *Vehicle Power and Propulsion Conference (VPPC), 2014 IEEE*. IEEE, 2014, pp. 1–6.

[21] X. Hu, F. S. A. Rodriguez, and A. Gepperth, "A multi-modal system for road detection and segmentation," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 1365–1370.

[22] L. Xiao, B. Dai, D. Liu, T. Hu, and T. Wu, "Crf based road detection with multi-sensor fusion," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015, pp. 192–198.

[23] J. Fritsch, T. Kuhnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 1693–1700.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[25] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[26] L. Xiao, R. Wang, B. Dai, Y. Fang, D. Liu, and T. Wu, "Hybrid conditional random field based camera-lidar fusion for road detection," *Information Sciences*, 2017.

Robot navigation balancing safety and time to goal in dynamic environments*

María-Teresa Lorente and Luis Montano[1]

***Abstract*— This work addresses a technique for robot motion planning and navigation in dynamic environments. First, a model to represent the future evolution of the moving obstacles in the environment is defined in a *robocentric* reference, which maps the obstacle motion on the control space, the *velocity-time space*, of the robot. Second, a planning technique working on that model for navigation and maneuvering in this scenario is performed. The planned motion commands balance safety and time to goal criteria, applying every control sampling time a velocity command satisfying the criteria from the planned strategy. Unlike other classic reactive strategies, the novel method developed allows to plan and execute safe motions according to the further evolution of the moving objects in the robot field of view. The robot maneuvers among the obstacles using the concepts of *time to collision* and *time to escape* from the perceived moving objects. The technique is evaluated in randomly generated simulated scenarios, based on metrics defined using safety and time to goal criteria.**

I. Introduction

Optimal techniques for navigating robots in changing, dynamic, and unforeseen scenarios, where autonomous robots have to coexist with other robots and humans in a friendly way (for example in crowded scenarios such as airport or hospital halls), require using planning and navigation strategies different from the techniques used for static scenarios. Many efforts during the last years have been dedicated to improve planning and reactive navigation algorithms for static scenarios, some of them also applied to dynamic environments. However, the use of pure reactive techniques jointly with classical global planners for navigation leads to unexpected, avoidable collisions, and even robot blocking.

In this work we develop a model to represent the environment dynamism on the control space of the robot. The model informs about the further evolution of the moving objects or people, in such a way that planning and navigation techniques using it can exploit all the represented information. We name this model *DOVTS* (Dynamic Object Velocity-Time Space), in which the forbidden and free robot velocities are represented as a function of the time. The kinodynamic constraints of the robot are also taken into consideration to compute a feasible motion, implicitly represented in the model.

In dynamic environments, the free space for moving the robot changes every sampling time, which leads to a more reduced effective free space for a motion planner. So, planners and navigation techniques have to be tailored to the dynamism of the scenario. The work presented here develops a robocentric motion planner, which works on *DOVTS* model to compute safe and near-optimal motions towards the goals. The planner evaluates on the model the time to collision or to escape from collision among the surrounding objects. This way the time to reach a goal can be optimized. Every control sampling time the planner computes the motion action that optimizes a hybrid criterion balancing safety, measured from the free velocity space, and time to goal, manoeuvring among the moving obstacles.

The three major contributions of this work are:

- A model that incorporates to a Velocity-Time space *DOVTS* the dynamic information and further evolution of the obstacles in the scenario.
- A planner that, working on the *DOVTS* space, plans motions for a midterm horizon in the field of view of the on-board sensors by optimizing a hybrid safety/time-to-goal criterion, re-planning the motion every sampling time.
- The model exhibits a lower computational complexity when compared with other modelling techniques in the literature.

Section II summarizes some of the related works found in the literature. In section III the model used to represent the dynamic environment is formally defined. Section IV introduces the planning and navigation algorithm, and section V evaluates the method through simulations. Finally, section VI presents some conclusions and future work.

II. Related Work

Iterative motion planners, such as [1], [2], [3], [4], [5], [6], and [7], combine global planning towards the goal and reactive avoidance of obstacles to navigate a robot. However, safe motion planning in dynamic environments requires explicitly computing the motion evolution of the obstacles and the kinodynamics of the robotic system. Consolidated approaches found in the literature refer to the *Velocity Obstacle* (*VO*) [8], expanded in [9], and the *Inevitable Collision States* (*ICS*) introduced by [10]. *VO* computes the set of velocities from the current state that would provoke a collision between a holonomic robot and an obstacle at a time in the future. Planning of collision-free trajectories using this model requires looking ahead, simulating several steps forward. *ICS* computes the robot states which lead to inevitable collision, i.e., there is no control the robot can apply to avoid a collision for a set of *evasive manoeuvres*.

*This work was partially supported by Spanish projects DPI2016-76676-R-AEI/FEDER-UE and T04-DGA-FSE.

[1]Instituto de Investigación en Ingeniería de Aragón, University of Zaragoza, C/Mariano Esquillor s/n 50018, Zaragoza, Spain. mlorente@unizar.es, montano@unizar.es

A different approach is carried out in [11], where authors deal with motion planning through a B-spline parametrization. The method exploits B-splines properties to replace constraints over the entire time horizon by finite sets of constraints, suitable for real time optimization.

In [12] the authors generalize the *VO* concept so that dynamic constraints can be easily addressed. Recently, [13] presented a new *VO*-based approach (*OVVO*) which improves the velocity selection compared to the *VO* approach. The method defines a grid from the velocity window and uses a cost function to decide on the best velocity for the robot. In [14] the authors extend relevant work addressing multi-robot systems based on *VO*, generalizing the reciprocal collision avoidance for robots with different kinematic constraints.

Motion safety has been formally studied with the *ICS* concept. Several criteria are defined in [15] to address motion safety for autonomous robotic systems, which establish that the robotic system should consider i) its own dynamics, ii) the environment objects' future behaviour, and iii) to reason over an infinite time-horizon, in order to compute its future motion. Computationally costly, an approximation of *ICS* is thus used in practice. In [16], [17], [18], [19] and [20] the *ICS* concept is exploited in different ways.

Motion safety issue in the *VO* framework using the *ICS* is addressed in [21] by calculating the proper time horizon as the minimum time for the robot velocity to exit the velocity obstacle. An adaptive computation of the time horizon is proposed in [22], which changes with respect to the relative information of the environment and the robot. In [23], the authors adapt the *DWA* approach ([24]) to consider moving obstacles, represented as moving cells in a grid map. Cells intersecting robot trajectories are considered forbidden.

The work presented in this paper extends the model defined in a preliminary work [25] to represent the dynamic information in the environment, and develops a robot motion planner based on this model, which is evaluated in different dynamic scenarios. An environment model is developed, the *DOVTS* (Dynamic Object Velocity-Time Space), which computes trajectories determining the set of *free* velocities which leads the robot to the goal without collision within the space horizon given, e.g. the sensor's field of view. In this method, there is no need for an imposed *time horizon* as in the referenced techniques, given that it is implicit to the *DOVTS* model definition. Moreover, when planning the next motion action for the robot, *DOVTS* provides a look-ahead of the trajectories with respect to the evolution of the moving objects, hence helping the decision process for searching the best motion.

III. *DOVTS* ENVIRONMENT MODEL

The developed environment model computes the relative dynamic information between the robot and the moving obstacles. Unlike other methods referenced in the literature, the method does not represent the Workspace-Time or Configuration-Time spaces, but the maximum and minimum velocities and associated times for reaching or escaping from the *collision band*, respectively, in the Velocity-Time

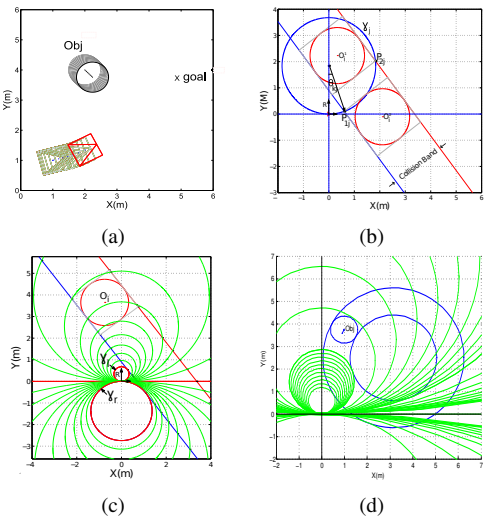


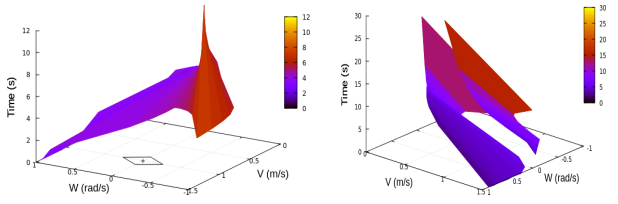
Fig. 1: (a) Workspace; (b) Collision Band swept by object $\mathcal{O}_i$, object $\mathcal{O}_i$ in positions $\mathcal{O}_i^1$ and $\mathcal{O}_i^2$, path γ_j, and collision points P_{1j} and P_{2j} in the robocentric (R) Configuration Space; (c) multiple paths Γ through the collision band; (d) robocentric representation for an obstacle moving following a circular curve (blue), and the robot circular trajectories (green).

space. The collision band is the configuration space swept by an obstacle while moving on a trajectory. We develop the method for a non-holonomic robot, thus circular trajectories for planning are computed, which are controlled from angular and linear velocities (ω, v).

Figure 1 shows the basic idea. In Figure 1a the robot Workspace and a moving object are depicted. Fig. 1b shows the Configuration Space for this situation in the robocentric reference, in which the object moves in a straight line trajectory. A circular robot trajectory γ_j intersects the limits of the collision band, corresponding to $P_{1j}(x_{1j}, y_{1j})$ and $P_{2j}(x_{2j}, y_{2j})$ which represent the robot locations for collision or escape in the Configuration Space. Then, the times for the obstacle to get to those points are computed, t_{1j} and t_{2j}, which, in turn, represent the collision and escape times for the robot. The associated velocities for the robot to reach those points after or before the obstacle are thus computed. The velocities refer to the control variables for the robot, i.e. the angular and linear velocity, which are obtained from Eq. (1). The resultant calculations, as illustrated in figure 2, lead to an obstacle in the Velocity-Time Space ($\mathcal{V} \times \mathbb{R}$) of the robot, the *DOVT* (Dynamic Object Velocity-Time). This computation is extended to a set of n circular trajectories Γ , so $\gamma_j \in \Gamma$, $j = 1..n$ (see Fig. 1c). Note that hereinafter, γ_j is used indistinctly to refer to both a circular trajectory and its radius.

$$\omega_{kj} = \frac{\theta_{kj}}{t_{kj}} = \frac{\text{atan2}\ (2\, x_{kj}\, y_{kj},\ x_{kj}^2 - y_{kj}^2)}{t_{kj}} \qquad (1)$$
$$v_{kj} = r_j\, \omega_{kj}, \quad k = 1, 2$$

This method for modelling is completely general for different trajectories of the objects. See figure 1d as an example of an object following a non-linear trajectory. Details for the computation process can be found in [25] for obstacles



(a) (b)

Fig. 2: (a) Velocity-time space *DOVTS* for one moving obstacle; the surface represents the corresponding velocity-time obstacle *DOVT* including the dynamic information. (b) *DOVTS* for two obstacles; free velocities can be selected between both surfaces representing the *DOVT* obstacles.

moving in linear trajectories, and in Appendix for obstacles moving in circular trajectories.

A. *Formal definition of* DOVTS *space*

The *DOVTS* model is formally defined as follows. Let be $\mathcal{R}(\mathbf{v}_j, t)$ the robot state, (x, y, θ), under the control action $\mathbf{v}_j = (\omega, v)$ applied at time t.

*Definition 1 (***Dynamic Object Velocity-Time):** The Dynamic Object Velocity-Time (*DOVT*) for a particular moving object $\mathcal{O}_i$ with respect to the set of feasible trajectories of the robot Γ is defined as the set of velocities that produce a collision with $\mathcal{O}_i$ at time t,

$$DOVT(O_i, \Gamma) = \{(\omega, v, t) \in \mathcal{V} \times \mathbb{R} \mid \exists \gamma_j \in \Gamma, \frac{v}{\omega} = \gamma_j, \ \mathbf{v}_j = (\omega, v), \ \mathcal{R}(\mathbf{v}_j, t) \cap \mathcal{O}_i(t) \neq \emptyset\} \quad (2)$$

Accordingly, *DOVT*$(\mathcal{O}, \Gamma)$ represents *DOVT* for all the obstacles in the environment,

$$DOVT(\mathcal{O}, \Gamma) = \cup_{\mathcal{O}_i} DOVT(\mathcal{O}_i, \Gamma) \quad (3)$$

As a consequence, velocities belonging to *DOVT* are unsafe, leading to collision at a future time.

*Definition 2 (***Free Velocity-Time):** The Free Velocity-Time (FVT) is the set of velocities outside $DOVT$,

$$FVT = \{(\omega, v, t) \in \mathcal{V} \times \mathbb{R} \mid \mathbf{v}_j = (\omega, v, t) \notin DOVT, \mathcal{R}(\mathbf{v}_j, t) \cap \mathcal{O}(t) = \emptyset\} \quad (4)$$

Any velocity inside *FVT* will be safe in the long term or, if it lies under the *DOVT* surface, during the time until collision, given that the initial conditions remain.

*Definition 3 (***Dynamic Object Velocity-Time Space):** The Dynamic Object Velocity-Time Space (*DOVTS*) is defined as the control space of the robot which contains controls belonging to *DOVT* and *FVT*,

$$DOVTS = \{DOVT \cup FVT\} \quad (5)$$

In short, velocities in *FVT* can be selected by a motion planner to compute safe motions without collision during a time horizon according to their collision time. Figure 2 shows *DOVTS* for one (a) and two (b) moving obstacles. Choosing velocities in between dynamic objects *DOVT*, time to goal could be improved.

Algorithm 1 Compute Goal in the Velocity Space

Require:
1: $(x, y)_g$, goal position relative to robot in workspace
2: **function** COMPUTEVELOCITYGOAL($(x, y)_g$)
3: γ_g = GetRadiusGoal($(x, y)_g$);
4: $\mathbf{v}_g$ = GetVelocity($(x, y)_g$, γ_g) : Equation 1
5: **return** $\mathbf{v}_g$
6: **end function**

IV. ROBOCENTRIC PLANNER ON *DOVTS*

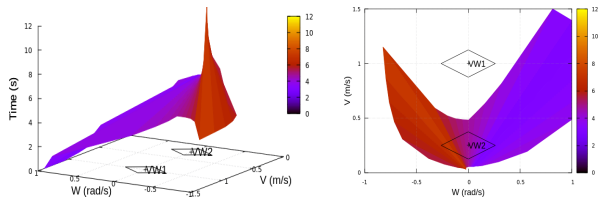
Any planner that exploits the information included in *DOVTS* could compute safe and quick trajectories towards the goals. This section introduces a motion planning and navigation technique that works directly on the *DOVTS* model explained in the previous section. As said before, using path planners for static environments in dynamic scenarios can lead to unsatisfactory robot behaviours, and in general to no optimal trajectories, due to the need of avoiding moving obstacles reactively. We claim that a motion planner using *DOVTS* model, which plans further robot motions based on the dynamism of the obstacles instead of acting in a pure reactive way, can strongly improve the performance of navigation in such kind of scenarios.

A. *The planner*

As the planner works in the Velocity-Time space, the goal location to be reached in the workspace has to be projected in *DOVTS* as a velocity goal, $\mathbf{v}_g$. From the current robot location in the configuration space, a circular trajectory towards the goal is computed, which is modeled into *DOVTS* as the set of velocities which leads to the goal following Eq. (1), and bounded by the velocity constraints. The maximum velocity in the set refers to the velocity the robot should have to reach the goal at the next sampling step. Then, $\mathbf{v}_g$ corresponds to the maximum reachable velocity in the set within the bounds. Algorithm 1 reproduces the computation in pseudo-code. Then, the planner searches for a plan towards $\mathbf{v}_g$ along the whole available time horizon.

In order to manoeuvre the robot, extremal controls can provide near time-optimal trajectories in environments with obstacles. In [26], time-optimal trajectories were calculated as extremal controls in free space for differential-drive and car-like robots with bounded linear and angular velocities but without acceleration limitations. In this work, either maximum angular or linear acceleration are selected as extremal controls to navigate, leading to clothoid or anti-clothoid trajectories ([25]).

The kinodynamic constraints are also considered in the planner (a rhomb for a differential drive robot in figure 3). A velocity window *VW* is defined according to the maximum angular and linear accelerations (α_m, a_m). Formally, $VW = \{(\omega_c \pm \alpha_m T, v_c \pm a_m T)\}$, where (ω_c, v_c) is the current robot velocity, and T the sampling time in the algorithm. These maximum accelerations are thus used as the cell size in the grid explored by the planner.



(a) (b)

Fig. 3: Two alternative velocities for the robot, *VW1* and *VW2*, in (a) *DOVTS* and (b) its projection into the velocity space *DOVS*. They represent the kind of velocities eligible when planning, i.e. velocities which do (*VW2*) and do not (*VW1*) have a bounded time of application given by the time until collision with the obstacle. The planner will combine safety and time to goal criteria to select the best motion.

The planner developed uses information mapped on *DOVTS* to select the more suitable commands. These commands are velocities $\mathbf{v} = (\omega, v)$ applied every sampling time to the robot. The planner searches for safe velocities in *FVT*, the free Velocity-Time space in the model, and executes every sampling time to adapt to changes in the scenario. Simple circular trajectories are considered when planning, which may lead to non optimal solutions if they are applied in the long term. In addition, the *DOVTS* model offers a look-ahead at an instant which is valid while velocity conditions remain invariable. Not allowing the planner to be executed to update the changes observed in the environment would result in sub-optimal trajectories and be unsafe for the robot, requiring a forward simulation of the system, which is costlier. However, it could be taken into consideration, if some safety guarantees could be found. The selection of the best velocity command is conducted by safety and near-optimal time to goal criteria. To illustrate the idea, figure 3a shows a situation where an obstacle and two candidate velocities defining the robot in *DOVTS* are plotted, *VW1* and *VW2*. These represent two examples of velocities eligible for planning. The projection of this situation into the velocity space *DOVS* is also shown in figure 3b to clearly state that *VW2* can only be temporally applied, as it lies under a *DOVT*.

For computing near-optimal time commands among safe velocities, an A*-like algorithm in the 3d-space *DOVTS* is proposed. The planner constructs a tree by expanding velocity-time safe nodes in the search space ordered by a cost function $f(n) = g(n) + h(n)$, which estimates the cost to reach the velocity goal, $\mathbf{v}_g$. $g(n)$ is the cost to reach the current node n, and $h(n)$ is a heuristic cost which estimates the cost to get to the goal from n. In our method, the search space is defined as a velocity-time grid obtained from *DOVTS*, and the nodes of the tree correspond to the cells in the grid, $n = (\omega_n, v_n, t_n)$. The cost function $g(n)$ reflects the number of velocity changes taken and their closeness to the velocity goal. The algorithm selects commands that get the robot away from the obstacles, but as little as possible from the velocity goal. The heuristic term $h(n)$ is the time to goal, measured as the number of sampling steps to reach the goal velocity in absence of obstacles.

Algorithm 2 Compute Robot Motion

Require:

DOVTS, Velocity-Time space of the robot
$\mathbf{v}_c$, current velocity of the robot
$\mathbf{v}_g$, velocity goal
s, strategy used for planning (*Conservative*, *Time-Aware*)
function COMPUTEMOTION(*DOVTS*, $\mathbf{v}_c$, $\mathbf{v}_g$, s)
    mapGrid = ComputeGrid(*DOVTS*)
    $cell_r$ = CellGrid($\mathbf{v}_c$)
    $cell_g$ = CellGrid($\mathbf{v}_g$)
    mapGrid = mapGrid $\cup$ $cell_r \cup$ $cell_g$
    $velPlan$ = MotionPlan(mapGrid, s)
    (ω, v) = FirstElement($velPlan$);
    return (ω, v)
end function

Formally, for a node $n = (\omega_n, v_n, t_n)$ being evaluated,

$$\begin{aligned} g(n) &= g(parent(n)) + 1 + OpCost(n) \\ h(n) &= t_{\mathbf{v}_g} - t_n \\ OpCost(n) &= 1 + \\ &\quad + \text{dist}(cell_n, cell_{\mathbf{v}_g}) \\ &\quad - \text{dist}(cell_{parent(n)}, cell_{\mathbf{v}_g}) \end{aligned}$$

The cost function $g(n)$ is interpreted as follows: the unity cost represents one velocity change opportunity at maximum acceleration (hence making shorter plans better); $OpCost$ models an opportunity cost of staying as close to the velocity goal as possible, with the euclidean distance between cells given in grid coordinates (which represent velocity changes at maximum acceleration) used to prefer velocities closer to the goal and the 1 term used to ensure an always positive increase in cost.

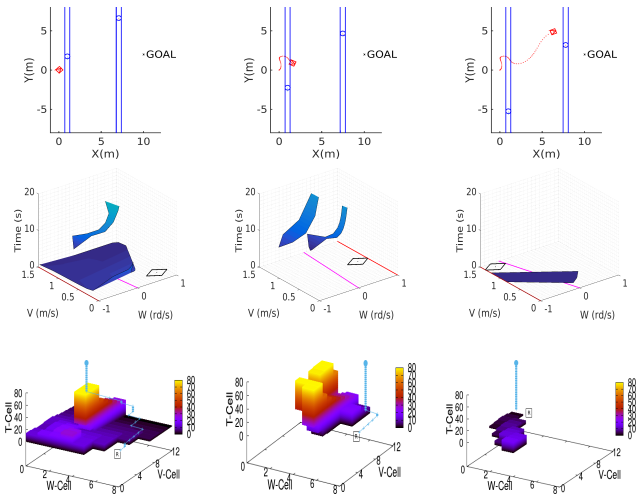
In other words, the planner minimizes the velocity changes while trying to stay close to the velocity goal, simultaneously avoiding Velocity-Time collision configurations. The heuristic, in turn, represents the minimum possible remaining steps in the plan given the height in time steps of the search space, used to accelerate the search without having an influence in the optimal solution.

Algorithm 2 represents the process to compute the motion command (ω, v) to be applied for the robot during a sampling time, for the plan found at every sampling step.

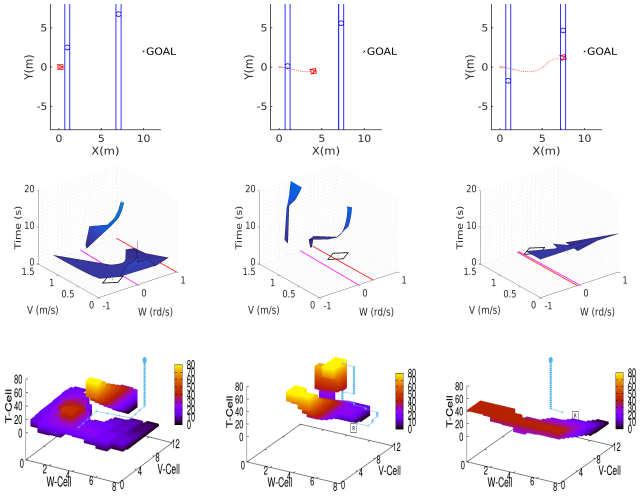
Figure 4 depicts an example of different navigation solutions proposed. The workspace with two obstacles following linear trajectories (the collision bands are plotted), the trajectory of the robot, the corresponding *DOVTS* and velocity-time grid at several instants are represented. The coloured grid cells correspond to the obstacles in *DOVTS*, i.e., forbidden velocity-time commands for the robot.

B. Navigation strategies

The planning and navigation algorithm can be applied using different strategies. We consider here two of them: the *Conservative* strategy, in which the velocities lying under any *DOVT* are considered as always occupied, thus they are



(a) *Conservative* strategy. Any velocity under any *DOVT* is forbidden for planning, i.e. any velocity which leads to collision at any future time. Thus, the planner computes motions above the first *DOVT* and around the second one. As a result, the robot has to deviate and wait until the obstacles pass to move towards the goal, leading to longer trajectories.



(b) *Time-Aware* strategy. Velocities between the *DOVT*s can be selected for planning. The planner selects free cells in between the *DOVT*s to plan motions which lead the robot to pass before both obstacles.

Fig. 4: a) *Conservative* and b) *Time-Aware* strategies. The workspace, which shows the trajectory followed by the robot and the moving obstacles, and the situation represented both in *DOVTS* and in the velocity-time grid are shown at different instants during navigation. The sequence of velocities computed in *FVT* from the current node (R) towards the goal are also plotted (blue line). Using information about time to collision (*Time-Aware* strategy) allows the robot reaching the goal in near-optimal time.

not eligible (see figure 4a); and the *Time-Aware* strategy, in which all the possible free velocities are considered for planning, even those that could lead to collision at a future time (see figure 4b). Both strategies are evaluated in Section V.

C. Grid parameters

The size of the cells in the grid is also an important issue, because it will determine the precision of the velocity obstacles in the grid, and the smoothness of the trajectories obtained from the command application. The computation time will fix also the minimum time step needed for planning. In the current implementation the sampling time is $T = 250ms$. This parameter, jointly with the maximum available robot velocities and the desired precision, determine the 3d velocity-time grid size. In this work, we have chosen to divide the velocity space $\mathcal{V}$ considering the maximum angular and velocity accelerations for the sampling time used, and the time dimension with respect to the sampling time. The time horizon considered during simulation ($20s$) is not a time horizon imposed in the method, it corresponds to the workspace horizon considered for modelling the obstacles in the field of view of the sensors. The maximum velocities for the robot are $v_{max} = 1.5m/s$ and $[w_{min}, w_{max}] = [-1, 1]$ rad/s.

D. Model complexity

The process to model the dynamic environment is analyzed and compared with respect to the referenced *VO* and *ICS* approaches.

Computing the set of *ICS* of a robot for different control trajectories and several obstacles involves a complexity of $\mathcal{O}(mnt_{max}o)$, where m is the number of obstacles, n the number of evasive manoeuvres, t_{max} the time considered for computing *ICS* states, and o the sum of the number of vertices of the robot and the most complex object, which are involved in the Minkowski difference operation required between each pair robot-obstacle. By using Graphics Processing Unit (GPU) the complexity can be reduced to $\mathcal{O}(mnt_{max})$ (see [27]).

The *VO* approach concerns mainly three operations: computing each individual VO_j for $j = 1..m$ obstacles, the multiple velocity obstacle MVO, and the set of safe velocities. The most complex operation is the second one, because it requires updating the points of each VO_i, $i = j + 1..m$, with the intersection points between VO_j and each VO_i, and ordering them clockwise, which leads to a complexity of $\mathcal{O}(m^2)$. This can be reduced to $\mathcal{O}(m \log m)$ by storing the points within a more efficient structure (see [28]). However, this approach does not provide information about future motion actions beyond the next sampling time, usually needing an exhaustive exploration of the search space for planning the best motion. The need for a t_{max} parameter to consider a look-ahead is also present in this approach.

In our technique, the complexity of modelling each moving object in its $DOVT_j$, $j = 1..m$, for a set of n robot planned trajectories (Γ), is $\mathcal{O}(mn)$, where m is the number of obstacles. The set of trajectories selected for computing a $DOVT_j$ depends on the position of the obstacle and its motion with respect to the robot, and on the desired discretization, focusing the calculation on the area where the obstacle will be moving. Thus, several robot trajectories are considered for each obstacle. Modelling objects moving in non-linear trajectories (circular paths have been developed in this work) implies the same order of complexity, because only intersection points with the collision band are needed. This approach provides enough information for computing plans in the defined workspace horizon, and it is not affected

by the parameter t_{max} of the other methods given that the time for which obstacles are considered is implicit within the model construction.

V. Simulation Results

Several simulations in randomly generated scenarios have been performed to test robot navigation under different conditions. The performance of the planner is evaluated in terms of safety and time to goal criteria. A video showing different simulations can be downloaded from [1].

Two scenarios have been chosen for evaluation: the first one more structured, the *road-crossing*, in which the robot crosses an area where obstacles moving in linear trajectories traverse the scenario from one extreme to the other in both directions; in the second one, the *non-linear*, the obstacles move in random directions following non-linear trajectories.

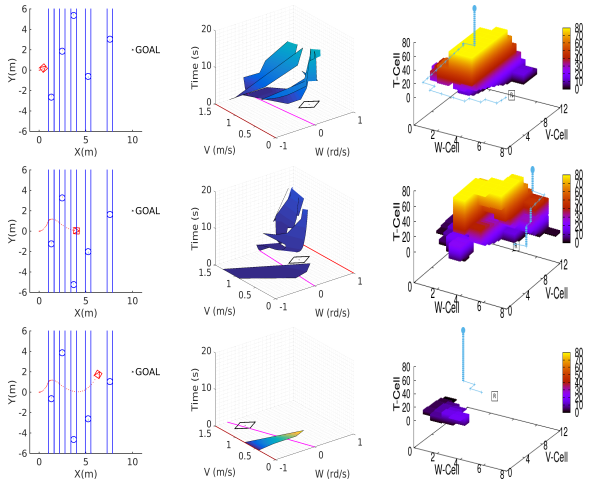
In addition, the number of moving obstacles and their velocities are changed so that we cover a range of conditions to generalize the evaluation performed. The density of moving obstacles is kept during all the simulations, in order to maintain the complexity of the scenario for navigation. For each condition we run 10 simulations, and for each same scenario we execute both *Time-Aware* and *Conservative* strategies of Section IV-B, to emphasize the improvement when time to collision is available for planning. For the *road-crossing* scenario, where obstacles move with non-zero linear velocity, the conditions defined are $V1 = \{\omega = 0, v = 0.2\}$ and $V2 = \{\omega = 0, v = 0.5\}$. In the *non-linear* scenario, the conditions established are $V1 = \{\omega = -0.15..0.15, v = 0.2\}$ and $V2 = \{\omega = -0.38..0.38, v = 0.5\}$. In both cases, the number of obstacles considered are 5 and 10.

Figure 5 shows several snapshots during simulation of the two scenarios designed. The trajectory followed by the robot and the collision band of each obstacle in the workspace are plotted, together with the obstacles modeled in *DOVTS* and the velocity-time grid used for planning. Once the information about the environment is modelled in *DOVTS*, it is mapped into the velocity-time grid.

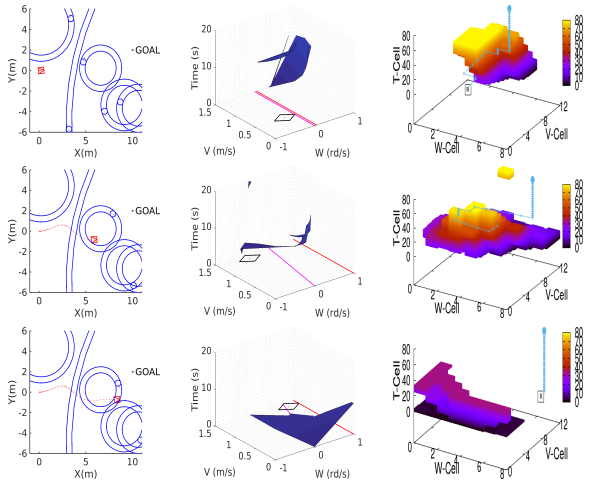
Figures 6 and 7 show the time needed to traverse the scenario with respect to the optimal case, i.e. in absence of obstacles, comparing both *Time-Aware* and *Conservative* strategies, for each kind of scenario considered. Figures 8 and 9 illustrate the profiles for linear velocity and distance to the obstacles. As expected, *Time-Aware* strategy leads to quicker trajectories and lower time to goal than the other strategy. The mean distance to obstacles is similar with both strategies for few obstacles, although as the number of obstacles increases, this distance decreases. This is because in scenarios with fewer obstacles there is more free space for the robot to move.

In general, collisions may appear during navigation. Table I shows the number of collisions produced out of the total simulations performed (10). The number of collisions using *Conservative* strategy are in general higher than with *Time-Aware* strategy. This is because, even though the method

[1] http://robots.unizar.es/data/videos/simulations-dovts.mp4



(a) *Road-crossing* scenario.



(b) *Non-linear* scenario.

Fig. 5: (a) *Road-crossing* scenario and (b) *Non-linear* scenario. Simulations with 5 obstacles moving with velocities in set $V1$ and executing *Time-Aware* strategy. The workspace with the robot trajectory and the collision bands of the obstacles, the *DOVTS* space and the velocity-time grid at different instants during robot navigation are illustrated. The planner searches for commands in *FVT* which lead the robot towards the goal. The circular trajectory which leads to the goal is represented in *DOVTS* as the set of velocities which describe the circumference arc towards the goal (magenta line in the images). The goal velocity $\mathbf{v}_g$ is the maximum value of the set within bounds. The colour of the cells in the grid identify the time to collision with an obstacle.

restricts the velocities which may lead to collision, this fact produces a great reduction of the free space to plan safe commands. In the *road-crossing* environment appear more collisions than in the *non-linear* case. In such a structured scenario, the robot has little room for crossing among the obstacles, mainly in the case of high density of obstacles and relative high obstacle velocities. It may get blocked in a situation where there is no opportunity for the robot to cross and reach the goal, ending in collision.

It can be concluded that, in general, the *Time-Aware* strategy performs better in the considered scenarios than the *Conservative* one, both in terms of safety and lower time to goal. This is because this strategy considers more

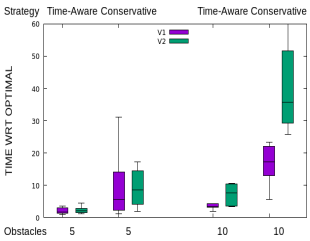


Fig. 6: *Road-crossing* environment. Time cost with respect to optimal case.

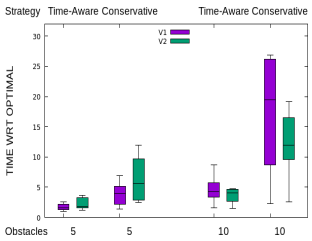
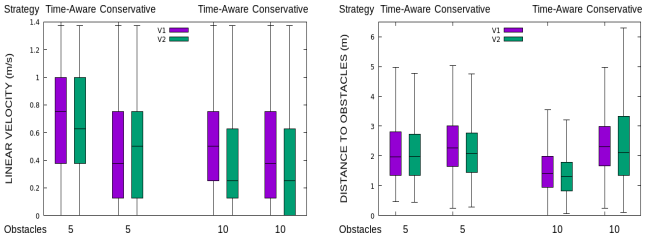
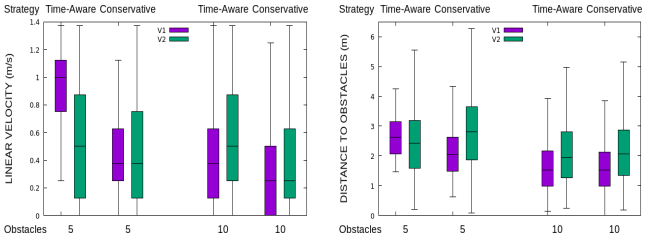


Fig. 7: *Non-linear* environment. Time cost with respect to optimal case.



(a) (b)

Fig. 8: *Road-crossing* environment. (a) Linear velocity and (b) distance to obstacles during simulation, comparing *Time-Aware* and *Conservative* strategies.



(a) (b)

Fig. 9: *Non-linear* environment. (a) Linear velocity and (b) distance to obstacles during simulation, comparing *Time-Aware* and *Conservative* strategies.

velocity commands that can be selected to move among the obstacles. However, in very dense dynamic scenarios, the *Time-Aware* strategy may result more risky than the *Conservative* strategy, leading to more collisions. During robot navigation, *unnatural* motions, as spinning or turning back, are observed sometimes. This is due to the fact that the robot needs time to reach a velocity given its dynamic constraints, especially when linear and angular velocities are high and the robot has to decelerate, and complete rotation is allowed when linear velocity is zero in order to explore potential free velocities to escape. This behaviour could be reduced by limiting the set of highest velocities and the turning angle of the robot in this exploration. But it may result in the robot not finding ways of escape due to this limitation. Further work will be devoted to analyse and improve all these issues.

TABLE I: Number of collisions during simulation.

Scenario	*Road-crossing*				*Non-linear*			
Obstacles	5		10		5		10	
	v1	v2	v1	v2	v1	v2	v1	v2
Time-Aware	0	0	1	4	0	0	1	1
Conservative	2	2	3	2	0	2	1	2

VI. Conclusions

The work presented in this paper develops a robocentric motion planner for dynamic environments. A model of the dynamic environment in the Velocity-Time space, *DOVTS*, is built. This model describes information about the future evolution of the obstacles. Minimum and maximum robot velocities for passing before the obstacles or for decelerating until the obstacles pass before the robot, and time to collision are used for motion planning.

The technique implements an A*-like algorithm in the *DOVTS* space. Two alternatives of the algorithm, *Time-Aware* and *Conservative* are compared. The first one uses explicitly the remaining time until collision, providing quicker trajectories towards the goal among the obstacles. The second strategy avoids selecting any velocity which leads to collision at any time, thus restricting the set of velocities to plan a motion. The technique optimizes a function that balances the time to goal and the time to collision to obtain a motion plan for the robot's field of view horizon, applying the first motion command of the plan every sampling time, and resuming the planning procedure. The model exhibits lower computational complexity than other techniques proposed in the literature for dynamic environments.

Improving the technique to reduce the number of collisions in difficult situations found in the experiments, and extending this work to a multi-robot decision-making context, are some subjects for future work.

APPENDIX

Computation of the robot collision velocities: Obstacle circular trajectories

Figure 10 illustrates a moving object positioned at $\mathcal{O}_i(t_0) = (x_{obj}, y_{obj})$ and describing a circular trajectory. Positions $\mathcal{O}_i^1$ and $\mathcal{O}_i^2$ refer to the instants at which the robot reaches points $P_{1j}(x_{1j}, y_{1j})$ and $P_{2j}(x_{2j}, y_{2j})$, following a circular trajectory γ_j. Points P_{1j} and P_{2j} are the intersection points between the circular robot trajectory γ_j and the circular collision band swept by the obstacle, delimited by $\mathcal{C}_{in}$ and $\mathcal{C}_{out}$ (6). P_{1j} is the position at which the robot should arrive after the object has just passed it, at time t_{1j} ($\mathcal{O}_i(t_{1j}) = \mathcal{O}_i^2$), whereas P_{2j} is the position at which the robot should arrive just before the object reaches it, at time t_{2j} ($\mathcal{O}_i(t_{2j}) = \mathcal{O}_i^1$).

$$\begin{aligned}(x-x_c)^2+(y-y_c)^2 &= r_{C_{out}}^2 = (r_c - r_{rob})^2\\ (x-x_c)^2+(y-y_c)^2 &= r_{C_{in}}^2 = (r_c + r_{rob})^2\\ (x-x_{rob})^2+(y-y_{rob})^2 &= r_{rob}^2\end{aligned} \tag{6}$$

$$(x-x_{obj})^2+(y-y_{obj})^2 = r_{obj}^2 \tag{7}$$

$\mathcal{O}_i^1$ and $\mathcal{O}_i^2$ are the object positions for computing the collision times t_{1j} and t_{2j}. Applying tangency conditions between robot trajectory γ_j and the circles corresponding to

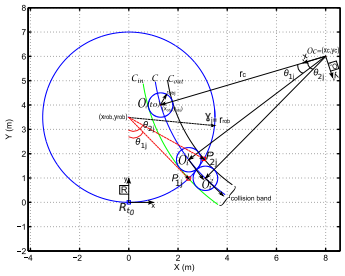


Fig. 10: Collision band, path γ_j with curvature radius r_{rob} and collision points P_{1j} and P_{2j} with a circular object that moves along a circular trajectory in the robocentric (R) Configuration Space.

the object in both positions, these positions can be computed (7). The times to collision are calculated from the angular displacements θ_{ij} and the angular velocity of the object ω_{obj} (8).

$$\begin{aligned} \theta_{cij} &= atan2(2\, x_{obj_i}\, y_{obj_i}, x_{obj_i}^2 - y_{obj_i}^2) \\ t_{ij} &= \theta_{cij}/w_{obj},\ i = 1, 2 \end{aligned} \tag{8}$$

REFERENCES

[1] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 341–346 vol.1.

[2] T. Fraichard, "Trajectory Planning in Dynamic Workspace: a 'State-Time Space' Approach," INRIA, Tech. Rep. RR-3545, Oct. 1998. [Online]. Available: https://hal.inria.fr/inria-00073139

[3] O. Brock and O. Khatib, "Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 550–555 vol.1.

[4] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002. [Online]. Available: http://ijr.sagepub.com/content/21/3/233.abstract

[5] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 1, 2001, pp. 43–49 vol.1.

[6] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, 2002, pp. 508–513 vol.1.

[7] J. Minguez and L. Montano, "Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios," *Robotics and Autonomous Systems*, vol. 52, no. 4, pp. 290 – 311, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188900500093X

[8] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. [Online]. Available: http://ijr.sagepub.com/content/17/7/760.abstract

[9] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: obstacles moving along arbitrary trajectories," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4, 2001, pp. 3716–3721 vol.4.

[10] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, Oct 2003, pp. 388–393 vol.1.

[11] T. Mercy, W. V. Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *2016 European Control Conference (ECC)*, June 2016, pp. 1586–1591.

[12] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 5573–5578.

[13] M. Kim and J.-H. Oh, "Study on optimal velocity selection using velocity obstacle (ovvo) in dynamic and crowded environment," *Autonomous Robots*, vol. 40, no. 8, pp. 1459–1470, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9520-6

[14] D. Bareiss and J. Van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015. [Online]. Available: http://ijr.sagepub.com/content/34/12/1501.abstract

[15] T. Fraichard, "A short paper about motion safety," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1140–1145.

[16] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2210–2215.

[17] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 704–710.

[18] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: An ics-based solution and its comparative evaluation," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 100–105.

[19] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 285–302, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10514-011-9257-9

[20] S. Bouraine, T. Fraichard, O. Azouaoui, and H. Salhi, "Passively safe partial motion planning for mobile robots with limited field-of-views in unknown dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3576–3582.

[21] Z. Shiller, O. Gal, and T. Fraichard, "The Nonlinear Velocity Obstacle Revisited: the Optimal Time Horizon," in *Guaranteeing Safe Navigation in Dynamic Environments Workshop*, Anchorage, United States, May 2010. [Online]. Available: https://hal.inria.fr/inria-00562249

[22] Z. Shiller, O. Gal, and A. Raz, "Adaptive time horizon for on-line avoidance in dynamic environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 3539–3544.

[23] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 1986–1991.

[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[25] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 2833–2838.

[26] D. J. Balkcom and M. T. Mason, "Time optimal trajectories for bounded velocity differential drive vehicles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 199–217, 2002. [Online]. Available: http://ijr.sagepub.com/content/21/3/199.abstract

[27] L. Martinez-Gomez, "Safe Navigation for Autonomous Vehicles in Dynamic Environments: an Inevitable Collision State (ICS) Perspective," Theses, Université de Grenoble, Nov. 2010. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00600578

[28] P. Fiorini, "Robot motion planning among moving obstacles," Ph.D. dissertation, Los Angeles, CA, USA, 1995, uMI Order No. GAX95-19816.

Introduction and Initial Exploration to an Automatic Tennis Ball Collecting Machine

Jialiang Zhao, Hongbin Ma*, Jiahui Shi , Yunxuan Liu

***Abstract*— This paper describes a preliminary development to an automatic tennis ball collecting machine. The framework adopts a roller to collect tennis balls when moving. Tennis balls are detected and tracked automatically by computer vision while hardware acceleration is used to improve efficiency. For navigation, fuzzy control and 6 ultrasonic sensors are used to avoid obstacles when approaching the targets. A web-based monitoring and teleoperation system is added for human-robot interaction.**

I. Introduction

There are more and more people start playing tennis nowadays [1]. According to statistics [2], there are more than 17 million tennis players in US. When playing tennis, a player commonly spend hours in practicing, however, during which collecting balls all around the tennis court can be a very tedious and time-consuming work. Some tools are invented to alleviate this problem.

A. Prior Work

There are some passive tennis ball collecting devices that can help players to pick tennis balls up. The most common tennis ball collecting tools can be seen in the market are passive tube-like gadgets, commonly called Tennis Tube. This kind of device is composed of a tube, used as the handle, which is also a container of tennis balls, and an end cap installed in the front end of the tube, used to collect balls. When working, users should hold the tube, approach a tennis ball, and push the tube towards the tennis ball and force it to be squeezed into the tube via the end cap. This kind of device is convenient and cheap, however, drawbacks of this design are obvious. For convenience and portability, its capacity is too limited, thus users should regularly empty the tube manually. What's more, users need to approach every tennis ball in the field, while there may be hundreds of them during practice. Thus this design can only free users from stooping down, but can't thoroughly solve this problem.

Mobile gadgets generally are more efficient than tube-like gadgets. This kind of device has a roller in its front end, and a handle is used to manipulate. The roller is used to collect tennis balls, one common design of which is shaped like a squirrel cage. The distance between two neighboring beams is slightly smaller than the diameter of a tennis ball. During working, user pushes this mobile gadgets, and the roller starts rolling because of friction of the ground. Tennis balls in front of the roller will be squeezed into it and will not go out because of the beams. The drawback of this design is that, users still need to approach every tennis ball, and it's still a time-consuming process, although much more efficient that tube-like gadgets. Thus the problem still remains unsolved.

There are also some research on autonomous or semi-autonomous electrical active tennis ball collectors, like RapidBallBoy, Har-Tru Ball Mower, and TENNIBOT. RapidBallBoy and Har-Tru Ball Mower both have electrical collectors. When it approaches a tennis ball, these devices will collect it automatically. Users still need to push these devices around the court. TENNIBOT is based on Computer Vision. It can automatically detect a tennis ball, approach it and finally pick it up. The main problem is that this robot doesn't have the ability to avoid obstacles and navigate itself.

B. Our Contribution

In this paper, we present a novel design of a fully autonomous tennis ball collecting machine, which can automatically detect, track, collect tennis balls and navigate itself in tennis court. The visual part is achieved by a RGB camera. The navigation part is achieved by 6 ultrasonic distance sensors. A web sever is built for human-robot interaction.

The framework of this robot is described in Section II. Software part, including the visual algorithms and the web server, is described in Section III. Control strategy is presented in Section IV. Performance and tests of different configurations are discussed in Section V.

Arch Linux is chosen as the operating system. The design approach of the development team follows the KISS principle ("keep it simple, stupid") as the general guideline, and focuses on elegance, code correctness, minimalism and simplicity [3]. It is easy for developers to customize their system to the maximum [4].

The open source robotic framework, Robot Operating System (ROS) [5], is adopted as the communication tool in this system. This framework provides a low-latency communication platform, many useful tools and protocols [6]. Indigo Igloo is the first Long-term Support (LTS) distribution of ROS, thus it is chosen for stability.

This research was partially supported by the National Natural Science Foundation of China under Grant No. 61473038, and the Training Program of the Major Research Plan of the National Natural Science Foundation of China No. 91648117

To whom all correspondences should be addressed. Email: mathmhb@bit.edu.cn

The authors are with School of Automation, Beijing Institute of Technology, Beijing, China, 100081. Hongbin Ma is also with State Key Laboratory of Intelligent Control and Decision of Complex Systems, Beijing Institute of Technology, Beijing, China, 100081.

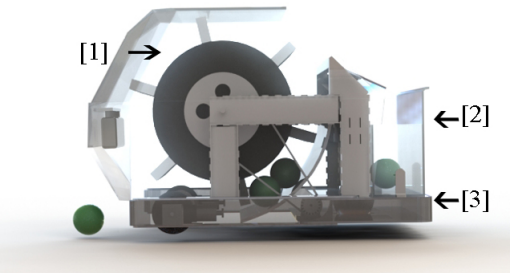


Fig. 1: Schematic of the tennis ball collecting machine. Part [1] is the roller, which is used to collect tennis balls. Part [2] is the ball container, which has a capacity of about 20 tennis balls. Part [3] is the chassis. During working, the roller starts rolling when it approaches a tennis ball, and the brushes installed on it will sweep the balls into the container. The outer shell is made by acrylic. A camera is installed on the front of it, and 6 ultrasonic sensors are installed around it.

II. Framework

A. Mechanical Structure

Carbon fibers (CF) are popular in mechanical structures for their high stiffness, high tensile strength, low weight, and high temperature tolerance [7]. Thus the main mechanical framework is based on CF.

The framework consists of 3 main components, a roller, a chassis, and a tennis ball container.

1) Roller: The roller illustrated in Fig.2 is used to collect tennis balls. There are 6 brushes, made by nylon, installed on it, which are used to sweep tennis balls. When the robot approaches a tennis ball, the roller starts rolling. The brushes will sweep the target into the container.

The roller is driven by a 12V DC 60RPM motor installed on the chassis. They are connected via a pair of pulleys and a drive belt.

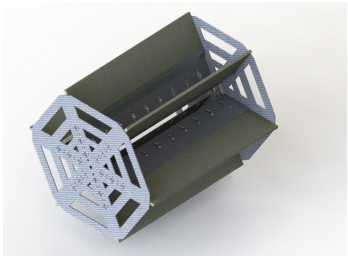


Fig. 2: The roller has a length of 400mm, and a diameter of 350mm. In order to guarantee that the tennis ball won't be missed, the front end of the bottom brush touches the ground when it's in a position.

2) Chassis: The chassis is made by two CF plates, which are connected to each other with some copper pillars. It is shaped like a character *U*. The *mouth* is used to install the roller. The two side bars of the *mouth* are used to place the two front motors. These two bars are narrow, in order to make a larger *mouth*, which will improve efficiency during working. Dynamic system, control boards and batteries are placed between the two plates.

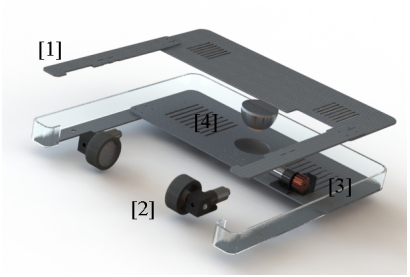


Fig. 3: The chassis. Part [1] is the U-shaped CF plate. Part [2] is the dynamic set. Part [3] is a motor used to drive the roller. Part [4] is an universal driven wheel.

3) Container: The container is made by acrylic, which is lite, cheap and beautiful. It is installed behind the roller and on the chassis. When a ball is collected, it will go into the container with the force of the brushes.

B. Dynamic System

The robot is powered by two T-shaped 12V DC motors. This kind of motor is widely used. Some manufactures, like TAMAGAWA^{TM}, maxon motor^{TM}, and CM motors^{TM} produce them. In this robot, 31ZY from CM motors^{TM} are used. Two encoders are installed in each motor for velocity measurement. These two electrical motors are separately installed on each of the side bar of the chassis. In the rear end of the chassis, a driven wheel is installed. Thus both direction and speed are determined by the two front driving wheels.

The physical robot is shown in Fig.4.

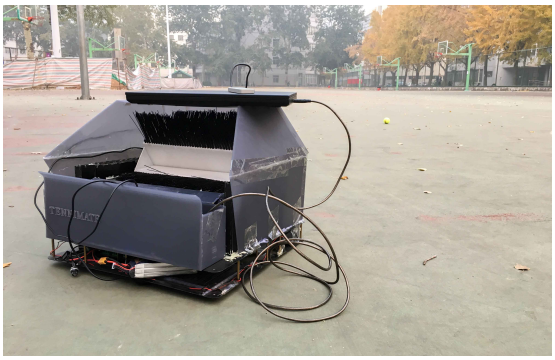


Fig. 4: The physical robot.

III. Software

This section consists two parts, the visual detection part and the web-based human-robot interaction part.

A. Visual Detection

In order to detect tennis balls and find their position, a high-speed and low-latency tennis ball detecting system is needed. What's more, this system should be able to accurately locate several tennis balls at a time.

Tennis balls can be detected by a variety of methods, e.g. wireless modules that are inserted in tennis balls in order to send coordinates; however, this kind of modules need power, and charging them frequently is unrealistic. Also, installing this kind of module in every tennis ball costs much. There are also researchers who let the collecting machine run randomly to traverse the target field, which is of low efficiency. Especially in tennis courts, traversal costs too much time [8].

Vision-based detecting system needs only a processor and a camera, and an efficient set of algorithms can locate tennis balls immediately. With the rapid development of CV, more and more efficient algorithms arise.

Before the recognition algorithm, some pre-processing procedures are necessary.

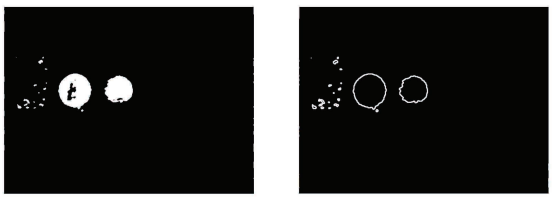
1) Convert Color Space: The native color space of the video stream captured by camera is the RGB color space. Because the color of a tennis ball is one of its most significant features, we need to do color splitting in the following step. HSV color space is more efficient in extracting a color block's hue, and less sensitive to different light conditions [9], so we convert the video stream from RGB to HSV.

After that, three thresholds are set to Hue, Saturation and Value respectively. It's obvious that the performance of detection is hugely influenced by these color thresholds. Thresholds with larger range will provide better performance in different environments with different light intensities, but will decrease the accuracy.

2) Blur: After picking the possible color blocks out, we obtain a gray-scale mask. In this mask, shade of a block indicates the possibility of this block to be a component of a tennis ball. Because of the uncontrollable quality of cameras, light conditions, and mottle on a tennis ball, a noise elimination procedure is needed. We add a median blur filter into the mask. Median blur can protect the image edges and is very effective in smoothing sharp noise [10].

3) Shape Identification: A pixel is treated to be a possible block if it's white, while dark means it belongs to background. Next we determine whether a cluster of possible blocks indicates a tennis ball by measuring its shape. Two algorithms are adopted in this procedure: Suzuki85 and Hough Transform.

- *Suzuki85:* Suzuki85 algorithm determines the surroundness relations among the borders [11] of a binary image. Firstly we use this algorithm to find every contour of the mask image. A result is shown in Fig.5.



The Gray-scale Mask The Contours

Fig. 5: Contours extracted from the mask.

Then we calculate the area of every contour picked from the mask. In this procedure, Green's Theorem is used. Green's Theorem gives Eq.1.

$$\oint_C (Ldx + Mdy) = \iint_D (\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y})\, dx\, dy \quad (1)$$

where C is a simple close curve, D is a double integral over the plane region, L and M are functions of (x, y) defined on D that have continuous partial derivatives [12]. From Green's Theorem, the area of C is given by Eq.2.

$$S = \oint_C (Ldx + Mdy) \quad (2)$$

when $\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} = 1$ [13]. Ideally, the contour of a tennis ball is a circle whatever the shooting angle is, so the area and the perimeter have a relationship of Eq.3;

$$\frac{C}{2\pi} = \sqrt{\frac{S}{\pi}} \quad (3)$$

where C denotes perimeter and S denotes area. Algorithm.1 is adopted to identify a contour.

Algorithm 1 Contour Identification

Input: sample contours $\{\xi_k\}$ of a total of n contours
Output: $Con\{\xi_k\}$ that indicates whether $\{\xi_k\}$ is a circle

```
for k = 1 to n do
    if C{ξ_k} and S{ξ_k} is consistent with Eq.3 then
        Con{ξ_k} = true
    else
        Con{ξ_k} = false
return Con{ξ_k}
```

- *Hough Transform:* Hough Transform is another algorithm to determine a contour's shape. It is a method to detect curves by exploiting the duality between points on a curve and parameters of that curve [14]. We use function from OpenCV to apply this algorithm.

Compared with Hough Transform, Suzuki85 algorithm runs faster but is less accurate. We use Suzuki85 first, and when a contour is detected to be a circle, we use Hough Transform on this small region to confirm it. One result in medium light intensity is shown in Fig.6, and the target tennis ball is labeled.

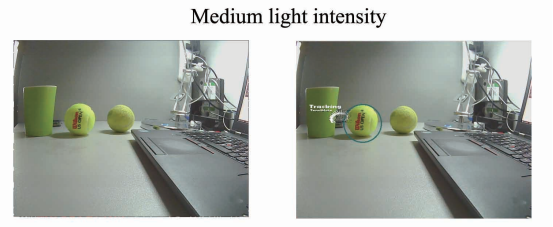


Fig. 6: The closest (biggest) tennis ball is marked, while the green cup is ignored.

B. Hardware Acceleration

Our image processing procedure takes lots of computing resources. This procedure is highly repeatable, thus parallel computing can significantly improve the performance [15]. Open Computing Language (OpenCL) is used in this system. OpenCL is a framework for writing programs that are executed on across heterogeneous platforms [16]. We use it to run some of our image processing algorithms in Graphics Processing Units (GPUs).

Several computing kernels are set up to compute independently. Hough Transform and median blur are computing with hardware acceleration.

After hardware acceleration, the performance is significantly improved compared with computing only with CPUs. Some experiment results are listed in Table I. With each computing method, three different scenes are tested. The test environment is 64-bit Intel Atom E3826 (dual-core, 1.46 GHz) with Intel HD Graphics Integrated, 2GB DDR3 RAM. OpenCL 2.0 and OpenCV 2.4.13.

TABLE I: Results of Hardware Acceleration

Scene	No.	CPU	GPU	FPS
Scene 1	1	√	×	73
	2	√	√	137
Scene 2	1	√	×	47
	2	√	√	110
Scene 3	1	√	×	24
	2	√	√	89

C. Web-based Human-robot Interaction

To have better interactions with this robot, we designed a web-based interface to control and monitor it. A python micro web framework, namely flask, is chosen to build the server for it's light and convenient.

The remote monitoring system should be real-time and of low-latency, to achieve that, we choose Socket.IO as the transmission media. Socket.IO is the cross-browser WebSocket for real-time applications, it enables real-time bidirectional event-based communication [17]. By using Socket.IO the system will be able to communicate between front-end and back-end in real-time. The whole process is shown in Fig.7. Based on C/S framework, Socket.IO is driven by event.

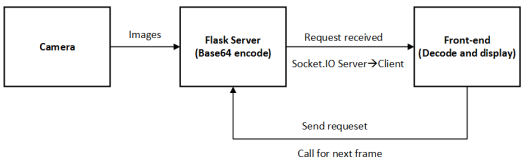


Fig. 7: Remote Monitoring System

In our system, when users open the web the first time, a request for images will be sent from front-end to back-end. When receiving the request, the back-end server will collect images from the camera. After encoding them in Base64 format, image codes will be sent to the front-end by Socket.IO. Then the Base64 codes will be decoded and displayed at the front-end, and a new request for next frame will be sent to server. The whole communication process will last until users close the web. The request-reply design makes full use of the characteristics of Socket.IO, which guarantee the low latency in transmission. Another advantage of this design is that the server sends images only if users visit the web. The transmission will be shut down when users close the web.



Fig. 8: Image Streaming



Fig. 9: Controlling Buttons

Based on the real-time images, users can teleoperate robots via buttons (shown in Fig.9) on web. The left five buttons send moving commands (in four directions) and stop command. The right button is used to start or stop the collecting roller. When a button is pressed, the corresponding color will be changed and a "post" request will be sent from the front-end and detected at the back-end. Commands will then be sent to ROS and robots will move accordingly.

IV. Navigation

We adopt 6 ultrasonic sensors and fuzzy control algorithm to navigate.

A. Sensors

One of the major challenges of the autonomous navigation for mobile robots is to detect obstacles during the robotic navigation task. Considering the main obstacles are humans and chairs on tennis court, we choose to use ultrasonic sensors because they are cheap and accurate enough in this specific environment. We use 6 ultrasonic sensors separated by 36° to get the real-time distance information. For each sensor, it can measure distance between 2cm to 4m with 15° measuring angle. The distribution of ultrasonic sensors is shown in Fig.10.

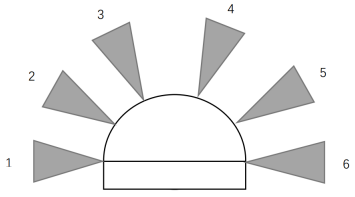


Fig. 10: Sensors Distributions

B. Fuzzy Control

In real-time navigation systems, controllers must be robust enough to deal with different situations, fuzzy control is therefore introduced in our system. Fuzzy control systems are rule-based or knowledge-based systems containing a

collection of fuzzy IF-THEN rules based on the domain knowledge or human experts [18]. The process of fuzzy logic is shown in Fig.11.

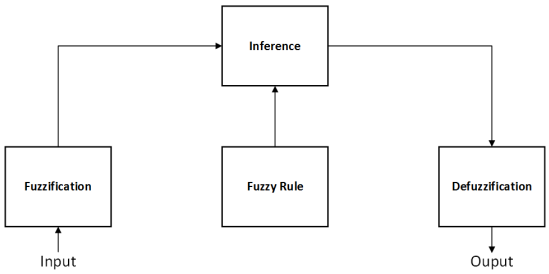


Fig. 11: Fuzzy Controller Structure

1) Fuzzyfication: In our fuzzy control system, variables are shown in Table II. The fuzzyfication process is to transform the continuous values variables into linguistic variables, which is implemented by member functions shown in Fig.12. TableIII shows the meaning of each linguistic variables.

TABLE II: Variables defined in fuzzy control

Variable	Input/Output	Description
LD	Input	Distance from left
FD	Input	Distance from front
RD	Input	Distance from right
θ	Input	Angle to destinations
LV	Output	Output velocity for left wheel
RV	Output	Output velocity for right wheel

TABLE III: Linguistic Variables

Continuous Value Variables	Linguistic Variables	Description
Distance	N	Distance is near
	M	Distance is middle
	F	Distance is far
Angle	P	Angle is from 0° to 60°
	Z	Angle is from 60° to 120°
	N	Angle is from 120° to 180°
Velocity	S	Velocity is very slow
	MS	Velocity is slow
	MF	Velocity is fast
	F	Velocity is very fast

2) Fuzzy Rules: Fuzzy rules are built based on knowledge. To simplify the control rules, we make a transformation to the sensor information.

$$\begin{aligned} LD &= min(Sensor[1], Sensor[2]) \\ FD &= min(Sensor[3], Sensor[4]) \\ RD &= min(Sensor[5], Sensor[6]) \end{aligned} \tag{4}$$

For each input variable, it is divided into three parts, so the total number of rules can be reduced to $3^4 = 81$. One part of rules is shown in Table IV. Taken rule1 as an example, it can be expressed as *"If (LD is F) and (FD is F) and (RD is F) and (Theta is N) then (LV is MS)(RV is F)"* in fuzzy logic rules, which means if distance from 3 directions are all big enough and the destination angle lies between 120° and 180 °, we will set LV to be slow and RV to be very fast, so the robot will turn left to reach the destination.

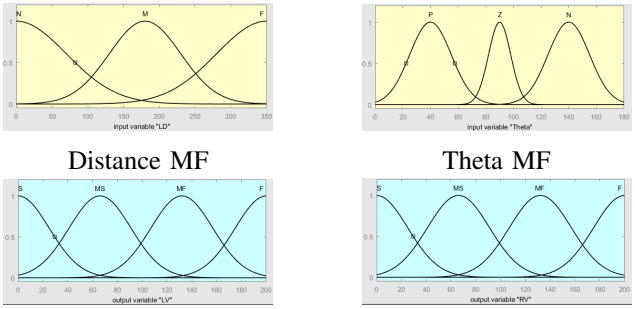


Fig. 12: Member Function

TABLE IV: Variables defined in fuzzy control

Number	LD	FD	RD	θ	LV	RV
1	F	F	F	N	MS	F
2	F	F	F	Z	F	F
3	F	F	F	P	F	MS
4	F	F	M	N	MS	F
5	F	F	M	Z	F	F
6	F	F	M	P	MF	F
...	...	...	...	...	...	...
81	N	N	N	P	MS	S

C. Defuzzification

We use centroid method [19] to defuzzify the member functions for outputs, the output velocities are defined by

$$v = \frac{\int_0^{200} v\mu_D(v)dv}{\int_0^{200} \mu_D(v)dv} \tag{5}$$

where $\mu_D(v)$ is the output member function.

V. Experimental Results

We tested different sets of configuration in the detecting system and the navigation system. Stability and regulating speed are used as criterion during tests.

A. Detecting System

The sensitivity of the detecting system is fundamentally determined by the color threshold and the area threshold. As mentioned early, the color threshold is used to determine whether a color block is a part of a tennis ball. The wider color threshold will show better performance in different conditions with different light intensities. As a result, the detecting accuracy will decrease. The area threshold indicates the minimum area of a possible contour. Smaller area threshold will show better performance when a tennis ball is far away from the robot. However, if this threshold is too small, noises may be erroneously treated as tennis balls. We tested different sets of these two parameters. The best parameters are shown in Table V

TABLE V: Values of HSV threshold

Parameter	H_{th_sub}	H_{th_sup}	S_{th_sub}	S_{th_sup}
Value	43	55	107	240
Parameter	V_{th_sub}	V_{th_sup}	$Area_{th_sub}$	$Area_{th_sup}$
Value	30	250	200	1800

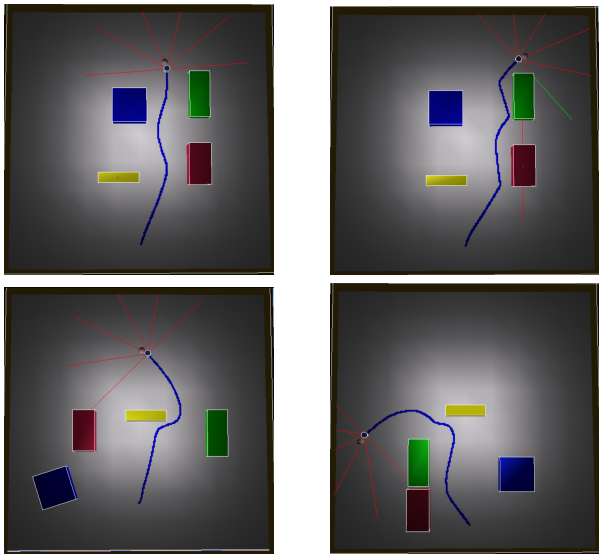


Fig. 13: Simulation Results

B. Navigation System

Webot Pro simulation software is used to test the controller. We build a world and a model for this robot, which is equipped with 6 ultrasonic sensors ranging from 0 to 350mm. Control program is written in MATLAB^{TM} to drive the simulation.

Fig.13 lists some of the most common situations on tennis court, the gray ball is set as destination and the blue footprint is the track of robot's movement. Red lines on robot is the range for ultrasonic sensors. We can see that this robot can collect the ball successfully without running into the obstacles.

VI. Future Work

This machine can now collect tennis balls by one camera. In the future, we are going to add more cameras and extend the collecting functions to other balls.

A. Collecting in Various Ball Games

This vision-based automatic control system can be applied into different ball games, like table tennis, golf and badminton. Theory behind these different functions are similar, thus we are going to make the image processing node able to detect other balls.

B. Multi-camera Support

Now this collecting machine only has one camera installed, and only the scenes in front of this machine can be observed. We are going to add multi-camera support, for example, one camera in each direction, to improve the efficiency. If the collecting machine can observe scenes from every direction at a time, its decision will be more intelligent and the collecting process will be faster.

VII. Conclusions

In this work, we have established a basic vision-based automatic tennis ball collecting system. This system is able to detect tennis balls and control the collecting machine to move towards them and then pick them up. Hardware acceleration and timing synchronization are applied to improve the performance. A web-based remote monitoring and controlling system is used for users to monitor the collecting machine's working state and environment. This web page also provides a method of manual intervention. Such a system may help tennis participants to collect tennis balls and save their time. In future work, we will make this system capable of collecting more balls rapidly in various ball games and add multi-camera support to improve efficiency.

References

[1] E. Wilson, *Love Game: A History of Tennis, from Victorian Pastime to Global Phenomenon*. University of Chicago Press, 2016.

[2] Statista. (2016) Number of participants in tennis in the united states from 2006 to 2014 (in millions). [Online]. Available: http://www.statista.com/statistics/191966/participants-in-tennis-in-the-us-since-2006/

[3] (2009) The arch way. [Online]. Available: http://wiki.archlinux.org

[4] I. Devolder, *Arch Linux Environment set-up How-To*. Packt Publishing, 2012.

[5] (2014) Robot operating system(ROS). [Online]. Available: http://wiki.ros.org

[6] S. Edwards and C. Lewis, "Ros-industrial: applying the robot operating system (ros) to industrial applications," in *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop*, 2012.

[7] J.-B. Donnet and R. C. Bansal, *Carbon fibers*. CRC Press, 1998.

[8] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning robot," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 1. IEEE, 1994, pp. 610–617.

[9] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with hsv color information," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 2001, pp. 334–339.

[10] D. Zhang, Y. Xue, X. Ye, and Y. Li, "Research on chipsdefect extraction based on image-matching," *International Journal on Smart Sensing and Intelligent Systems*, vol. 7, no. 1, pp. 321–336, 2014.

[11] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[12] G. Green, *An essay on the application of mathematical analysis to the theories of electricity and magnetism*. author, 1828.

[13] J. Stewart, *Calculus*. Cengage Learning, 2015.

[14] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[15] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Redwood City, CA, 1994, vol. 400.

[16] A. Munshi, "The opencl specification," in *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 2009, pp. 1–314.

[17] R. Rai, *Socket. IO Real-time Web Application Development*. Packt Publishing Ltd, 2013.

[18] S. H. Tang, D. Nakhaeinia, and B. Karasfi, *Application of Fuzzy Logic in Mobile Robot Navigation*. InTech, 2012.

[19] L.-X. Wang, "Fuzzy systems are universal approximators," in *Fuzzy Systems, 1992., IEEE International Conference on*. IEEE, 1992, pp. 1163–1170.

On Solution of the Dubins Touring Problem

Jan Faigl, *Member, IEEE*, Petr Váňa, Martin Saska, Tomáš Báča, Vojtěch Spurný

***Abstract*— The Dubins traveling salesman problem (DTSP) combines the combinatorial optimization of the optimal sequence of waypoints to visit the required target locations with the continuous optimization to determine the optimal headings at the waypoints. Existing decoupled approaches to the DTSP are based on an independent solution of the sequencing part as the Euclidean TSP and finding the optimal headings of the waypoints in the sequence. In this work, we focus on the determination of the optimal headings in a given sequence of waypoints and formulate the problem as the Dubins touring problem (DTP). The DTP can be solved by a uniform sampling of possible headings; however, we propose a new informed sampling strategy to find approximate solution of the DTP. Based on the presented results, the proposed algorithm quickly converges to a high-quality solution, which is less than 0.1% from the optimum. Besides, the proposed approach also improves the solution of the DTSP, and its feasibility has been experimentally verified in a real practical deployment.**

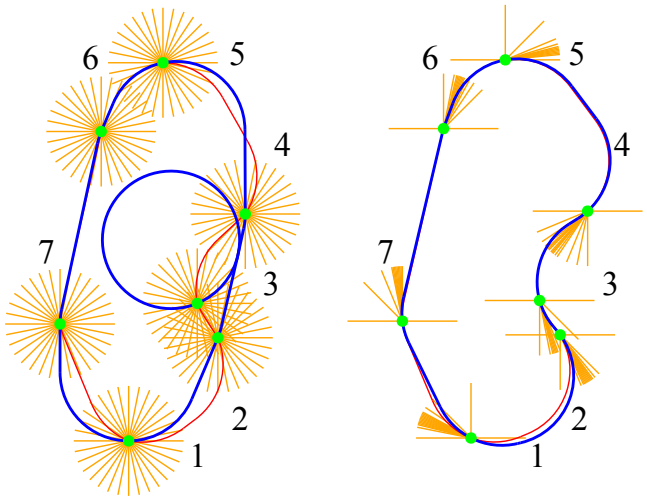
I. Introduction

Curvature-constrained path planning aims to provide a cost efficient path for a non-holonomic vehicle such as Dubins vehicle [1] that models car-like or aircraft vehicles with the minimal turning radius ρ moving with a constant forward velocity. Probably the most utilized problem formulation for surveillance missions with Dubins vehicles is the *Dubins traveling salesman problem* (DTSP) [2], [3] which is a variant of the NP-hard combinatorial optimization TSP. The DTSP stands to find a closed shortest path to visit a given set of target locations in a plane while the path satisfies the motion constraints of Dubins vehicle [4]. The DTSP is also NP-hard [5] as it includes a solution of the TSP; however, it includes additional challenge related to the determination of the optimal heading of the vehicle at each target location. The total tour length depends not only on the sequence of the visits but also on the vehicle heading at each target location.

The difficulty of simultaneous determination of the headings with finding the optimal sequence of the visits motivated researchers to address the DTSP by relaxing the mutual dependency of these two subproblems and assume a sequence of the targets is given, or finite sets of possible headings are assumed in sampling-based approaches [6]. In this paper, we focus on the approach that relies on a given sequence such as, e.g., *Alternating algorithm* (AA) [4], receding horizon methods [7], or *Local iterative optimization* (LIO) [8].

Having the sequence, the problem of determining the optimal headings can be called the *Dubins touring problem* (DTP). Although the DTP can be considered as a subproblem of the DTSP, we believe the DTP is the fundamental building block of routing problems with Dubins vehicle, and thus it deserves a dedicated formulation. For example in the recently introduced *Dubins orienteering problem* (DOP) [9], a solution of the DTP is a part of the target insertion/deletion step, and the solution of the DOP depends on the sum of the collected rewards, and thus it may not necessarily depend only on the final tour length as in the DTSP. Therefore, in this paper, we focus on the solution the DTP to quickly find a high-quality solution with the estimation of its gap to the optimal solution. The presented approach is motivated by practical needs of the robotic competition MBZIRC [10], [11], where a high-quality solution found in the shortest time possible is desirable because of limited time to plan how to quickly collect as many of high rewarding object as possible.

In particular, we investigate a sampling of the headings in the DTP to reduce the number of required samples. Based on the recent results on the so-called *Dubins interval problem* (DIP) [12] utilized to establish a lower bound of the DTP solution, we developed a new informed sampling-based strategy to quickly determine the most promising headings for the optimal solution of the DTP. The proposed approach quickly converges to a solution of high-quality, and it is less computationally demanding than using a uniform sampling of the headings utilized in [13]. The practical influence of the guided sampling is demonstrated in Fig. 1.



(a) Uniform sampling – $N = 224$, $\mathcal{L} = 19.8$, $\mathcal{L}_U = 13.8$, $t = 128$ ms

(b) Proposed sampling – $N = 128$, $\mathcal{L} = 14.4$, $\mathcal{L}_U = 14.2$, $t = 76$ ms

Fig. 1. A solution of the DTSP for a given sequence of the targets (the green disks) with the total number of samples N, final path length $\mathcal{L}$, and lower bound $\mathcal{L}_U$. The found solution is the blue curve, and the red curve is its lower bound determined as a solution of the Dubins interval problem (DIP) with the cost $\mathcal{L}_U$. The uniform sampling utilizes 32 heading values per each target. The required computational time is denoted t.

The authors are with the Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S.

II. Related work

One of the first DTP-based approaches to the DTSP is the AA [4] in which headings are first established for even edges of the sequence by straight line segments, and then, the optimal Dubins maneuvers are determined for the odd edges. Later on, this approach has been improved by a metaheuristic procedure based on a greedy randomized adaptive search [14]. In [15], the same authors consider a distance between two consecutive targets to improve the basic idea of the AA. Only two consecutive targets in the sequence are considered in all these approaches, and thus these algorithms are computationally efficient and their time complexity for n targets can be bounded by $O(n)$.

A look-ahead approach based on more targets in the sequence has been proposed in [7] where three consecutive targets are considered, and the authors report improved results over the simple AA. The idea has been further investigated in [16] for a combination of the k-look-ahead technique with the local improvement of the 2-Opt heuristic; however, the authors do not report on the number of utilized headings per each target and also do not report on the computational time.

An optimal solution of the DTP based on the convex optimization has been proposed in [17] for instances where each pair of consecutive targets are more than four times the minimal turning radius apart. The authors reduce the DTP to a family of n-dimensional convex optimization sub-problems where the number of sub-problems can be bounded by 2^{2n-2}.

The aforementioned heuristic approaches provide a solution of the DTP, but none of them provides a tight-lower bound. The first systematic procedure to provide both a solution and its lower bound has been presented in [18] and further evaluated in [13], but without presenting computational requirements. The herein proposed informed sampling strategy directly builds on the results of the lower bound presented in [13] and also on the solution of DIP introduced by the same authors in [12]. The main contribution of the current paper is in the increased computational efficiency by avoiding dense uniform sampling of the headings, and thus a better solution can be found for a limited computational time than for the uniform sampling.

III. Problem Statement

The *Dubins touring problem* (DTP) stands to determine a shortest curvature-constrained tour to visit a given sequence of n target locations, $\mathcal{P} = (p_1, \ldots, p_n)$, $p_i \in \mathbb{R}^2$. The state q of Dubins vehicle [1] is represented as a triplet $q = (x, y, \theta)$ and $q \in SE(2)$, where $(x, y) \in \mathbb{R}^2$ is the vehicle position in a plane and $\theta \in \mathbb{S}^1$ is the vehicle heading at (x, y). Dubins vehicle is constrained to move only forward at the constant speed v and has the minimum turning radius restricted to ρ. The motion of the vehicle can be described as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, \quad |u| \leq 1, \tag{1}$$

where u is the control input.

In the DTP for the DTSP, Dubins vehicle is requested to visit a given sequence of n target locations $\mathcal{P}$ and return to the starting location. Since the order of the targets is given, the problem is to find a particular heading for each target while the tour constructed from Dubins maneuvers [1] is the shortest possible. This can be formulated as an optimization problem for n variables representing particular headings $T = \{\theta_1, \ldots, \theta_n\}$ with the piecewise continuous cost function:

$$\mathcal{L}(T, \mathcal{P}) = \sum_{i=1}^{n-1} \mathcal{L}(q_i, q_{i+1}) + \mathcal{L}(q_n, q_1), \tag{2}$$

where $\mathcal{L}(q_i, q_j)$ is the length of the shortest Dubins maneuver between the configurations q_i and q_j. The optimization problem can be stated as follows.

Problem 3.1 (DTP):

$$\begin{aligned} &\textit{minimize}_{\,T} \quad \mathcal{L}(T, \mathcal{P}) = \sum_{i=1}^{n-1} \mathcal{L}(q_i, q_{i+1}) + \mathcal{L}(q_n, q_1) \\ &\textit{subject to} \quad q_i = (p_i, \theta_i), \;\; p_i \in \mathcal{P}, \; i = 1, \ldots, n \end{aligned}$$

The proposed approach for the optimal solution of the DTP is based on a solution of DIP, which is detailed in Section III-A. Therefore, we further distinguish a particular value of the heading θ and an interval of heading values Θ in this paper. Moreover, based on the heading intervals we can establish a lower bound of the optimal solution of the DTP, while a feasible solution represents an upper bound.

A. Dubins Maneuvers and Dubins Interval Problem (DIP)

In [1], Dubins shows that for two states q_i and q_j the optimal path for a vehicle with the minimal turning radius ρ is one of the six possible maneuvers that consist of the straight line segment S and a part of the circle with the radius ρ denoted by the C-segment, which is further distinguish based on the orientation of the circle as L and R. Each optimal path can have at most three segments (zero length segments are allowed) which provide two types of paths:

- CCC type: LRL, RLR;
- CSC type: LSL, LSR, RSL, RSR.

The optimal path connecting two states is called Dubins maneuver, and it can be computed by a closed-form expression. The optimal path is easy to compute if headings at the locations are prescribed. However, the length of Dubins maneuver is only a piecewise continuous with the discontinuity between CCC and CSC maneuvers, depending on the mutual distance of the states and headings.

A generalization of this simple Dubins planning for intervals of possible headings at the locations instead of a single heading value at the particular state has been proposed in [12]. The authors call the problem as the *Dubins interval problem* (DIP), and it is detailed in the next paragraph.

DIP stands to find the shortest Dubins maneuver for two locations p_i and p_j for which the departure angle from p_i is in the given interval $\theta_i \in \Theta_i$ and the arrival angle at p_j must be in $\theta_j \in \Theta_j$, where $\Theta_i = [\theta_i^{min}, \theta_i^{max}]$ and $\Theta_j = [\theta_j^{min}, \theta_j^{max}]$, see Fig. 2. The authors of [12] provide

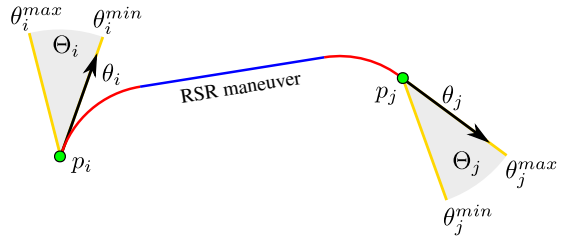


Fig. 2. An instance of the Dubins interval problem to connect p_i and p_j using the departure angle $\theta_i \in \Theta_i$ and the arrival angle $\theta_j \in \Theta_j$

a list of all possible Dubins maneuvers for particular cases of the departure and arrival angles. The types of Dubins maneuvers S, R, and L are further classified as special cases of R and L if parts of the circling maneuver are longer (in an angular distance) than π and they are called as R_ψ and L_ψ, respectively. The optimal solution of DIP is one of the following nine maneuvers according to the particular values of the headings $\theta_i \in \Theta_i$ and $\theta_j \in \Theta_j$:

1) S or L_ψ or R_ψ [1]
2) LSR for $\theta_i = \theta_i^{max}$ and $\theta_i = \theta_j^{max}$;
3) LSL or $LR_\psi L$ for $\theta_i = \theta_i^{max}$ and $\theta_j = \theta_j^{min}$;
4) RSL for $\theta_i = \theta_i^{min}$ and $\theta_j = \theta_j^{min}$;
5) RSR or $RL_\psi R$ for $\theta_i = \theta_i^{min}$ and $\theta_j = \theta_j^{max}$;
6) LS or LR_ψ for $\theta_i = \theta_i^{max}$ and $\theta_j \in \Theta_j$;
7) RS or RL_ψ for $\theta_i = \theta_i^{min}$ and $\theta_j \in \Theta_j$;
8) SR or $L_\psi R$ for $\theta_i \in \Theta_i$ and $\theta_j = \theta_j^{max}$;
9) SL or $R_\psi L$ for $\theta_i \in \Theta_i$ and $\theta_j = \theta_j^{min}$.

The particular optimal maneuver can be selected regarding the shortest Dubins maneuver for these nine cases, which is a bit more complex than a solution of the original Dubins maneuver for single heading values, but the optimal solution of DIP is still determined by a closed-form expression.

Notice, if we allow a full range of heading values, i.e., $\Theta_i = \Theta_j = [0, 2\pi)$, the solution is the CSC maneuver with the zero length circle parts and the length of the straight line segment equals to the Euclidean distance between p_i and p_j.

IV. Proposed Sampling Strategy for the DTP

Based on the analysis and solution of DIP, we propose a new iterative sampling-based algorithm to solve the DTP. The key idea of the proposed approach is to sample heading at the targets by the informed way using a lower bound of the DTP [18]. The lower bound solution is utilized for determining the promising candidates of the heading intervals. Such candidate intervals are iteratively refined, and the process is repeated until a selected angular resolution ϵ is reached or after a finite number of refinements. Moreover, since a more precise estimation of the lower bound is determined at each iteration, the iterative refinement can be terminated once the ratio of the length of the found DTP solution to the value of the lower bound reaches the requested approximation ratio α. Particular components of the proposed algorithm are detailed in the following paragraphs.

[1] Authors of [12] claimed that $L_\psi R_\psi$ or $R_\psi L_\psi$ could also be the optimal solution of DIP, but it is not necessary to consider this case because this is not locally optimal in any instance of DIP.

A. Heading Intervals

The lower bound of the DTP is computed from a solution of DIP for which the heading values at the target locations are constrained to particular intervals [18]. Thus, for each target location $p_i \in P$, a set of heading intervals H_i is maintained. H_i splits the whole range of possible heading values $\theta_i \in [0, 2\pi)$ into k_i not necessarily equally sized heading intervals $H_i = \{[\theta_i^1, \theta_i^2], [\theta_i^2, \theta_i^3], \ldots, [\theta_i^{k_i}, \theta_i^1]\}$. For a better readability, we denote the symbol Θ_i^l to a particular heading interval $\Theta_i^l = [\theta_i^l, \theta_i^{l+1}]$. Hence, for each target location p_i, the set of heading intervals is $H_i = \{\Theta_i^1, \ldots, \Theta_i^{k_i}\}$. Having this notation, we consider the problem of finding a solution of the DTP as a problem to efficiently create a set of particular heading intervals $\mathcal{H} = \{H_1, \ldots, H_n\}$.

B. Lower Bound Solution of the DTP

For the heading intervals $\mathcal{H}$ with up to k intervals per each target, the lower bound of the DTP solution is determined by computing an optimal solution of the corresponding DIP [18]. Since the number of intervals is finite, it is possible to create an oriented graph with n layers. Each layer corresponds to one target location p_i and consists of k_i nodes, each for one heading interval in H_i. The graph nodes are connected by edges representing a solution of DIP, i.e., the weight associated with each edge is the length of the solution of DIP. The graph is visualized in Fig. 3.

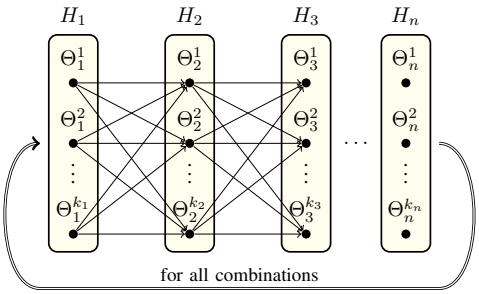


Fig. 3. An example of the search graph for the Dubins touring problem

In this graph, the lower bound $\mathcal{L}_U(\mathcal{P}, \mathcal{H})$ is determined for the current heading intervals $\mathcal{H}$ using a feed-forward search evaluating all possible paths from the starting target p_1 with up to k headings per each target in the sequence $\mathcal{P}$. Since the problem contains n target locations with up to k heading intervals for each target, the graph can have up to nk^2 edges. Therefore, the time complexity to find the shortest tour in the graph can be bounded by $O(nk^3)$. Notice, if a single heading value is used for each heading interval $\theta_i^l \in \Theta_i^l$ for $1 \leq i \leq n$ and $1 \leq l < k_i$, the same procedure can be used to find a feasible solution of the DTP.

C. Refinement of the Heading Intervals

The current lower bound solution is a correct estimation of the minimum tour length; however, it is not necessarily a feasible solution. In fact, the lower bound solution is rarely feasible because of a discontinuity in the heading at each target. Therefore, we introduce the *angular resolution* ϵ to

denote the size of the intervals which directly limits the maximal discontinuity of the vehicle heading.

The algorithm starts with a relatively high ϵ and iteratively refines promising intervals to avoid dense uniform sampling, and thus decrease the computational burden. Having a lower bound solution of the DTP for the current $\mathcal{H}$, the algorithm splits only the intervals presented in the current lower bound solution without losing candidate headings presented in other intervals. By repeating this procedure, the length of the lower bound solution can increase and, eventually, it converges to the optimal solution. Moreover, a single heading value can be sampled for each interval and determine a feasible solution of the DTP for the current $\mathcal{H}$ by the same procedure as the determination of the lower bound.

Notice that even though the solution of DIP may cause discontinuities in particular heading values at the waypoints, any heading value from the interval can be selected and fixed for solving the DTP. Therefore, a feasible solution of the DTP for such fixed heading values is always determined.

D. Proposed Informed Sampling Algorithm for the DTP

The proposed algorithm for solving the DTP is directly based on the aforementioned refinement procedure denoted by $\mathrm{refineDTP}(\mathcal{P}, \epsilon, \mathcal{H})$, which refines $\mathcal{H}$ up to the angular resolution ϵ. Then, for the refined $\mathcal{H}$, a feasible solution is found by the $\mathrm{solveDTP}(\mathcal{P}, \mathcal{H})$ procedure, i.e., using the forward search graph in Fig. 3. The value of ϵ is gradually decreased to the requested ϵ_{req}, and for each ϵ, both the lower and upper bound solutions are determined. Hence, the proposed algorithm has any-time property, and an updated feasible solution is available at the end of each iteration. The refinement procedure is summarized in Algorithm 1.

Algorithm 1: Proposed Iterative Algorithm for the DTP

Input: P – Target locations to be visited
Input: ϵ_{req} – Requested angular resolution
Input: α_{req} – Requested quality of the solution
Output: T – A tour visiting the targets

$\epsilon \leftarrow 2\pi$ `// initial angular resolution`
$\mathcal{H} \leftarrow \mathrm{createIntervals}(P, \epsilon)$ `// initial intervals`
$\mathcal{L}_L \leftarrow 0$ `// init lower bound`
$\mathcal{L}_U \leftarrow \infty$ `// init upper bound`
while $\epsilon > \epsilon_{req}$ and $\mathcal{L}_U / \mathcal{L}_L > \alpha_{req}$ **do**
  $\epsilon \leftarrow \epsilon / 2$
  $(\mathcal{H}, \mathcal{L}_L) \leftarrow \mathrm{refineDTP}(\mathcal{P}, \epsilon, \mathcal{H})$
  $(T, \mathcal{L}_U) \leftarrow \mathrm{solveDTP}(\mathcal{P}, \mathcal{H})$
end
return T

The algorithm is terminated after reaching the requested angular resolution ϵ_{req}, which guarantees a termination after a finite number of iterations because there is a finite number of the heading intervals for ϵ_{req}. Alternatively, the refinement can be terminated when the solution quality of the feasible solution is below α_{req}. However, this may not properly stop the algorithm in a reasonable time for very small α_{req} if the solution requires very fine sampling. Therefore, it is convenient to combine both conditions together. If the anytime property of the algorithm is utilized, the refinement loop can be terminated after a given computational time as the first solution is found in a few milliseconds.

V. Results

The performance of the proposed algorithm has been evaluated in randomly generated instances of the DTP and compared with the uniform sampling strategy to verify if the sampling strategy guided by the solution of the lower bound of the DTP provides better solutions with lower computational requirements than the uniform sampling. Then, the proposed DTP solver has been deployed in solving the DTSP for a given sequence of visits to the targets and compared with the existing heuristics for the DTSP in instances with increasing number of target locations n. For brevity and without loss of generality, we consider Dubins vehicle with $v = 1$ m.s^{-1} and $\rho = 1$ m in the evaluated problems. Finally, a feasibility of the solution found by the proposed solver has been verified in an experimental deployment with the real vehicle that follows the planned path. All the evaluated algorithms have been implemented in C++, and the presented results have been obtained using a single core of the AMD Phenom^{tm} II X6 1090T CPU running at 3.2 GHz, and thus the required computational times can be directly compared.

A. Computational Requirements of the DTP Solvers

The DTP can be directly used in the solution of the DTSP, and therefore, we studied computational requirements of the sampling-based solvers of the DTP for randomly generated DTSP instances for which the sequence of the targets is determined as an optimal solution of the Euclidean TSP found by Concorde [19], e.g., similarly as for the AA approach [4]. It is known that a solution of the DTSP and also DTP depends on the mutual distance of the targets with respect to the minimal turning radius ρ. Therefore, we generate random instances of the DTSP according to the relative density of the targets d inside an area with the dimensions $s \times s$, where s is determined as $s = (\rho\sqrt{n})/d$. Due to the limited space, $d = 0.5$ is considered in the presented results, and 20 random instances are created for $n \in \{10, 20, 50, 70, 100\}$, which gives 100 instances in total.

The real computational requirements of the proposed algorithm are mostly related to the number of headings for the current intervals $\mathcal{H}$. It can be expected that for a finer angular resolution or a low requested ratio α, the computational time will increase. Therefore, we consider the any-time property of the algorithm and evaluate its computational requirements as the quality of found solutions α for the given computational time. The value of α is the ratio of the feasible path length $\mathcal{L}$ and the length $\mathcal{L}_u$ of the lower bound solution, $\alpha = \mathcal{L}/\mathcal{L}_U$.

Summarized results are shown in Fig. 4, where the solution quality is presented as average values from 20 trials of the sequences with 50 target locations and the error bars denote the standard deviations. The results indicate the proposed

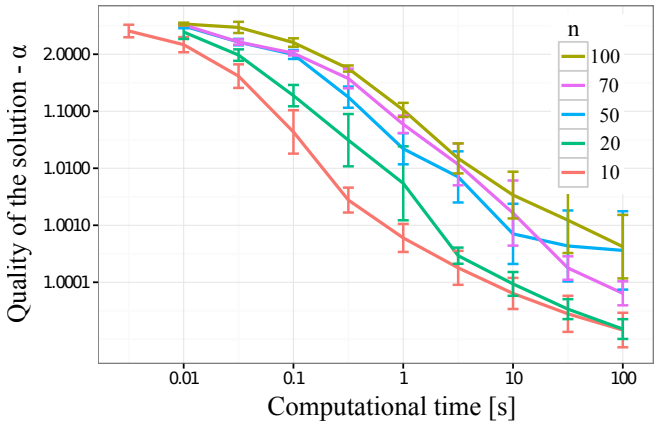


Fig. 4. Average solution quality α computed as the ratio of the solution cost to its lower bound cost determined by the proposed algorithm according to the given computational time. Notice both axes are in log-scale.

algorithm is able to find solutions that are less than 0.1% from the optimal solution (lower bound) in about 10 seconds. Further improvement of the solution for more computational time can be observed; however, from a practical point of view, a solution closer than 0.01% from the optima might be considered as the optimum.

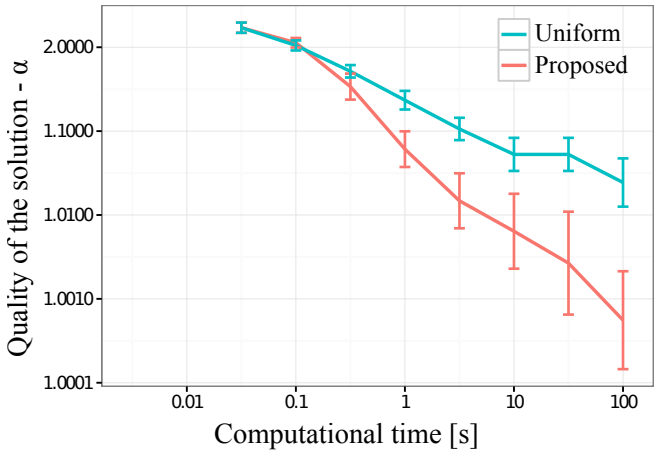


Fig. 5. Average solution quality α according to the given computational time. The results are average values of 20 trials of the DTP instances with $n = 50$ targets and the relative density $d = 0.5$. Notice both axes are in log-scale.

The proposed approach has been compared with the uniform sampling utilized in [13], where the authors use 128 samples of the heading values per each target location to uniformly split possible heading intervals (two times each step). Since a solution of the DTP with 128 samples per each of 50 targets can be computationally demanding, we incrementally increase the number of samples (two times each step) and solve the problem for each individual number of uniformly distributed headings up to the resolution for which the solution of the DTP is found in less than 100 seconds. The results of the evaluation depicted in Fig. 5 indicate that even though the uniform sampling with a high number of samples provides high-quality solutions, from the practical point of view, the proposed refinement converges to the optimal solutions much faster. In particular, the proposed refinement strategy provides solutions in less than 10% from the optimal solution in less than one second, while the uniform sampling needs about 10 seconds. Moreover, the proposed refinement strategy is capable of providing solutions that are less than 1% from the optima in less than 10 seconds on the utilized computer, and it is even capable of providing solutions that are less than 0.1% from the optima. To find such high-quality solutions with uniform sampling, a high number of samples is needed, and the computational requirements are significantly higher. Therefore, for high-quality solutions, e.g., in solving the Orienteering problem [9] where the evaluation of the Dubins tour is used to add or remove particular targets to/from the current reward collecting tour, it is preferable to utilize the proposed refinement strategy.

B. Performance of the Proposed Algorithm in the DTSP

The proposed algorithm has been compared with the existing heuristic solvers for the DTSP, where the sequence of the visits to the targets is determined by a solution of the underlying Euclidean TSP. We consider the same problems as in the previous evaluation and the AA [4] and LIO [8] heuristic algorithms for this evaluation. In addition, the Memetic algorithm for the DTSP [20] is considered to evaluate an influence of the high-quality solution of the DTP provided by the proposed algorithm with the DTSP algorithm that does not rely on the sequence of the visits to the targets.

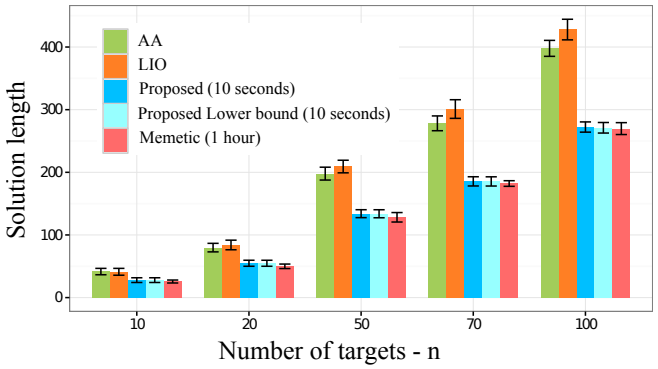


Fig. 6. Average solution length (from 20 trials) for the DTSP instances with n targets and $d = 0.5$. The computational time for the proposed algorithm has been restricted to 10 seconds and for the Memetic algorithm to 1 hour.

The heuristic algorithms AA and LIO found solutions in less than few seconds for a given sequence, and the solutions are not further improved if more computational time is available. On the other hand, the Memetic algorithm can provide high-quality solutions at the cost of high computational requirements. Therefore, we restrict its computational time to 1 hour per each trial and use a computational grid (with the Intel Xeon CPUs). The results are presented in Fig. 6.

It can be observed that the proposed algorithm provides the best solution of the underlying DTP of the DTSP instances. Regarding the lower bound and the feasible solution provided by the proposed algorithm, the found solutions are very close

to the optimal solution of the DTP, albeit the computational time of the proposed algorithm has been limited to 10 seconds. Moreover, quite surprisingly, the solutions provided by the proposed DTP algorithm are very close to the solutions provided by the computationally very demanding Memetic algorithm, which in addition optimizes the sequence of the visits to the targets. This is a source of motivation to employ the proposed DTP solver in the DTSP to further improve the solution once a sequence is determined, especially for instances with high values of the targets density d.

C. Real Experiment

A feasibility of the found solution has been experimentally verified with the hexacopter Unmanned Aerial Vehicle (UAV) with forwarding velocity limited to $v = 4$ m.s^{-1}, the minimal turning radius $\rho = 6$ m, and limited acceleration $a_{max} = 2.67$ m.s^{-2}. The scenario is motivated by a visual inspection of objects of interest that have to be captured by a downward-looking camera in the MBZIRC competition [10]. A visualization of the real deployment is shown in Fig. 7, where the objects of interest can be seen as small light regions inside the disk-shaped target locations with the diameter corresponding to the field of view of the used camera.

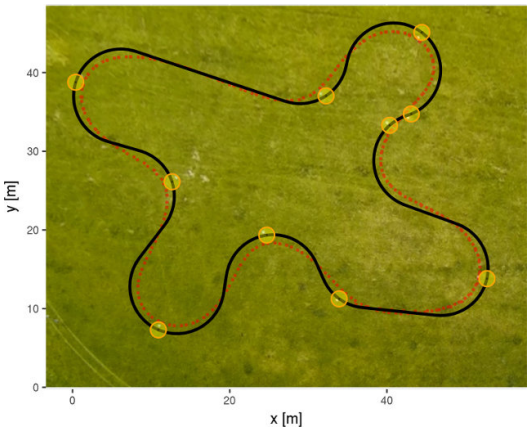


Fig. 7. Planned path (in black) and real executed path captured by the DGPS (in red) of the hexacopter UAV in the DTSP problem with 10 targets

VI. Conclusion

In this paper, we investigate the Dubins touring problem (DTP) as the fundamental building block of routing problems with Dubins vehicle. The DTP is an important subproblem of the DTSP approaches that rely on a sequence of visits to the targets. A new informed sampling-based algorithm for the DTP has been proposed. It uses an iterative refinement of the possible heading intervals where the optimal headings can be found, and it provides high-quality solutions of the DTP. The proposed algorithm utilizes a tight lower bound of the DTP to guide sampling of the suitable heading intervals, and thus the algorithm can provide a quality guarantee of the found solution. The presented results support the feasibility and quick convergence of the proposed algorithm. Moreover, the comparison with the Memetic algorithm indicates that a near-optimal solution of the DTP can significantly improve a solution of the DTSP, albeit a single sequence is utilized in the DTP-based approaches. Based on the presented results, our further work aims to use the proposed algorithm in a solution of the DTSP, where it can provide a quick evaluation of the candidate sequences of the visits to the targets.

Acknowledgments

Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. No. CZ.1.05/3.2.00/08.0144.

References

[1] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, pp. 497–516, 1957.

[2] K. J. Obermeyer, "Path planning for a uav performing reconnaissance of static ground targets in terrain," in *AIAA Guidance, Navigation, and Control Conference*, 2009, pp. 10–13.

[3] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 4, pp. 70–77, 2010.

[4] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *American Control Conference*, 2005, pp. 786–791.

[5] J. Le Ny, E. Feron, and E. Frazzoli, "On the Dubins Traveling Salesman Problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[6] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based path planning for a visual reconnaissance unmanned air vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 619–631, 2012.

[7] X. Ma and D. A. Castanon, "Receding horizon planning for Dubins traveling salesman problems," in *45th IEEE Conference on Decision and Control*, 2006, pp. 5453–5458.

[8] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *IROS*, 2015, pp. 4029–4034.

[9] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem," *Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[10] "MBZIRC team of the Czech Technical University," [cited 17 Mar 2017]. [Online]. Available: http://mrs.felk.cvut.cz/projects/mbzirc

[11] "Mohamed Bin Zayed International Robotics Challenge (MBZIRC)," [cited 17 Mar 2017]. [Online]. Available: http://www.mbzirc.com

[12] S. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Shortest Paths of Bounded Curvature for the Dubins Interval Problem," *arXiv preprint arXiv:1507.06980*, 2015.

[13] S. Manyam, S. Rathinam, and D. Casbeer, "Dubins paths through a sequence of points: Lower and upper bounds," in *ICUAS*, 2016, pp. 284–291.

[14] D. G. Macharet, A. A. Neto, V. F. da Camara Neto, and M. F. Campos, "Nonholonomic path planning optimization for dubins' vehicles," in *ICRA*, 2011, pp. 4208–4213.

[15] D. G. Macharet and M. F. Campos, "An orientation assignment heuristic to the dubins traveling salesman problem," in *Advances in Artificial Intelligence–IBERAMIA*, 2014, pp. 457–468.

[16] P. Isaiah and T. Shima, "Motion planning algorithms for the Dubins Travelling Salesperson Problem," *Automatica*, vol. 53, pp. 247–255, 2015.

[17] X. Goaoc, H.-S. Kim, and S. Lazard, "Bounded-curvature shortest paths through a sequence of points using convex optimization," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 662–684, 2013.

[18] S. Manyam and S. Rathinam, "A Tight Lower Bounding Procedure for the Dubins Traveling Salesman Problem," *arXiv preprint arXiv:1506.08752v2*, 2015.

[19] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "CONCORDE TSP Solver," 2003, [cited 14 May 2017]. [Online]. Available: http://www.tsp.gatech.edu/concorde.html

[20] X. Zhang, J. Chen, B. Xin, and Z. Peng, "A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets," *Chinese Journal of Aeronautics*, vol. 27, no. 3, pp. 622–633, 2014.

Review of Integrated Vehicle Dynamics Control Architectures

Moad Kissai[1], Bruno Monsuez[1], and Adriana Tapus[1]

***Abstract*— Most of the chassis systems are developed by automotive suppliers to improve a specific vehicle performance. Drivers, and consequently vehicle manufacturers, are more concerned by the overall behaviour of the Vehicle. Many coordination architectures have been proposed in the literature in order to integrate different chassis systems in a single vehicle. In this paper, these architectures are compared and discussed. Two major classes are proposed: *Downstream* and *Upstream Coordination*. The purpose of this classification is to help car manufacturers and suppliers standardize an Integrated Vehicle Dynamics Control architecture for faster and more flexible designs.**

***Index Terms*— Control Architectures, Chassis Systems, Systems Coordination, Vehicle Dynamics Control, Control Allocation.**

I. Introduction

In today's automotive sector, driving experience, vehicle safety, and environmental protection are key competition criteria between automakers. In this context, Original Equipment Manufacturers (OEMs) offer constantly new attractive subsystems. Advances in automotive Electronic Control Units (ECUs), sensors, and actuators make the cost of vehicle embedded systems constantly decreasing. The number of chassis control systems grows rapidly as well as the number of competing systems in a single vehicle [1].

However, each vehicle subsystem has an independent control logic to accomplish a specific objective. These objectives compete and may implement contradictory logic: if a brake-based yaw control is used to reduce oversteer or understeer, the longitudinal acceleration demand in a cornering operation will deteriorate. Since new hardware is always expensive, improvements must be provided by the synergies that already exist between subsystems [2]. If instead of implementing competing systems, subsystems get coordinated, over-actuation would offer new opportunities to improve the overall system's safety and performance [3]. This could be done by developing a supervised overall vehicle control system, where information is shared by many subsystems, less resources are required, and computational costs are decreased therefore avoiding unnecessary duplication [2]. This has been given the name of *interoperability* in [4].

This will influence the vehicle's overall behavior. Thus, the high-level control architecture should be designed by car manufacturers. As different suppliers are concerned, a standardization of actuators, sensors, and software components interfaces is crucial. The development of this architecture requires a close collaboration between vehicle manufacturers and their suppliers. Therefore, any proposed solution should respect the intellectual property rights (IPR) of both sides [5]. In this context, three essential capabilities have been cited in [2]:

- *Adaptability and dynamic reconfiguration* to face environment changes, drivers behaviors, and failures [6],
- *Plug-and-play extensibility* to rapidly insert additional technologies without redesigning the whole architecture,
- *Openness* to support various systems from different sources.

The main purpose of this work is to propose a new classification of integrated control architectures to help designers choose a combination approach for a specific need or for the overall system. This would not be the first try, as in [4] a classification of integrated control methodologies for road vehicles has been made. The architectures have been differentiated as *centralized*, *supervisory*, *hierarchical*, and *coordinated control*. However, no discussion is provided so as to assign an architecture to a specific goal. Another recent classification was proposed in [7], where the classification has been simplified to affect the architectures to the desired goals. Two classes have been proposed: *single-criterion* and *multi-criterion* integration motion control. In the single-criterion integration motion control approach, the integrated systems are combined to improve one single aspect of the vehicle. In the multi-criterion integration motion control approach, the combined systems aim to manage different concurrent vehicle dynamics. However, for each class, we could find different combination methodologies depending on the problem's complexity.

A main difference if not the main difference that we distinguished from the different architectures reviewed, is the position of the coordination layer in the control flow [8]. Two main approaches could be adopted to deal with the subsystems combination problem: either we treat the interactions and the eventual conflicts after the subsystems' operations or we control the commands distribution to the different subsystems in order to generate the desired synergies and avoid conflicts. We call the first approach *systems downstream coordination*, and the second, *systems upstream coordination*.

We start with a presentation of the downstream coordination approach in Section II and we explain why it was preferred by auto-makers until these days. We then give a presentation of the upstream approach and justify

[1]Moad Kissai, Bruno Monsuez, and Adriana Tapus are with ENSTA ParisTech, Department of Computer and System Engineering, 828 Boulevard des Marchaux, 91762 Palaiseau Cedex, France {moad.kissai, bruno.monsuez, adriana.tapus}@ensta-paristech.fr

why it should be investigated and replace the downstream coordination approach (see Section III). In Section IV, we propose a general architecture where the different researches seem to converge to and we discuss the open challenges arising from its use. Finally, conclusion is be drawn and some future works are outlined (see Section V).

II. Systems Downstream Coordination

We choose the appellation "*systems downstream coordination*" because this approach consists of studying the stand-alone chassis systems interactions after the commands generation by these subsystems. The coordination is made downstream. To schematize this approach, in Fig. 1 we reproduce the simple structure given by [8], where this approach is called the "bottom-up approach".

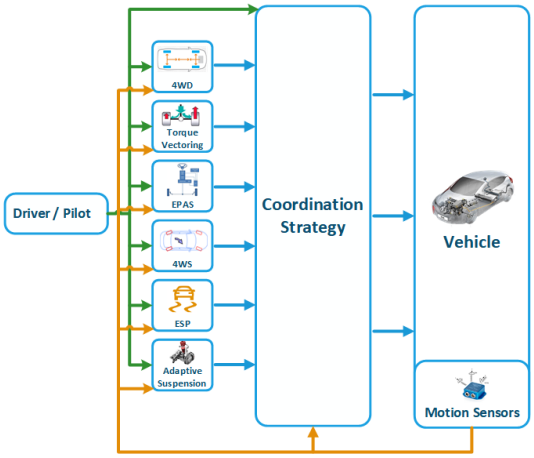


Fig. 1. Structure of the downstream coordination approach (adapted from [8]).

In [4] and [9], this is rather called a *decentralized control*. It follows a parallel structure where each subsystem works separately. Controllers have to work simultaneously. They have their own information system and Electronic Control Unit (ECU), which require more costs and space [10]. The only way to add "integration" in the process is by allowing additional communications between the controllers as the Fig. 2 shows.

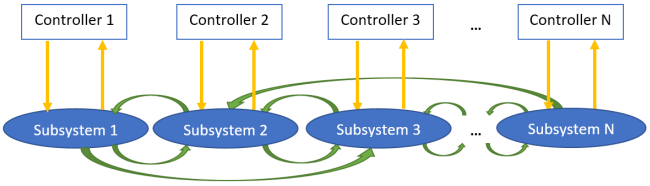


Fig. 2. Decentralized control structure (adapted from [4]).

This architecture is more suited for some manufacturing systems [9]. where it is called *heterarchical*. The fact that there is no global supervision makes this architecture particularly less suitable for vehicle safety control.

In order to develop a coordination strategy, interactions between the subsystems are studied. Automakers engineers use their "*expert knowledge*" to develop handling controllers. This is achieved through a rule-based controllers arbitration deduced from the subsystems interactions studies [8]. In this context, four coordinator types have been distinguished in [4]: *Pure Subsumption*, *Largest Modulus Activation*, *Artificial Neural Network*, and *Fuzzy Logic Control*.

A. Pure Subsumption

Regarding the *pure subsumption* approach, the highest level non-zero command takes precedence over of the other sub-commands (see Fig. 3).

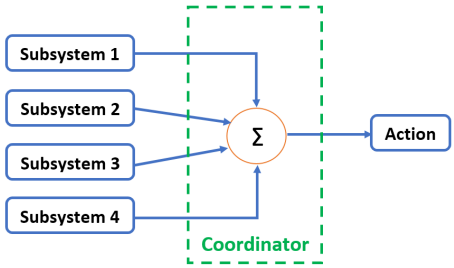


Fig. 3. Pure Subsumption coordination (adapted from [4]).

In this context, the work in [11] used Active Differential (AD), Electronic Stability Control (ESC), and Torque Vectoring (TV) to improve the vehicle lateral performances. A simple method based on prioritizing one system over another has been used. If the yaw torque demand can be satisfied by the AD, then the ESC and TV will not be activated. Otherwise, the rest of the yaw torque demand will be equally shared between the ESC and the TV systems. Both vehicle performance and safety have been improved within this method.

B. Largest Modulus Activation

In the *largest modulus activation*, several high level commands are considered, and the one with the highest modulus takes precedence over the rest (see Fig. 4).

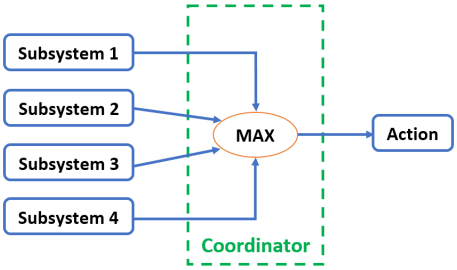


Fig. 4. Largest Modulus Activation coordination (adapted from [4]).

Both pure subsumption and largest modulus activation methods are characterized by modes switching. These switches generate undesirable transients that could destabilize the overall system. In this context, the *Artificial Neural Network* and the *Fuzzy Logic Control* were introduced.

C. Artificial Neural Network

The *Artificial Neural Network* consists of simple averaging or via a non-linear interpolation function weights (see Fig. 5).

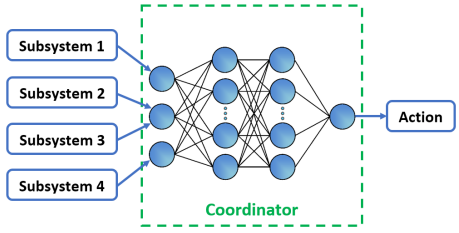


Fig. 5. Artificial Neural Network coordination (adapted from [4]).

These functions could be chosen to ensure smooth transitions between coordination modes, actuator saturation avoidance, etc. [12].

D. Fuzzy Logic

The *Fuzzy Logic* uses "easily understood" rule-based coordination functions (see Fig. 6).

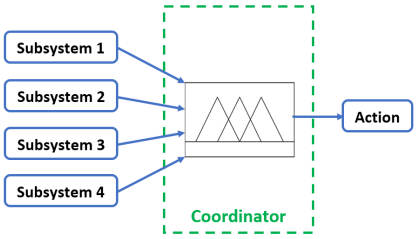


Fig. 6. Fuzzy Logic coordination (adapted from [4]).

Here again, the highest level predominates but smooth transitions are ensured [4]. In [13] for example, a fuzzy logic scheme and weighting factors are used to coordinate the different systems. The controller agents computes combined control signals for the steering angle and the wheel torque depending on the targeted performances priority. The main advantage is the conflict mitigation, for example, when the braking controller has to track simultaneously the yaw rate reference and the longitudinal acceleration demand.

E. Architecture's Compatibility

From the examples in the previous subsections, it appears that the downstream coordination approach is more suitable for single objective coordination control. For multi-objective control, Other coordination algorithms could be cited as the *Linear Quadratic Regulator (LQR)*, *Lyapunov functions*, *sliding mode control*, *adaptive control*, etc. These optimization methods have the major advantage of dynamic coordination, which ensure dynamic transitions. They offer possibilities for robust and fault-tolerant vehicle control systems without *a-priori* knowledge of anticipated failure modes [4]. Thus, the redundancy (that could be very expensive) only concerns the basic hardware needs. These methods are rather used in an upstream coordination approach.

III. Systems Upstream Coordination

Here, the coordination is carried out upstream the subsystems. The commands are distributed in a way to avoid the conflicts downstream the subsystems. A multivariable controller is placed between the driver/pilot commands and the chassis systems. The controller design is based on a coupled nonlinear vehicle model that give insights about the possible conflicts before reaching them. In [8], this approach is called "top-down approach". To illustrate this approach, we reproduce the structure given in [8] (Fig. 7).

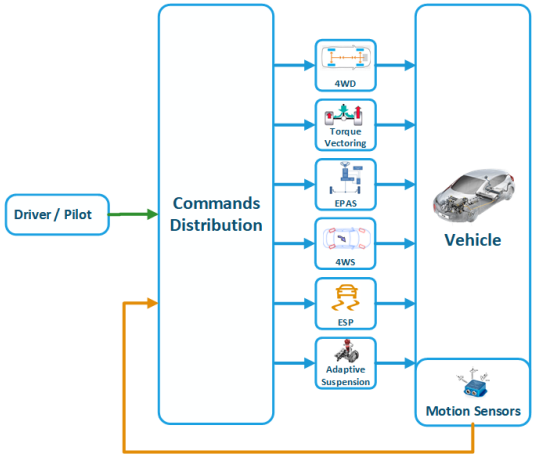


Fig. 7. Structure of the downstream coordination approach (adapted from [8]).

In accordance to this approach, three architectures have been distinguished in [4] and [9]: *centralized control*, *supervisory control*, and *decentralized control*. As we have mentioned, the decentralized control rather corresponds to downstream coordination. We present only the centralized control (A) and the supervisory control (B).

A third class could be distinguished, which is more of an extension of the supervisory control, called the multi-layer architecture (C).

A. Centralized Control

A central global controller is responsible of taking all the control decisions. In general, this controller follows the global multivariable control formalism as it has been realized in [14]-[17]. Fig. 8 illustrates this concept.

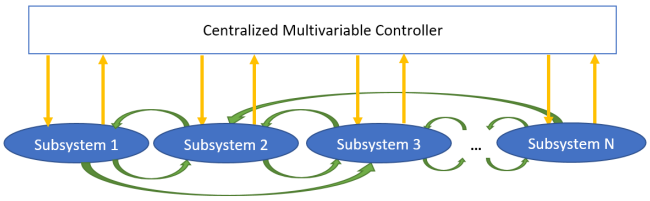


Fig. 8. Centralized control structure (adapted from [4]).

However, it has been pointed out in [18] that any desired fail-safe redundancy of micro-controllers or power converters increases rapidly the cost of the control components. For

that matter, a distributed control method has been preferred, which can be assimilated to a supervisory control [4].

B. Supervisory Control

Supervisory control represents an intermediate between the centralized and decentralized control. Indeed, a supervisory layer is added to a decentralized structure to add more information in the process. Fig. 9 illustrates this approach.

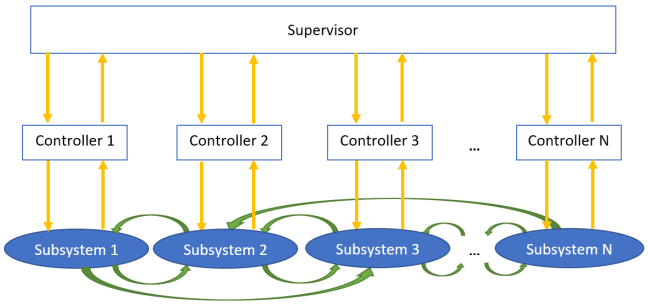


Fig. 9. Supervisory control structure (adapted from [4]).

Three advantages could be deduced from this architecture:

- **Fault-tolerance**: it ensures a minimum of operations safety even if the high-level controller fails,
- **Extensibility**: it can be evolved to a multi-layer hierarchical structure to add more functionalities,
- **Modularity**: it allows manufacturers and suppliers develop independently complementary control algorithms.

Using this structure, [10] used three main layers with two levels of abstraction:

1) *Decision Layer*: Identifies the current driving situation first, then decide how to coordinate the subsystems actions,
2) *Control Layer*: Transforms the control objectives generated by the Decision Layer to references for each of the local controllers,
3) *Physical Layer*: Contains simply the different actuators and sensors.

It should be noted that the decision layer plays a major role to ensure the overall system safety. It is responsible of two main tasks: classifying the current driving situation and deciding how coordination should be made. For example, in [10], a k-means data-based algorithm and a decision logic module based on a set of heuristic rules have been used[1].

C. The Multi-layer architecture

For more flexibility and more comprehensive control, the functional requirements should rather be separated while ensuring a supervised control. Fig. 10 illustrates this method.

Each layer has a specific function [19]:

1) Layer 1: Generation of vehicle motion reference,
2) Layer 2: Decision on the control mode based on the vehicle state recognition,
3) Layer 3: Calculation of the generalized forces and moments at the vehicle's center of gravity through the high-level controllers,
4) Layer 4: Distribution of the commands to the available actuators in an optimal or sub-optimal way through control allocation logic,
5) Layer 5: Control of stand-alone subsystems to follow the commands that comes from the layer 4,
6) Layer 6: Execution of the various operations through smart actuators composed of low-level effectors (e.g. electric motor, hydraulic valve ...etc.) and their own controllers.

D. Architecture's Compatibility

This approach can be used for single objective coordination control. For example, lateral performances have been improved by integrating Active Front Steering (AFS) and the brake-based Dynamic Yaw Control (DYC). In this context, model predictive control [20], model-following control [21], and even fuzzy logic control in an upstream coordination structure [22], [23] were used. These studies showed not only improvement of lateral stability, but also manoeuvrability and agility were enhanced. However, the current cost of this architecture do not justify the achieved gains.

Most of interesting studies carried out using this approach concern the multiple objective coordination control. In this context, various integration methods have been adopted. For example, to deal with lateral and vertical integration, active suspension has been combined with brake-based control in [24]. A Linear Parameter Varying (LPV) design with fault-tolerant control has been used. The commands distribution is made by using three weighting functions for lateral acceleration, heave acceleration, and suspension deflection. The results showed attractive improvements in rough surface conditions.

In the past years, control allocation techniques became more preponderant, e.g. [19], [25]-[27]. For example the authors of [26] used an Integrated Chassis Control (ICC) strategy to improve cornering performance in high speed by combining the ESC, the 4-Wheel Drive (4WD) and the Active Roll Control (ARC) systems. The control architecture is composed of thee parts: a supervisory controller that determines the target vehicle motions, the upper-level controller that calculates the target forces and moment and the lower-level controller that optimally distributes the actuator inputs. The same architecture has been adopted in [27]. The subsystems coordination algorithm uses restriction weights that changes according to the performances targeted. The investigations were carried out by a hardware-in-the-loop test rig, which demonstrated potential enhancement for different performances. Regarding the optimization methods, we can cite 3 approaches [28]: *Update laws* [29], *Repeated optimization* [30], [31] and *Precomputed laws* [32].

However, as pointed out by [7], all the results are made by simulation only. There is a clear lack of experimental results and benchmark requirements that allow comparison between the different methods.

[1]Comparison of decision techniques is beyond the scope of this paper.

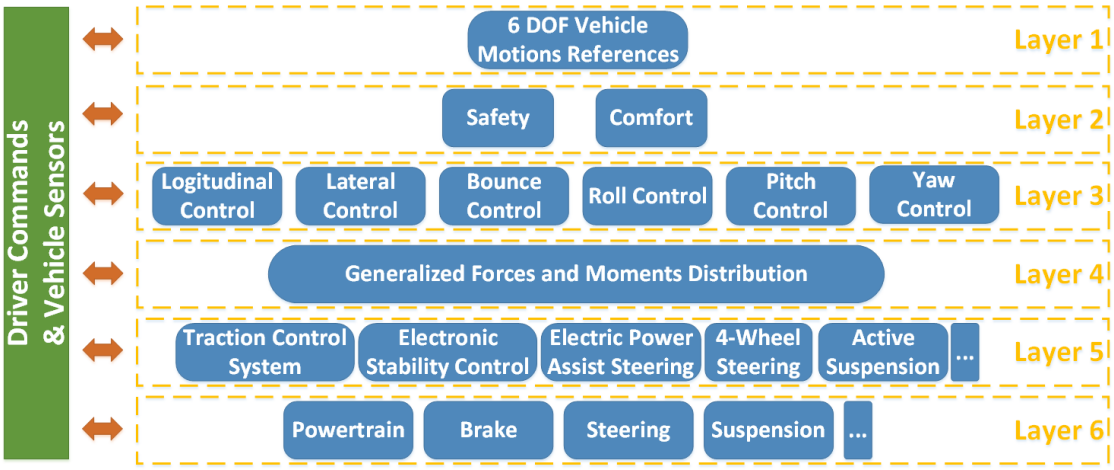


Fig. 10. Multi-layered control architecture [19].

IV. Comparison and Challenges

A. Relevance of the new classification

We recall that two enriching reviews of integrated architectures have already been proposed in [4] and [7]. In [4], emphasis has been put on the architectures' topology. The authors pointed out two major extremes, *the fully decentralized control* and *the fully centralized control*, considering the *supervisory control* as an intermediate between them. This allowed the authors to highlight the benefits of a multi-layer architecture and to draw specific requirements related to the architecture topology: *modularity*, *simplicity*, *fault-tolerance*, and *openness*. However, no relations were proposed between the different structures and control objectives. As "*simplicity*" should be one of the architecture's requirements, a multi-layer architecture may seem exaggerated for single-objective control problems.

The authors of [7] focused rather on control objectives. Two classes have been proposed: *single-criterion* and *multi-criterion*. They studied a large amount of examples for both classes, but conclusions were limited to the control methods and challenges. They particularly highlighted that most *single-criterion* solutions are using rule-based methods (e.g. fuzzy logic control), while recent researches around *multi-criterion* control are using optimization methods, especially control allocation. Unlike [4], no discussion about the architectures topology is provided. We recall that fuzzy logic could be used downstream the stand-alone subsystems for single-objective control [8], or upstream the stand-alone subsystems for multi-objective control [22].

In order to gather the different stakeholders to start thinking on the standardization of integrated vehicle dynamics control architectures, we have to make a bridge between the two types of classifications. For this reason, we believe that the architecture topology should be related to the control objective. As we are interested in subsystems coordination, the topologies classification should be linked to the coordination layer position with respect to stand-alone subsystems. In this context, two approaches have been distinguished in [8]: the "bottom-up" approach (which is called in this paper downstream coordination) approach, and the "top-down" approach (referred to the upstream coordination). However, in [8], only the centralized control structure was given to the top-down approach. The concluding remarks were that the top-down approach is unfavourable from an industrial perspective for its difficulty and cost[2]. The bottom-up approach was then chosen and Fuzzy Logic with β-phase plane control was selected. The main differences that we provide with respect to [8] are a broader distinction of each approach and their suitability for control objectives. Therefore, the control designer would be able to automatically choose a standardized architecture depending on its objectives. The new classification proposed in this paper seems to be relevant within this framework. *Upstream* or *downstream coordination* not only designate where, but also how the coordination is managed. This has allowed us to link each approach to control objectives. When the coordination is made downstream, rule-based control design is used. This technique can handle well single-objective control problems. As more interactions are added, it is hard to foresee the different couplings induced, so it is more complicated to formalize additional rules to achieve safe coordination. When the coordination is made upstream, the control design is based on a coupled vehicle model. Different interactions could be predicted. With an adapted optimization technique, multi-objective control problems could be handled.

B. Comparison of the two approaches

1) Complexity: The main advantage of the downstream approach is the low complexity as long as the number of interactions between competing systems are low. The design methodology consists on first studying the interactions between two or more systems and then establishing adequate rules to benefit from their potential synergies. This could be done for example by using a fuzzy logic approach or the β-phase plane control [8]. On the other hand, complexity

[2]This has changed today.

is the main drawback of the upstream approach. Firstly, the coordination lies on a MIMO (Multiple Inputs Multiple Outputs) controller based on a coupled non-linear vehicle model. Then, optimization techniques are used to solve the problem, in real-time. However, the more complex the interaction between the competing systems get nonetheless with regards to emerging behaviour prediction, the more this upstream approach is pertinent.

2) Cost: Here, also the downstream approach is more attractive. In fact, without modifying the structure of the subsystem control logic, automobile manufacturers can proceed to bulk purchasing from OEMs taking advantage from the economy of scale. Moreover, the architecture does not require additional controllers, but simply a coordination strategy made downstream. Unfortunately, this is not the case for the upstream approach. Not only additional high-level controller(s) is needed, but it may also require additional sensors or estimators [42]. For these reasons, auto-makers are reluctant to implement this approach in real vehicles.

3) Potential: The downstream approach is based on "expert knowledge" methods [8]. This consists in using some preliminary use-case studies and control designers' expertise to develop arbitration strategies. This approach is not well formalized because, theoretically, we cannot cover all the possible use-cases. It is hard to measure this approach potential, so we are not able to know if optimal results are achieved, or at least if better results could be obtained. Another issue is the fact that the subsystems control laws may then be based on different reduced vehicle models[3]. For instance, the development of an Active Rear Steering (ARS) is based on just the simple bicycle model [14]. As a result, putting together different systems based on different behaviour models does not ensure proper operation of the overall vehicle system. In contrast, the upstream approach depicts mathematically the dynamic interactions, which is more suitable for numerical processes. Couplings can be quantified, so we can have some insights about the possibility of finding an optimum solution, a sub-optimal solution or no solution at all. The conflicts are rather prevented than mitigated. For this reason, upstream coordination techniques have more potential in handling multi-objective control problems.

C. Summary

It appears that a compromise should be made between complexity, cost, and potential. To avoid complexity, the control designer could prefer a downstream approach as long as the problem consists on a single-objective control. As more interactions are added, upstream approach becomes necessary for safety matters.

Although the upstream approach may seem expensive, the ECUs have became faster, less cumbersome, and cheaper in the past years. Moreover, various algorithms have been developed and tested. According to [19], [33], [34]-[37], control allocation methods are more suitable for this problem, especially for over-actuated vehicles. These methods have been reviewed and compared in [30], [38]. Linear Programming (LP) and Quadratic Programming (QP), could be executed in a few milliseconds with a limited number of iterations, which real-time computations require. These advances may finally convince auto-mobile manufacturers to adopt this approach.

As the main aim of this paper is to invite auto-makers and suppliers to standardize one overall architecture, one approach should be put on the spotlight. With the arrival of autonomous vehicles, the virtual pilot should take into account multiple objectives at the same time. As we have mentioned, downstream coordination is more suitable for single objective control where the human pilot deals with the other performances. For example, in a high speed cornering manoeuvre, the driver controls the longitudinal acceleration while the active steering could be combined with the brake-based yaw control to control the lateral dynamics and stabilize the vehicle. In autonomous driving, these two objectives should be fulfilled by the virtual pilot. Interactions should be predicted, and conflicts should be avoided rather than mitigated. Consequently, as we are moving towards the autonomous driving, the upstream coordination approach will become necessary.

For this reason, vehicle manufacturers and suppliers should prepare a common overall architecture. This architecture should have the following criteria:

- *Adaptability* to face environment changes and drivers behaviors [2],
- *Fault-tolerance* to propose some degraded modes and ensure a minimum of safety,
- *Dynamic reconfiguration* to ensure soft switching and prevent loss of stability [6],
- *Extensibility* to rapidly insert additional technologies without redesigning the whole architecture [2],
- *Modularity* to ensure flexibility,
- *Openness* to support various systems from different sources without jeopardizing the intellectual property rights of the different stakeholders [2].

As a consequence, the multi-layer architecture with the control allocation method seems to be a better choice to fulfill these criteria [25]-[28], [39]-[43]. For example, the advanced approach presented in [40]-[43] uses vehicle state estimator based on *extended Kalman filter*, *high-level controller* of vehicle general motion, *middle-level control allocation* and *lower-level controllers* for each subsystem as the Fig.11 shows.

It should be noted that this architecture has been validated in a Hardware-In-the-Loop (HIL) procedure [42]. For all these reasons, we believe that this architecture is worth investigating and should be implemented in a real vehicle.

D. Open challenges

1) Real vehicle implementation: As we mentioned, the first challenge is to be able to implement this architecture in a real vehicle. Real-time computation of an optimization method is usually a challenging task. In Integrated Vehicle Dynamics Dynamics Control, the challenge gets bigger as

[3]This is certainly the case when produced by different suppliers but may also be the case for a single supplier.

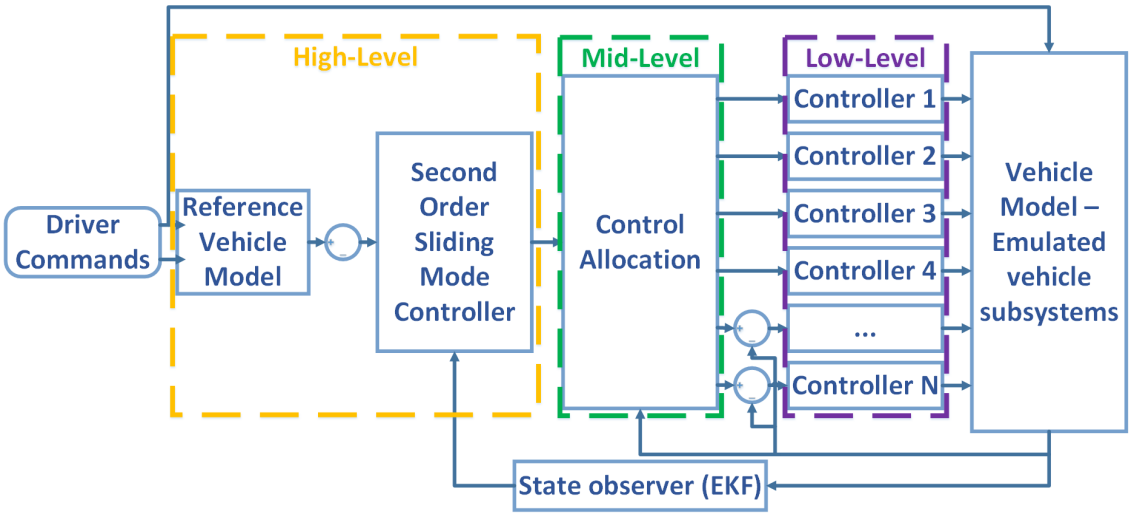


Fig. 11. General architecture of Integrated Motion Control [42].

the coordination technique is located in an inner loop. Consequently, a higher rate is required (around 100 Hz according to [30]). Control allocation with linear or quadratic programming could be the solution [30], [38]. Another issue is to bring together manufacturers and suppliers to collaborate. That goes beyond the scope of our work. Nevertheless, attractive results for both sides could be the first step.

2) "Adaptability": As long as over-actuated systems are concerned, multiple solutions could be found for an optimization problem. Secondary objectives could therefore be achieved. Allocation can be used to favour one solution over another according to the desired behaviour of the vehicle. Consequently, different "feelings" can be generated to realize the same manoeuvre. While controlling the yaw rate using the 4-Wheel Drive system could give the vehicle a sporty behaviour, using the 4-Wheel Steering system for the same manoeuvre could rather give a comfortable behaviour. This is of major importance for autonomous vehicles where the challenge is not only the trajectory following but also how the vehicle follows this trajectory. Control allocation introduces new opportunities to make drivers accept autonomous vehicles. Using specific weighting functions to favour different subsystems combinations could be used to generate feelings of comfort and security, and therefore make drivers trust more their vehicles. However, the intra and inter-individual variability between the drivers is also a challenge. Motions that generate excitement for some people could generate fear among others. So allocation should be adaptable and change over time and maybe even learn from its driver's preferences. Evolutionary algorithms and artificial intelligence could be an interesting approach to investigate in this field too.

V. Conclusion and Future Works

In this paper, a new classification of integrated vehicle dynamics control architectures has been proposed. Two major classes related to the coordination logic have been outlined: the downstream coordination architecture and the upstream coordination architecture. These two classes have been compared. The downstream coordination requires less cost and less complexity but it is limited to single-objective control coordination. In contrast, the upstream coordination can handle multi-objective control coordination but it is more complex and requires more costs. These last drawbacks could be overcame by the advances made in electronic and software engineering.

The literature shows that the integrated vehicle dynamics control architectures are still an object of research. There is a clear lack of benchmark standards and common test procedures to validate the integrated subsystems coordination methods. This paper aims to invite auto-mobile manufacturers and suppliers to adopt an upstream approach and to standardize its structure.

We recognize that more evidence are needed to convince the different stakeholders to favour one architecture over another. That is why our future works will concern the development of the multi-layer architecture with control allocation methods and their comparison with the downstream approach.

The first step of control synthesis is modelling. Regarding ground vehicles, tires are the sole effectors. Distribution of control commands is mainly constrained by tires' potential. As far as combined slip is concerned, this potential varies. Tires' stiffness and maximum efforts should be updated on-line. In this way, we can favour the tires with greater potential. For these reasons, a new tire model is under development. This model is linear in order to facilitate control synthesis, and parameter varying to depict combined slip.

Acknowledgement

The authors would like to thank Dr. Xavier Mouton and Dr. Didier Martinez from the Group Renault for their enriching discussions about today's automotive industry challenges.

REFERENCES

[1] B. Heibing and M. Ersoy (Eds.), "Chassis Handbook - Fundamentals, Driving Dynamics, Components, Mechatronics, Perspectives," Vieweg+Teubner Verlag — Springer Fachmedien Wiesbaden GmbH 2011.

[2] E. Coelingh, P. Chaumette and M. Andersson, "Open-interface definitions for automotive systems - Application to a Brake by Wire System," in *SAE 2002 World Congress*, 2002-01-0267.

[3] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," in *Annual Reviews in Control*, 32(2), 229-252, 2008.

[4] T. Gordon, M. Howell and F. Brandao, "Integrated control methodologies for road vehicles," in *Vehicle Syst. Dyn., Int. J. Vehicle Mech. Mobility*, vol. 40, nos. 1-3, pp. 157-190, 2003.

[5] N. Navet and F. Simonot-Lion, "Automotive embedded systems handbook," CRC press, 2008.

[6] L. Wills, S. Kannan, S. Sander, M. Guler, M., B. Heck, J. Prasad, D. Schrage and G. Vachtsevanos, "An open platform for reconfigurable control," in *IEEE Control Systems Magazine*, 21(3), (2001), pp.49-64.

[7] V. Ivanov and D. Savitski, "Systematization of Integrated Motion Control of Ground Vehicles," in *IEEE Access*, vol. 3, no. , pp. 2080-2099, 2015.

[8] M. A. Selby, "Intelligent Vehicle Motion Control," PhD thesis, University of Leeds, Feb. 2003.

[9] N.A. Duffie, R. Chitturi and J. Mou., "Fault-tolerant heterarchical control of heterogeneous manufacturing system entities," in *Journal of Manufacturing Systems*, 7 (1988), pp.315-328.

[10] C.A. Vivas-Lopez, J.C. Tudon-Martinez, D. Hernandez-Alcantara and R. Morales-Menendez, "Global Chassis Control System Using Suspension, Steering, and Braking Subsystems," in *Mathematical Problems in Engineering*, vol. 2015, Article ID 263424, 18 pages, 2015.

[11] M. Velardocchia and A. Vigliani, "Control systems integration for enhanced vehicle dynamics," in *The Open Mech. Eng. J.*, vol. 7, 2013, pp. 58-69.

[12] C. O. Nwagboso, X. Ouyang, and C. Morgan, "Development of neural-network control of steer-by-wire system for intelligent vehicles," in *Heavy Vehicle Systems*, vol. 9, 2002, pp. 1-26.

[13] J.-X. Wang, N. Chen, D.-W. Pi and G.-D. Yin, "Agent-based coordination framework for integrated vehicle chassis control," in *Proc. Inst. Mech. Eng., D, J. Automobile Eng.*, vol. 223, no. 5, pp. 601-621, 2009.

[14] S. Brennan and A. Alleyne, "Integrated vehicle control via coordinated steering and wheel torque inputs," in *American Control Conference*, (2001), pp.7-12.

[15] M. Nagai, S. Yamanaka and Y. Hirano, "Integrated control of active rear wheel steering and yaw moment control using braking forces," in *JSME International Journal*, 42 (2) Series C (1999), pp.301-308.

[16] M. Harada and H. Harada, "Analysis of lateral stability with integrated control of suspension and steering systems," in *JSAE Review*, 20 (1999), pp.465-470.

[17] M. Nagai, Y. Hirano and S. Yamanaka, "Integrated robust control of active rear wheel steering and direct yaw moment control," in *Vehicle System Dynamics*, 28 (1998), pp.416-421.

[18] N.A. Kelling and W. Heck, "The BRAKE project centralized versus distributed redundancy for brake-by-wire systems," in *SAE 2002 World Congress*, (2002) 2002-01-0266.

[19] A. Soltani, "Low Cost Integration of Electric Power-Assisted Steering (EPAS) with Enhanced Stability Program (ESP)," PhD thesis, Cranfield University, 2014.

[20] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari and D. Hrovat, "Integrated braking and steering model predictive control approach in autonomous vehicles," in *Proc. 5th IFAC Symp. Adv. Automot. Control*, Seascape Resort, CA, USA, 2007, pp. 273-278.

[21] M. Abe and O. Mokhiamar, "An integration of vehicle motion controls for full drive-by-wire vehicle," in *Proc. Inst. Mech. Eng., K, J. Multi-Body Dyn.*, vol. 221, no. 1, pp. 116-127, 2007.

[22] M. J. L. Boada, B. I. Boada, A. Muoz and V. Daz, "Integrated control of front-wheel steering and front braking forces on the basis of fuzzy logic," *Proc. Inst. Mech. Eng., D, J. Automobile Eng.*, vol. 220, no. 3, pp. 253-267, 2006.

[23] Y. Hou, J. Zhang, Y. Zhang and L. Chen, "Integrated chassis control using ANFIS," in *Proc. IEEE Int. Conf. Autom. Logistics, Qingdao*, China, Sep. 2008, pp. 1625-1630.

[24] P. Gspr, Z. Szab and J. Bokor, "Active suspension in integrated vehicle control," in *Switched Systems*, J. Kleban, Ed. Rijeka, Croatia: InTech, 2009.

[25] G. Knobel, A. Pruckner and T. Bunte, "Optimized force allocation - A general approach to control and to investigate the motion of over-actuated vehicles," in *Proc. 4th IFAC Symp. Mechatronic Syst.*, Heidelberg, Germany, 2006, pp. 366-371.

[26] H. Heo, E. Joa, K. Yi and K. Kim, "Integrated chassis control for enhancement of high speed cornering performance," in *SAE Int. J. Commercial Vehicles*, vol. 8, no. 1, pp. 102-109, 2015.

[27] B. Shyrokau, D. Wang and M. Lienkamp, "Integrated vehicle dynamics control based on control allocation with subsystem coordination," in *Proc. 23rd IAVSD Int. Symp. Dyn. Vehicles Roads Tracks*, Qingdao, China, 2013, pp. 1-10.

[28] E.H. van den Berg, "Global chassis control and braking control using tyre forces measurement". TU Delft, Delft University of Technology, 2011.

[29] M. Gerard, "Optimal Control Allocation on Over-Actuated Vehicles - Maximizing vehicle performance and safety through advanced vehicle control strategies". Delft Center for Systems and Control (DCSC), Delft University of Technology, August 10, 2016.

[30] M. Bodson, "Evaluation of optimization methods for control allocation," in *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 703711, 2002.

[31] O. Hrkegrd, "Dynamic control allocation using constrained quadratic programming," in *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 6, pp. 10281034, 2004.

[32] F. Borrelli, A. Bemporad, M. Fodor and D. Hrovat, "An mpc/hybrid system approach to traction control," in *IEEE Transactions on Control Systems Technology*, vol. 14, pp. 541552, May 2006.

[33] W. Chen, H. Xiao, L. Liu, J. W. Zu and H. Zhou, "Integrated control of vehicle system dynamics: Theory and experiment," in *Advances in Mechatronics*, H. Martnez-Alfaro, Ed. Rijeka, Croatia: InTech, 2011.

[34] J. Tjnns, "Nonlinear and Adaptive Dynamic Control Allocation," Norway: NTNU, Trondheim, 2008.

[35] Y. Seongjin, "Coordinated Control with Electronic Stability Control and Active Steering Devices," in *Journal of Mechanical Science and Technology*, Vol 29 (12) (2015) 5409 5416.

[36] H. Zhao, B. Gao1, B. Ren, H. Chen, and W. Deng, "Model Predictive Control Allocation for Stability Improvement of Four-Wheel Drive Electric Vehicles in Critical Driving Condition," in *IET Control Theory Appl.*, 2015, Vol. 9, Iss. 18, pp. 26882696.

[37] Z. Shuai, H. Zhang, J. Wang, J. Li, M. Ouyang, "Lateral motion control for four-wheel-independent-drive electric vehicles using optimal torque allocation and dynamic message priority scheduling," in *Control Engineering Practice*, Vol. 24, March 2014, pp. 5566.

[38] T. A. Johansen and T. I. Fossen, "Control Allocation - A survey," in *Automatica*, Vol. 49, Issue 5, May 2013, Ppp. 10871103.

[39] P. Reinold and A. Traechtler, "Closed-loop control with optimal tire-force distribution for the horizontal dynamics of an electric vehicle with single wheel chassis actuators," in *Proc. Amer. Control Conf. (ACC)*, Washington, DC, USA, Jun. 2013, pp. 2159-2164.

[40] B. Shyrokau and D. Wang, "Control allocation with dynamic weight scheduling for two-task integrated vehicle control," in *Proc. 11th Int. Symp. Adv. Vehicle Control*, Seoul, Korea, 2012, pp. 1-6.

[41] B. Shyrokau and D. Wang, "Coordination of steer angles, tyre inflation pressure, brake and drive torques for vehicle dynamics control," in *SAE Int. J. Passenger Cars-Mech. Syst.*, vol. 6, no. 1, pp. 241-251, 2013.

[42] L. Heidrich, B. Shyrokau, D. Savitski, V. Ivanov, K. Augsburg and D. Wang, "Hardware-in-the-loop test rig for integrated vehicle control systems," in *Proc. 7th IFAC Symp. Adv. Automot. Control*, Tokyo, Japan, 2013, pp. 683-688.

[43] B. Shyrokau, D. Wang, D. Savitski, K. Hoepping and V. Ivanov, "Vehicle motion control with subsystem prioritization," in *Mechatronics*, pp. 1-12, Dec. 2014.

Extrinsic 6DoF Calibration of 3D LiDAR and Radar

Juraj Peršić, Ivan Marković, Ivan Petrović

Abstract— **Environment perception is a key component of any autonomous system and is often based on a heterogeneous set of sensors and fusion thereof, for which extrinsic sensor calibration plays fundamental role. In this paper, we tackle the problem of 3D LiDAR–radar calibration which is challenging due to low accuracy and sparse informativeness of the radar measurements. We propose a complementary calibration target design suitable for both sensors, thus enabling a simple, yet reliable calibration procedure. The calibration method is composed of correspondence registration and a two-step optimization. The first step, reprojection error based optimization, provides initial estimate of the calibration parameters, while the second step, field of view optimization, uses additional information from the radar cross section measurements and the nominal field of view to refine the parameters. In the end, results of the experiments validated the proposed method and demonstrated how the two steps combined provide an improved estimate of extrinsic calibration parameters.**

I. Introduction

Robust environment perception is one of the essential tasks which an autonomous mobile robot or vehicle has to accomplish. To achieve this goal, various sensors such as cameras, radars, LiDAR-s, and inertial navigation units are used and information thereof is often fused. A fundamental step in the fusion process is sensor calibration, both intrinsic and extrinsic. Former provides internal parameters of each sensor, while latter provides relative transformation from one sensor coordinate frame to the other. The calibration can tackle both parameter groups at the same time or assume that sensors are already intrinsically calibrated and proceed with the extrinsic calibration, which is the approach we take in the present paper.

Solving the extrinsic calibration problem requires finding correspondences in the data acquired by intrinsically calibrated sensors, which can be challenging since different sensors can measure different physical quantities. The calibration approaches can be target-based or targetless. In the case of target-based calibration, correspondences originate from a specially designed target, while targetless methods utilize environment features perceived by both sensors. Former has the advantage of the freedom of design which maximizes the chance of both sensors perceiving the calibration target, but requires the development of such a target and execution of an appropriate offline calibration procedure. The latter has the advantage of using the environment itself as the calibration target and can operate online by registering structural correspondences in the environment, but requires both sensors to be able to extract the same environment features. For example, calibration of a 3D-LiDAR and a camera can be based on line features detected as intensity edges in the image and depth discontinuities in the point cloud [1]. In addition, registration of structural correspondences can be avoided by odometry-based methods, which use the system's motion estimated by individual sensors to calibrate them [2], [3]. However, for all practical means and purposes, the targetless methods are hardly feasible due to limited resolution of current automotive radar systems, as the radar is virtually unable to infer the structure of the detected object and extract features such as lines or corners. Therefore, we focus our research on target-based methods.

Target-based 3D LiDAR calibration commonly uses flat rectangles which are easily detected and localized in the point cloud. For example, extensive research exists on 3D LiDAR-camera calibration with a planar surface covered by a chequerboard [4]–[7] or a set of QR codes [8], [9]. Extrinsic calibration of a 2D LiDAR-camera pair was also calibrated with the same target [10], while improvements were made by extracting centerline and edge features of a V-shaped planar target [11]. Furthermore, an interesting target adaptation to the working principle of different sensors was presented in [12], where the authors proposed a method for extrinsic calibration of a 3D LiDAR and a thermal camera by expanding a planar chequerboard surface with a grid consisting of light bulbs. Concerning automotive radars, common operating frequencies (24 GHz and 77 GHz) result with reliable detections of conductive objects, such as plates, cylinders, and corner reflectors, which are then used in calibration methods [13]. In [14] authors used a metal panel as the target for radar-camera calibration. They assume that all radar measurements originate from a single ground plane, thereby neglecting the 3D nature of the problem. The calibration is found by optimizing homography transformation between the ground and image plane. Contrary to [14], in [15] authors take into account the 3D nature of the problem. Therein, they manually search for detection intensity maximums by moving a corner reflector within the field of view (FoV). They assume that detections lie on the radar plane (zero elevation plane in the radar coordinate frame). Using these points a homography transformation is optimized between the radar and the camera. The drawback of this method is that the maximum intensity search is prone to errors, since the return intensity depends on a number of factors, e.g., target orientation and radar antenna radiation pattern which is usually designed to be as constant as possible in the FoV. In [16] radar performance is evaluated using a 2D LiDAR as a ground truth with a target composed of radar tube reflector

Authors are with University of Zagreb Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Unska 3, HR-10000, Zagreb, Croatia, juraj.persic@fer.hr, ivan.markovic@fer.hr, ivan.petrovic@fer.hr

and a square cardboard. The cardboard is practically invisible to the radar, while enabling better detection and localization in the LiDAR point cloud. These complementary properties were taken as an inspiration for our target design.

While the above described radar calibration methods provide sufficiently good results for the targeted applications, they lack the possibility to fully assess the placement of the radar with respect to other sensors.Therefore, we propose a novel method which utilizes a 6 degrees of freedom (DoF) extrinsic calibration of a 3D LiDAR-radar pair. The proposed method involves special calibration target design, correspondence registration, and two-step optimization. The first step is based on reprojection error optimization, while the second step uses additional information from the radar cross section (RCS), a measure of detection intensity. RCS distribution across the radar's FoV is used to refine a subset of calibration parameters that were noticed to have higher uncertainty.

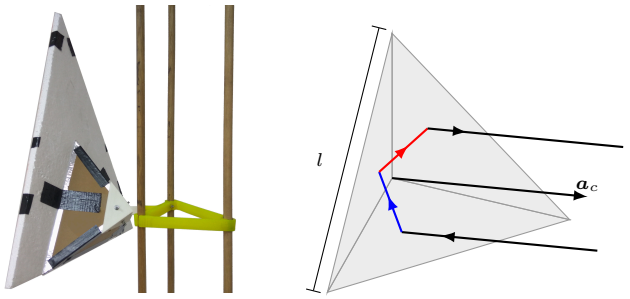
The paper is organized as follows. Section II elaborates the calibration method including calibration target design II-A and data correspondence registration II-B. Section III explains two steps of optimization: reprojection error optimization III-A and FoV optimization III-B. Section IV-A provides details on the setup of experiment conducted to test the method, while the results are given in IV-B. We give final remarks and propose future work in section V.

II. Extrinsic Radar-LiDAR Calibration Method

The proposed method is based on observing the calibration target placed at a range of different heights, both within and outside of the nominal radar FoV. It requires the 3D LiDAR's FoV to exceed the radar's vertical FoV, which is the case in most applications. In addition, due to the problems associated with radars such as ghost measurements from multipath propagation, low angular resolution etc., data collection has to be performed outdoor at a set of ranges $(2 - 10\,\mathrm{m})$ with enough clear space around the target.

A. Calibration Target Design

Properties of a well-designed target are (i) ease of detection and (ii) high localization accuracy for both sensors. In terms of the radar, a target with a high RCS provides good detection rates. Formally, RCS of an object is defined as the area of a perfectly conducting sphere whose echo strength would be equal to the object strength [13]. Consequently, it is a function of object size, material, shape and orientation. While any metal will suffice for the material, choosing other properties is not trivial. Radars typically estimate range and angle of an object as a centroid in the echo signal. Therefore, in order to accurately localize the source of detection, the target should be as small as possible, but which implies a small RCS. Thus, a compromise between the target size and a high enough RCS has to be considered. Radar reflectors, objects that are highly visible to radars, are used not only in intrinsic calibration, but also as marine safety equipment resulting in numerous designs. Given the previous discussion, we assert that one of these designs can be considered as a good compromise and we chose the triangular trihedral corner reflector which consists of three orthogonal flat metal triangles.



(a) Calibration Target (b) Corner reflector

Fig. 1: Constructed calibration target and the illustration of the working principle of the triangular trihedral corner reflector

The constructed radar calibration target and an illustration of the working principle is shown in Fig. 1a. It has an interesting property that any ray reflected from all three sides is returned in the same direction as illustrated in Fig. 1b. The reason behind this is that normals of the three sides form an orthonormal basis. Namely, reflection causes direction reverse of incident ray's component parallel to the surface normal, while the component parallel to the surface tangent plane remains the same. After three reflections, which form an orthonormal basis, the ray's direction is reversed. Due to this property, regardless of the incident angle, many rays are returned to their source, i.e., the radar. Unlike a single flat plate, which has a high RCS but is highly sensitive to orientation changes, trihedral corner reflector provides a high and stable RCS. When the axis of the corner reflector, $\mathbf{a_c}$, points directly to the radar, it reaches its maximum RCS value:

$$\sigma_c = \frac{\pi l^4}{3\lambda^2}, \tag{1}$$

where l is a hypotenuse length of a corner reflector's side and λ is radar's operating wavelength.

Analytical description of the reflector RCS as a function of the orientation is nontrivial. However, from experiments presented in [13], it can be seen that orientation changes of $\pm 20^\circ$ result in a slight decrease of RCS, which can be approximated as a constant, while $\pm 40^\circ$ causes a decrease of $-3\mathrm{dBm}^2$. Furthermore, authors in [17] show that all the rays which go through multiple reflections travel the same length as the ray which is reflected directly from the corner centre. This results in a high localization accuracy.

Corner reflector is visible to the LiDAR, but is difficult to accurately localize it at greater distances due to its small size and complex shape. This problem is solved by placing a flat styrofoam triangle board in front of the reflector. Styrofoam is made of approximately 98% air resulting with low permittivity (around 1.10) and nonconductiveness. These properties make it virtually invisible to the radar, but still visible to the LiDAR. However, instead of a common rectangular shape, we choose a triangular shape with which we can solve localization ambiguity issues caused by finite LiDAR

resolution. Namely, LiDAR azimuth resolution is commonly larger than the elevation resolution, which results with the 'slicing' effect of an object; thus, translating the rectangle along the vertical axis would yield identical measurements until it becomes visible to the next LiDAR layer (which is not the case for the triangle shape). This effect has a stronger impact on localization at greater distances which are required by our method.

Finally, target stand should be able to hold the target at a range of different heights (0–2 m). Additionally, it must have a low RCS not to interfere with the target detection and localization. We propose a stand made of three thin wooden rods which are fixed to a ground wooden plane and connected with a plastic bridge (Fig. 1a). Target attached to the bridge can be slided and tilted to adjust its height and orientation.

B. Correspondence Registration

Correspondence registration in the data starts with the detection of the triangle in the point cloud. The initial step is to segment plane candidates from which edge points are extracted. Afterwards, we try to fit these points to the triangle model. Levenberg–Marquardt (LM) algorithm optimizes the pose of the triangle by minimizing the distance from edge points to the border of the triangle model. A final threshold is defined based on which we accept or discard the estimate. Position of the corner reflector ${}^{l}\boldsymbol{x}_{l}$[1] origin is calculated based on the triangle pose estimate and the known target configuration.

Radar data of interest is a list of detected objects described by the detection angle ${}^{r}\phi_{r,i}$, range ${}^{r}r_{r,i}$ and RCS $\sigma_{r,i}$. The i-th object from the list is described by the vector ${}^{r}\boldsymbol{m}_{i} = [{}^{r}\phi_{r,i}\ {}^{r}r_{r,i}\ {}^{r}\sigma_{r,i}]$ in the radar coordinate frame, $\mathcal{F}_r$: $({}^{r}x, {}^{r}y, {}^{r}z)$. The only structural property of detected objects is contained within the RCS, which is influenced by many other factors; hence, it is impossible to classify a detection as the corner reflector based solely on the radar measurements. To find the matching object, a rough initial calibration is required, e.g., with a measurement tape, which is used to transform the estimated corner position from the LiDAR coordinate frame, $\mathcal{F}_l$: $({}^{l}x, {}^{l}y, {}^{l}z)$, into the $\mathcal{F}_r$, and eliminate all other objects that fall outside of a predefined threshold. The correspondence is accepted only if a single object is left.

The radar correspondence groups are obtained as follows. The target is observed at rest for a short period while the registered correspondences fill a correspondence group with pairs of vectors ${}^{r}\boldsymbol{m}_{i}$ and ${}^{l}\boldsymbol{x}_{l}$. Variances of the radar data $({}^{r}\phi_{r,i}, {}^{r}r_{r,i}, {}^{r}\sigma_{r,i})$ within the group are used to determine the stability of the target. If any of the variances surpasses a preset threshold, the correspondence is discarded, since it is likely that the target detection was obstructed. Otherwise, the values are averaged. In addition, we create unregistered groups where radar detections are missing. These groups are used in the second optimization step where we refine the FoV. Hereafter, we will refer to the mean values of the groups as radar and LiDAR measurements.

[1]In the article, we use left superscript r and l to denote that the value belongs to the $\mathcal{F}_r$ and $\mathcal{F}_l$, respectively

III. Two-Step Optimization

A. Reprojection Error Optimization

Once the paired measurements are found, alignment of sensor coordinate frames is performed. To ensure that the optimization is performed on the radar measurements originating from the calibration target, we perform RCS threshold filtering. We choose the threshold ζ_{RCS} close to the σ_c so that we encompass as many strong and reliable radar measurements while leaving out the possible outliers.

The optimization parameter vector includes the translation and rotation part, i.e., $\boldsymbol{c}_r = [{}^{r}\boldsymbol{p}_l\ \boldsymbol{\Theta}]$. For translation, we choose position of the LiDAR in the $\mathcal{F}_r$, ${}^{r}\boldsymbol{p}_l = [{}^{r}p_{x,l}\ {}^{r}p_{y,l}\ {}^{r}p_{z,l}]^T$. For rotation, we choose Euler angles representation $\boldsymbol{\Theta} = [\theta_z\ \theta_y\ \theta_x]$ where rotation from $\mathcal{F}_r$ to $\mathcal{F}_l$ is given by:

$$ {}^{l}_{r}R(\boldsymbol{\Theta}) = {}^{l}_{2}R_x(\theta_x)\,{}^{2}_{1}R_y(\theta_y)\,{}^{1}_{r}R_z(\theta_z). \tag{2} $$

Figure 2 illustrates the calculation of the reprojection error for the i-th paired measurement. As discussed previously, radar provides measurements in spherical coordinates lacking elevation ${}^{r}\boldsymbol{s}_{r,i} = [{}^{r}r_{r,i}\ {}^{r}\phi_{r,i}\ \sim]$, i.e., it provides an arc ${}^{r}a_{r,i}$ upon which the object potentially resides. On the other hand, LiDAR provides a point in Euclidean coordinates ${}^{l}\boldsymbol{x}_{l,i}$. Using the current transformation estimate, LiDAR measurement ${}^{l}\boldsymbol{x}_{l,i}$ is transformed into the radar coordinate frame:

$$ {}^{r}\boldsymbol{x}_{l,i}(\boldsymbol{c}_r) = {}^{l}_{r}R^{T}(\boldsymbol{\Theta}) \cdot {}^{l}\boldsymbol{x}_{l,i} + {}^{r}\boldsymbol{p}_l, \tag{3} $$

and then ${}^{r}\boldsymbol{x}_{l,i}$ is converted to spherical coordinates ${}^{r}\boldsymbol{s}_{l,i} = [{}^{r}r_{l,i}\ {}^{r}\phi_{l,i}\ {}^{r}\psi_{l,i}]$. By neglecting the elevation angle ${}^{r}\psi_{l,i}$, we obtain the arc ${}^{r}a_{l,i}$ upon which LiDAR measurement resides and can be compared to the radar's. Reprojection error $\epsilon_{r,i}$ is then defined as the Euclidean distance of points on the arc for which ${}^{r}\psi_{r,i} = {}^{r}\psi_{l,i} = 0°$:

$$ \epsilon_{r,i}(\boldsymbol{c}_r) = \left\| \begin{bmatrix} {}^{r}r_{r,i}\cos({}^{r}\phi_{r,i}) \\ {}^{r}r_{r,i}\sin({}^{r}\phi_{r,i}) \end{bmatrix} - \begin{bmatrix} {}^{r}r_{l,i}\cos({}^{r}\phi_{l,i}) \\ {}^{r}r_{l,i}\sin({}^{r}\phi_{l,i}) \end{bmatrix} \right\|. \tag{4} $$

Using the LM algorithm, we obtain the estimate of the calibration parameters $\hat{\boldsymbol{c}}_r$ by minimizing the sum of squared reprojection errors from N measurements:

$$ \hat{\boldsymbol{c}}_{\mathbf{r}} = \underset{\boldsymbol{c}_r}{\arg\min} \left(\sum_{i=1}^{N} \epsilon_{r,i}^{2}(\boldsymbol{c}_r) \right). \tag{5} $$

Optimization of described reprojection error yields unequal estimation uncertainty among the calibration parameters. Namely, translation in the radar plane and rotation around it's normal causes significant changes in the radar measurements. Therefore, parameters ${}^{r}p_{x,l}$,${}^{r}p_{y,l}$ and θ_z can be properly estimated. In contrast, the change in the remaining parameters ${}^{r}p_{z,l}$,θ_y and θ_x causes smaller changes in the radar measurements, e.g. translation of radar along ${}^{r}z$ introduces only a small change in the range measurement. Therefore, these parameters are refined in the second step.

Due to the filtering in the correspondence registration, not many outliers are present in the data. The remaining outliers are removed from the dataset by inspection of the reprojection error after the optimization. Measurements

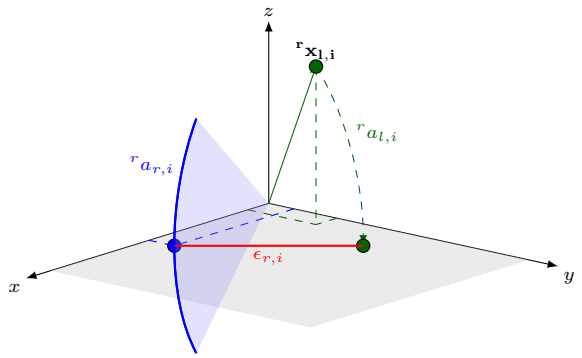
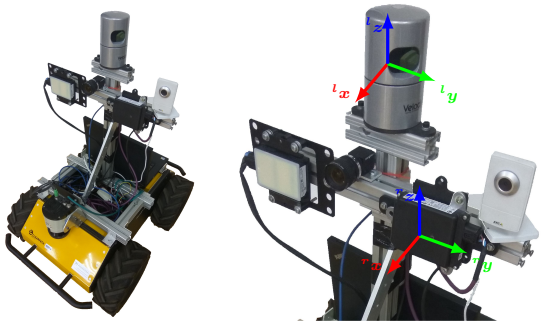


Fig. 2: Illustration of reprojection error calculation. Green: LiDAR's measurement; blue: radar's; red: reprojection error.



(a) Mobile robot (b) Sensor placement

Fig. 3: Mobile robot and sensors used in the experiment.

that surpass the radar's accuracy are excluded from the dataset and optimization is performed again on the remaining measurements.

B. FoV optimization

To refine the parameters with higher uncertainty we propose a second optimization step which uses additional information from RCS. We try to fit the radar's nominal FoV in the LiDAR data by encompassing as many measurements with high RCS as possible. Definition of RCS is such that it is independent of the radar's radiation. However, radar estimates the object RCS based on the intensity of the echo which is dependent on the radiated energy. Intrinsic calibration of a radar ensures that RCS is correctly estimated only within the nominal FoV where it is fairly constant. As the object leaves the nominal FoV, less energy is radiated in its direction, which then results in decrease of RCS until the object becomes undetectable. This effect is used to estimate the pose of the nominal FoV based on the RCS distribution across the LiDAR's data.

Vertical FoV of width $2\psi_f$ is defined with two planes that go through the origin of $\mathcal{F}_r$, $\mathcal{P}_U$ and $\mathcal{P}_D$, with elevation angles $\pm\psi_f$. We propose an optimization in which we position radar's nominal FoV, so that as many as possible strong reflections fall within it, while leaving the weak ones out. The optimization parameter vector consist of a subset of transformation parameters and an RCS threshold, $\boldsymbol{c}_f = [^r p_{z,l}\ \theta_y\ \theta_x\ \zeta_{RCS}]$, whereas other parameters are kept fixed.

After transforming a LiDAR measurement $^l\boldsymbol{x}_{l,i}$ to $\mathcal{F}_r$, the FoV error of i-th measurement $\epsilon_{f,i}$ is defined as:

$$\epsilon_{f,i}(\boldsymbol{c}_f) = \begin{cases} 0 & \text{if inside FoV and } \sigma_i > \zeta_{RCS} \\ d & \text{if inside FoV and } \sigma_i > \zeta_{RCS} \\ 0 & \text{if outside FoV and } \sigma_i < \zeta_{RCS} \\ d & \text{if outside FoV and } \sigma_i < \zeta_{RCS}, \end{cases} \tag{6}$$

where

$$d = \min\{\text{dist}(\mathcal{P}_U, {}^r\boldsymbol{x}_{l,i}), \text{dist}(\mathcal{P}_D, {}^r\boldsymbol{x}_{l,i})\}. \tag{7}$$

Error is greater than zero only if the LiDAR measurement falls inside the FoV when it should not according to the threshold, and vice versa. Function $\text{dist}(\mathcal{P}, x)$ is defined as an unsigned distance from plane $\mathcal{P}$ to point x.

An estimate of calibration parameters is obtained by minimizing the following cost function:

$$\hat{\boldsymbol{c}}_\mathbf{f} = \arg\min_{\boldsymbol{c}_f} \left(\sum_{i=1}^{N} \epsilon_{f,i}^2(\boldsymbol{c}_f) \right). \tag{8}$$

Dependence of the cost function is discrete with respect to the RCS threshold, since change of the threshold does not affect the cost function until at least one measurement falls in or out of the FoV. This results in many local minima and the interior points method was used for optimization, since it was found to be able to converge in majority of analysed cases.

IV. Experiment

A. Experiment Setup

An outdoor experiment was conducted to test the proposed method. A mobile robot Husky UGV, shown in Fig. 3, was equipped with a Velodyne HDL-32E 3D LiDAR and two short range radars from different manufacturers, namely the Continental SRR 20X and Delphi SRR2.

Commercially available radars are sensors which provide high level information in the form of detected object list. Raw data, i.e., the return echo, is processed by proprietary signal processing techniques and is unavailable to the user. However, from the experiments conducted with both radars, we noticed that they follow the behaviour as expected from our calibration method. The only noticed difference is that the target stand without the target was completely invisible to Continental, while the Delphi was able to detect it at closer ranges ($^r r_{r,i} < 5$ m). This effect was present because the Delphi radar accepts detections with lower RCS. However, this did not present an issue, because the stand has a significantly lower RCS than the target and it was easily filtered out. Since the purpose of the experiment is evaluation of the method and not radar performance, in the sequel we only present results for the Continental radar.

Continental radar technical data of interest is given in Table I. Based on the analysis of the reprojection error, radar measurements outside of the azimuth angle range of $\pm 45^\circ$ were excluded from the reprojection error optimization, because they exhibited significantly higher reprojection errors

TABLE I: Continental SRR 20X specifications

Continental SRR 20X	Value
HFoV × VFoV	150° × 12°
Range Accuracy	0.2m
Azimut Accuracy @ HFoV	±2°@±20°; ±4°@±60°; ±5°@±°75

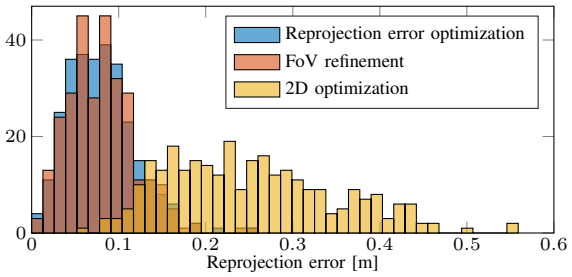


Fig. 4: Histogram of reprojection errors for the two steps of the calibration and the 2D calibration

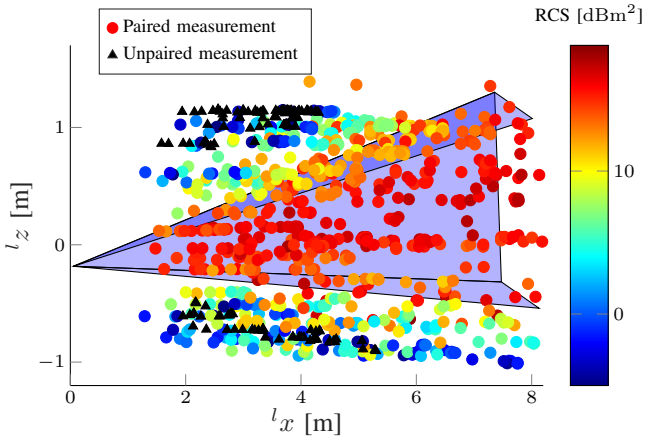


Fig. 5: RCS distribution across LiDAR 3D data and placement of the radar's FoV.

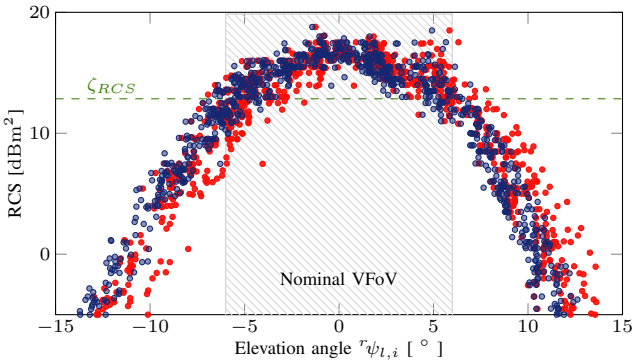


Fig. 6: RCS distribution across radar's VFoV. Red: reprojection error optimization; blue: FoV optimization.

than those inside the range. Considering FoV optimization, we noticed that outside of the azimuth angle range ±60° radar detections were occasionally missing. Therefore, they were excluded from the FoV optimization.

The calibration target was composed of a corner reflector with side length $l = 0.32\,\text{m}$ with a maximum RCS of $\sigma_c = 18.75\,\text{dBm}^2$. Based on vertical resolution of the Velodyne HDL-32E LiDAR (1.33°) we used styrofoam triangle of height $h = 0.65$ m. It ensured extraction of at least two lines from the target, which is a prerequisite to unambiguously determine the pose. Data acquisition was done by driving a robot in the area up to 10 m of range with target placed at 17 different heights ranging from ground up to 2 m height. In total, 880 registered radar-LiDAR measurements were collected, together with 150 LiDAR measurements unregistered by the radar.

B. Results

To assess the quality of calibration results we conducted four experiments. First, we examined the distribution of the reprojection error after both optimization steps and compared it to a 2D optimization, which minimizes reprojection error by optimizing only the calibration parameters with lower uncertainty, i.e., translation parameters ${}^rp_{x,l}$ and ${}^rp_{y,l}$, and rotation θ_z. Secondly, we inspect FoV placement with respect to the distribution of RCS over the LiDAR's data. Afterwards, we examine the correlation between RCS and the elevation angle. Lastly, we run Monte Carlo simulations by randomly subsampling the dataset to examine reliability of the estimated parameters and potential overfitting of data.

Parameters estimated by reprojection error optimization are $\hat{\boldsymbol{c}}_r = [-0.047, -0.132, 0.079\text{m}; -2.07, 3.58, -0.02°]$, while FoV optimization estimates $\hat{\boldsymbol{c}}_f = [0.191, \text{m}; 4.19, -0.84°; 12.85\text{dBm}^2]$. Carefully measured translation by hand between the sensors ${}^r\tilde{\boldsymbol{p}}_l = [-0.08, -0.14, 0.18]^T$ m is given as a reference.

Figure 4 shows distribution of the reprojection error and is composed of three histograms, where we can see how the reprojection error of both steps of calibration is compared to the case of 2D calibration. We notice that neglecting the 3D nature of the problem causes higher mean and greater variance of the reprojection error which implies poor calibration. Furthermore, the FoV optimization is bound to degrade the overall reprojection error because it is not a part of the optimization criterium. However, resemblance between the distributions after the first and the second optimization steps implies low degradation of reprojection error.

In Fig. 5, distribution of the RCS across LiDAR's data is shown. LiDAR's measurements are color-coded with the RCS of the paired radar measurement, accompanied with the black-dyed markers which indicate the lack of registered radar measurements. We can see that within the nominal FoV, target produces a strong, fairly constant reflections. As the elevation angle of the target leaves the radars FoV, the RCS decreases until the point where it is no longer detectable.

To examine the effect of decrease in the target's RCS as a function of the elevation angle after both optimizations, we use Fig. 6. It shows elevation ${}^r\psi_{l,i}$ of each LiDAR measurement transformed into the $\mathcal{F}_r$ and RCS of the paired radar measurement. In the ideal case, i.e. if the transformation was correct and the axis of corner reflector always pointed directly to the radar, the data would lay on the curve which describes radar's radiation pattern in respect to the elevation angle. The dispersion from the curve is present in the both steps due to the imperfect directivity of the target in the

TABLE II: Monte Carlo Analysis Results

	Reprojection Error Optimization	FoV optimization
${}^{r}p_{x,l}$	$\mathcal{N}(-0.047\text{m}, 1.53 \times 10^{-5})$	
${}^{r}p_{y,l}$	$\mathcal{N}(-0.132\text{m}, 6.12 \times 10^{-5})$	
${}^{r}p_{z,l}$	$\mathcal{N}(0.078\text{m}, 2.53 \times 10^{-3})$	$\mathcal{N}(0.174\text{m}, 9.10 \times 10^{-4})$
θ_z	$\mathcal{N}(-2.08°, 1.12 \times 10^{-2})$	
θ_y	$\mathcal{N}(3.59°, 9.50 \times 10^{-1})$	$\mathcal{N}(4.00°, 9.93 \times 10^{-2})$
θ_x	$\mathcal{N}(-0.03°, 8.08 \times 10^{-1})$	$\mathcal{N}(-0.93°, 1.44 \times 10^{-1})$

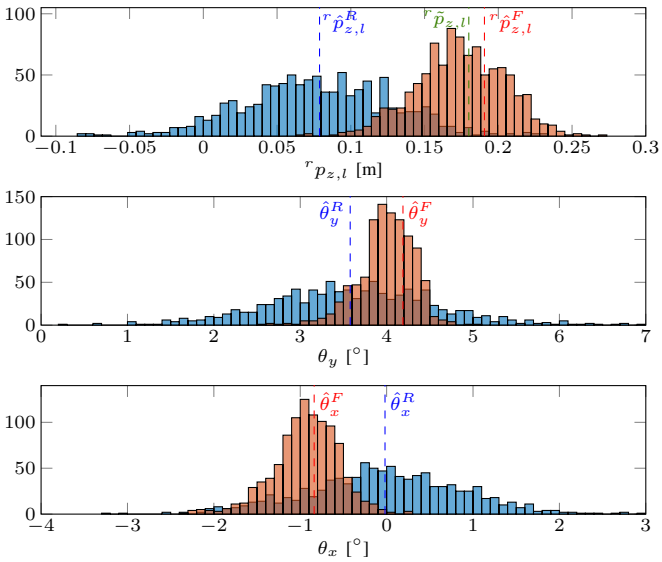


Fig. 7: Monte Carlo Analysis histograms. Red: calibration after reprojection error optimization; blue: with FoV optimization

measurements. In addition, we notice a higher dispersion using only reprojection error optimization which indicates miscalibration.

Lastly, Monte Carlo analysis is done by randomly subsampling our dataset to half of the original size and performing 1000 runs of optimization on different subsampled datasets. The results follow a Gaussian distribution whose estimated parameters are given by the Table II. As expected, distributions of parameters ${}^{r}p_{x,l}$, ${}^{r}p_{y,l}$ and θ_z obtained by the reprojection error optimization have a significantly lower variance than the rest. Figure 7 illustrates how the FoV optimization refines parameters ${}^{r}p_{z,l}$, θ_y and θ_x. We can see overall decrease in variance, as well as the shift in the mean. Estimation of parameter ${}^{r}p_{z,l}$ using reprojection error optimization is clearly further away from the measured value, unlike the FoV optimization's estimate.

V. Conclusion

In this paper we have proposed an extrinsic calibration method for a 3D-LiDAR-radar pair. A calibration target was designed in a way which enabled both sensors to detect and localize the target within their operating principles. The extrinsic calibration was found by a two-step optimization: (i) reprojection error optimization, which was the followed by (ii) FoV optimization which used additional information from RCS to refine the estimate of the calibration parameters. Results of the experiments validated the proposed method and demonstrated how the two steps combined provide an improved estimate of extrinsic calibration parameters. In the future work, we plan to include a camera in the extrinsic calibration. In addition, we plan to improve the results of the calibration by introducing the sensor uncertainty models as radars typically have variable accuracy across the FoV.

Acknowledgment

This work has been supported from the Unity Through Knowledge Fund (no. 24/15) under the project Cooperative Cloud based Simultaneous Localization and Mapping in Dynamic Environments (cloudSLAM). This research has also been carried out within the activities of the Centre of Research Excellence for Data Science and Cooperative Systems supported by the Ministry of Science and Education of the Republic of Croatia.

References

[1] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," in *Robotics: Science and Systems (RSS)*, 2013.

[2] S. Schneider, T. Luettel, and H. J. Wuensche, "Odometry-based online extrinsic sensor calibration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1287–1292.

[3] N. Keivan and G. Sibley, "Online SLAM with any-time self-calibration and automatic change detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5775–5782.

[4] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3D laser scanner and an omnidirectional camera," in *IFAC Symposium on Intelligent Autonomous Vehicles*, 2010, pp. 336–341.

[5] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot." in *IEEE Conference on Robotics and Automation (ICRA)*, 2012, pp. 3936–3943.

[6] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of RGB Camera With Velodyne LiDAR," *WSCG 2014 Communication Papers*, pp. 135–144, 2014.

[7] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 642–648, 2012.

[8] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, "3D LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization," *Int. Journal of Robotics Research*, vol. 31, no. 4, pp. 452–467, 2012.

[9] J. L. Owens, P. R. Osteen, and K. Daniilidis, "MSG-cal: Multi-sensor graph-based calibration," in *IEEE International Conference on Intelligent Robots and Systems (ICRA)*, 2015, pp. 3660–3667.

[10] Q. Z. Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2301–2306.

[11] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *IEEE International Conference on Intelligent Robots and Systems (ICRA)*, 2011, pp. 3283–3289.

[12] D. Borrmann, H. Afzal, J. Elseberg, and A. Nüchter, "Mutual calibration for 3D thermal mapping," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 605–610, 2012.

[13] E. F. Knott, *Radar Cross Section Measurements.* ITP Van Nostrand Reinhold, 1993.

[14] T. Wang, N. Zheng, J. Xin, and Z. Ma, "Integrating millimeter wave radar with a monocular vision sensor for on-road obstacle detection applications," *Sensors*, vol. 11, no. 9, pp. 8992–9008, 2011.

[15] S. Sugimoto, H. Tateda, H. Takahashi, and M. Okutomi, "Obstacle detection using millimeter-wave radar and its visualization on image sequence," in *International Conference on Pattern Recognition (ICPR)*, 2004, pp. 342–345.

[16] L. Stanislas and T. Peynot, "Characterisation of the Delphi Electronically Scanning Radar for Robotics Applications," in *Australasian Conference on Robotics and Automation (ARAA)*, 2015.

[17] C. G. Stephanis and D. E. Mourmouras, "Trihedral rectangular ultrasonic reflector for distance measurements," *NDT&E international*, vol. 28, no. 2, pp. 95–96, 1995.

People Tracking and Re-Identification by Face Recognition for RGB-D Camera Networks

Kenji Koide*[1], Emanuele Menegatti[2], Marco Carraro[2], Matteo Munaro[2], and Jun Miura[1]

***Abstract*— This paper describes a face recognition-based people tracking and re-identification system for RGB-D camera networks. The system tracks people and learns their faces online to keep track of their identities even if they move out from the camera's field of view once. For robust people re-identification, the system exploits the combination of a deep neural network-based face representation and a Bayesian inference-based face classification method. The system also provides a predefined people identification capability: it associates the online learned faces with predefined people face images and names to know the people's whereabouts, thus, allowing a rich human-system interaction. Through experiments, we validate the re-identification and the predefined people identification capabilities of the system and show an example of the integration of the system with a mobile robot. The overall system is built as a Robot Operating System (ROS) module. As a result, it simplifies the integration with the many existing robotic systems and algorithms which use such middleware. The code of this work has been released as open-source in order to provide a baseline for the future publications in this field.**

***Index Terms*— RGB-D Camera Network, People Tracking, Person Identification, Face Recognition.**

I. Introduction and Related Work

Camera network-based people tracking systems have attracted much attention in the computer vision community. They have been applied to various tasks, such as surveillance and human-robot interaction. One of the essential problems for people tracking is person re-identification. To keep track of people who left the camera view, systems have to re-identifies them when they re-appear in the view. In addition to that, a capability of identifying people over days (Long-term re-identification) is required in long-term service scenarios.

Many works proposed person re-identification methods for camera networks [1], [2], [3]. By combining appearance features which are robust to pose, illumination, and camera characteristics changes and sophisticated feature comparison methods, they achieved good people identification performance over multiple cameras. However, the proposed methods rely on the RGB information of each subject, thus, on the appearance of their clothes. Therefore, it is not possible to identify people over days and not applicable to long-term service scenarios.

Several soft biometrical features, such as gait [4], [5] and skeletal lengths [6], have been proposed for overcoming this problem. Since such biometrical features are specific to individuals and invariant over time, they enable to identify people over days. However, those features may be indiscriminative when several people have similar physiques. To reliably identify people, features for person re-identification have to be discriminative and robust to viewpoint changes. One of the most discriminative and reliable features to identify people is the face [7]. However, face features had not been widely used for person re-identification [2] since faces are not always visible, and we need to deal with pose, illumination, and expression (PIE) changes to apply it to camera networks. Some works used face features for person re-identification [8]. However, the use of face features was limited to assist appearance features, and those methods intended to be applied to short-term re-identification tasks.

It was difficult to solve the PIE issues by using traditional face features, such as EigenFace [9], Local Binary Patterns [10], and Scale-Invariant Features Transform [11]. On the other hand, recent deep neural network-based face representations [12], [13], [14] provide robust and discriminative face features and allow to reliably identify a person in presence of those issues. Although face visibility is still a hard problem, recent inexpensive consumer cameras allow to have a dense camera network. A person's face might be visible to any of the cameras under such settings.

In this paper, we propose a people tracking and re-identification system for RGB-D camera networks. The proposed system tracks people by using *OpenPTrack* [15], an RGB-D camera network-based people tracking framework, and learns their faces online to re-identify people who left the camera view once. For real-time performance, we employ a distributed people detection and face feature extraction structure. A PC is connected to each distributed RGB-D camera, and it detects people, extracts face features, and sends those data to a master PC. The master PC aggregates the information to track and re-identify people. This system also provides a predefined people identification capability: a set of people names and face images is given to the system, and it makes the association between tracks and the predefined people to know the names of the tracked people.

The contributions of this paper are two-fold. First, we propose a Bayesian inference-based face classification method. It allows to reliably classify a face according to the confidence of deep neural network-based face comparison results. Secondly, this work is an open source ROS [16] platform-based project [1], and it can easily be integrated with robotic

[1]Kenji Koide and Jun Miura are with the Department of Computer Science and Engineering at the Toyohashi University of Technology, Japan

[2]Emanuele Menegatti, Marco Carraro, and Matteo Munaro are with the Department of Information Engineering at the University of Padua, Italy

*indicates the corresponding author, email{koide@aisl.cs.tut.ac.jp}

A supplementary video is available at `https://goo.gl/yGRdEZ`

[1]The code is available at `https://goo.gl/ypILr7`

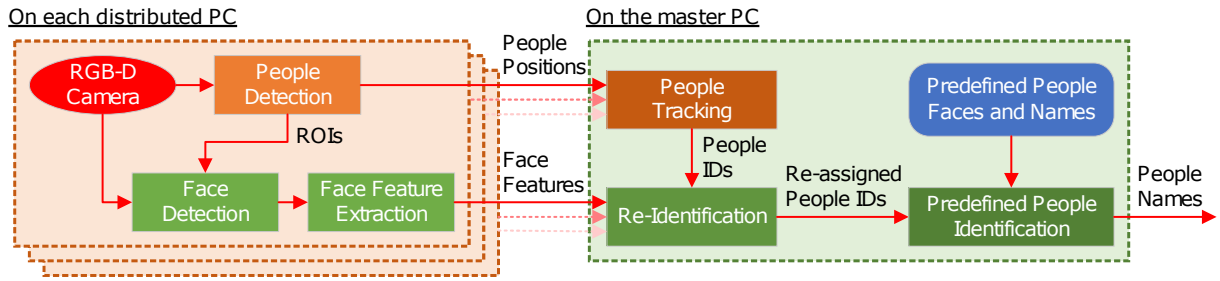


Fig. 1. System overview.

systems. We show the possibility of the integration of the system and robotic systems through the experiment.

The remainder of this paper is organized as follows. Sec. II describes an overview of the proposed method. Secs. III describes the proposed face recognition framework. Sec. IV describes the integration of the proposed system and a mobile robot system. Sec. V presents the evaluation and the experiment of the proposed system. Sec. VI concludes the paper and discusses future work.

II. System Overview

Fig. 1 shows an overview of the proposed system. The proposed system tracks people by using *OpenPTrack* [17], an RGB-D image-based people tracking framework, and re-identify people by face recognition based on *OpenFace* [18], a deep convolutional neural network-based face representation.

On each distributed PC connected to an RGB-D camera, we first perform RGB-D image-based people detection [19]. Then, we calculate ROIs from the detected people positions and detect faces on the ROIs. Face features are extracted from the detected face regions by using a deep neural network provided by *OpenFace*. In this system, the distributed PCs do not send raw camera images, but send only the detected people positions and extracted face features to the master PC. In this way, since those data is very light, we allow our system to be efficient and scalable to a large and dense camera network without bandwidth problems.

While the *OpenPTrack* master node tracks people from the detection positions it receives, our proposed system re-identifies people by face recognition. The re-identification part takes advantage of the tracks, which connect several frames from the same (still unknown) person and the face features associated to those frames. By integrating this information, the system can build online a descriptor for each person face. Then, it associates the tracks of the people with the learned faces. By referring the ID of the face associated with the track of a person, we can keep tracking the identity of the person even if he/she leaves and re-enters the camera view.

This system also provides a predefined people identification capability. A set of people names and faces are given to the system in advance, and the system associates the predefined people faces with the online learned faces. This capability allows to know a specific person's whereabouts and have a rich human-system interaction for applications like housework robots [20].

The proposed system is built on top of the Robot Operating System (ROS), thus, it can be easily combined with other robotic systems.

III. Proposed Method

A. People Detection and Tracking

We integrated the face recognition-based re-identification algorithm with *OpenPTrack* [15], a people tracking framework for RGB-D camera networks. This framework first detects people from point clouds obtained from RGB-D cameras placed in an environment. It removes the ground plane from a point cloud and then applies euclidean clustering to extract candidate clusters of human. Then, it judges whether a cluster is human or not by using an image-based SVM classifier on HOG features. This detection process is performed on each PC connected to an RGB-D camera, and then the detection results are aggregated on a master PC. The detected people are tracked by the combination of global nearest neighbor data association and Kalman filtering with a constant velocity motion model.

Fig. 2 shows a snapshot of *OpenPTrack*. As long as a person is visible from any of the RGB-D cameras, it provides reliable people tracking results. However, once a person leaves the field of view of all cameras, the system will assign a new track ID to the person whenever he/she will re-appear. In such cases, a person re-identification capability is necessary to recover the track of the person in such case.

B. Face Detection

To detect people faces in real-time, we take a top-down face detection approach. We first calculate ROIs from the people detection results provided by *OpenPTrack*. We project the rectangle with a fixed metric dimension (e.g., 0.2 m) to a detected human cluster's top position in the image. A HOG and cascaded SVM-based face detector [21] runs on the ROIs, and face features are extracted from the detected face regions. Fig. 3 shows an example of the face detection results. The green and red boxes indicate the ROIs calculated from the top positions of the detected people clusters, and the detected face regions, respectively.

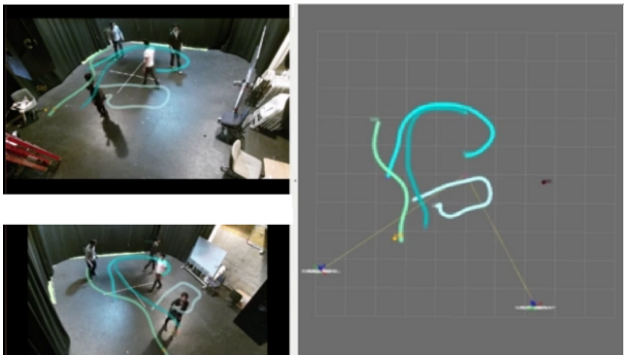


Fig. 2. OpenPTrack, a people tracking framework using a RGB-D camera network, provides robust people tracking. However, person re-identification capability is necessary to recover the track of a person left from the camera view. [17]

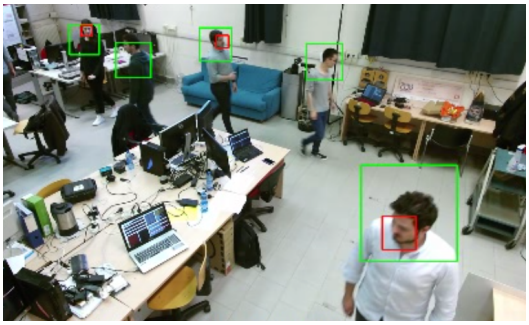


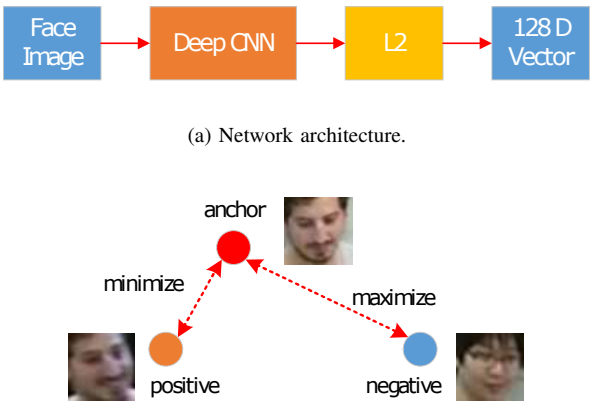
Fig. 3. An example face detection result. The green boxes indicate the ROIs calculated from the detected people positions. The red boxes indicate the detected face regions. Some faces are not detected due to their poses. However, by integrating the face detection results of multiple cameras, most faces will be observable from any of the cameras.

C. Deep Neural Network-based Face Features

We utilize *OpenFace* [18], a face recognition framework, to extract face features. The framework provides an implementation of *FaceNet* [13], a state-of-the-art deep convolutional neural network for face feature extraction. Fig. 4(a) shows the network architecture of *FaceNet*. The network first transforms a face image to a 128-dimensional feature vector by applying a deep convolutional neural network. Then, the feature vector is normalized so that its L2 norm becomes 1. The network is trained to minimize the triplet loss function [13]. This function takes three training data (i.e., *triplet*), a training data to be an *anchor*, a positive data with the same identity as the anchor, and a negative data with a different identity. The triplet loss function minimizes the L2 distance between the anchor-positive pair and maximizes the distance between the anchor-negative pair. As a result, faces of the same person are embedded close together, and face features of different persons are placed far from each other. Thereby, we can judge whether two faces have the same identity or not by calculating the L2 distance between their face features.

D. People Re-identification by Face Recognition

The system learns people faces online and associates the tracks of people and the learned faces to re-identify people who left the camera view by means of Algorithm 1. *face_list* is the list of online learned faces, and *track_list* is the list of people tracks. We first classify face images observed from the track of a person into the online learned faces. If the observed face images are classified as one of the online learned faces with high confidence, we assign the face's ID to the track. If the track is classified as an unknown person, we consider that he/she is a newly appeared person and give a new face ID to the track. While the track is alive, the system keeps the assigned face ID and learned face images. If the track of a person is lost, the system removes the track from *track_list* and moves the assigned face ID and images of the track to *face_list* to make it assignable to new tracks.



(a) Network architecture.

(b) Triplet loss function.

Fig. 4. *FaceNet* framework [13]. This framework employs a deep convolutional neural network and the triplet loss function to obtain discriminative face representations.

Algorithm 1 Assign a face ID to the track of a person

```
face_list is a set of (face_ID, face_images)
track_list is a set of (track_ID, face_ID, face_images)
for all track in track_list do
  add observed face images to track.face_images
  if track.face_ID has not been assigned then
    result ← classify(track.face_images, face_list)
    if result.confidence < threshold then
      continue
    else if result is known person then
      track.face_ID ← result.face_ID
      add result.face_images to track.face_images
      remove result from face_list
    else
      track.face_ID ← new face_ID
    end if
  end if
  if track is not alive then
    remove track from track_list
    add (track.face_ID, track.face_images) to face_list
  end if
end for
```

E. Bayesian Inference-based Face Classification

The simplest way to classify an observed face into registered faces is thresholding: if the distance between the observed face and a registered face is less than a threshold, the observed face is classified as the registered one. If several faces have the distances less than the threshold, the observed face is classified as the face with the minimum distance. Usually, this classification is performed for a certain number of observations (e.g., 5 face images) and the decision is made by majority voting. However, this method ignores the difference of the distances, and it may affect the classification accuracy when several faces have similar distances. The classification method should take into account the difference of the distances as the confidence of the classification for robust classification results.

To reliably classify observed faces, we propose a Bayesian inference-based face classification. Let $p(x_{ij})$ be the probability that the i-th track has the same identity as the j-th learned face, $p(x_{i0})$ be the probability that the i-th track's face has not been registered to the face list (the person is a new person), and $p(x_{0j})$ be the probability that the person who has the j-th registered face is not tracked (the person does not exist in the camera view). From this definition, we obtain the following constraints.

$$\sum_{k=0}^{N} p(x_{kj}) = 1 \quad (j \neq 0) \tag{1}$$

$$\sum_{k=0}^{M} p(x_{ik}) = 1 \quad (i \neq 0) \tag{2}$$

The probabilities can be represented as a table (see Fig. 5). We update this probability table with face images observed from people tracks. According to Bayesian theorem, the posterior probability of $p(x_{ij})$ under an observation y is given as:

$$p(x_{ij}|y) \propto p(x_{ij})p(y|x_{ij}) \tag{3}$$

We calculate the distances between the observed face image and the registered faces as follows and consider the distances as observations.

$$d_{ij} = \min_k \|F(x_i^t) - F(x_j^k)\|, \tag{4}$$

where, x_i^t is the face image observed from the tracker i at time t, x_j^k is the k-th face image of the j-th learned face, and $F(x)$ is the feature extraction function. To model the likelihood function $p(y|x_{ij})$, we randomly sampled correct and wrong face pairs from the LFW face database [22]. The number of the correct and wrong pairs are about 30000 and 60000, respectively. Fig. 6 shows the distributions of the L2 distances between the features extracted from the face pairs. We fit a skew normal distribution to each distribution by using maximum likelihood estimation and gradient decent method. The solid lines in Fig. 6 indicate the fitted skew normal distributions. We denote by $N_p(x)$ and $N_n(x)$ the skew normal distributions of the correct and wrong pairs, respectively. We calculate the likelihood $p(y|x_{ij})$ from a distance d_{ij} as:

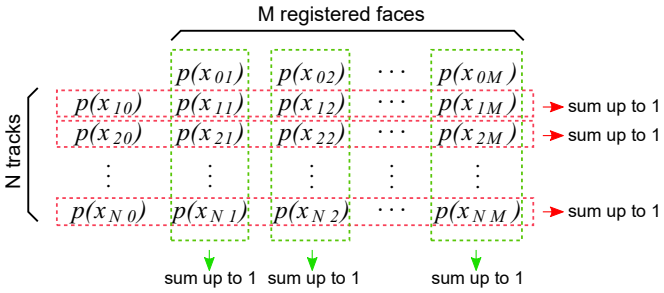


Fig. 5. The posterior probability table. The element at (i, j) is the posterior probability that i-th track has the same identity as the j-th learned face. Rows and columns are iteratively normalized so that they sum up to 1 by using Shinkhorn iteration.

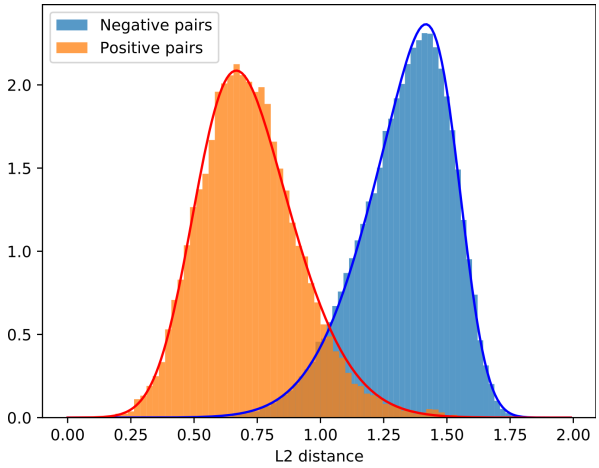


Fig. 6. The distributions of L2 distance of the positive and the negative face pairs. The solid lines indicate the skew normal distributions fitted to the distributions.

$$p(y|x_{ij}) = \begin{cases} N_p(d_{ij}) & (j = i) \\ c \cdot N_n(d_{ij}) & (j = 0) \\ N_n(d_{ij}) & otherwise \end{cases} \tag{5}$$

where, c is a constant which models the tendency that an observed face image is produced from an unknown person. If we take a large c, the system tends to classify an observed face as an unknown face. In this work, we just use $c = 1$, however, it works well for most cases.

After updating the posterior probabilities, we normalize the probabilities so that they satisfy the equation (1) and (2) by using Shinkhorn iteration [23]. It first normalizes every row and then normalizes every column so that they sum up to 1. This normalization is repeated until the probabilities converge. After the normalization, if $p(x_{ij})$ is larger than a threshold (e.g., 0.95), we classify the i-th track as the j-th learned face.

F. Predefined People Identification

To allow a rich human-system interaction, the proposed system provides a predefined people identification capability.

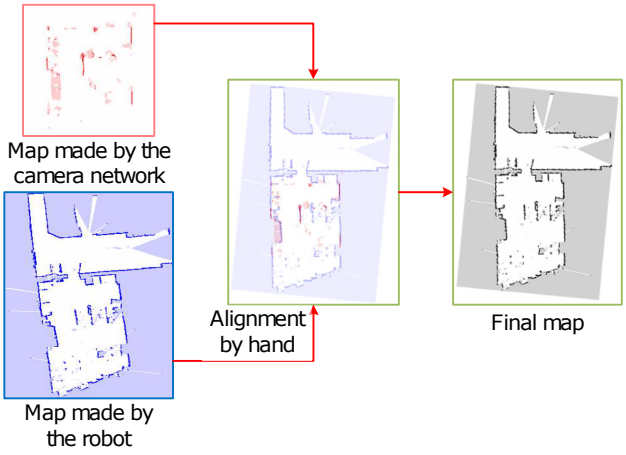


Fig. 7. Aligning an environmental map made by the robot to the one made by the camera network. An occupancy grid map is created from point clouds obtained from the camera network, and then the maps are manually aligned.

Predefined people data are given to the system as a set of people names and face images, and the system associates online learned faces and the predefined people.

We calculate the distance between face images of a predefined person and learned face images, and if the distance is smaller than a threshold, the system associates the predefined person and the learned face. The distance is calculated as the minimum distance of the distances between all combinations of the predefined person's face images and the learned face images.

IV. Cooperation with a Mobile Robot

Since the proposed system has been implemented as ROS modules, it can easily cooperate with robotic systems. In this work, we integrate the proposed system with a mobile robot to demonstrate a human-system interaction service. We use a Turtlebot2 robot equipped with a laser range finder (see Fig. 11). By using ROS navigation stack [24], we made the robot create an environmental map and estimate its current pose. To operate the robot with the camera network, it is necessary to know the transformation between the created map and the camera network coordinates. We first extract points in a certain height range (e.g., 0.1 - 0.3m) from the camera network and create an occupancy grid map. Then, we manually align the map made by the robot to the one made by the camera network (see Fig.7). Note that this map alignment can be semi-automated by giving an initial guess by hand and applying ICP matching between the maps. Fig. 8 shows a snapshot of the integrated system. We can see the robot, the point clouds obtained by the Kinects, and the tracked people in the same view.

V. Experiments

A. Person Re-identification Experiment

We conducted long-term person re-identification experiments. Two RGB-D image sequences were recorded at different days. The subjects changed their clothes (and hair styles) between the recordings, thus, appearance-based re-identification is not effective for this dataset. In each of the sequences, six subjects are walking in the environment. Four of them appear in both the sequences. Fig. 9 shows a snapshot of the experiments. We put three Kinects so that they cover the environment. Table I shows a summary of the dataset. The durations of the sequences are 162 and 218 [s]. The subjects often moved out from the camera's field of view, and the system lost track of people 49 and 58 times in the sequence 1 and 2, respectively.

The proposed method is applied to the sequences. For comparison, we also applied the proposed method with

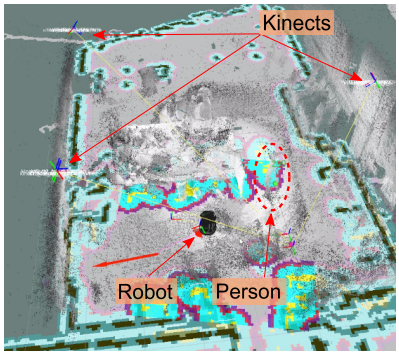


Fig. 8. The integration of the system and a mobile robot. We can see the robot, the point clouds obtained from the Kinects, and the tracked person in the same view.

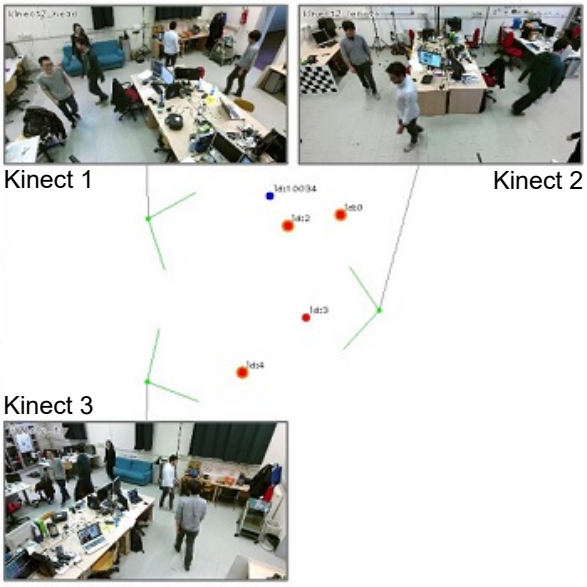


Fig. 9. A snapshot of the experiments. Three Kinects are placed so that they cover the environment. The dots indicate the tracked people.

TABLE I
Long-term re-identification dataset summary

	duration [s]	# of subjects	# of lost
Seq. 1	162	6	49
Seq. 2	218	6	58

thresholding instead of the Bayesian inference and a traditional face recognition method with a landmark detection and SIFT features [25]. In this experiment, the face detection and the feature extraction took about 15 msec per frame on each distributed PC, and the re-identification took about 10 msec on the master PC.

Table II shows a summary of the re-identification results. *success* and *failure* in Table II mean the number of tracks successfully and wrongly re-identified. Since *OpenFace* discards a face image when it couldn't detect the landmarks of the face, it yields fewer face features than [25], and it took much time to re-identify a person from when he/she appears than the SIFT-based method. However, the recognition accuracy of the proposed methods significantly outperforms [25]. While the limitation of the face view faces make the SIFT-based method fail to identify people, the proposed methods achieve the high identification accuracies thanks to the robustness of the deep neural network-based face representation to the face view change.

With the simple thresholding scheme, the system has to make a decision when it observes a certain number of face images. This affects the re-identification accuracy when several registered faces show similar distances to an observed face image. On the other hand, with the proposed Bayesian inference, the system can wait to make a decision until a significant difference of the faces is observed. In addition to that, once a significant difference is observed, the system can immediately classify the observed face. As a result, the proposed method with the Bayesian inference could improve both the recognition accuracy and the average re-identification time. Fig. 10 shows examples of the long-term re-identification results (i.e., re-identification between the sequence 1 and 2). The system successfully re-identified most of the subjects even if they changed their clothes (and hair styles). However, it wrongly identified a newly appeared person as an existing person due to their similar face appearances (beards and brows). It was the only failure case in this experiment. *recover rate* in Table II shows the rate of the successfully re-identified tracks among all the tracks. It shows a limitation of the face recognition-based re-identification: while it shows a good re-identification accuracy, it cannot re-identify a person when his/her face is not visible. One possible way to address this problem is combining the face recognition-based re-identification with other re-identification methods, such as appearance-based and soft biometric-based methods. It allows taking advantage of the high recognition accuracy of the face recognition and visibility independence of such methods.

TABLE II

EVALUATION OF RE-IDENTIFICATION ACCURACY

	success	failure	re-id time	recover rate
[25]	34	54	2.12	0.318
ours w/o Bayesian	51	7	4.59	0.477
ours w/ Bayesian	60	1	4.24	0.561

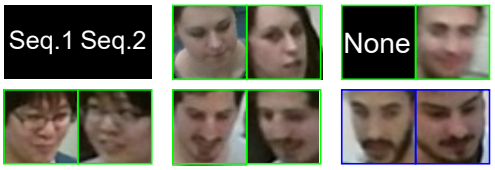


Fig. 10. Examples of the long-term re-identification results. The green boxes indicate that the person is correctly re-identified. The blue boxes indicate that the right side face is wrongly identified as the left side face.

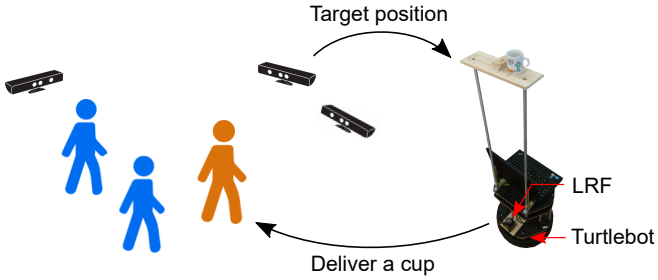


Fig. 11. The cup delivery experiment. The system tells the target person position to the robot, and the robot delivers a cup to him.

B. Experiment with a Mobile Robot

To show the possibility of the integration of the proposed system and a mobile robot, we conducted an experiment with a daily service robot scenario. The task of the robot is to deliver a cup to a specific target person. A pair of the target person's name and a face image is given to the system in advance. The system identifies the target person among the others and tells his/her position to the robot. Then, the robot moves toward the target person's positions to deliver the cup to him/her (see Fig. 11).

Fig. 12 shows the experimental setting. The number of the subjects is three, and each subject stands in front of a Kinect and stares at it. We give a list of people names and face images of the subjects and other three people. Thus, the number of the predefined people is six. We conducted the experiment three times while changing the target.

Fig. 13 shows a snapshot of the experiment. While the target person stared at the Kinect, the system successfully identified him as the target person (Fig. 13 (a)). Then, the system told his position and name to the robot, and the robot moved toward him (Fig. 13 (b)). After arriving at his position, the robot called his name and told him to take the cup (Fig. 13 (b)), and the target person took the cup on the robot (Fig. 13 (d)).

In all the experiments, the system successfully identified the target person, and the robot delivered the cup to the target person. The experimental result shows that the proposed system can be integrated with a mobile robot system, and it allows to have a rich human-system interaction.

VI. CONCLUSIONS AND FUTURE WORK

This paper has described a face recognition-based people tracking and re-identification system for RGB-D camera networks. The proposed system utilizes *OpenPTrack* and *OpenFace* to track and recognize their faces. We proposed

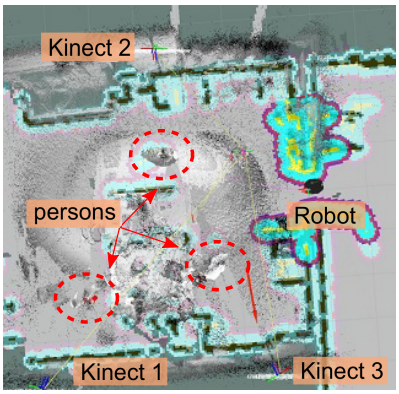
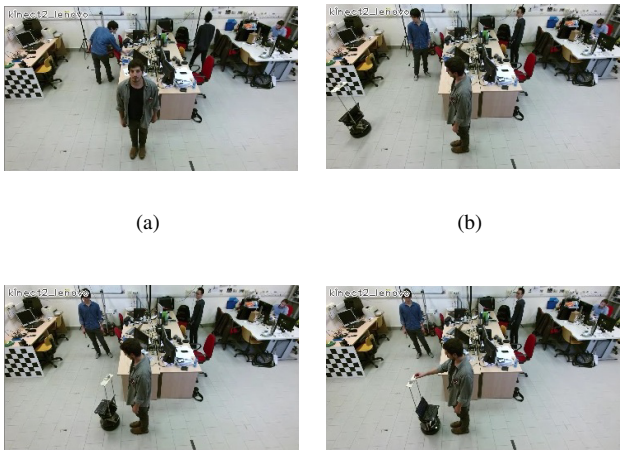


Fig. 12. The setting of the experiment with a mobile robot. A person stands in front of each Kinect and stare at it. The system identifies a target person and tells his position to the robot, and then the robot moves toward him to deliver a cup.



(a) (b)

(c) (d)

Fig. 13. A snapshot of the cup delivery experiment. The target person is identified by the camera network, and then his position is told to the robot. The robot moves toward the target person, and after arriving, the robot calls his name and tells him to take the cup on the robot.

a Bayesian inference-based face classification method for reliable re-identification. We evaluated the re-identification performance of the proposed system and conducted an experiment to show the possibility of the integration of the proposed system with robotic systems.

Face visibility is still a hard problem. The system cannot identify a person if the person hides his/her face intentionally. To deal with such situations, we are planning to combine the face features with features which do not suffer from the visibility problem, such as appearance and skeletal features.

Acknowledgement

This research was partially supported by Omitech srl under the O-robot grant and the Leading Graduate School Program R03 of MEXT.

References

[1] R. Mazzon, S. F. Tahir, and A. Cavallaro, "Person re-identification in crowd," *Pattern Recognition Letters*, vol. 33, no. 14, pp. 1828–1837, oct 2012.

[2] A. Bedagkar-Gala and S. K. Shah, "A survey of approaches and trends in person re-identification," *Image and Vision Computing*, vol. 32, no. 4, pp. 270–286, apr 2014.

[3] L. Nanni, M. Munaro, S. Ghidoni, E. Menegatti, and S. Brahnam, "Ensemble of different approaches for a reliable person re-identification system," *Applied Computing and Informatics*, vol. 12, no. 2, pp. 142–153, jul 2016.

[4] A. Bedagkar-Gala and S. K. Shah, "Gait-assisted person re-identification in wide area surveillance," *Computer Vision - ACCV 2014 Workshops*, pp. 633–649, 2015.

[5] K. Koide and J. Miura, "Identification of a specific person using color, height, and gait features for a person following robot," *Robotics and Autonomous Systems*, vol. 84, pp. 76–87, oct 2016.

[6] M. Munaro, A. Fossati, A. Basso, E. Menegatti, and L. V. Gool, "One-shot person re-identification with a consumer depth camera," in *Person Re-Identification*. Springer, 2014, pp. 161–181.

[7] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, dec 2003.

[8] A. Bedagkar-Gala and S. K. Shah, "Part-based spatio-temporal model for multi-person re-identification," *Pattern Recognition Letters*, vol. 33, no. 14, pp. 1908–1915, oct 2012.

[9] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 1992, pp. 71–86.

[10] G. Zhang, X. Huang, S. Z. Li, Y. Wang, and X. Wu, "Boosting local binary pattern (LBP)-based face recognition," in *Advances in Biometric Person Authentication*. Springer, 2004, pp. 179–186.

[11] C. Geng and X. Jiang, "Face recognition using SIFT features," in *International Conference on Image Processing*. IEEE, 2009, pp. 3277–3280.

[12] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 Classes," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 1891–1898.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 815–823.

[14] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*. BMVA, 2015, pp. 41.1–41.12.

[15] M. Munaro, F. Basso, and E. Menegatti, "OpenPTrack: Open source multi-camera calibration and people tracking for RGB-d camera networks," *Robotics and Autonomous Systems*, vol. 75, pp. 525–538, jan 2016.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. IEEE, 2009, p. 5.

[17] "Openptrack — towards collective computing." [Online]. Available: http://openptrack.org/

[18] A. Brandon, B. Ludwiczuk, and S. Mahadev, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[19] M. Munaro and E. Menegatti, "Fast RGB-d people tracking for service robots," *Autonomous Robots*, vol. 37, no. 3, pp. 227–242, 2014.

[20] M. Carraro, M. Antonello, L. Tonin, and E. Menegatti, "An open source robotic platform for ambient assisted living," in *Italian Workshop on Artificial Intelligence and Robotics*, 2015, pp. 3–18.

[21] "Dlib c++ library." [Online]. Available: http://dlib.net/

[22] G. B. H. E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," University of Massachusetts, Amherst, Tech. Rep. UM-CS-2014-003, May 2014.

[23] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda, "Softassign and EM-ICP on GPU," in *International Conference on Networking and Computing*. IEEE, nov 2010.

[24] "ROS navigation stack." [Online]. Available: http://wiki.ros.org/navigation

[25] G. Pitteri, M. Munaro, and E. Menegatti, "Depth-based frontal view generation for pose invariant face recognition with consumer RGB-D sensors," in *Intelligent Autonomous Systems Conference*. Springer, 2016.

Multimodal Visual-Inertial Odometry for navigation in cold and low contrast environment

Axel Beauvisage and Nabil Aouf
Centre for Electronic Warfare Information and Cyber, Defence Academy of the United Kingdom
Cranfield University, Shrivenham, SN6 8LA United Kingdom
Email: a.beauvisage@cranfield.ac.uk

***Abstract*— Multispectral setups have a great potential to tackle problems such as collision avoidance or pedestrian detection. However, multispectral odometry is nowadays less precise than most stereo visible setups, so there is a need to improve such systems. With this work, we investigate the fusion of inertial data with visual information to improve the performance of multispectral setups. Inertial data are included in the process model of an extended Kalman filter to estimate the pose of a vehicle. But inertial measurement units drift rapidly when integrated for a certain period of time. Visual odometry is used to correct the predicted pose and reduce this drift. IMU data are also used to provide a pose estimation when images are too noisy to compute a motion (e.g. motion blur or low contrast). Therefore, we present a new multispectral (visible/LWIR) navigation system able to cope with fast motions and to operate in cold environments, where the contrast of infrared images is reduced. We demonstrate the robustness of the setup on a series of semi-urban datasets acquired from a car. An average error inferior to 3% of the distance traveled between the estimated trajectory and the GPS track is achieved on all the datasets.**

I. Introduction

Visual Odometry (VO) and Simultaneous Localization And Mapping (SLAM) have become an important part of autonomous navigation research because they provide a convenient and reliable alternative to classic robot/vehicle localisation solutions. Indeed, compared to other localisation sensors, cameras are cheaper and can easily be placed on a huge variety of transportation systems from ground vehicles to unmanned air vehicles (UAVs). More accurate than inertial measurement units (IMUs), VO also possesses several advantages over GPS navigation. Indeed, the 3D pose of the camera is estimated for each picture taken which means that in addition to its position, the orientation of the vehicle can also be retrieved. Moreover, VO is not prone to jamming and can be run indoor.

Visual Odometry [1] consists of estimating the pose of one [2] or several [3]–[5] cameras in motion. Unless SLAM approaches [6] which are computing and updating a map of the environment, VO techniques are estimating the vehicle pose relative to its original pose. Stereo setups are usually more precise than monocular systems as they are able to compute the exact 3D position of each feature from the disparity between images. With a single camera on the other hand, the depth of features is not known and the trajectory obtained is only estimated up to a scale. Concerning stereo systems, two main approaches can be distinguished, dense and feature-based estimations. The former is maximizing the photo-consistency in consecutive images [7], whereas the latter is matching relevant features of the scene (points [3], [8] or lines [9]).

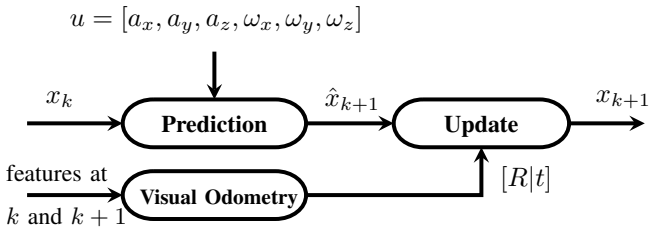


Fig. 1: flowchart of the EKF

VO is one of the primary tasks of any driving assistance or autonomous system and is usually used by other navigation-related applications such as obstacle avoidance or decision making. In that sense, multispectral setups start to get investigated as they provide more information about the scene [10], [11]. However, such systems suffer from the lack of similarity between images. New techniques have been developed to match images coming from different part of the electromagnetic spectrum [2]–[4] but multispectral VO systems are currently less precise than other standard visible setups such as [5]. Thermal imaging on the other hand, is a good solution to tackle one of the main problem affecting visible cameras: dark illumination conditions. And it is already used for some specific applications such as pedestrian detection [11]–[13]. Its performance is yet diminished when the temperature of the scene is homogeneous or during fast motion (thermal images are usually blurrier and more prone to motion blur than visible images due to the bigger wavelength of infrared light).

The work presented in this paper therefore aims at improving the accuracy of long wave infrared (LWIR)/visible stereo setups by using additional information from an Inertial Measurement Unit (IMU). However, this information is not always reliable and should be used carefully. Acceleration for example has to be integrated twice to obtain the distance traveled and can drift very quickly (just a few seconds) which makes IMU devices not suitable for localization purposes when used independently. The orientation on the other hand, is computed from the angular velocity (integrated only once)



Fig. 2: The top row represents an average stereo pair. Bottom left picture: thermal image subject to motion blur. Bottom right picture: thermal image in low contrast environment (low temperature).

and can be estimated more precisely when used in a filter framework such as a Kalman Filter [14]. Therefore, the uncertainty of each sensor has to be estimated properly. IMU data have already been combined with visual information to create visual-inertial setups (mono [15] or stereo [16]) but it has never been done with multispectral images. To fuse information from different sensors, an Extended Kalman Filter (EKF) is used with a similar approach to [17]. As shown on Fig. 1, the accelerations and angular velocities obtained from the IMU are included in the input vector of the process model to predict the state of the system at step $k+1$. VO is then performed independently and estimates the transformation matrix $T_{k,k+1}$ between two consecutive camera poses. This transformation, made of a rotation R and a translation t, is then fed to the EKF as observation to correct the estimated state.

The main objective of this work is to evaluate the limits of our LWIR/visible stereo setup by testing it in tough conditions, when the temperature is approaching 0°C and during fast motions. As a consequence, the quality of thermal images decreases drastically and VO becomes remarkably noisy which is critical for relative navigation where the current pose depends on the previously estimated poses. An error at any moment will be accumulated. Example of noisy thermal images can be seen in Fig.2.

The innovation of this work is coming from the combination of two noisy motion estimation techniques, namely multispectral VO and IMU to create a new, more robust multispectral setup able to reduce the drift from IMU data by correcting the measurements with visual information and to recover the pose when the VO is too noisy to compute a motion.

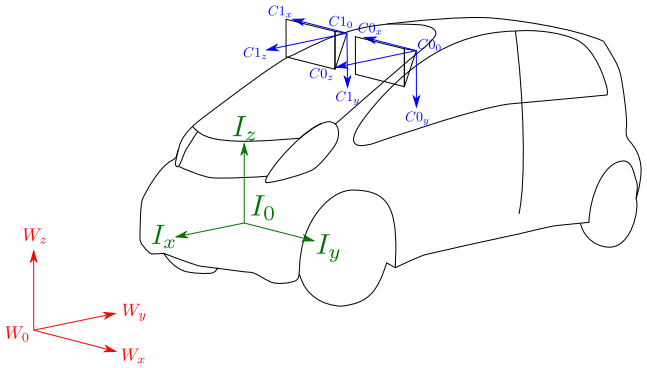


Fig. 3: representation of the different coordinate frames

II. Kinematics of the vehicle

A. coordinate system and notation

In this section the different coordinate systems and notation used in the paper are defined and illustrated in Fig. 3. The vehicle is evolving in a world coordinate system $W(W_x, W_y, W_z)$. The origin is chosen to match the inertial frame of the vehicle I_0 at the start of the acquisition. $I_k(I_x, I_y, I_z)$ represents the inertial frame linked to the vehicle at step k. Its origin is located at the centre of rotation of the vehicle and the IMU should ideally be placed as close as possible to this point. Finally, each camera possesses its own coordinate frame $C0_k(C0_x, C0_y, C0_z)$ and $C1_k(C1_x, C1_y, C1_z)$ at step k. $C0$ corresponds to the left camera and $C1$ to the right one. A value expressed in the world, inertial or camera frame will be represented respectively with the indices W, I or C_i.

B. equations of motion

The inertial sensors studied in this paper are the 3-axis accelerometer which provides an acceleration (a_x, a_y, a_z) along each axis and the 3-axis gyroscope which provides an angular velocity $(\omega_x, \omega_y, \omega_z)$ along each axis. A strapdown technique is employed to track the orientation and position of the vehicle from the IMU. First, the orientation is defined over time by the equation:

$$\theta_W(t+dt) = \int_t^{t+dt} \omega_W(t+dt)\ dt + \theta_W(t) \qquad (1)$$

In addition to the forces generated by the movements of the car, accelerometers are also measuring the gravitational force created by the earth. This force, represented by the vector $g_W = (0, 0, 9.81)$ has to be subtracted to the acceleration values expressed in the world coordinate system. The formula to track the velocity over time is then:

$$v_W(t+dt) = \int_t^{t+dt} (a_W(t+dt) - g_W)\ dt + v_W(t) \qquad (2)$$

Finally, position can be derived from velocity:

$$p_W(t+dt) = \int_t^{t+dt} v_W(t+dt)\ dt + p_W(t) \qquad (3)$$

It can be noticed that the acceleration is expressed in the world coordinate frame but the values obtained from

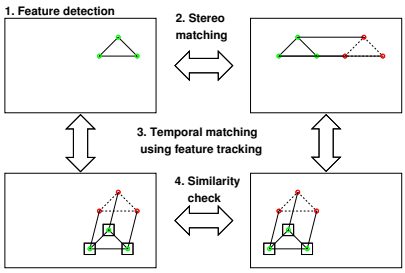


Fig. 4: Matching process between two consecutive pairs

the sensor are expressed in the inertial frame. They have thus to be transposed to the world coordinate frame before being used. Moreover, time is continuous but the filter is updated only when a new pair of image is available so the above equations need to be converted to discrete steps. As the frequency of acquisition of the IMU data (120Hz) is much higher than the acquisition of the images (10 Hz), the acceleration a_{I_k}, measured at step k, represents the average of all acceleration values over the interval $d\tau$ between two consecutive frames. This way, the value used is more representative of the acceleration over the interval of time than just a single measurement. The discrete equations of motion are defined as:

$$\theta_W(k+1) = \begin{bmatrix} 1 & sin(\omega_x)tan(\omega_y) & cos(\omega_x)tan(\omega_y) \\ 0 & cos(\omega_x) & -sin(\omega_x) \\ 0 & \frac{sin(\omega_x)}{cos(\omega_y)} & \frac{cos(\omega_x)}{cos(\omega_y)} \end{bmatrix} d\tau + \theta_W(k) \tag{4}$$

$$v_W(k+1) = R_{W,I}^{-1}(a_{I_{k+1}} - g_W)\ d\tau + v_W(k) \tag{5}$$

$$p_W(k+1) = v_W(k+1)\ d\tau + p_W(k) \tag{6}$$

Where $R_{W,I}^{-1}$ is the rotation matrix representing the orientation $\theta_W(k)$.

C. State representation

To estimate the pose of the vehicle at a certain moment in time, its position, velocity and orientation are encapsulated in a state vector:

$$x_k = [X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}, \phi, \theta, \psi]_W \tag{7}$$

The state vector is expressed in the world coordinate system. The orientation is parameterized by the Euler angles along the 3 axis W_x, W_y, W_z, respectively roll, pitch and yaw.

III. Visual Odometry

To compute the motion of the vehicle from multispectral (LWIR/visible) images the algorithm previously developed in [3] has been utilized and improved. This technique consists in finding the same features in two consecutive multispectral stereo pairs, and then using a bundle adjustment approach, to estimate the transformation between the two camera poses. Two types of matching are required to get the same feature in the 4 images: stereo matching and temporal matching (see Fig. 4). The multispectral stereo matching is the most challenging because of the difference in modalities. It is performed by looking for a stereo corresponding pixel on the epipolar line (same row when images are stereo rectified). Each candidate is tested and a similarity score is computed based on phase congruency and mutual information. It is a winner takes all approach which means that the candidate with the highest score is selected as the match. Phase Congruency is an image frequency analysis technique developed by Peter Kovesi [18] and is used here as an edge detector for the stereo matching part and as a corner detector during the feature detection process. Mutual information [19] on the other side, is a statistical approach derived from information theory and which evaluates the similarity between images regardless of their pixel intensities. The combination of the two techniques produces a robust cross-modality matching technique [3]. Temporal matching is performed using the pyramidal KLT algorithm [20]. The method, based on optical flow, is widely used for visual odometry and robust when applied to images of the same modality. That is why it has been chosen to perform the temporal matching, when images are coming from the same camera. Once all features have been matched, motion is estimated using a window bundle adjustment framework. The motion is parameterized by a rotation and a translation (6 DoF in total):

$$X = [x, y, z, \phi, \theta, \psi]_{C_0} \tag{8}$$

The best X vector that minimizes the reprojection errors is then computed. The minimization problem can be expressed as follows:

$$\min_X \sum_{i=0}^{N} \|f(p_i, X) - z_i\|^2 \tag{9}$$

Where N represents the total number of features, p_i is the position of the ith feature in the previous stereo pair and z_i is the position of the observed feature in the next pair. The function f is defined as:

$$f(p_i, X) = h^{-1}(R\ h(p_i) + t) \tag{10}$$

Where h corresponds to the function that project a stereo feature into a 3D point and R and t are respectively the rotation matrix and translation vector corresponding to the vector X. To optimize the function f over X, the Levenberg-Maquart method has been preferred over Gauss-Newton as it more robust to bad initialization. To initialize X with values close to the real solution and thus to converge quickly, the transformation estimated during the prediction step of the Kalman filter is used as an initial guess. The quality and speed of the algorithm is also boosted by keeping the remaining matches after the RANSAC outlier rejection scheme onto the next iteration. Indeed, those are reliable matches and so they are used to bring more inliers to the RANSAC process and help increasing its performance. Moreover, those inliers contain already matched features so there is no need to perform the stereo matching again (which is time consuming compared to the temporal matching).

Also, any new feature detected in a 3px radius from one of those inliers is discarded as it probably describes the same physical element and has a risk of being wrongly matched. Finally, when no motion can be computed because of a lack of features or because the optimization failed, the motion provided by the IMU is selected instead of assuming that the vehicle did not move. This way, the VO will not affect the filter during the correction step if it fails. However, this solution only works for short disruptions. The algorithm can only rely on the IMU information for a short period of time (few seconds) or it will start drifting.

IV. Extended Kalman Filter

The Kalman filter is used to model the uncertainties generated by each sensor and to fuse the information obtained from the IMU and the VO. The non-linearities involved in the motion of the vehicle and the difference of coordinate systems (IMU data are measured in the inertial frame whereas the state vector is expressed in the world frame) makes the standard Kalman Filter inappropriate for this system as it breaks the linearity assumption. Instead, an extended Kalman filer is used where the process model is linearized.

A. Process model

As shown in Fig. 1, the process model is including the IMU data as input to predict the state at the next iteration. The input vector contains the acceleration and angular velocity information:

$$u = [a_x, a_y, a_z, \dot{\phi}, \dot{\theta}, \dot{\psi}]_I \tag{11}$$

This vector represents the evolution of the system from step k to $k+1$ according to the IMU data. Knowing the state of the system x_k at step k, the future state can be estimated by a function depending on u and x_k:

$$\hat{x}_{k+1} = f(x_k, u) + w_{k+1} \tag{12}$$

The process model is assumed to be perturbed by a white gaussian noise w_k with zero mean. From the equations of motion derived in section II-B we obtain:

$$\begin{bmatrix} X \\ Y \\ Z \\ \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \phi \\ \theta \\ \psi \end{bmatrix}_{W,k+1} = \begin{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}_{W,k+1} \\ R_k^{-1} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_I - \begin{bmatrix} 0 \\ 0 \\ 9.81 \end{bmatrix}_W \\ \begin{bmatrix} 1 & sin\phi \cdot tan\theta & cos\phi \cdot tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & \frac{sin\phi}{cos\theta} & \frac{cos\phi}{cos\theta} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_I \end{bmatrix} \delta\tau + \begin{bmatrix} X \\ Y \\ Z \\ \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \phi \\ \theta \\ \psi \end{bmatrix}_{W,k} \tag{13}$$

Where R_k is the rotation matrix representing the attitude of the vehicle based on x_k. Because f is not linear, the predicted covariance matrix $\hat{P}_{k+1}$ is computed using F_k, the jacobian of $f(x_k, u)$:

$$\hat{P}_{k+1} = F_k P_k F_k^T + Q_k \tag{14}$$

Where P_k is the covariance matrix of the system and Q_k is the noise covariance matrix.

TABLE I: camera characteristics

Model	mvBlueFOX-ICG	Gobi-640
Resolution	752x480 px	640x480 px
Max. frame rate	93 Hz	50 Hz
Spectral band	0.4-0.7 μm	8-14 μm



Fig. 5: experimental setup. Cameras can be seen on each part of the IMU (orange box)

B. Observation model

As mentioned earlier, VO is providing a measurement of the pose of the vehicle that is used by the filter to correct the estimated state. But it is only computing the motion between two consecutive frames, so it has to be concatenated to the previous motions in order to obtain a guess of the state. To do so, the measured position and orientation are derived from the VO output and the current state of the system as it represents the pose that results from all the previous motions. According to section II-B, the estimated state is only reliable if the current position and velocity are estimated properly because it depends on the current state of the system. The velocity estimation is more important than the position estimation as it is also used to compute the new position of the vehicle. That is why, even though the VO algorithm is providing a distance between two consecutive frames, it is the velocity that is taken into account in the observation model, to offer a more accurate velocity estimation, and therefore a better integration of the IMU data at the next iteration. Thus, the observation model is the following:

$$Z_k = \begin{bmatrix} \dot{X}_{vo} \\ \dot{Y}_{vo} \\ \dot{Z}_{vo} \\ \phi_{vo} \\ \theta_{vo} \\ \psi_{vo} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{X}_k \tag{15}$$

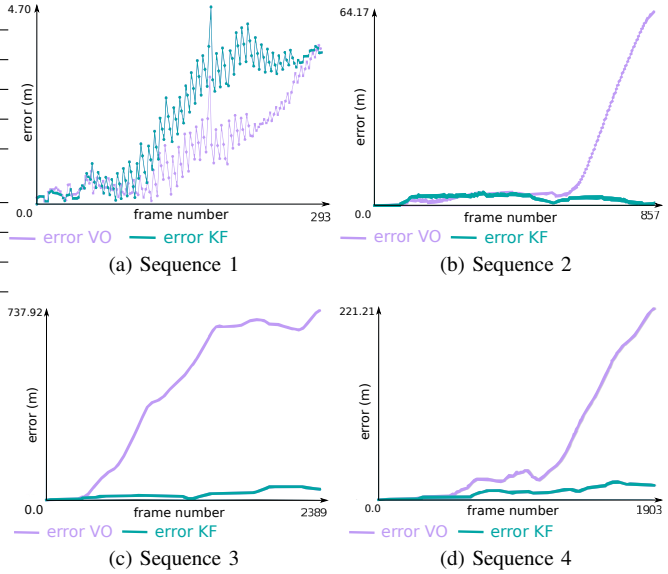
V. Experiments

A. Stereo setup

The algorithm presented in this work has been tested on different datasets. The images have been acquired with a multispectral stereo setup made of a BlueFOX-ICG (Matrix Vision) and a Gobi-640 (Xenics). More information Concerning the characteristics of the cameras can be found in TABLE I. A wide baseline was used to separate the two sensors in order to get a good precision during the projection of features into 3D points and to be able to compute the depth

TABLE II: Errors obtained for all sequences.

Datasets	Seq. 1	Seq. 2	Seq. 3	Seq. 4
Distance	97m	122m	879m	330m
Error on final points (VO)	3.71m	64.17m	737.93m	221.21m
Error on final points (EKF)	3.62m	0.89m	41.97m	16.61m
Error in % of the traveled distance (EKF)	3.73%	0.72%	4.77%	5.04%
Mean error (VO)	1.24m	12.20m	419.08m	58.4m
Mean error (EKF)	2.02m	2.3m	22.64m	8.85m
Mean error in % (EKF)	2.08%	1.88%	2.58%	2.68%



(a) Sequence 1 (b) Sequence 2

(c) Sequence 3 (d) Sequence 4

Fig. 6: Error between the estimated position and the GPS coordinate. The purple curve represents the error obtained with VO alone, the green curve represents the error using the EKF.

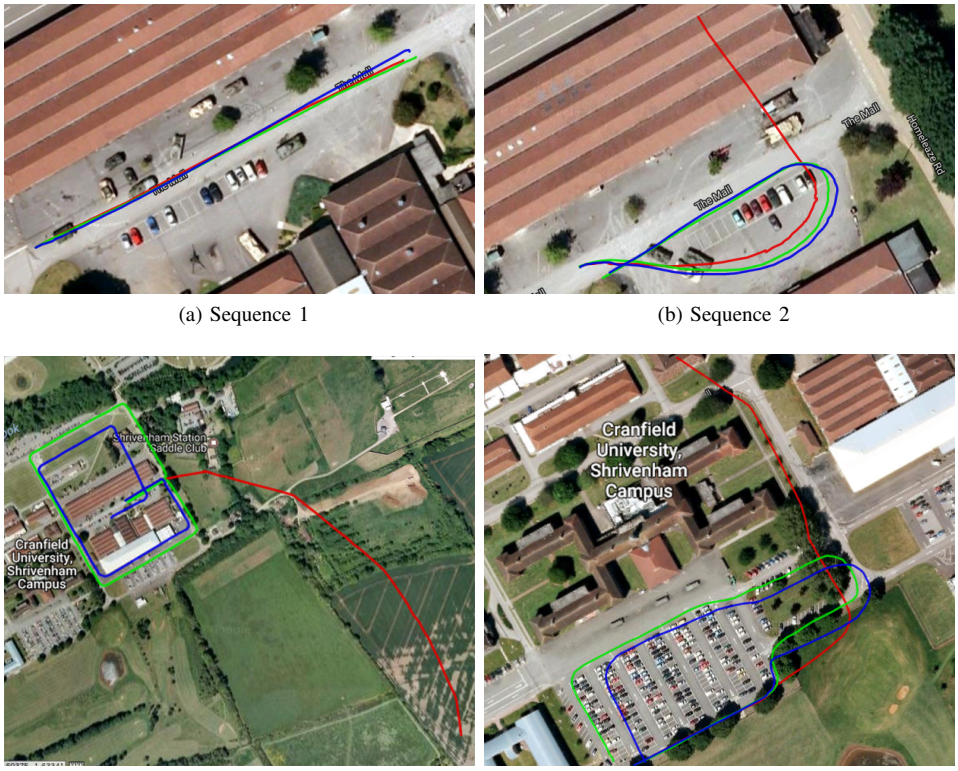
of far features (up to 400m). The calibration of the stereo setup was performed with a chessboard made of aluminium and cardboard, two materials with different properties, so it can be seen in both modalities. Intrinsic and extrinsic parameters were estimated using the AMCC toolbox [21]. IMU data acquisition was performed using an MTi-G, GNSS/INS device manufactured by Xsens. The MTi-G possesses MEMS solid state inertial sensors and a GPS receiver. It also contains a navigation system able to track the position and orientation but only raw data were used here. Accelerometers have a range of $\pm 50\ m/s^2$ (5g) and a $0.02\ m/s^2$ bias stability. Gyroscopes run at a rate of $\pm 300\ deg/s$ and have a $1\ deg/s$ bias stability. GPS coordinates were recorded to be compared with the trajectories estimated by our algorithm. The setup was mounted on a car that was driven on public roads to simulate real navigation conditions. The environment where the experiments took place was mainly semi-urban. Some of the tests were also carried out on a car park. The images were acquired with a cloudy weather and cold temperature (approximately 0°C). Different trajectories have been tested, including straight lines and loops. 90° and 180° turns have been taken intentionally to test the robustness and limit of the proposed solution during fast motion, when the whole image is moving from one frame to the other. Longer trajectories than [3] have been experimented.

B. Results

In total, 4 trajectories with different shapes and and lengths (from 100m up to 900m) are presented. They can be visualized in Fig. 7. TABLE II shows the different errors obtained for each trajectory. VO alone (purple trajectories) offers good performance on straight lines as seen on the first sequence. A small drift can be perceived but it does not exceed a few meters. It can however be noticed on the other datasets that the VO trajectory start drifting considerably after each important turn. Indeed, most of the corners are not estimated properly because one or more frames could not be used to estimate a proper motion, or if it is the case, the poor quality of the matching is altering the trajectory computation, which results in a wrong orientation estimation after the turn. This can be explained by the fast motion involved by the turn. As illustrated in Fig. 2, thermal images are easily affected by motion blur, which reduces considerably the number of features matched and thus the performance of the algorithm. The failures can be easily visualized on the error graphs (see Fig. 6) as they produce a peak. The Kalman filter approach, used to fuse IMU data, can compensate for that drift by providing a motion when VO fails or by correcting it when images are noisy (motion blur, low contrast, etc...). TABLE II summarized the overall results. Errors have been computed by taking the euclidean norm between a state or a camera pose and the GPS coordinate of the vehicle when the picture was taken. It shows that the EKF approach reduces considerably the final and mean errors for sequence 2,3 and 4. It can however be noticed that more error is produced on Sequence 1 (straight line). Indeed, an average error of 2.02m is obtained with the EKF whereas 1.24m is acheived with the visual information alone. This shows that the use of IMU data produces some drift than can affect the final trajectory estimation but the difference between the two techniques is minimal ($<$ 1m) and the EKF still improves greatly the motion estimation when trajectories become more complex.

VI. Conclusion and Future Work

Globally, the results obtained are satisfying as the mean error for each dataset does not exceed 3% of the traveled distance. Combining IMU data with visual information helped reducing the drift generated by VO when the image quality is deteriorated due to low contrast in temperature or fast motions. It is also useful for failure recovery, when no motion can be estimated from the cameras for a short period of time. In this paper, we have therefore introduced a robust multispectral navigation system able to work in tough conditions. The work presented here was more focused on the visual-inertial fusion to create a reliable setup. We intend to improve its accuracy in future works by performing a better matching process with the help of the IMU. Having a first



(a) Sequence 1 (b) Sequence 2

(c) Sequence 3 (d) Sequence 4

Fig. 7: Experimental datasets. The GPS tracks, VO trajectories and EKF trajectories are represented respectively by green, red and blue curves.

motion computed from inertial data could help predicting the location of features on the next stereo pair and thus facilitate the matching process.

REFERENCES

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2004.*, 2004, pp. 652–659.

[2] J. Poujol, C. A. Aguilera, E. Danos *et al.*, "A Visible-Thermal Fusion Based Monocular Visual Odometry," in *Proc. 2nd Iberian Rootics Conference, ROBOT 2015*, 2016, pp. 517–528.

[3] A. Beauvisage, N. Aouf, and H. Courtois, "Multispectral Visual Odometry for Unmanned Air Vehicles," in *Proc. IEEE Int. Conference on Systems Man and Cybernetics, SMC 2016*, 2016, pp. 1–7.

[4] T. Mouats, N. Aouf, A. D. Sappa, C. Aguilera, and R. Toledo, "Multi-Spectral Stereo Odometry," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1–15, 2015.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–61.

[6] B. Williams and I. Reid, "On combining visual SLAM and visual odometry," in *Proc. IEEE International Conference on Robotics and Automation*, 2010.

[7] K. S. Shankar and N. Michael, "Robust direct visual odometry using mutual information," in *Proc. IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR*, oct 2016, pp. 9–14.

[8] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, jun 2011, pp. 963–968.

[9] J. Witt and U. Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in *Proc. IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, nov 2013, pp. 4164–4171.

[10] S. Krotosky and M. Trivedi, "Multimodal stereo image registration for pedestrian detection," in *Proc. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2006, pp. 109–114.

[11] S. Hwang, J. Park, N. Kim *et al.*, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*. IEEE, jun 2015, pp. 1037–1045.

[12] P. Borges and S. Vidas, "Practical Infrared Visual Odometry," *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 8, pp. 1–9, 2016.

[13] S. J. Krotosky and M. M. Trivedi, "Mutual information based registration of multimodal stereo videos for person tracking," *Computer Vision and Image Understanding*, vol. 106, no. 2-3, pp. 270–287, 2007.

[14] JoanSolà, "Quaternion kinematics for the error-state KF," 2016.

[15] M. Bloesch, S. Omari, M. Hutter *et al.*, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 298–315.

[16] Y. Song, S. Nuske, and S. Scherer, "A Multi-Sensor Fusion MAV State Estimation from Long-Range Stereo, IMU, GPS and Barometric Sensors," *Sensors*, vol. 17, no. 1, 2016.

[17] A. Nemra, L. M. Bergasa, E. López *et al.*, "Robust Visual Simultaneous Localization and Mapping for MAV Using Smooth Variable Structure Filter," in *Proc. Int. Symposium on Robotics Research, ROBOT 2015*, 2015, pp. 557–569.

[18] P. Kovesi, "Phase congruency : A low-level image invariant," *Psychological Research*, vol. 64, no. 2, pp. 136–148, 2000.

[19] P. Viola and W. M. Wells, "Alignment by Maximization of Mutual Information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[20] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[21] M. Warren, D. McKinnon, and B. Upcroft, "Online Calibration of Stereo Rigs for Long-Term Autonomy," in *Proc. Int. Conference on Robotics and Automation, ICRA 2013*, Karlsruhe, 2013, pp. 3692–98.

City-scale continuous visual localization

Manuel Lopez-Antequera[1,2], Nicolai Petkov[1] and Javier Gonzalez-Jimenez[2]

***Abstract*— Visual or image-based self-localization refers to the recovery of a camera's position and orientation in the world based on the images it records. In this paper, we deal with the problem of self-localization using a sequence of images. This application is of interest in settings where GPS-based systems are unavailable or imprecise, such as indoors or in dense cities.**

Unlike typical approaches, we do not restrict the problem to that of sequence-to-sequence or sequence-to-graph localization. Instead, the image sequences are localized in an image database consisting on images taken at known locations, but with no explicit ordering. We build upon the Gaussian Process Particle Filter framework, proposing two improvements that enable localization when using databases covering large areas: 1) an approximation to Gaussian Process regression is applied, allowing execution on large databases. 2) we introduce appearance-based particle sampling as a way to combat particle deprivation and bad initialization of the particle filter. Extensive experimental validation is performed using two new datasets which are made available as part of this publication.

I. Introduction

Performing self-localization with a single camera is of great interest in applications where GPS is unavailable or imprecise, as is the case in urban environments or indoor settings. Since it is a thriving research topic, many advances have been made recently [1], however, there are still limitations when dealing with:

- **Unconstrained topology of the database**: In order to develop systems that work online, the localization problem is usually posed as sequence-to-sequence or sequence-to-graph matching (especially in the case of appearance-based methods). Localizing efficiently in a database of unordered images is an open topic.
- **Changes in appearance** due to illumination or weather conditions. This leads to difficulties when comparing the input images to those from the database. This is particularly noticeable when using local feature descriptors such as SIFT.

To improve performance on these situations, we propose a method that leverages state-of-the-art convolutional neural network (CNN)-based descriptors to localize an image sequence taken from a monocular camera, using as reference an unordered, GPS-tagged collection of images (such as those readily available through Google Street View). Our proposal builds upon Gaussian process particle filters (GPPFs), in which Gaussian processes (GPs) are used as observation models for particle filters (PFs).

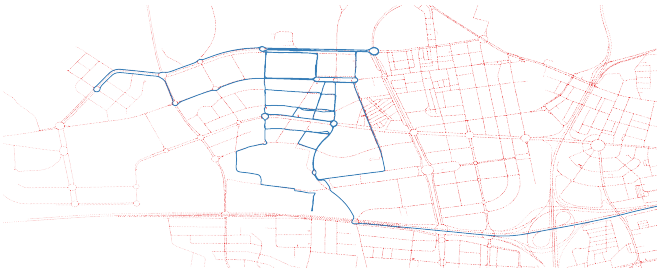


Fig. 1. Our contributions allow GPPFs to localize image sequences (blue, Málaga Urban Dataset [6] on large unordered georeferenced image databases (red, "Málaga Street View 2016" dataset, spanning 8 km^2).

GPPFs were introduced for signal strength-based robot localization in [2] and other modalities in [3], but their practical value for visual egocentric localization was limited at the time, as adequate image processing methods to exploit egocentric images within the framework were not available then. Now, recent advances from the computer vision community can be leveraged to enable egocentric localization through GPPFs. Specifically, we propose to use on whole-image descriptors extracted from convolutional neural networks trained for place recognition [4]. These representations are the state of the art in terms of robustness to illumination, weather, and long-term seasonal changes. An advantage of some of these features [5] is that they are trained so that their representations behave smoothly with respect to pose changes, that is, the distance between descriptors grows with increasing changes in camera pose. This behavior makes the descriptors amenable to interpolation over the pose space, which is desirable when used in a GPPF.

We expand upon previous work [7], in which GPs are used as an observation model for egocentric visual localization in an indoor scenario. Here, we introduce significant improvements to allow localization in large outdoor environments (8 km^2, Fig. 1) at interactive frame rates, while also enabling the system to handle global localization. Due to the small size of the image representations (8 kB per image), the system is scalable and feasible for portable applications. The main contributions of this paper are thus:

- The use of an approximation for GP regression (section III-A), enabling localization using GPPFs on large environments.
- The introduction of an appearance-based particle sampling scheme to enable the filter to initialize from an unknown location with a low number of particles

This work has been supported by the Spanish Government (contract DPI2014-55826-R), and the EU-H2020 project MOVECARE (Grant N. 732158).

[1]MAPIR-UMA group, University of Málaga, Instituto de Investigación Biomédica de Málaga (IBIMA), Spain (mlopezantequera/javiergonzalez)@uma.es

[2]Johann Bernoulli Institute of Mathematics and Computing Science, University of Groningen, The Netherlands n.petkov@rug.nl

(section III-B).
- The collection of two new datasets: an unordered collection of 172.000 Google Street View images which serves as a map, and a collection of 50 sequences gathered from Mapillary[1].

We experimentally demonstrate our contributions in section IV by performing experiments which highlight the nature of these contributions and their effect on the success rate of global localization.

II. Related work

Pose representations

Space is continuous. However, for practical reasons, it is common to simplify appearance-based localization problems ("*where am I?*") by replacing them with classification problems ("*in which place am I?*"). Representing space as a discrete collection of places simplifies the problem: given a measure of image similarity, the most likely location is the one that is most similar to the current input. With this philosophy, FAB-MAP [8] is an approach to solve the place recognition problem by building a probabilistic model on top of a bag-of-words representation of images. Other methods exploit the sequentiality of the recorded images in the database and the live sequence, improving performance. In this line, SeqSLAM and its extensions [9], [10], [11] pose the problem as a sequence to sequence matching procedure, obtaining good results even with drastic appearance changes due to changing seasons. Similar work in [12] introduces efficient binary descriptors that allow direct sequence to sequence matching as a single hamming distance operation. The CAT-SLAM [13] system performs continuous localization: instead of discretizing the world into distinct places, they model the world as a continuous trajectory on which localization is performed. Although the probabilistic estimate of the position is a one-dimensional probability density function, however, localization is restricted to a sequence.

All of the previous methods constrain the problem to that of sequence-to-sequence localization, in which the database is formed by an ordered sequence of images. This restriction becomes problematic when dealing with scenarios where different trajectories are possible such as in a city, where many intersections exist and many routes cover the same locations. Some recent work deals with localization in such scenarios: In [14], the authors achieve localization of a moving camera in a city, however, they achieve this by representing the space as a dense grid, over which a Bayesian filter is applied. Although they achieve good results, representing the probability mass as a categorical distribution sets an upper bound of the size of the map. The authors of [15] achieve localization of a moving camera in a city by modelling the location of the vehicle as a categorical distribution on a graph of the road network. Using a graph representation of the city instead of a grid representation is advantageous, as memory and computation are not wasted on grid cells that represent non-transitable areas.

Image representations

Extracting representations that are useful for place recognition and visual localization is fundamental for any localization system. As many other applications within computer vision, visual localization has been improved dramatically by the use of CNNs, producing image representations that are robust to changes in illumination, weather and even the seasons: starting with [16], where the authors explored the use of internal representations of CNNs trained for object recognition. Later, [17] and [4] trained networks using semi-supervised, tripled-based training schemes to improve place recognition performance. Recently, the authors of [18] push the state of the art in place recognition by collecting a massive database of images from stationary webcams to train a CNN in a fully-supervised manner. Complementary to these advances, the work in [5] also applies CNNs to extract image representations that are tied to camera pose changes by linear transformations.

Gaussian processes for localization

GPs have also been used as an observation model to perform indoor Bayesian localization using WiFi signal strength [2], egocentric omnidirectional images [19] and egocentric monocular video [7]. More specifically, GPs within a PF-based localization (GPPFs) were introduced to the field of robot visual localization in [3], where the pose of a robotic blimp was tracked from an external viewpoint through a fixed camera. We build upon these works and extend the approach to large outdoor environments.

III. Gaussian Process Particle Filters

GPPFs are defined in [3] as PFs which use GPs for both the observation model and the transition model[2]. However, for self-localization of vehicles, it is not necessary to learn the transition model since wheel odometry is more reliable and is commonly available. Moreover, if the input frame rate is high enough, visual odometry (VO) can be used. The error incurred when estimating egomotion through VO is also well understood and does not need to be learned [20], [21].

GPs are a powerful tool to perform regression. It is out of the scope of this paper to introduce them[3], save for a short description: An intuitive view of GP regression is that predictions are calculated as a weighted average of neighboring points, where the weights are assigned according to a kernel function which provides a measure of distance or similarity of the query point to the neighboring training set points. GPs present two key features:

[1]Mapillary offers a crowdsourced collection of videos which are geotagged with poses refined using structure-from-motion techniques

[2]In a PF, an *observation model* predicts the observation for each particle. This prediction is compared to the real observation and determine the likelihood of a particle surviving. A *transition model* moves the particles according to some motion input. In some cases (for example, several degree of freedom actuators), the motion model can be learned from data, to help predict the actual motion from indirect sensing

[3]See [22] for a thorough reference on Gaussian processes

- GPs are non-parametric: instead of learning model parameters, the training data is used for regression.
- GPs output a probabilistic estimate of the uncertainty of the prediction.

As an observation model for a PF, the GP performs probabilistic regression, obtaining an estimate $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ of the image descriptor $\mathbf{y} \in \mathbb{R}^D$ at any pose $\mathbf{p}_i = (x_i, y_i, \theta_i)$ in the plane. To this effect, a kernel function $k(\mathbf{p}_i, \mathbf{p}_j)$ must be defined to yield a measure of similarity. As in [7], we use the following kernel function to combine rotation and translation:

$$k(\mathbf{p}_i, \mathbf{p}_j) = \beta \exp\big(-\alpha_t \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 - \alpha_r \|\mathbf{r}_i - \mathbf{r}_j\|_2^2\big), \quad (1)$$

where $\mathbf{r}_i = (\cos(\theta_i), \sin(\theta_i))$, $\mathbf{x}_i = (x_i, y_i)$ and $\beta, \alpha_t, \alpha_r$ are the kernel parameters[4]. The observation model for the GPPF is the likelihood of the point belonging to the predicted Gaussian distribution. If all of the D dimensions of the descriptor $\mathbf{y}$ are assumed to be i.i.d, with standard deviation σ, we have:

$$p(\mathbf{y} = \mathbf{z}|\mathbf{x}) \propto \exp\big(-\frac{D}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}||\mathbf{z} - \boldsymbol{\mu}||_2^2\big) \quad (2)$$

In simple terms, particles whose predicted appearance is similar to the observation score high as long as there is confidence about the predicted appearance. For this observation model to work properly, the chosen image descriptor must be amenable to interpolation, that is, the values of the elements of the descriptor should behave smoothly with small camera pose changes. Descriptors extracted with CNNs trained to perform place recognition are well suited for this [7].

To perform localization, the GPPF iteratively carries out the following steps. 1: Particles are moved, following some motion input (e.g. wheel odometry). 2: Particles are scored with the observation model (eq. 2). 3: Particles are resampled: Those with higher score have bigger chances of being sampled. We now introduce two improvements to this system to enable online global localization in large environments.

A. Fast GP regression

GP regression becomes intractable when the size of the database n increases, due to their quadratic and cubic increase on compute time and memory use, respectively. In the context of outdoor visual localization in a city where the state can be any pose (x, y, θ), we can expect that a certain density of data points will be required to achieve localization. The value of this density will define an upper bound on the size of the world that the system can work in. Several approaches to reduce the time and memory requirements of GPs are discussed in [22], most of which reduce the complexity by replacing the training set with a different, smaller set of points $m < n$ that is used for inference. We choose the simplest of these, called Subset of Datapoints approximation in [22]. In this approximation, only a subset of the datapoints is used to perform inference.

[4]Although the GP kernel parameters and noise variance can be learned from data, we have empirically picked the following values for all of the experiments: $\alpha_t = 12$, $\alpha_r = 0.025$, $\beta = 0.5$, $\sigma_n^2 = 4$

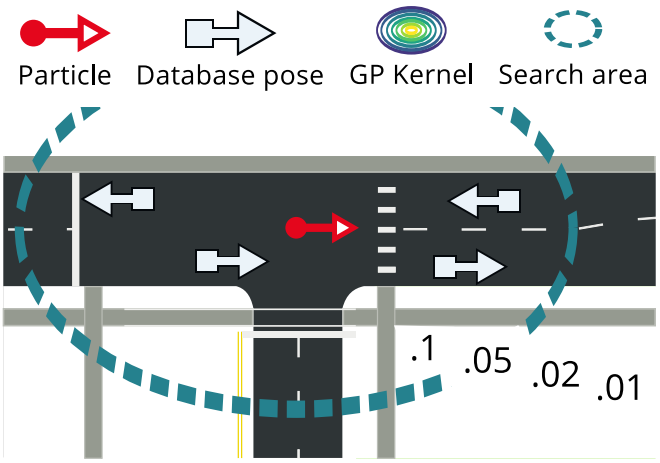


Fig. 2. Approximated GP regression allows the filter to work in large environments. The approximation only uses points that are close (in x, y, θ) to the particle being weighted. The value of the GP kernel is used to define a region from which to select these points. In this illustration, simplified to two dimensions x, y, only points in the area with kernel values under .05 are included. The shaded database point, as well as any other points in the database not seen in the figure, are not used to weight this particle.

In the general case, this approximation can be difficult to implement correctly: the criterion for selecting which subset of points to use is not always simple. However, for this application and the selected Gaussian kernel, selecting which datapoints to use can be done effectively and efficiently, since that only points that are located close enough to a given particle will have an effect in the regression of the descriptor at that particle's location. This can be seen intuitively: images that are far away in position or orientation (for example, rotated more than 90 degrees or 1 km away) have nothing to contribute to the output. We implement this by indexing the locations of the images of the database in a k-d tree. During the execution of the PF, the neighboring datapoints for each particle are searched (Fig. 2) and used as part of the GP observation model, while the rest of the database is ignored. Since the datapoints from the reference database are evenly spread over the map, the weighting phase of the PF executes in constant time regardless of the area of operation. The time of the search does depend on the size of the map, but it is small and grows, at worst, linearly with the number of datapoints in the map [23].

B. Appearance-based particle sampling

When the filter is initialized with an unknown position of the camera, particles are scattered over the map. After that, at least one particle must be close to the right location for the filter to be able to converge. If the map is large, this means that a large number of particles must be used so that the space x, y, θ is densely covered.

Adapting the number of particles so that they are reduced when the filter converges has been a successful solution for indoor, laser-based localization systems [24]. However, on a large outdoor environment like a city, the amount of memory and computation time required to cover the pose

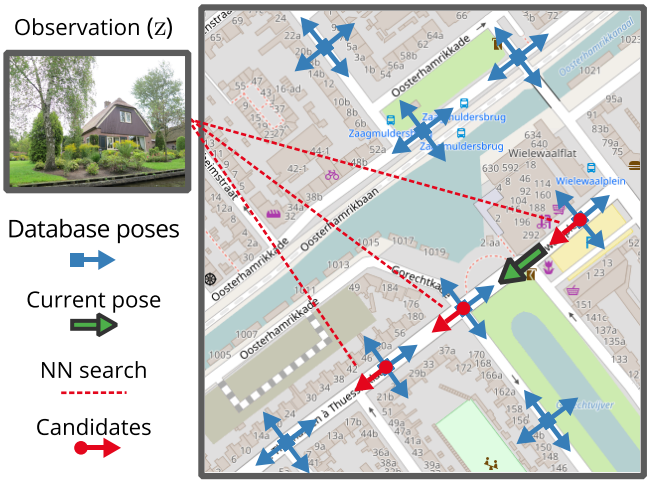


Fig. 3. Drawing new particles from appearance-based nearest neighbor proposals allows the filter to perform global localization and to escape wrong convergence.

space sufficiently makes this unfeasible. Another common problem with PFs is that they can converge to a wrong solution, leaving the filter in an unrecoverable state.

Traditionally, these issues have been relieved by introducing particles at random locations at every evaluation of the PF. We also propose to sample particles at new locations not previously represented by the probability mass. However, instead of sampling randomly, we generate candidates at locations which are visually similar to the current observation (see fig. 3), exploiting the fact that descriptors extracted from CNNs are suitable for appearance-based image retrieval [25].

During the resampling phase of the particle filter, images similar to the current observation in the database are searched: The n_a nearest neighbors of the descriptor z of the current image are retrieved. Then, with probability p_a, particles' poses are set to one of these nearest neighbors (chosen randomly), instead of being resampled from the existing probability mass. This method allows the filter to perform global localization and to recover from incorrect convergence. Another advantage is that the system does not need to explicitly detect that it is lost: the same operations are performed at every PF iteration. This search is also accelerated by means of a k-d tree, so that its time complexity is, at worst, linear with the size of the image database.

IV. Experimental evaluation

In this section, we first introduce the datasets used to perform our experiments: two new datasets and an already existing one. We then perform experiments analyzing the effects of fast GP regression and appearance-based sampling. Finally, we test our system on a challenging crowdsourced collection of sequences.

Datasets and image representation

All our experiments are performed with datasets from the city of Málaga (Spain). We have gathered two new datasets and also use an existing sequence.

Málaga Street View 2016: In order to have a database of images covering a large surface in which to localize video sequences, we collected images in an area of 8 km^2 surrounding the main campus of the university of Málaga using Google Street View. Four images were collected at each location where a Street View panorama was available: facing the vehicle's orientation, and at 90, 180 and 270 degrees. The database, shown as red points in figure 1, is composed of 172.000 images from 43.000 locations.

Málaga Mapillary 2017: We downloaded 50 sequences of images from Mapillary, selected so that they overlap with the Málaga Street View 2016 dataset (used as reference). We selected sequences whose ground truth poses met either one of these criteria: a) Sequences of 20 or more frames in which at least 80% of the images are within the bounding box of the reference database. b) Sequences where 100 or more frames are within the bounding box of the reference database, regardless of the total length. We discarded sequences with wrong or no compass information[5]. This dataset is intended to be used as a difficult test case for localization, as the sequences are recorded in uncontrolled conditions: different cameras, modes of transport, times of day, points of view, speeds, etc.

Málaga Urban Dataset (2013): We also rely on the Málaga Urban Dataset [6] as an easier sequence on which to localize (when compared to the Mapillary sequences), as it is long and recorded from a forward-facing viewpoint on a stable platform. It is sourced from video recorded with a Bumblebee 2 stereo camera mounted on a car. The sequence was recorded on a single 37 km run and includes precise ground truth location from RTK GPS.

Image representation: On all our experiments, we extract NetVLAD [4] descriptors to represent images, following preliminary results where "off-the-shelf" CNN representations and other compact descriptors for place recognition [17] did not work as reliably. The dimensionality of the NetVLAD descriptors is reduced from 1024 to 128 elements through principal component analysis (PCA). This reduction is computed on the reference database (Málaga Streetview 2016) and applied online to the images of the test sequence.

Experiment 1: Fast GP regression

To evaluate the effect of the subset of data approximation, we select random entries (image descriptors and poses) from the Málaga Street View 2016 dataset. We then predict their values through GP regression, using a variable number of neighboring points as data. We compare the result of performing GP regression using a small number of points $\mathbf{y}_{fastGP}$ with the result obtained using a large number of points $\mathbf{y}_{GP}$ (since using the whole dataset is not possible on a normal desktop computer due to memory constraints, we select a 'large' number of points by picking all points within 100 m of the query). We record the normalized euclidean distance from the result of the approximated GP regression

[5]We assumed wrong orientation if it differed by more than 30 degrees, on average, from the orientation of the vectors pointing from the location of each point to the next one in the sequence

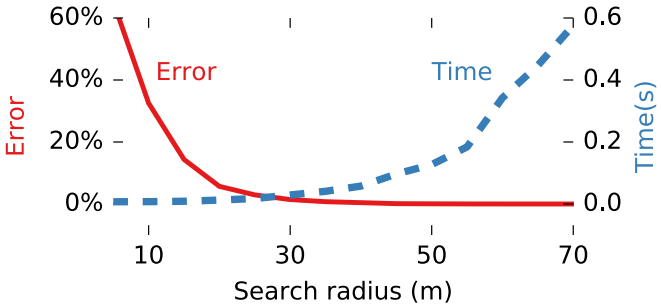


Fig. 4. Using only the neighboring points for GP regression is sufficient on the Málaga Street View 2016 dataset and enables timely execution.

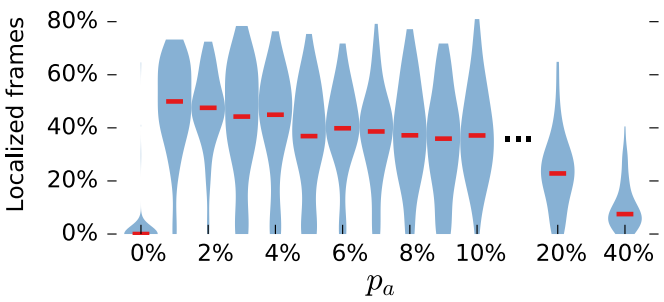


Fig. 5. Sampling a few particles from the reference database at each iteration based on their appearance enables global localization. If too many particles are sampled this way, the filter degenerates into frame-by-frame appearance-based place recognition

to that of the 'full' GP, $||\mathbf{y}_{fastGP} - \mathbf{y}_{GP}||/||\mathbf{y}_{GP}||$ for each test case. Results are averaged over 100 test samples and shown in figure 4. As expected, error decreases when the search radius is increased, also increasing the computational demand. More importantly, selecting a radius larger than 30 m yields almost no error reduction, validating the use of this approximation for localization. We fix the search radius to this value in the following experiments.

Experiment 2: Appearance-based particle sampling

We now test the added value of appearance-based sampling of new particles as introduced in section III-B. We do this by evaluating the full localization system, using the Málaga Street View 2016 dataset as reference, and the Málaga Urban Dataset as the test sequence (both shown in figure 1). The problem is reduced to 2D localization by projecting the poses of the database and the test sequences onto a 2D plane tangential to Earth's surface at the mean point of the locations in the reference database. The PF is initialized by uniformly scattering particles on the map. The size of the filter is set to 500 particles in all our experiments. To simulate errors in motion sensing, the ground truth motion between consecutive frames in the test sequence is perturbed by noise[6] before being used as the odometry input. Particles are moved with the same motion model doubling the amount of position and rotation noise that is added to the actual input. This is done in order to enforce diversity in the particles' poses. The particle filter is evaluated (weighting and resampling) after every 5 m of motion according to this simulated odometry. The output of the system is calculated as the mode of the distribution, estimated by running mean shift on the position of the particles with a Gaussian kernel of $\sigma = 20\ m$. The system is considered to have localized correctly if this estimate is within 15 m of the ground truth position. In each run of the simulation, a randomly selected section of the Málaga Urban Dataset sequence is used, effectively testing on different subsets of the test sequence. Each simulation is executed over 1000 consecutive frames.

We test the effect of appearance-based sampling by varying the values of the parameters p_a and n_a and observing their effect on the localization performance. In fig. 5, we plot the fraction of localized frames in the sequence over 100 particle filter simulations for each value of p_a. The figure shows how completely disabling appearance-based sampling (p_a = 0) makes it very difficult for the PF to localize, as it is highly unlikely that a particle is randomly sampled at the correct pose during initialization. Enabling appearance-based sampling by selecting a small value of p_a allows the newly sampled particles to drive the distribution close to the ground truth location, however, if p_a is large, then many particles are sampled based on image appearance on every step, making the distribution of particles frequently 'jump' from location to location, discarding any accumulated evidence. The effect of the value for n_a is not shown in the figure, since we found the method to be quite robust to the specific value of the number of neighbors within a range $2 < n_a < 10$.

Experiment 3: Localization of crowdsourced sequences

We evaluate the localization system using both improvements (fast GP regression and appearance-based particle sampling) by performing localization of the sequences from the Málaga Mapillary 2017 dataset. This experiment has the same structure as experiment 2, fixing $p_a = 1\%$ and $n_a = 2$. These sequences are more challenging than the Málaga Urban Dataset [6], since they were captured in unconstrained conditions and vary in length from 100 m to 5.6 km, the shorter ones being more difficult to localize as the filter has less chances to accumulate evidence.

We test our system on these sequences and compare against a baseline where each particle is directly weighted using the descriptor distance to the closest image in the database, that is: $w = e^{-||\mathbf{z}-\mathbf{y}_{\mathrm{NN}}||_2}$, where $\mathbf{y}_{\mathrm{NN}}$ is the descriptor of the image in the database closest to the particle being weighted[7]. This baseline uses the same image representation as our proposal (PCA-reduced NetVLAD). We also endow it with appearance-based particle sampling (Sec. III-B). Otherwise, global localization is nearly impossible on this dataset. This comparison thus highlights the advantage of performing probabilistic regression instead of a simple

[6]Gaussian noise with $\sigma_d = 0.1d$ is added to both elements x, y of the motion vector, where $d = ||(x, y)||_2$. The orientation of the particles is also perturbed by Gaussian noise with $\sigma_r = 0.05|r|$, where r is the angle of rotation of the ground truth motion.

[7]we first search for the four closest images and then pick the one with the most similar orientation

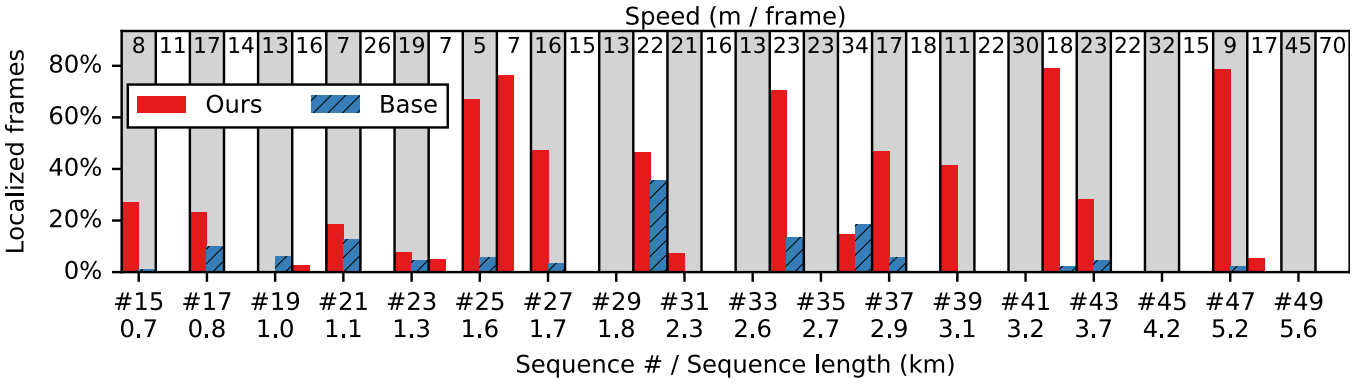


Fig. 6. Fraction of localized frames in sequences 15 to 50 of the Málaga Mapillary 2017 dataset, averaged over 20 runs.

image-to-image comparison when performing localization, as all other aspects (particle filter, motion model, image description, resampling scheme...) are the same. Results are shown in figure 6 as the average number of localized frames for 20 runs on sequences 15 to 50. Sequences 1 to 14 are shorter (under 700 m) and neither the baseline nor our method achieved localization.

V. Conclusions

In large environments, global localization with a standard particle filter is infeasible using a normal GPPF. The appearance-based sampling introduced in section III-B enables global localization with a small number of particles by exploiting appearance-based retrieval techniques. The use of a subset of data approximation allows evaluating the observation model in linear time instead of quadratic time, making GPPFs feasible in large environments.

Experimental validation shows that these advances enable the use of GPPFs for practical, online localization based on egocentric images. As part of this publication, we offer the Málaga Street View 2016 and Málaga Mapillary 2017 datasets online at `mapir.isa.uma.es`.

References

[1] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual Place Recognition: A Survey," *IEEE Transactions on Robotics (TRO)*, 2016.

[2] B. Ferris, D. Haehnel, and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proceeding of Robotics: Science and Systems*, 2006.

[3] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[4] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] D. Jayaraman and K. Grauman, "Learning image representations tied to egomotion," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[6] J.-L. Blanco, F.-A. Moreno, and J. González-Jiménez, "The Málaga Urban Dataset: High-rate Stereo and Lidars in a realistic urban scenario," *The International Journal of Robotics Research (IJRR)*, 2014.

[7] M. Lopez-Antequera, N. Petkov, and J. Gonzalez-Jimenez, "Image-based localization using Gaussian processes," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016.

[8] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research (IJRR)*, 2008.

[9] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[10] E. Pepperell, P. Corke, and M. Milford, "All-environment visual place recognition with SMART," *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[11] ——, "Routed roads: Probabilistic vision-based place recognition for changing conditions, split streets and varied viewpoints," *The International Journal of Robotics Research (IJRR)*, 2016.

[12] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Towards Life-Long Visual Localization using an Efficient Matching of Binary Sequences from Images," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[13] W. Maddern, M. Milford, and G. Wyeth, "CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory," *The International Journal of Robotics Research (IJRR)*, 2012.

[14] G. Vaca-Castano, A. R. Zamir, and M. Shah, "City scale geo-spatial trajectory estimation of a moving camera," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[15] A. Taneja, L. Ballan, and M. Pollefeys, "Never Get Lost Again: Vision Based Navigation Using StreetView Images," in *Asian Conference on Computer Vision (ACCV)*, 2015.

[16] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional Neural Network-based Place Recognition," *arXiv*, 2014.

[17] R. Gomez-Ojeda, M. Lopez-Antequera, N. Petkov, and J. Gonzalez-Jimenez, "Training a Convolutional Neural Network for Appearance-Invariant Place Recognition," *arXiv*, 2015.

[18] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep Learning Features at Scale for Visual Place Recognition," *arXiv*, 2017.

[19] T. Schairer, B. Huhle, P. Vorst, A. Schilling, and W. Straßer, "Visual mapping with uncertainty for correspondence-free localization using Gaussian process regression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[20] R. Gomez-Ojeda and J. González-Jiménez, "Robust Stereo Visual Odometry through a Probabilistic Combination of Points and Line Segments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[21] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine (RAM)*, 2011.

[22] C. E. Rasmussen, *Gaussian processes for machine learning*, 2006.

[23] S. Maneewongvatana and D. M. Mount, "It's okay to be skinny, if your friends are fat," *Center for Geometric Computing 4th Annual Workshop on Computational Geometry*, 1999.

[24] D. Fox, "KLD-sampling: Adaptive particle filters," in *Advances in neural information processing systems (NIPS)*, 2001.

[25] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Visual Instance Retrieval with Deep Convolutional Networks," *CoRR*, 2014.

Algorithms for limited-buffer shortest path problems in communication-restricted environments

Alessandro Riva, Jacopo Banfi, Arlind Rufi, and Francesco Amigoni

***Abstract*— In several applications, a robot moving from a start to a goal location is required to gather data along its path (e.g., a video feed in a monitoring scenario). The robot can have at its disposal only a limited amount of memory to store the collected data, in order to contain costs or to avoid that sensible data fall into the hands of an attacker. This poses the need of periodically delivering the data to a Base Station (BS) through a deployed communication infrastructure that, in general, is not available everywhere. In this paper, we study this scenario by considering a variant of the shortest path problem (which we prove to be NP-hard) where the robot acquires information along its path, stores it into a limited memory buffer, and ensures that no information is lost by periodically communicating data to the BS. We present and evaluate an optimal exponential time algorithm, an efficient feasibility test, and a polynomial time heuristic algorithm.**

I. Introduction

In several applications, a robot moving from a start to a goal location may be required to gather data along its path. This happens, for instance, in some monitoring applications, where the robot acquires a video feed to be later processed at a Base Station (BS) [1] or in those civilian or military settings where *a posteriori* processing of log files is required to ensure that the robot has not been hijacked by a malicious attacker [2].

The robot can have at its disposal only a limited amount of memory to store the collected information. In civilian settings, this could be motivated by the need to reduce costs, while, in military settings, this could be enforced to avoid that a large amount of sensible data fall into the hands of a malicious attacker. This memory limitation poses the need of periodically transmitting data to a BS in order not to overfill the available memory buffer.

In most application settings, it is unrealistic to assume the presence of a robust communication infrastructure able to uniformly cover the environment with the same (high) transmission rate. Typical settings can instead rely only on the presence of a limited number of "communication zones" from where robots can reliably communicate with the BS [1]. Moreover, such zones could display a significant variability in the data transfer rate, also according to the distance from the communication device [3]. The combination of requirements imposed by incremental data acquisition, limited memory, and restricted communication define a challenging path planning scenario.

The authors are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy {alessandro.riva, jacopo.banfi, francesco.amigoni}@polimi.it, arlind.rufi@mail.polimi.it

In this paper, we study a variant of the shortest path planning problem with the goal of planning high-level paths (defined in terms of waypoints to traverse) under the constraints imposed by the above scenarios. Specifically, the problem we consider is to compute the shortest path between given start and goal locations in a graph representing the environment, while (a) accounting for the storage of the gathered information into a buffer of limited size and (b) ensuring that no information is lost through periodic transmissions of data from communication zones to the BS.

In particular, after formalizing the problem and proving its NP-hardness, we present three original contributions: (i) an optimal exponential time solving algorithm, (ii) an efficient feasibility test that can be applied to all problem instances to check if they admit a solution, and (iii) a polynomial time heuristic algorithm based on the refinement of a "raw" feasible solution. Experiments, conducted in simulations, show that the proposed algorithms can effectively be used to trade-off optimality of the solution against running time.

II. Related Work

The problem we address in this paper shares some similarities with some optimization problems studied both in robotics and in other research fields. In this section, we provide a brief overview of related work.

Planning high-level paths is a fundamental ability that a mobile robot needs to possess [4]. Without constraints, this is commonly accomplished by resorting to well-known algorithms, like Dijkstra's for graph-based planning [5] or variants of A* search for geometric planning [6]. In the literature, different works try to incorporate in the path planning problem some additional (possibly application-dependent) constraints. For instance, [7] is one of the first works to propose a path planning algorithm (inspired to A*) able to cope with additional constraints, such as time, risk, energy, and uncertainty. An example of a more recent work is [8], which studies path planning for a solar-powered robot subject to time and energy constraints. However, communication issues in path planning have been mainly investigated for multirobot systems, especially in the context of maintaining (more or less periodically) the team of robots globally connected through a multi-hop network while pursuing a primary mission objective (see, for instance, the solution devised in [9] for informative path planning under periodic connectivity constraints).

The communication paradigm we use in this work is introduced in [1] in the context of multirobot patrolling. It assumes the presence of a number of "communication zones"

that robots can exploit to communicate with the BS. In this paper, we further refine the model by associating different zones with (possibly) different data transmission rates.

At an algorithmic level, the problem we investigate can be framed in the class of the *shortest path problems with resource constraints* [10]. In particular, it shares similarities with variants of the Constrained (or restricted) Shortest Path problem (CSP) [11]. The CSP is a generalization of the shortest path problem on graphs in which each edge is associated not only with a distance, but also with an additional *weight*, and the objective is to find the shortest path between given start and target locations subject to a constraint on the total weight accumulated along the path. In this regard, the literature offers some particularly relevant works. The work in [12] investigates a generalization of the CSP where edges may be associated to binary indicators able to reset the accumulated weight when the corresponding edge is traversed. This model is not applicable in our case, since we have to deal with the possibility of transmitting only a portion of the accumulated data. Authors of [13], instead, propose a model that partially generalizes ours in the context of planning shortest paths with charging stops for electric vehicles (the filling of our buffer can be naively thought as dual to the draining of the battery). In particular, their model is able to cope with continuous, increasing, and concave functions describing how batteries recharge, even when going downhill. However, the existence of cycles with negative cumulative weight, namely the possibility of traveling along cycles while indefinitely recharging the battery, is clearly ruled out by the laws of physics. In our case, we must also contemplate such a possibility (think of a robot which takes a detour towards a transmission area with an increasingly higher transmission rate, and then goes back to its original path). A Fully Polynomial-Time Approximation Scheme (FPTAS) is presented in [14] for a restricted version of the model studied in [13].

III. Problem Setting

We model the environment as a connected, simple, weighted graph $G = (V, E)$, where V represents physical locations the robot can occupy and E represents the connections between those locations. Edges are associated with a weight function $t : E \to \mathbb{N}^+$, called *time function* and representing their traveling time. We assume that time evolves in discrete steps $\mathbb{N}^+$, as well. Between two subsequent time steps, the robot can either *stay still* at its current vertex or *move along a graph edge*. In the latter case, the robot is not allowed to interrupt an edge traversal once started, but it must reach the destination vertex before making another decision. In any case, at each time step, the robot stacks 1 unit (or, equivalently, any constant amount) of information into a buffer of size $B \in \mathbb{N}^+$.

The environment has some communication zones modeled as a set of *transmission vertices* $V_T \subseteq V$. To formalize the transmission of data, we define a *transmission-rate function* $r : V^2 \to \mathbb{Q}$ that describes the amount of information a robot can send to the BS *between two consecutive time steps* when moving from a vertex to another one or staying at a vertex. To consistently model transmissions according to reasonable assumptions about communication and legal moves in G, we impose some constraints on $r()$:

1. (**vertex transmission capability**) $r(v, v) > 0 \iff v \in V_T$;
2. (**legal moves**) $r(u, v) = 0$ for each $u \neq v$ s.t. $(u, v) \notin E$;
3. (**edge conservativeness**) for each $(u, v) \in E$, $r(u, v) \leq \max\{r(u, u), r(v, v)\}$.

To compact the notation, we also define the net amount of information unstacked from the buffer in a time step:

$$\bar{r}(u, v) = r(u, v) - 1.$$

If $\bar{r}(u, v) > 0$ the amount of information transmitted exceeds the amount of information stacked in a time step and, thus, the robot is able to unfill information from the buffer while moving from u to v.

The robot must move from a start vertex $s \in V$ to a goal vertex $g \in V$, without any constraints on the initial and final amount of data contained into the buffer. A *solution* S of our problem consists of a sequence of k pairs $p_i \in V \times \mathbb{N}$, representing the number of steps in which the robot remains still at a vertex. A solution starts from s and ends in g:

$$S = [p_1 = (s, t_s), p_2, \ldots, p_k = (g, 0)],$$

where two subsequent pairs $p_i = (v_i, t_i)$ and $p_{i+1} = (v_{i+1}, t_{i+1})$ implicitly define the traversal of the graph edge $(v_i, v_{i+1}) \in E$ after having remained still for t_i steps at v_i. This solution encoding defines a sequence of pairs of values $[(b_1^I, b_1^O), (b_2^I, b_2^O), \ldots, (b_k^I, b_k^O)]$ representing the amount of data present into the buffer when arriving (I) and leaving (O) from each of the k vertices composing a solution.

We say that a solution S is *feasible* iff $b_i^I, b_i^O \leq B$ for each $i = 1, 2, \ldots, k$. The objective is to reach the goal in the least possible time, i.e., to minimize:

$$T = \sum_{i=1}^{k-1} \left[t_i + t(v_i, v_{i+1}) \right]. \tag{1}$$

We call this problem *Limited-Buffer Shortest Path problem* (*LBSP*).

A. NP-hardness

We now give strong evidence to the fact that LBSP is a hard problem by proving that the corresponding decision version, called LBSP-D, is NP-hard. In LBSP-D, the aim is to decide whether a given instance of LBSP admits a feasible solution with total time less than a given $\overline{T}$, with $\overline{T} \in \mathbb{N}^+$. To this aim, we construct a reduction from the decision version of CSP, which is NP-complete [15]:

CSP-D

INSTANCE: a graph $\hat{G} = (\hat{V}, \hat{E})$, an edge length function $l : \hat{E} \to \mathbb{N}^+$, an edge weight function $w : \hat{E} \to \mathbb{N}^+$, start and goal vertices $\hat{s}, \hat{g} \in \hat{V}$, positive integers $L, W \in \mathbb{N}^+$.

QUESTION: is there a path from $\hat{s}$ to $\hat{g}$ in $\hat{G}$ that has total length at most L and total weight at most W?

Algorithm 1: Graph Transformation

Input: A simple undirected graph $G = (V, E)$, a buffer size B, the rate function $r()$, the time function $t()$
Output: A weighted directed graph $G_B = (V_B, A_B, w)$

1 **function** transformGraph(G, B, r, t)
2 $V_B \leftarrow \{\}$
3 $A_B \leftarrow \{\}$
4 **foreach** $v \in V$ **do**
5 $V_B \leftarrow V_B \cup \{v^0, v^1, \ldots, v^B\}$
6 **foreach** $u, v \in V$ **do**
7 **for** $b = 0$ to B **do**
8 $x \leftarrow \max\{0, b - t(u,v)\bar{r}(u,v)\}$
9 **if** $x \leq B$ **then**
10 $A_B \leftarrow A_B \cup \{(u^b, v^x)\}$
11 $w(u^b, v^x) \leftarrow t(u,v)$
12 **return** $G_B = (V_B, A_B, w)$

Without loss of generality, we consider only CSP-D instances in which $l(u,v) \geq w(u,v), \forall (u,v) \in \hat{E}$. Indeed, any problem instance can be turned into an instance satisfying such a constraint by simply multiplying all the lengths by a proper constant. For the reduction, we set $G = \hat{G}$ (implying $V = \hat{V}$ and $E = \hat{E}$), $V_T = V$, $B = W$, and $\overline{T} = L$. The edge time function t of LBSP-D is set equal to the edge length function l of CSP-D. The rate function, for each $u \neq v \in V$, is defined as:

$$r(u,v) = \begin{cases} 1 - \dfrac{w(u,v)}{l(u,v)} & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$

Also, for each $v \in V_T$, the rate function is defined as:

$$r(v,v) = \max\{r(u,v) \mid (u,v) \in E\}.$$

Notice that all the transmission rates are lower than or equal to 1. This means that there is no advantage in staying still at any vertex, since the buffer value would not decrease. More formally, if there exists a solution of LBSP-D whose stop time t_i at a vertex is not 0, there also exists a not worse solution whose stop time t_i is 0. Also, because $l(u,v) \geq w(u,v)$, $\forall (u,v) \in E$ there are no "negative cycles", i.e., cyclic paths allowing to decrease the buffer value when traveled. More generally, if there exists a solution where the robot travels along a cycle, then there exists a not worse solution without cycles.

Given what said above, it is straightforward to check that the constructed LBSP-D instance admits a *yes* answer iff the original CSP-D instance admits a *yes* answer.

IV. Algorithms

A. Optimal Algorithm

We now present an exponential algorithm for solving to optimality the LBSP defined in the previous section.

Let us notice that, despite the amount of information stacked and unstacked could be a rational number $\mathbb{Q}$, any problem instance can be turned into an equivalent instance where all the possible *buffer states* are – arbitrarily large – integer numbers. In particular, let M be the least common multiple of all the $\bar{r}()$'s denominators. From now on, we assume that all the values of the time function t and the buffer size B are defined as multiple of M and thus only integer buffer states are allowed (the original time values can be obtained, once a solution is found, dividing by M).

The algorithm leverages a transformation of the input graph G, whose pseudo-code is reported in Algorithm 1. The *directed* graph obtained, G_B, is an expanded version of G in which the state of the buffer is explicitly represented for each vertex through a set of "buffer-expanded" vertices. An optimal solution to the LBSP on G is then obtained by simply finding a shortest path on the new graph G_B.

In order to simplify the pseudo-code, it is assumed that, for each $u, v \in V$, $t(u,v) = \infty$ if $(u,v) \notin E$. Also, we set $t(v,v) = M$. The algorithm starts by creating $B+1$ vertices $\{v^0, v^1, \ldots, v^B\}$ for each vertex in V (lines 4-5): these vertices univocally identify the state of the robot, i.e., v^i means that the robot is at v with buffer state i.

The algorithm then builds the set of arcs, which are of two types: those connecting vertices of G_B corresponding to the same vertex of G, but with different buffer states (i.e., connecting v^i and v^j), and those connecting vertices of G_B corresponding to different vertices in G (and possibly different buffer states). In the pseudo-code, when $u = v$, the first case is handled, otherwise, the second case is covered (lines 6-11). These arcs correspond to temporal transitions in the system state. Following an arc, the robot can either stay still on a vertex of G and change its current buffer state, or move to another vertex of G (possibly changing his buffer state too). Notice that, for each vertex in V_B, there is exactly one arc of the first type and at most $|V|-1$ arcs of the second type. This means that $|A_B|$ is lower than or equal to $MB|V|^2$ (where B is the buffer size).

Once the graph G_B is returned, a shortest path algorithm is applied, e.g., Dijkstra's algorithm, to find a shortest path from s^{b_0} to any g^x, where s and g are the start and the goal vertices on G, respectively. (x could be restricted to a set of values, if additional constraints to the final state of the buffer are imposed.) The correctness of the algorithm follows from the fact that we are explicitly representing all the possible buffer states. The whole computing time of the transformation and the shortest path seeking is clearly exponential, and both upper-bounded by $O(MB|V|^2)$.

B. Feasibility Test

To decide the feasibility of a given problem instance, one could apply Algorithm 1 and check whether the obtained graph G_B contains at least an (s,g)-path. If not, the instance does not admit any feasible solution. However, since the computing time of such a procedure could be large (recall the above complexity bound), it could be useful to have at hand a faster method to decide feasibility.

Algorithm 2: Find Times

Input: A feasible walk $\mathbf{w} = [v_1, \ldots, v_k]$ on G, a buffer size B, the rate function $r()$, the time function $t()$
Output: A sequence of times T

```
function findTimes(w, B, r, t)
    b^I, b^O, T ← k-length array initialized to 0
    for i = 0 to k − 1 do
        b^O_i ← max{0, b^I_i − t_i r̄(v_i, v_i)}
        b^I_{i+1} ← max{0, b^O_i − t(v_i, v_{i+1}) r̄(v_i, v_{i+1})}
        if b^I_{i+1} > B then
            updateTimes(T, b^I, b^O, i)
            updateBuffers(T, b^I, b^O, i)
    return T
```

We now present a simple method that leverages two additional graphs $G_1 = (V_1, E_1, w_1)$ (weighted) and $G_2 = (V_2, E_2)$ (unweighted). To obtain G_1, we set $V_1 = V$ and $E_1 = E$. Then, for each $(u, v) \in E_1$, we set $w_1(u, v) = \max\{0, -t(u, v)\bar{r}(u, v)\}$. The weights of G_1 represent the amount of stacked data (i.e., the amount of increase of the buffer value, if any) the robot attains when traversing (u, v). For what concerns G_2, the set V_2 is the set of vertices of G whose transmission rate is strictly greater than 1, plus s and g. We add an edge (u, v) to E_2, with $u, v \in V_2$, iff the length of the shortest path from u to v in G_1 (with weights w_1) is less than or equal to B. This is equivalent to say that there exists a u-v path in G such that it is always possible to travel from u to v without overfilling the buffer (for a sufficiently low buffer state in u).

Given the graph G_2 constructed as above, it can be easily shown that an (s, g)-path in G_2 exists if and only if the problem instance admits at least a feasible solution. The computing time of this procedure is bounded by the complexity of finding a shortest path between each pair of vertices in G_1 (to compute E_2), that is, $O(|V|^3)$.

C. A Heuristic Algorithm

We now present a heuristic algorithm based on the refinement of an initial solution, which can be obtained from a weighted variant of the graph G_2 used above for the feasibility test. In particular, we start by constructing the "skeleton" of the heuristic solution as the walk (a sequence of possibly-repeated vertices) $\mathbf{w} = [s = v_1, v_2, \ldots, v_k = g]$ associated to the shortest (s, g)-path on G_2 according to a set of weights w_2. Specifically, the weight $w_2(u, v)$ of an edge $(u, v) \in E_2$ is defined as the traveling time of the path associated to the satisfaction of the buffer constraint (i.e., the traveling time of the path obtained by minimizing the weights w_1 defined previously). We now present a method to compute the *sequence of stopping times* for $\mathbf{w}$. (In fact, this method allows to compute the stopping times for any given sequence of vertices underlying a feasible solution.)

Formally, in order to obtain a solution $S = [p_1 = (s, t_s), p_2, \ldots, p_k = (g, 0)]$, we have to find a sequence of times $T = [t_1 = t_s, t_2, \ldots, t_k = 0]$ such that the buffer constraint is satisfied and the objective function (1) is minimized. To this aim, we use the procedure of Algorithm 2.

The algorithm iterates through the vertices of the walk, iteratively updating the buffer state b^I, b^O at each vertex. The idea is that, at a given vertex v_i, if the robot cannot reach the next vertex v_{i+1} in the walk, it has to necessarily transmit a certain amount of information before proceeding further. Such an amount is transmitted by the updateTimes() function, which iteratively tries to exploit the vertices with the higher transmission rates. More precisely, the algorithm makes use of two functions, which have access to the input data B, r, t.

- updateTimes(T, b^I, b^O, i): it starts computing the amount to transmit $q = b^I_{i+1} - B$, in order to proceed further through the walk. This function tries to transmit q units of information at each already traversed vertex (index less than $i + 1$), starting from the one with the highest transmission rate and considering only those vertices whose rate is greater than 1. At each transmission attempt, for each $j < i$, we check two constraints: (1) at vertex v_j, the robot cannot transmit more than the current value of b^O_j and (2) moving backward from v_{i+1} to v_j (without ever stopping) the buffer has to never exceed B. At this point, if q units of information are completely transmitted, the sequence of times T is updated.
- updateBuffers(T, b^I, b^O, i): given an updated sequence of times T, it straightforwardly updates the buffer states b^I and b^O, up to the vertex v_{i+1}.

The algorithm has a complexity of $O(|\mathbf{w}|^2)$, since updateTimes() can be run in $O(|\mathbf{w}|)$ by pre-sorting the vertices in the walk by transmission rates and pre-computing the distances between each pair of vertices. Therefore, the whole heuristic complexity (including finding a feasible solution) is $O(|V|^3 + |\mathbf{w}|^2)$.

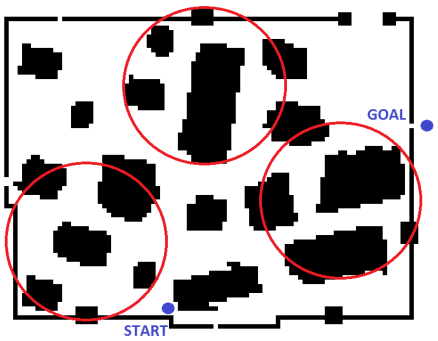


Fig. 1: Experimental environment (size 400×300 m). Red discs represent the communication zones (e.g, areas covered by RF transceivers).

V. Experiments

We implemented the algorithms described in the previous section in C++ (using the LEMON graph library [16]) and in this section we evaluate their performance in monitoring the simulated military outpost depicted in Fig. 1 (size 400×300

m), where we are interested in keeping the buffer size as small as possible. All the experiments are run on a laptop equipped with an Intel Core i7@2.70 GHz CPU and 16 GB RAM.

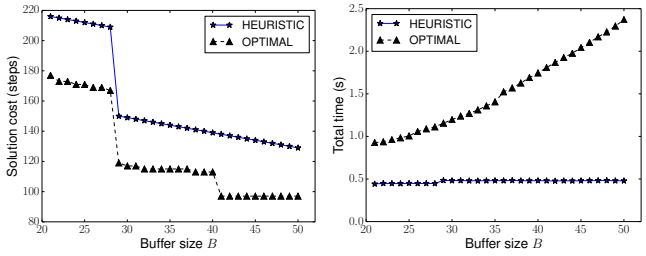
Planning takes place on the vertices of a uniform 4-connected grid, where two adjacent vertices are separated by a length of 4 m. The robot moves at 4 m/s, collecting 8 Mbit/s of data (for instance, a video feed at low resolution), which will represent our basic buffer unit (i.e., the "+1" unit of data stacked into the buffer at each time step). The environment presents three RF transceivers (e.g., WiFi), with maximum range of 20 cells (80 m) whose bitrate changes across the different experiments according to the distance from the transceiver location.

In the first set of experiments, we keep fixed start and goal locations as shown in Fig. 1 and study how the solution cost (expressed as number of time steps) and the runtime of our algorithms vary as a function of the buffer size B. We model a conservative scenario where the available bitrate at the center of the communication zone is 32 MBit/s, and decreases each 4 grid cells of 8 MBit/s: therefore, the robot can transmit at rates $4, 3, 2, 1$ w.r.t. the buffer unit, and the maximum range is reduced to 16 cells. (When moving between cells having a different rate, we assume that the robot can transmit at the minimum of the two rates.)

Fig. 2 summarizes the results obtained. Examining Fig. 2(a), we can immediately notice how the optimal algorithm behaves significantly better than the heuristic. At the same time, note that both the algorithms are able to obtain feasible solutions from a buffer size $B = 21$. Also, in both cases, the solution cost decreases for increasingly larger buffer sizes. While it is expected for the optimal algorithm, this fact suggests the soundness of the heuristic algorithm. Fig. 2(b) shows the computing time required by the two algorithms. Clearly, the heuristic algorithm runs faster, but its efficiency does not always compensate for the loss in solution quality. Fig. 4(a) reports three example paths (we do not explicitly show where the robot stops in a cell for 1 or more steps to transmit data) computed by the optimal algorithm and corresponding to different buffer size values: the smallest value for which we obtain a solution ($B = 21$) and the two values corresponding to sharp changes in the solution cost ($B = 29$ and $B = 41$). Notice how these paths vary in length according to the number of different transmission regions traversed. Moreover, note that the three paths belong to three different "classes", taking routes that cannot be reduced to each other.

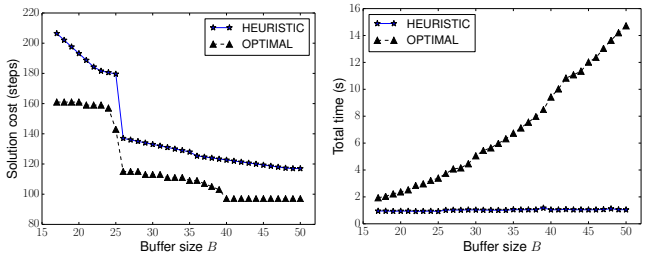
In the second set of experiments, we keep fixed the start and goal locations as before, but we model a less conservative scenario where the available bitrate at the center of the communication zones is 32 MBit/s, remains fixed at this value within a distance of 4 cells from the center, and decreases of 2 MBit/s for each subsequent cell. Again, we study how the solution cost and the runtime of the algorithms vary as a function of the buffer size B.

Fig. 3 shows the results. Looking at the solution costs of Fig. 3(a), we observe the same trends of the previous set



(a) (b)

Fig. 2: Results of the first set of experiments (conservative scenario).



(a) (b)

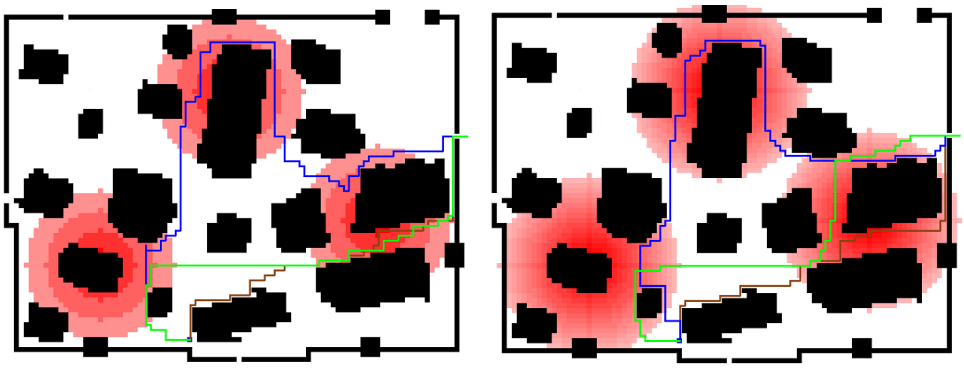
Fig. 3: Results of the second set of experiments (non-conservative scenario).

of experiments. Compared to the first set of experiments, the runtime of our algorithms (Fig. 3(b)) increases, but not dramatically. Fig. 4(b) shows three example paths computed by the optimal algorithm and corresponding to interesting buffer size values. The same considerations made before hold also in this case.

In the final set of experiments, we consider both the scenarios above and we compare the solution quality of the optimal algorithm against that of the heuristic one on 100 randomly selected pairs of start-goal locations for each buffer size. For each instance, we keep fixed start and goal locations among the two rate scenarios. Figs. 5(a)-(b) show the number of instances for which the two algorithms return a solution. (Note that, by construction, both the algorithms return a feasible solution whenever there is at least one.) In both scenarios, this number increases with the buffer size, as expected. Clearly, for a fixed buffer size, we are able to solve a larger number of instances in the non-conservative scenario. Figs. 5(c)-(d) show the average gap (in percentage) between the solution returned by the optimal algorithm and the heuristic, plotted with 95% confidence interval bars. In both cases, the average gap increases as the buffer size increases. Also, note how the average gap of the conservative scenario is significantly smaller than the gap of the non-conservative one for a given buffer size, possibly because it represents a simpler setting.

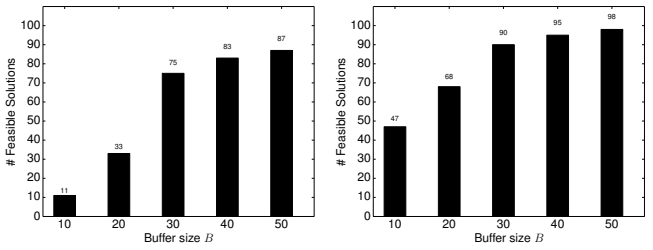
VI. Conclusions

In this paper we considered the problem of finding shortest paths under a limited-buffer constraint, for a robot that ac-

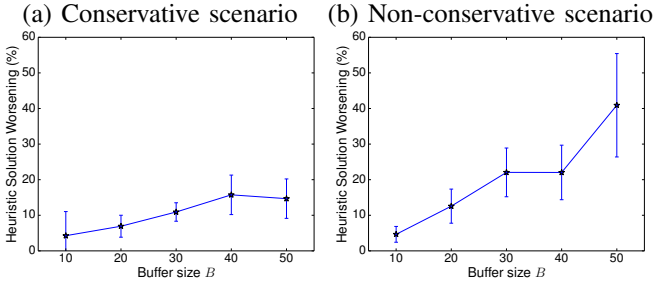


(a) (b)

Fig. 4: Example paths of the first two sets of experiments. (a) Firs set (blue: $B = 21$, green: $B = 29$, brown: $B = 41$); (b) Second set (blue: $B = 17$, green: $B = 26$, brown: $B = 40$).



(a) Conservative scenario (b) Non-conservative scenario



(c) Conservative scenario (d) Non-conservative scenario

Fig. 5: Results of the final sets of experiments on 100 randomly selected pairs of start-goal locations.

quires data while the time evolves and can transmit them only from some communication zones. We called this problem LBSP. We proposed an optimal algorithm and an heuristic algorithm, which leverages a feasibility algorithm and the optimal assignment of the transmission times. In the experiments, the cost of the solutions returned by the heuristic algorithm is comparable with that of solutions found by the optimal algorithm, but the gap increases with the complexity of the setting. The computing time of the heuristic algorithm is consistently shorter than that of the optimal one.

An interesting direction for future research is the investigation of the applicability of the approach proposed by [17] to our constrained path planning problem. From a theoretical point of view, it would also be worth it to investigate the NP-membership of the LBSP-D. Finally, we are planning the implementation of the proposed algorithms on real robots in order to further validate the feasibility of our approach.

References

[1] J. Banfi, N. Basilico, and F. Amigoni, "Minimizing communication latency in mutirobot situation-aware patrolling," in *Proc. IROS*, pp. 616–622, 2015.

[2] M. Hooper, Y. Tian, R. Zhou, B. Cao, A. Lauf, L. Watkins, W. Robinson, and W. Alexis, "Securing commercial wifi-based uavs from common security attacks," in *Proc. MILCOM*, 2016.

[3] J. Jansons and T. Dorins, "Analyzing IEEE 802.11 n standard: outdoor performance," in *Proc. ICDIPC*, pp. 26–30, 2012.

[4] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[5] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer Math*, vol. 1, no. 1, pp. 269–271, 1959.

[6] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *J Artif Intell Res*, vol. 39, pp. 533–579, 2010.

[7] O. Causse and J. Crowley, "Navigation with constraints for an autonomous mobile robot," in *Proc. IROS*, vol. 3, pp. 1899–1905, 1994.

[8] P. Plonski, P. Tokekar, and V. Isler, "Energy-efficient path planning for solar-powered mobile robots," *J Field Robot*, vol. 30, no. 4, pp. 583–601, 2013.

[9] G. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: theory and experiments," *IEEE Trans Robot*, vol. 28, no. 4, pp. 967–973, 2012.

[10] S. Irnich and G. Desaulniers, "Shortest path problems with resource constraints," in *Column generation* (G. Desaulniers, J. Desrosiers, and M. Solomon, eds.), pp. 33–65, Springer, 2005.

[11] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math Oper Res*, vol. 17, no. 1, pp. 36–42, 1992.

[12] M. Bolívar, L. Lozano, and A. Medaglia, "Acceleration strategies for the weight constrained shortest path problem with replenishment," *Optim Lett*, vol. 8, no. 8, pp. 2155–2172, 2014.

[13] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf, "Shortest feasible paths with charging stops for battery electric vehicles," in *Proc. ACM SIGSPATIAL GIS*, 2015. Paper 44.

[14] S. Merting, C. Schwan, and M. Strehler, "Routing of electric vehicles: constrained shortest path problems with resource recovering nodes," in *Proc. ATMOS*, 2015.

[15] M. Garey and D. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[16] B. Dezső, A. Jüttner, and P. Kovács, "LEMON - An open source C++ graph template library," *ENTCS*, vol. 264, no. 5, pp. 23–45, 2011.

[17] S. Bhattacharya and V. Kumar, "Persistent homology for path planning in uncertain environments," *IEEE T Robot*, vol. 31, no. 3, pp. 578–590, 2015.

Mobile robotics in arable lands: current state and future trends

Luis Emmi, Pablo Gonzalez-de-Santos

Abstract— **This paper presents a summary of the current state of mobile robotics oriented to perform precision agricultural tasks on arable lands. Two approaches of robot configurations are identified and some relevant examples are mentioned in addition to identifying the trend of robotics in agriculture, the current limitations, and the following steps as understood by the authors for reducing the gap for increased inclusion of robotics in everyday agricultural tasks.**

I. Introduction

Recent reports forecast annual growth in the global mobile robotics market between 2013 and 2019 [1]. Mobile robots have become widely utilized in outdoor applications where highly unstructured and rough terrains present a challenge to perform difficult tasks while maintaining a high level of accuracy, safety and robustness. Some commercial robotic systems can be found in this type of environment, such as the forestry and mining industries [2], equipped with advanced technologies for (1) positioning and orientation, (2) navigation and planning, and (3) sensing and identification. However, there are very few fully autonomous commercial-oriented solutions for the agricultural industry and more specifically, solutions for arable lands, despite the fact that the complexity of the environment is very similar and the required technologies are already available. On the other hand, the amount of robotics prototypes for precision farming tasks or precision agriculture (PA) [3] has been increased considerably in recent years. This is due to the latest technological advances that have allowed

- centimeter-level localization of mobile robots in outdoor environments (e.g., Real-Time Kinematics Global Positioning System or RTK-GPS [4]–[6]);
- precise identification of field and crop needs by using hyperspectral high-resolution cameras (e.g., remote sensing [7]–[9]); and
- decision making and information systems for crop management optimization (e.g., farm management systems [10]–[12]).

The advances in the aforementioned technologies have helped to apply precision farming techniques in arable lands, which have been studied for many years. For example, beginning in the 1980's, the benefits of treating crop fields selectively began to experience increased study [13], given that:

- field and crop needs have a considerable spatial variability, where the use of conventional techniques ceases to be efficient;
- infestations are found in patches in different areas and forms as well as in different types [14];
- required nutrients for crops are also distributed heterogeneously in diverse quantities throughout the field; and
- heavy machinery has a considerable impact in soil quality, and a significant amount of energy and resources is required to repair such damage.

However, until the achievement of the appropriate technological level to identify these field needs with the required precision and consecutively to act appropriately, the fieldwork is performed manually. Likewise, autonomous vehicles have demonstrated their potential in recent decades. In the early 1990s, initial developments at Carnegie-Mellon University demonstrated that an autonomous vehicle could be possible even when a Global Navigation Satellite System (GNSS) was not used [15]. These preliminary results encouraged the Defense Advanced Research Projects Agency (DARPA) to continue to support these types of projects by funding the DARPA Grand Challenges (DGCs) [16], which gradually achieved autonomous cars capable of navigating through urban areas while obeying all traffic regulations and negotiating other traffic and obstacles. Currently, Google is pursuing the Google Self-Driving Car project [17] with extraordinary results, and several states in the USA have begun to legalize driverless cars since 2011 [18]. This means that the relevant technology is ready to be used in urban, crowded areas. Therefore, it is ready to be applied in agricultural vehicles as well.

The commercial availability of GNSSs (currently, GPS and GLONASS; Galileo and Compass are expected to be operative in 2020) has provided easier methods of configuring autonomous vehicles or navigation systems to assist drivers (note that the DGCs did not allow the use of such positioning systems). Today, a large number of navigation systems for agricultural vehicles have become available, ranging from mere driving assistants that instruct the driver whether to move left or right (light bars) to highly accurate vehicle steering systems (Autopilot™, Trimble; AutoTrac™, John Deere). These systems aid the operator in the precise guidance of agricultural tractors using LASER or GPS technology (accurate to the centimeter level) but do not endow a vehicle or implement with any kind of autonomy; they require some kind of intelligence.

Therefore, the required technology is available today that will allow more robotic systems to be incorporated into the food industry and more specifically in the production and

L. Emmi and P. Gonzalez-de-Santos are with the Centre for Automation and Robotics (UPM-CSIC), 28500, Arganda del Rey; Madrid; Spain (corresponding author to provide phone: +34918711900; fax: +34918717050; e-mail: luis.emmi@car.upm-csic.es; pablo.gonzalez@car.upm-csic.es).

harvesting of the product in arable lands (outdoor environments), with the purpose of

- reducing the use of herbicides (polluting elements for water and soil);
- applying the required amount of fertilizers (representing a significant amount of energy input in agriculture) in the required zones;
- guiding robotic systems into the fields by minimally damaging both the soil and the crop; and
- performing harvesting tasks for both grains and fruits in a much more efficient way.

This article presents the current state of mobile robotics for agricultural tasks capable of operating in arable lands as well as the latest trends in automation and configuration of agricultural vehicles. Furthermore, based on our experience, this paper also addresses future guidelines to be followed to reduce the gap for the inclusion of robotics technologies on agricultural fields.

II. TRENDS OF AUTONOMOUS VEHICLES FOR AGRICULTURE

A. Automation of conventional vehicles

The tractor is the vehicle *par excellence* for the execution of most of the work required in a crop field, given its robustness and versatility. With the proper tools, this machine can till, plant, fertilize, spray, haul, mow, and even harvest. Such adaptability makes this vehicle a prime target for automation, enabling the productivity to be increased as well as improving safety and reducing operational costs. Fig. 1 presents an example of the technologies and requirements for the automation of agricultural tractors. An example of a hybrid control architecture is presented Fig. 1a, where the different levels of control and communications are illustrated (in red for low-level control −steering and speed control−; in green for the communications between the tractor and the implement; and in purple for high-level perception and sensor systems). Fig. 1b presents an example of the basic systems (actuators and sensors) to enable a tractor to be completely autonomous.

Several attempts to automate diverse types of tractors have been investigated and performed around the world in previous decades. At the University of Illinois, USA [19], a guidance system for an autonomous tractor based on sensor fusion was developed in 1998, including machine vision, RTK-GPS, and geometric direction sensor (GDS), achieving a lateral average error of approximately 8.4 cm at approximately 2.3 m/s. In 2006, on the Department of Agricultural and Biological Engineering, University of Florida, USA [20] an autonomous guidance system for use in a citrus grove was developed based on machine-vision (guidance average error of approximately 2.8 cm) and Laser Radar (LADAR guidance average error of approximately 2.5 cm.) and was tested in a curved path at a speed of approximately 3.1 m/s.

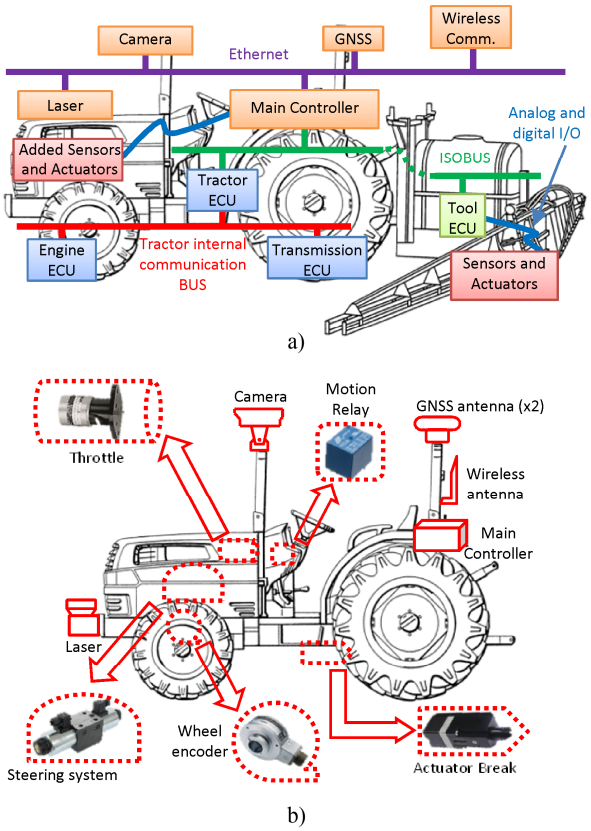


Figure 1. Example of agricultural tractor automation. a) Schematic diagram of the control elements and communication interfaces (ECU for Electronic Control Unit). b) Schematic diagram of the distribution of the diverse sensorial and actuation systems for adapting any agricultural tractor into an autonomous robot.

Other developments of automation of agricultural tractors can be found in Table I. Some research work has gone a step further and has integrated autonomous vehicles with automated tools. One relevant example is the work conducted by Nørremark [21], where an automatic intra-row weed control system [22] was connected to an unmanned tractor [23], linked via a hydraulic side-shifting frame attached to the rear three-point hitch of the vehicle. Currently, several companies offer the automation of conventional agricultural tractors, such as ASI (NAV), ATC (AUTODRIVE™) and Precision Makers (X-PERT). Essentially, these systems are installed in tractors owned by farmers, and they generally consist of a computer (the controller), a device for steering control, a localization system (mostly based on RTK-GPS) and a safety system (mostly based on a LIDAR). Most of these systems are compatible only with the most advanced tractors that feature ISOBUS (ISO 11783: Tractors and machinery for agriculture and forestry — Serial control and communications data network) control technology. Thus, the controllers connected to the ISOBUS are able to access other subsystems on the tractor (throttle, brakes, auxiliary valves, power take-off, linkage, lights, etc.). Another shortcoming is their lack of intelligence in solving problems, especially upon the detection of obstacles, given that they are not equipped with the proper technology to characterize and identify the type of obstacle and to estimate their possible behavior.

TABLE I. SOME EXAMPLES OF TRACTOR AUTOMATION AROUND THE WORLD

Author, year	*Country*	*Navigation technology and error obtained*	*Application*
O'Connor et al. [24], 1996	USA	CDGPS-based system. Speed: 0.9 m/s; heading error: 1°; lateral error: 2.5 cm	Tractor guidance following curved path using a unique Carrier Phase Differential GPS (CP-DGPS) sensor
Thuilot et al. [25], 2001	France	CDGPS-based system. Speed: 1.7 m/s; lateral error: 60 cm	
Blackmore et al. [23], 2004	Denmark	RTK-GPS and steering potentiometers; lateral error: 10 cm	Automatic steered tractor capable of following a predefined route plan
Bergerman et al. [26], 2012	USA	Laser sensors, driving and steering wheel encoders. Speed 0.9 – 1.3 m/s	Self-driving orchard vehicle for tree pruning and training, blossom and fruit thinning, fruit harvesting, mowing, spraying, and sensing.
Kayacan et al. [27], 2015	The Netherlands	RTK-DGPS. Speed: 0.6 m/s; lateral error: 40 cm	Tractor guidance using model predictive control for yaw dynamics

This information is essential for defining any behavior other than simply stopping and waiting for the situation to be resolved by itself. A shortcoming of this approach is that the conventional configuration of a standard tractor driven by an operator is designed to maximize the productivity per hour, and thus, the general architecture of the system (tractor plus equipment) is only roughly optimized. A conventional tractor must cover a wider range of equipment with the same fixed unit of a given power/weight. The complexity of the conventional tractor is also affected by operator needs with regard to ergonomics and safety, and the resultant productivity is related to the needs of the operator in terms of tiredness and the limited number of working hours in a day.

B. Specialized mobile platforms

The second approach to the introduction of mobile robots in agriculture is the development of specialized autonomous vehicles, where the researchers develop mobile platforms more like robots than tractors. One of the most promising robotic platforms under development is the BoniRob [28], a multi-purpose robotic platform for agriculture applications consisting of four independently steerable drive wheels and capable of adjusting its track width, making this platform adaptable to different scenarios in the agricultural field. The platform is equipped with the sensorial systems commonly used in robotic applications in agriculture, such as Lidar, inertial sensors, wheel odometry and GPS. Moreover, the robotic platform can be retrofitted and upgraded with exchangeable application modules or tools for crop and weed identification, plant breeding applications and weed control.

Nevertheless, this robot configuration limits the intervention area to the space below the robot, restraining its flexibility and versatility. Furthermore, given the robot morphology, this configuration presents a disadvantage when operating in irregular areas with considerable slopes or with the presence of ditches and gully erosion, where high stability and rollover safety is required. Moreover, articulated wheeled robots (AWR) present some drawbacks regarding the control of redundantly actuated systems, which exhibit complex interactions with the environment, limitations on the operational speed, and complex joint designs regarding control systems and brakes, making the control motion much more difficult than conventional wheeled mobile robots. Another drawback of this robot platform is that it is completely powered by electrical power, which diminishes its autonomy capacity regarding operational working time compared to conventional combustion systems. Other examples of mobile platforms under development that focus on performing precision agricultural tasks, are Rippa [29], Ladybird [30], Kongskilde Vibro Crop Robotti [31], and AgBot II [32]. Table II shows a summary of the diverse robotic platforms. These robots are oriented to fertilizing, seeding, weed control and gathering information and have similar characteristics between each other in terms of weight, load capacity, operation speed and morphology. Fig. 2 illustrates some pictures of those platforms. Tools, instrumentation equipment and intervention mechanisms are basically under the robots, and the task is performed in the area just below the robot, thereby limiting the maximum area of intervention. This feature also prevents these robots from intervening in farmlands with considerable (medium to high)

TABLE II. EXAMPLE OF SEVERAL SPECIALIZED PLATFORMS.

Vehicle	**Applications**	**Comments**
BoniRob [28]	Crop and weed identification, plant breeding, weed control	– It has four independently steerable wheels capable of adjusting its track distance to the crops – Its intervention area is limited to the space below the robot, limiting its flexibility and versatility. – It must control redundant actuated systems – It is powered only by batteries, which diminishes its working time
Rippa [29]	Fertilization, seeding, weed control and gathering of information	– The tools, sensors and intervention mechanisms are under the robots – The tasks are performed in the area just below the robot, limiting the maximum area of intervention. – They present difficulties working in farmlands with medium to high slopes, ditches or in the presence of gully erosion.
Ladybird [30]		
Vibro Crop Robotti [31]		
AgBot II [32]		
Cäsar [33]	Pest and soil management, fertilization, harvesting and transport	– It lacks flexibility, adaptability, robustness and intelligence to cope with diverse scenarios – Its safety features are basic and unintelligent (unable to reschedule or solve the problem by itself after detecting an obstacle)
Greenbot [34]	Fruit, horticulture and arable farming, urban sector, waterfronts, roadsides	– It lacks flexibility, adaptability, robustness and intelligence to cope with diverse scenarios – It stops functioning when it encounters unknown obstacles in its paths – It does not possess any detection system for weed or soil identification

slopes or in the presence of gully erosion. Moreover, two examples of commercial autonomous vehicles are the fruit robot "Cäsar" (Raussendorf Maschinen [33], Germany) and the Greenbot (Precision Makers [34], The Netherlands). The Cäsar is a remote-controlled special-purpose vehicle that can perform temporarily autonomous operations in orchards and vineyards. Pest management, soil management, fertilization, harvesting and transport are the tasks that can be performed in this specific environment. Meanwhile, the Greenbot is a self-driving machine that has been specially developed for all professionals in the agricultural and horticultural sectors who perform work tasks that are regularly repeated. This vehicle can be used for fruit farming, horticulture and arable farming but also in the urban sector and even at waterfronts or on roadsides. Nevertheless, these systems lack flexibility, adaptability, robustness and intelligence to cope with diverse scenarios, and their safety features are basic and unintelligent. For example,

- they are focused only on orchard and vineyard activities (Cäsar) or exhibit limitations with respect to ground clearance (Greenbot);
- they cease functioning when they encounter unknown obstacles in their paths and are unable to recognize or interpret what is happening, and thus, they are not capable of rescheduling or solving the problem by themselves (Cäsar and Greenbot);
- they must be manually guided to the working area rather than being capable of freely and autonomously moving to different working areas around the farm (Cäsar) and
- they do not possess any advanced detection system for weed or soil identification, thereby limiting their use to previously planned tasks related to selective treatment (Greenbot).

III. FLEETS OF ROBOTS

Although the scientific and technological bases of PA are mostly known and robust [35], the commercial application of these new technologies is still very limited. To overcome this situation, researchers have used existing Information and Communication Technologies (ICT) to design and build improved weed and crop sensors, enhanced actuators to perform proper pest control and autonomous mobile platforms to accurately move those sensors and required actuators all over the working field. One example is the RHEA project (Robot Fleets for Highly Effective Agriculture and Forestry Management, founded by the FP7 programme) [36][37], which consisted of a fleet of small/medium-sized mobile robots equipped with perception systems and agricultural tools to perform weed management tasks in cereal crops, wide-row crops and woody perennials [38] (see Fig. 3). These vehicles were a new generation of robots for effective chemical and mechanical management of a large variety of crops with the purpose of minimizing the use of agricultural inputs and decreasing environmental pollution while improving crop quality and safety and reducing costs.

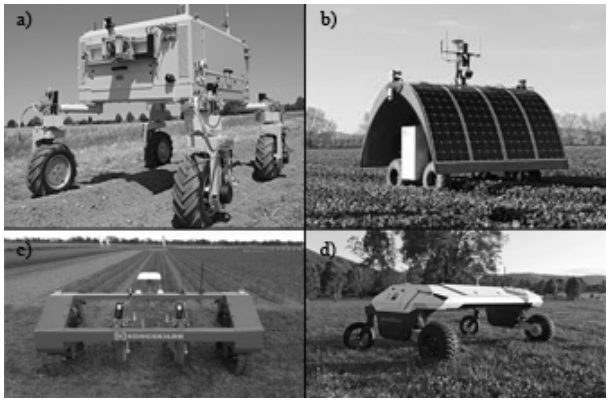


Figure 2. Examples of several agricultural platforms. a) BoniRob; b) Ladybird; c) Kongskilde Vibro Crop Robotti; and d) AgBot II.

To accomplish this aim, RHEA conducted research in (1) advanced perception systems to detect and identify crop status, including crop row detection, and (2) innovative actuation systems to apply fertilizers and herbicides precisely as well as to remove or eliminate weeds directly. Additional research was focused on the development of (3) a fleet of small, safe, reconfigurable, heterogeneous, and complementary mobile units to guarantee the application of the procedures to the entire operation field. This scientific activity was complemented with technical developments in (4) novel communication and location systems for robot fleets, (5) enhanced simulation systems and collaborative graphic user interfaces, and (6) pioneering fuel cells to build clean and efficient energy sources. This fleet of robots has proven to be better than traditional sized vehicles regarding productivity (better adaptation to small fields than large vehicles), safer (less mass, less physical impact) and fault tolerance (a failure in a robot does not stop the mission). Furthermore, they produce less ground compaction (less mass, less effect on the ground) and require a smaller workforce (an operator can supervise several robots at the same time). There are some related projects that have already been completed or are still under development, as in the cases of the European projects FutureFarm and Flourish, respectively.



Figure 3. The RHEA fleet (ground mobile units and implements).

FutureFarm (Meeting the challenges of the farm of tomorrow by integrating Farm Management Information Systems to support real-time management decisions and compliance to standards, founded by FP7 programme) [39] was focused on coordinating general aspects in agriculture farms. The project's main aim was to increase farm competence and integrate goods provided by farming into management strategies. Alternatively, the Flourish project (Aerial Data Collection and Analysis, and Automated Ground Intervention for Precision Farming, founded by H2020 programme) [40] is focused on bridging the gap between the current and desired capabilities of agricultural robots by developing an adaptable robotic solution for precision farming.

The aforementioned projects have in common the use of advance information for fleet management in agricultural scenarios, allowing a significant reduction in time and energy usage. Small vehicles ensure higher positioning accuracy during operation and are intrinsically lighter than big machines. This last feature reduces the soil compaction and makes the vehicles safer in terms of safety to others, vehicle safety and crop safety, all important features in agricultural equipment currently [41]. However, small robots manage smaller implements and payloads than big machines. Therefore, several small robots are needed to accomplish tasks that are similar to what one big machine can manage. This raises the concept of fleets of robots with additional advantages regarding price (it allows farmers to get high-technology equipment in an increasing manner), fault tolerance (failure in a small robot means one less robot at work, while failure in a big vehicle means the entire process on the field is stopped), mission coordination and reconfiguration (being able to change the fleet behavior at any time to optimize the mission, taking into account sudden changes in field conditions), etc.

IV. Discussion

Currently, it is still not clear which of the two tendencies could be the most suitable for introducing robotics into agriculture in regard to configuring a mobile robot for PA tasks (automated tractors or specialized platforms). What many authors agree on and what has also been demonstrated practically in the RHEA project is that the future of PA is the deployment of a multi-robot configuration to perform the most arduous tasks that require a greater precision and a better use of the resources.

Given that agriculture is a very broad economic area where many diverse goods are produced, some requiring more care and more agricultural inputs (in quantity applied) than others, it is clear that robotics for agriculture must follow a line of modularity, flexibility and adaptability. This is not new; for example, according to the strategic policies for both robotics and agricultural research in the EU defined by The Partnership for Robotics in Europe (SPARC) and the EU's Standing Committee on Agricultural Research (SCAR), the challenge in the coming years is to develop robots that respond more flexibly, robustly and efficiently to the everyday needs of workers and citizens in professional or domestic environments. However, today, there is a long way to go both to manage all the information needed to operate a fleet of robots and to define the best configuration of that fleet. Some elements that still require reinforcing of their robustness and reliability are the following:

- *The size and morphology of each mobile robot:* automated tractors have proven to be more adaptable for any agricultural task given their standards for mechanical connections —three-point hitch (TPH)— and communications —ISOBUS— and have a higher work rate, because they are able to cover a larger area of application. However, specialized platforms have proven to be more eco-friendly (less use of fuel or totally electric) and generate less soil compaction, and given their size, they are less likely to generate extensive damage (increased safety). The selection of one approach or another may be conditioned to the type of task to be performed and how profitable it can be, which has not been adequately demonstrated.

- *Type and level of precision of intervention tools*: the benefits of not treating the crop field as a homogeneous environment but rather identifying both its spatial and temporal variabilities have been demonstrated for the efficient usage of agricultural inputs. Nevertheless, the identification of the field and crop needs is an area of research that still requires a step change, which must be accompanied by the ability of the intervention mechanism or implements to act accurately.

- *Standards in communication*: although there are communication standards specially focused on agriculture machinery, such as the ISOBUS, there is still a long way to go to manage the data obtained in the crop field, to interpret these data, and to make them available for better decision-making. Farm Management Information Systems are tools that can contribute to the integration of data, including finances, economics, weather forecasts, soil-structures and requirements, and field historic behavior. Farmers need clear proof that robotic systems can be profitable, and obtaining and managing the data to demonstrate such benefits requires a further advance in the standardization of the collected information, both outside and inside the crop field.

- *Level of autonomy*: one of the great steps that robotics in agriculture must take is to ensure a high level of autonomy, almost eliminating the intervention of a human operator (to reach the highest level of automation according to the 10-level taxonomy developed by Endsley [42]). All the robotic systems presented in this paper require a direct intervention by an operator to move the robotic system to the working area or to solve a basic safety situation (such as encountering an obstacle). Systems of obstacles identification and classification, predictors of behavior of mobile obstacles (such as people, animals or vehicles), and perception systems not prone to changes in light conditions are necessary for the incorporation of robotics in the day-to-day operation of a farm.

V. Conclusion

The technologies that will allow an increasing number of robots to work in crop fields are now available. Steps should

still be taken to integrate such technologies, and the definitions of the best configurations for each of the tasks in crop fields remain to be identified. Demonstrating that robotic systems are cost-effective is one of the important steps that must be taken by researchers, but the acquisition and management of field data is necessary and still requires more progress and standardization.

ACKNOWLEDGMENT

The research leading to these results received funding from the CSIC under the AutoFarm project.

REFERENCES

[1] Transparency Market Research, "Global Mobile Robotics Market to Develop at 12.6% owing to Increasing Government Support," 2015. [Online]. Available: http://www.transparencymarketresearch.com/pressrelease/mobile-robotics-market.htm. [Accessed: 03-May-2017].

[2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Science & Business Media, 2008.

[3] N. Zhang, M. Wang, and N. Wang, "Precision agriculture—a worldwide overview," *Comput. Electron. Agric.*, 2002, pp. 113–132.

[4] F. Rovira-Más, I. Chatterjee, and V. Sáiz-Rubio, "The role of GNSS in the navigation strategies of cost-effective agricultural robots," *Comput. Electron. Agric.*, 112, 2015, pp. 172–183.

[5] G. Vellidis, B. Ortiz, J. Beasley, R. Hill, H. Henry, and H. Brannen, "Using RTK-based GPS guidance for planting and inverting peanuts," *Precision agriculture'13*, J. V. Stafford, Ed. Wageningen Academic Publishers, 2013, pp. 357–364.

[6] H. J. Heege, "Precision in Guidance of Farm Machinery," *Precision in Crop Farming*, Ed. Springer Netherlands, 2013, pp. 35–50.

[7] M. Vázquez-Arellano, H. W. Griepentrog, D. Reiser, and D. S. Paraforos, "3-D Imaging Systems for Agricultural Applications—A Review," *Sensors*, 16(5), p. 618, Apr. 2016.

[8] S. Haug, P. Biber, A. Michaels, and J. Ostermann, "Plant Stem Detection and Position Estimation using Machine Vision," In *Proceedings of the International Workshop on Recent Advances in Agricultural Robotics (RAAR2014)*, Padova, Italy, May 2016.

[9] D. C. Slaughter, "The Biological Engineer: Sensing the Difference Between Crops and Weeds," in *Automation: The Future of Weed Control in Cropping Systems*, S. L. Young and F. J. Pierce, Eds. Springer Netherlands, 2014, pp. 71–95.

[10] A. Kaloxylos et al., "Farm management systems and the Future Internet era," *Comput. Electron. Agric*., 89, 2012, pp. 130–144.

[11] R. Nikkilä, I. Seilonen, and K. Koskinen, "Software architecture for farm management information systems in precision agriculture," *Comput. Electron. Agric.*, 70(2), pp. 328–336, Mar. 2010.

[12] C. G. Sørensen, S. Fountas, E. Nash, L. Pesonen, D. Bochtis, S. M., Pedersen, and S. B. Blackmore, "Conceptual model of a future farm management information system," *Comput. Electron. Agric.*, 2010, pp. 37–47.

[13] P. C. Robert, "Precision agriculture: a challenge for crop nutrition management," *Plant Soil*, 247(1), 2002, pp. 143–149.

[14] P. K. Thornton, R. H. Fawcett, J. B. Dent, and T. J. Perkins, "Spatial weed distribution and economic thresholds for weed control," *Crop Prot.*, 9(5), pp. 337–342, Oct. 1990.

[15] C. E. Thorpe, "Outdoor visual navigation for autonomous robots," *Robot. Auton. Syst.*, 7(2), pp. 85–98, Aug. 1991.

[16] G. Seetharaman, A. Lakhotia, and E. P. Blasch, "Unmanned vehicles come of age: The darpa grand challenge," *Computer*, 39(12), 2006.

[17] M. Birdsall, "Google and ITE: The Road Ahead for Self-Driving Cars," *Inst. Transp. Eng. ITE J. Wash.*, 84(5), 2014, pp. 36–39.

[18] Clay Dillow, *Popular Science*, 2011. [Online]. Available: http://www.popsci.com/cars/article/ 2011-06/nevada-passes-driverless-car-legislation-paving-way-autonomous-autos. [Accessed: 03-May-2017].

[19] N. Noguchi, J. F. Reid, J. Will, E. R. Benson, and T. S. Stombaugh, "Vehicle automation system based on multi-sensor integration," *ASAE Pap.*, 983111, 1998.

[20] V. Subramanian, T. F. Burks, and A. A. Arroyo, "Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation," *Comput. Electron. Agric.*, 53(2), 2006, pp. 130–143.

[21] M. Nørremark, H. W. Griepentrog, J. Nielsen, and H. T. Søgaard, "The development and assessment of the accuracy of an autonomous GPS-based system for intra-row mechanical weed control in row crops," *Biosyst. Eng.*, 101(4), 2008, pp. 396–410.

[22] E. Wisserodt et al., "Gesteuerte Hacke zur Beikrautregulierung innerhalb der Reihe von Pflanzenkulturen. [Controlled Hoe for Weeding within Crop Rows]," in *Proceedings of the VDI-Tagung Landtechnik Braunschweig*, Düsseldorf, 1999, pp. 155–160.

[23] B. Blackmore, H. W. Griepentrog, H. Nielsen, M. Nøremark, and J. Resting-Jeppesen, "Development of a deterministic autonomous tractor," in *Proceedings of the International Commission of Agricultural and Biosystems Engineering*, Beijing, China, 2004.

[24] M. O'Connor, T. Bell, G. Elkaim, and B. Parkinson, "Automatic steering of farm vehicles using GPS," in *Proceedings of the 3rd International Conference on Precision Agriculture*, Minneapolis, USA, 1996, pp. 767–777.

[25] B. Thuilot, C. Cariou, L. Cordesses, and P. Martinet, "Automatic guidance of a farm tractor along curved paths, using a unique CP-DGPS," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2, 2001, pp. 674–679.

[26] M. Bergerman, S. Singh, and B. Hamner, "Results with autonomous vehicles operating in specialty crops," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2012)*, 2012, pp. 1829–1835.

[27] E. Kayacan, E. Kayacan, H. Ramon, and W. Saeys, "Towards agrobots: Identification of the yaw dynamics and trajectory tracking of an autonomous tractor," *Comput. Electron. Agric.*, 115, pp. 78–87.

[28] A. Ruckelshausen et al., "BoniRob–an autonomous field robot platform for individual plant phenotyping," *Precis. Agric.*, 9, 2009.

[29] Robert Bogue, "Robots poised to revolutionise agriculture," *Ind. Robot Int. J.*, 43(5), pp. 450–456, Aug. 2016.

[30] J. P. Underwood, M. Calleija, Z. Taylor, C. Hung, J. Nieto, R. Fitch, and S. Sukkarieh, "Real-time target detection and steerable spray for vegetable crops," in *Proceedings of IEEE ICRA, workshop on robotics in agriculture*, 2015.

[31] Kongskilde [Online]. Available: http://www.kongskilde.com/. [Accessed: 03-May-2017].

[32] O. Bawden, D. Ball, J. Kulk, T. Perez, and R. Russell, "A lightweight, modular robotic vehicle for the sustainable intensification of agriculture," in *Australian Conference on Robotics and Automation (ACRA 2014)*, University of Melbourne, December 2014.

[33] Raussendorf Maschinen, 2016. [Online]. Available: http://www.raussendorf.de/en/fruit-robot.html. [Accessed: 09-Jun-2016].

[34] Precision Makers, "Greenbot," 2016. [Online]. Available: http://www.precisionmakers.com/greenbot/. [Accessed: 09-Jun-2016].

[35] A. Srinivasan, "Handbook of precision agriculture: principles and applications," in *Food Products Press New York*, NY, USA, 2006.

[36] RHEA, 2017. [Online]. Available: http://www.rhea-project.eu/. [Accessed: 09-May-2017].

[37] P. Gonzalez-de-Santos, A. Ribeiro, and C. Fernandez-Quintanilla, "The RHEA Project: using a robot fleet for a highly effective crop protection," in *Proceedings of the International Conference of Agricultural Engineering, CIGR-Ageng 2012*, Valencia, Spain, 2012.

[38] L. Emmi, M. Gonzalez-de-Soto, G. Pajares, and P. Gonzalez-de-Santos, "Integrating Sensory/Actuation Systems in Agricultural Vehicles," *Sensors*, 14(3), 2014, pp. 4014–4049.

[39] FutureFarm, 2017. [Online]. Available: http://www.futurefarm.eu/about. [Accessed: 03-May-2017].

[40] "Flourish Project." [Online]. Available: http://flourish-project.eu/. [Accessed: 03-May-2017].

[41] B. S. Blackmore, H. Have, and S. Fountas, "A specification of behavioural requirements for an autonomous tractor," in *Proceedings of the 6th International Symposium on Fruit, Nut and Vegetable Production Engineering conference*, Germany, Institute für Agrartechnik Bornim e.V, 2001, pp. 25–36.

[42] M. R. Endsley, "Level of automation effects on performance, situation awareness and workload in a dynamic control task," *Ergonomics*, 42(3), 1999, pp. 462–492.

Human Robot Motion: A shared effort approach

Grimaldo Silva[1] and Thierry Fraichard[1]

Abstract— **This paper is about Human Robot Motion (HRM), i.e. the study of how a robot should move among humans. This problem has often been solved by considering persons as moving obstacles, predicting their future trajectories and avoiding these trajectories. In contrast with such an approach, recent works have showed benefits of robots that can move and avoid collisions in a manner similar to persons, what we call human-like motion. One such benefit is that human-like motion was shown to reduce the planning effort for all persons in the environment, given that they tend to solve collision avoidance problems in similar ways. The effort required for avoiding a collision, however, is not shared equally between agents as it varies depending on factors such as visibility and crossing order. Thus, this work tackles HRM using the notion of motion effort and how it should be shared between the robot and the person in order to avoid collisions. To that end our approach learns a robot behavior using Reinforcement Learning that enables it to mutually solve the collision avoidance problem during our simulated trials.**

I. Introduction

Human Robot Motion (HRM) is the study of how a robot should move among persons. In this context, robot motion must be safe and appropriate. While safety relates to guaranteeing collision-free motion [1], the term appropriate relates to respecting concepts such as social spaces [2], legibility and perceived safety [3].

Many recent studies have focused on tackling HRM by teaching a robot human-like behavior, such as in [4] and [5]. The justification for this approach is that it allows a robot to follow the flow of the persons [4], and also allows for better behavior legibility to persons around the robot. Legibility is important because it was shown that persons tend to solve collision avoidance problems in stereotypical ways under repeated conditions [6], which implies that a robot behaving in an uncommon way forces the person to actively plan its motion instead of relying on already learned motion plans, this means that human-like motion reduces planning effort for all the persons in the environment [7]. Furthermore, another argument is that unexpected motions can be perceived as unsafe by nearby persons even though in practice they may be collision free [5].

In order to create human-aware robots capable of navigating among persons, most current approaches in HRM, such as [8] and [9], operate in two steps. First the probable future behavior of the persons is predicted without considering the robot. Then the future robot motion is computed taking this prediction into account. As a result, the robot always yields, that is, it avoids to the best of its ability regions where a person is expected to go through. Collision avoidance among persons is, however, mutually solved [10]. This means that, depending on the current disposition of nearby persons, each person is expected to contribute a certain amount of what we call *effort* to avoid a collision. The amount of effort expected from each person and in which manner this effort is represented, as speed or path changes for example, depends on many factors [10], [11], [12], such as: who is first, angle of approach, speed and visibility.

In order to replicate human collision avoidance behavior, our objective is to allow the robot to share collision avoidance effort with people, when necessary, in a safe and appropriate manner, that is, in a way that is expected by its human peers. To that end our approach accounts for two facts: visibility and crossing order. The former represents the understanding of the robot regarding what nearby persons can see, while the latter represents which agent in crossing scenarios should give way to the other. Note, however, that in situations where the person is unwilling or unable to follow a stereotypical motion the robot in our approach will still be able to take full responsibility for avoiding collisions. An important aspect is how the effort needs to be shared between persons and robot. In some situations the person does not expect the robot to yield, such as when the person is behind the robot but intending to overtake. Whereas in other cases the person expects the other agent to give him priority and also to be responsible for most of the collision avoidance [10], as is the case when the front of the robot would collide into the side of a person during perpendicular crossing scenarios.

Predicting human behavior in reaction to a given robot motion in our approach depends on a human-like model (HLM), which unlike many works in HRM such as [8] and [5] does not use the Social Force Model (SFM) which was introduced in [13]. Instead we rely on a slightly modified version of Optimal Reciprocal Collision Avoidance (ORCA) [14], also called RVO 2. This HLM was chosen as it can be directly modified to accommodate different degrees of participation from a particular agent during collision avoidance.

Based on the persons' reaction to a given robot motion, we intend to use this information to avoid collisions with persons in a human-like way. To that end, our approach relies on reinforcement learning (RL) [15] to learn such behaviors, this technique was chosen for its ability to explore the state space and also to learn behaviors that can be recalled even in real-time situations [16].

[1] Grimaldo Silva jose.jgrimaldo@gmail.com and Thierry Fraichard thierry.fraichard@inria.fr are with Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

A. Outline of the Paper

This work is divided into six sections. Section II describes works with related concepts. Afterwards, in Sec. III a formal description of our approach is presented and also how to measure the additional effort required for collision avoidance. This is followed by Sec. IV where this additional effort measure is used to build a human-like collision avoidance strategy. Experimental results of our approach are presented in Sec. V. Finally, a discussion of our results, future works and final remarks are presented in Sec. VI.

II. Background and Contributions

Initial concepts in HRM focused mainly on allowing a robot to respect social spaces, which can be defined in a general sense as regions that for whatever reason a person considers as belonging to them [2].

There are many other concepts that have an influence in HRM, such as comfort. Comfort relates to the subjective feeling of a person that the body is relieved of negative stimuli [17]. Many factors affect comfort, one such factor is the visibility which has been tackled in [17] using a multi-layer costmap that factors the cost of visibility into a costmap in order to calculate the optimal trajectory of an autonomous wheelchair. A definition of comfortable motion that is more related to HRM was made in [5], it can be summarized as the perception of a person being able to walk in their preferred velocity and if their path felt collision free.

Among the several human-like models (HLM) that can approximate human behavior in these cases, we highlight the Extended Social Force Model [13], a method based on modeling each person as being attracted to their goal (in a preferred velocity) and being repulsed by other agents and also static objects in the environment. Another tool used in simulation of pedestrians, particularly in crowd simulation [18], [19], is the reciprocal velocity objects (RVO) [14] which is based on finding velocity choices for agents that guarantee collision avoidance.

Given one such HLM, its possible to calculate the reaction of a given person to a robot motion. This contrasts with many current approaches where the planned human motion is static [8] or probabilistic [9]. That is, in these works the robot avoids regions where persons are predicted to go in order to avoid disrupting their plans.

Another concept, defined in [9], was hindrance. This term relates to situations where a person natural behavior is disrupted due to a robot's proximity. To that end, a human-like planner using Markov Decision Process associates a probability for each of the several possible person trajectories to the goal (a distribution over trajectories), this planner is trained by observing human trajectories. Thus, a robot is able move towards its goal while avoiding high hindrance regions.

Our approach brings novel contributions in relation to those works as we focus on reproducing how persons share collision avoidance effort. To this end, it is necessary to forecast short-term human motion plan in reaction to a given robot action, which we accomplish with a modified ORCA.

III. Overview of the problem

A robot is tasked with reaching a given goal, in-between his current and desired positions any number of persons may cross his path. It is evident that collisions with persons have to be avoided whenever possible. However, persons have certain expectations about how this collision avoidance should take place. To solve this problem it is important to model how the collision avoidance effort should be distributed.

A. Formalization of the problem

Consider that W represents the environment, with $W \subset \mathbb{R}^2$. In this environment, each person p and robot r that belong to the set of dynamic objects D have positional properties: $\boldsymbol{q}_p = (x_p, y_p, \theta_p) \in \mathbb{R}^2 \times S^1$. Thus we define the state of a given person as $\boldsymbol{s}_p = (\boldsymbol{q}_p, \dot{\boldsymbol{q}}_p)$, where each person also has a goal, which is known a priori, $\boldsymbol{g}_p = (x_p^*, y_p^*, \theta_p^*) \in \mathbb{R}^2 \times S^1$. Additionally, the robot r is also an agent in this environment and as such also has positional properties s_r and a goal g_r.

Although human behavior can be the result of large cognitive effort, recent studies showed that realistic trajectories can be generated with simple models where an agent solely avoids local collisions [5]. Thus, our choice to utilize a reactive HLM to evaluate human reaction to a given robot motion over n time steps is reasonable.

One possible approach to the robot-person collision avoidance problem can be posed in terms of minimizing additional human effort. First, let $\boldsymbol{\pi}_{p,r} = \{\boldsymbol{q}_p(0), \ldots, \boldsymbol{q}_p(n)\}$ be the predicted trajectory of person p after interaction with a robot r trajectory within a prediction window of n time steps ahead. Moreover, consider that the additional effort of a given trajectory is represented by a value $\Gamma(\boldsymbol{\pi}) \to \mathbb{R}^*$ (detailed in Sec. III-B). Finally, consider one possible formulation to this problem

$$\boldsymbol{\pi}_{r*} = \underset{\boldsymbol{\pi}_r \in \Pi_r}{\arg\min} \sum_{p \in P} \Gamma(\boldsymbol{\pi}_{p,r}) \tag{1}$$

where Π_r is the set of admissible robot motions to the goal. In this model the robot avoids causing additional effort to the person whenever possible, that is, it will minimize the disruption of the person's motion plan while still reaching its goal. This approach is necessary in case the person is unaware of the robot or either unwilling or incapable of changing his motion plan. Conversely, in real scenarios, a person does not always yield. The additional effort required for collision avoidance is shared between the persons involved. In such context, a robot that acts unlike other persons can generate scenarios where, for example, persons are forced to actively think about the robot motion plan instead of relying on already learned stereotypical trajectories. As such, to achieve HRM it is also necessary for the robot to replicate the ability of persons to share necessary changes in planning between themselves in a socially aware manner in order to solve collision avoidance situations in stereotypical situations.

To account for the effort sharing between person and robot, the problem of collision avoidance is posed as an optimization problem in this manner

$$\boldsymbol{\pi}_{r*} = \underset{\boldsymbol{\pi}_r \in \Pi_r}{\arg\min} \sum_{p \in P} |(1 - \alpha_{r,p}) \cdot \Gamma(\boldsymbol{\pi}_{p,r}) - \alpha_{r,p} \cdot \Gamma(\boldsymbol{\pi}_r)| \tag{2}$$

where $\alpha_{r,p} \in [0, 1]$ is the effort distribution coefficient (EDC) between p and r. This coefficient indicates, at each time step, what is the relative cost of the robot's deviation from its baseline goal in relation to the person, a higher proportion engenders less deviation, this is detailed in the section IV.

B. Human trajectory cost function

Anticipating the human effort necessary to execute a given trajectory is a necessary step in order to properly divide effort between person and robot. Many models exist to measure this effort. One such function is the path length and also total time to the goal [20]. Another approach, is given by [21], which describes the cost of a trajectory as a combination of weighted acceleration controls.

Our work relies on the concept of understanding how collision avoidance requires additional effort in relation to the robot baseline motion. Baseline motion represents the trajectory that does not account for the presence of other agents in the environment. The interaction with other agents, however, requires change in the motion plan. To measure this change, the first step is calculating the distance of an agent r to the goal at time t using $d_t(r, g_r) = \sqrt{(x_r(t) - x_r^*)^2 + (y_r(t) - y_r^*)^2}$ where $x_r(t)$ and $y_r(t)$ are, respectively, the x and y coordinates of the agent r at time t. Thus, we can define the change in distance to the goal as $\Delta d_t(r, g_r) = d_t(r, g_r) - d_{t-1}(r, g_r)$. In our approach, at each time step, a baseline change in distance to the goal is estimated, that is, the agent plans its motion without accounting for other agents. This baseline change in distance to the goal at the current time step is represented by $\Delta \mathbf{B}_t(r, g_r)$ and can be understood as the desired progression to the goal.

However, interaction with other agents require additional effort, which impose changes into the baseline motion of an agent. Given this concept, we can define the additional effort of r for a given trajectory as

$$\Gamma(\boldsymbol{\pi}_r) = \sum_{t=1}^{n} \max\{0, \Delta d_t(r, g_r) - \Delta \mathbf{B}_t(r, g_r)\} \tag{3}$$

This cost function calculates its result based on the difference from the baseline motion to the actual motion. In this formulation, a given motion can only have an equal or smaller cost than the baseline motion at any time step. This definition guarantees that $\Gamma(\boldsymbol{\pi}) \to \mathbb{R}^*$, which is a property that is important in Sec. IV-B, when using it as part of a reward function during optimization.

IV. Presentation of the Approach

Given the initial state of the person and the robot (including position, goal and velocity), the robot wishes to find a trajectory $\boldsymbol{\pi}_{r*}$ that shares collision avoidance effort among them in a similar way as another person would. Thus, in this section we divide our approach to solve the optimization problem of shared effort presented in Eq. 1 and Eq. 2 in five main steps:

1) Receive information from sensors (world model/state)
2) Find $\forall p \in D$ the $\alpha_{r,p}$ based on current state
3) Plan collision avoidance actions up to n steps ahead
4) Send planned velocity (action) to wheels
5) Stop if goal reached, go to step 1 otherwise

As the robot receives input from its sensors it builds a representation of the world including position of the goal, position and velocity of nearby persons and also his own. This information can be used to generate what is called a model of its environment – a world model.

Information about position and velocity of nearby persons enables the robot to calculate the amount of effort it should share with each one for human-like collision avoidance. The effort distribution coefficient (EDC) and the steps necessary to calculate it are described in details in Sec. IV-A.

Given the world model and the EDC, the motion plans for future timesteps can be calculated. To that end, Reinforcement Learning (RL) is used to learn a motion plan capable of reaching a given goal while avoiding collision with a nearby person. Our formulation of this problem as RL problem is described in Sec. IV-B.

Based on this overview of our approach to solve the shared effort collision avoidance problem, in the upcoming subsections the aforementioned steps are detailed and some advantages and limitations of our approach are discussed.

A. Sharing effort

The proportion of effort shared during collision avoidance between person and a robot varies depending on crossing order and crossing angle. It is known that the person that is giving way has to contribute more to the avoidance than the one passing first [10]. One possible explanation for this comes from difference in visual stimuli that both agents have, as the person that gives way can more easily obtain visual information about the person passing first [10]. In our current formulation these two factors are taken into account to decide shared effort: crossing order and visibility.

The point of potential collision, which is the position where both agents would collide on in case they continue in their current velocity, forms an angle $\zeta_{r,p} \in [0, 2\pi]$ between the current position of the robot r and of person p. Henceforth, when analyzing angles of crossing scenarios, the angle that is being referenced is $\zeta_{r,p}$. Furthermore, the angle $\beta_{r,p}$ is formed from the bearing-angle of r in relation to the position of p. The derivative of the bearing angle $\dot{\beta}$ can be a strong indicator of potential collision and also of crossing order [22]. These angles are shown in Fig. 1.

Based on results found [10] through analysis of the perpendicular crossing scenarios, it was found that the person crossing first has a maximum of 40% contribution in collision avoidance effort, while the one crossing last has a maximum of 40%. Furthermore, it is intuitive that in most situations of head-on collision with similar velocities or when both person and robot see each other but have no clear crossing order,

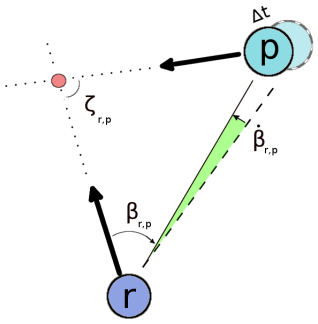


Fig. 1: Collision situation between r and p, where crossing angle ζ, bearing angle β and its derivative $\dot{\beta}$ are shown.

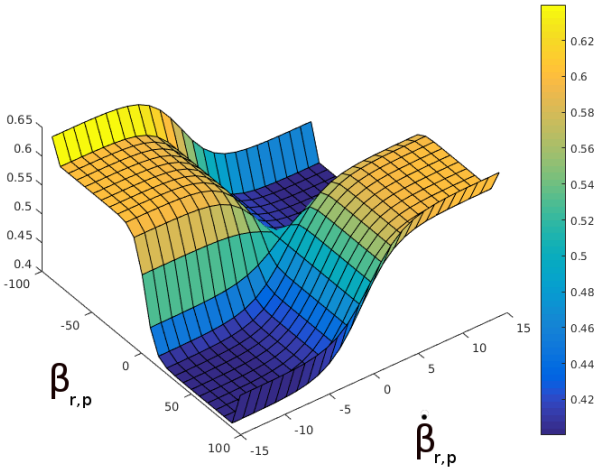


Fig. 2: Shared effort space defines $\alpha_{p,r}$ (both axis in degrees). Its value indicates the relative cost of the robot's deviation from its baseline motion in relation a person's deviation.

the effort is shared equally between participants. Conversely, in scenarios where one agent is potentially unaware of the other i.e. the passing agent is coming from behind; the responsibility shifts to the agent that sees the other. Recent results also indicate that agents are still able to avoid collisions against obstacles in peripheral vision [23].

This background allows us to correctly distribute effort during collision avoidance between a person and a robot. Thus let $\alpha_{r,p}$ represent the effort sharing coefficient between r in relation to p, which we define as a proportion that weights crossing order and visibility into the relative cost of the robot's deviation from its baseline motion in relation to the person. That is, the higher the proportion, the less deviations from baseline motions of the robot are done in comparison to the person.

The notion that agents do not react to other agents that are outside their field of view, which span around 180º (with both eyes) when looking ahead [24], is translated into our model as a function $\Psi : \mathbb{R} \rightarrow [0, 1]$. This model is used for the robot in order to find trajectories that respect humans expectations. Thus, Ψ is defined as

$$\Psi(\beta_{r,p}) = \begin{cases} 0 & \text{for } |\beta_{r,p}| \geq \frac{\pi}{2} \\ 1 - e^{-\lambda_1(|\beta_{r,p}| - \frac{\pi}{2})} & \text{otherwise} \end{cases} \quad (4)$$

where λ_1 is 15. Based on this model of visibility, the shared effort coefficient of r in relation to p that also accounts for the passing order can be defined as

$$\alpha_{r,p} = (1 - \Psi(\beta_{r,p})) + (0.5 + f(\beta_{r,p})) \cdot \Psi(\beta_{r,p}) \quad (5)$$

$$f(\beta_{r,p}) = \text{sgn}(-\beta_{r,p})(1 - \delta(\beta)) \left(A + \frac{K - A}{1 + \exp(-\lambda_2 \dot{\beta}_{r,p})} \right) \quad (6)$$

where the constants A, K and λ_2 are, respectively, 0.1, -0.1 and 30. Furthermore, $\dot{\beta}$ is the rate of change of β, sgn is the standard sign function that extracts the sign of a real number and $\delta : \mathbb{R} \rightarrow [0, 1]$ is a function that resembles a smooth approximation of the dirac delta distribution that maps β into $1 - |\tanh(\lambda_3 \beta)|$ in which $\lambda_3 = 8$ was chosen to appropriately control the rate of convergence from one to zero. The dirac-like distribution was used to guarantee that the effort is always shared evenly during head-on (or near head-on) collision scenarios. Additionally, a generalized logistic function represents the boundary between the head-on collision avoidance case and the perpendicular case (where there may be an unequal distribution of effort).

The function f, showcased in Fig. 2, is not applied in cases where there is no chance of collision, as there is no need to change its motion plan, or in cases where the person does not see the robot. In the latter case, for example, if a robot is trying to pass a person from behind it is not appropriate to expect the person to share effort with the robot as the robot is outside its field of view. Thus, in both cases the robot is responsible for the total motion effort.

B. Human-like collision avoidance

To correctly share effort between a person and a robot the optimization problem defined in Sec. 2 is presented in this section in a way can be solved using Reinforcement Learning [15]. The most usual way to represent reinforcement learning problems is as a Markov Decision Process (MDP) which defines a tuple containing $\langle Z, A, R, P \rangle$ that are, respectively, the set of possible states Z, the set of possible actions A, the reward function $R : Z \times A \times Z \rightarrow \mathbb{R}$ and also a transition function $P : Z \times A \rightarrow Z$. At each discrete time step the MDP observes the current state $z_0 \in Z$ and selects an action $a_0 \in A$, as a result, it reaches a new state z_1 and receives a reward r_1. Given this formulation, the goal of the MDP is to reach a given terminal state s_f with the best expected reward or maximize the expected reward within a certain time frame.

A particular robot behavior, that is, a relation between every state and action is defined as $\psi : Z \rightarrow A$ and called policy. The goal of a reinforcement learning is thus to learn a policy ψ^* that provides better reward than any other policy. Among available methods of Reinforcement Learning, TEXPLORE [16] was selected as our choice as it is robust to noise and able to handle continuous state features.

In order for ψ to make a decision about the future

robot motion, the robot represents its own internal state and the state of nearby persons into a form that can be used in RL. As such, its RL state, defined as z_t, is a tuple $\langle \beta_{r,g}, d_{r,g}, \zeta_{r,p}, \text{ttc}, \beta_{r,p}, \dot{\beta}_{r,p}, d_{r,p} \rangle$ that is used a person where its current motion has risk of collision with the robot, where ttc represents the number of time steps to collision (up to n steps ahead) given linear projection of current velocities, and $\dot{\beta}$ is the rate of change of the bearing angle (see Fig. 1). One limitation of this state space formulation is that it only allows for shared effort in the one person and one robot scenario, given that adding more persons would require an unbounded number of new states to the state space, according to the number of people in the environment.

Using the relative angle and distance to the goal allows the robot to learn what actions better leads him to the goal. For instance, in the absence of collision risk, maintaining the bearing angle of the robot to the goal, $\beta_{r,g}$, at near zero guarantees the reward is maximum. In a similar sense $\dot{\beta}$ is used to allow the agent to measure the risk of collision, the direction of the collision is given by $\beta_{r,p}$ and $\zeta_{r,p}$. When collision is detected within the visible range the ttc is set to the predicted amount of time steps, its value is an arbitrary maximum distance of collision detection otherwise.

The possible actions are a discretization of the control space, represented as forward motion and also left and right motions in 45º angles. The discretization was chosen in such way to reduce learning times. To avoid sharp turns as a result of this discretization, the generated trajectories are smoothed using a B-spline [25]. Given this control space, each action a_t in our model can be represented by a control $u(t)$. Furthermore, the motion $u(t)$ can be seen a trajectory of two points and one time step, where its cost can be expressed in terms of Γ, thus for each action a_t in state z_t its reward is given by

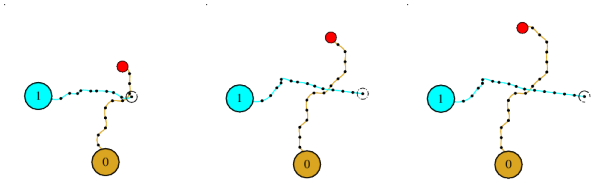
$$r_{t+1} = -\left|(1-\alpha_{r,p}) \cdot \Gamma(u_p(t)) - \alpha_{r,p} \cdot \Gamma(u_r(t))\right| \quad (7)$$

The reward in Eq. 7 is used in case the robot did not reach its goal and there was no collision, in case otherwise, the reward is set to, respectively twenty and minus twenty.

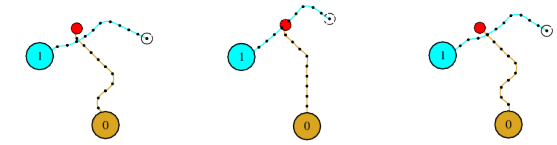
V. Results

In this section we evaluate our approach to shared effort in HRM. The tests were executed inside the ROS framework and its packages but most trajectory planning is done inside ORCA space. The persons are simulated as holonomic agents using ORCA and are able to change their speed, conversely, the robot has a discretized control space that is always at maximum speed. The robot is set to have maximum speed equal to a person's maximum speed, however our model is compatible with any particular proportion between these two speeds. Moreover, in our tests both simulated person and robot have a circular shape with diameter of 34 cm (similar to TurtleBot 2). The robot motion model used to find the trajectory is point mass but restricted to three acceptable actions, see section IV-B for details.

In these tests, the time step between t and $t+1$ of our prediction is equal 0.25 seconds.

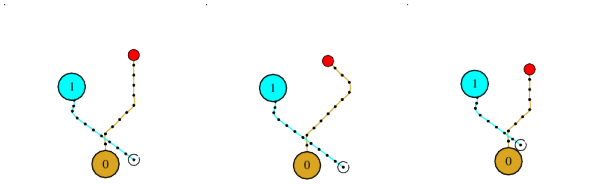


(a) Three difference scenarios where $\dot{\beta} < 0$

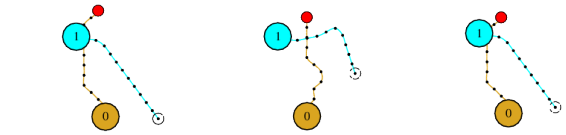


(b) Three difference scenarios where $\dot{\beta} > 0$

Fig. 3: Crossing angle of 90º, where zero indicates the effort-aware robot and one the human-like planner



(a) Three difference scenarios where $\dot{\beta} < 0$



(b) Three difference scenarios where $\dot{\beta} > 0$

Fig. 4: Crossing angle of 45º, where zero indicates the effort-aware robot and one the human-like planner

A. Trajectories based on crossing order

The trajectories presented were made accounting for different crossing angles and crossing order expectations in order to evaluate their feasibility. The goal of the person is a point with a fixed distance away while the goal of the robot is a random position away and an angle near the direction of their heading, this allows one to randomize the crossing order without altering relative velocities. This is so as the person and the robot are set to have near equal speeds.

It is important to note that there is no perceived order in crossing scenarios with angles of 0º, depicted in Fig. 5. Whereas in the case of crossing angle 45º (Fig .4) and 90º (Fig. 3), we showcase different available trajectories in the cases where a robot has crossing order priority ($\dot{\beta}_{r,p} < 0$) and in cases where a person has the priority ($\dot{\beta}_{r,p} > 0$).

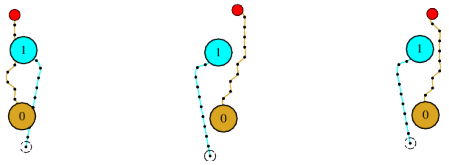


Fig. 5: Case with no crossing order, where zero indicates the effort-aware robot and one the human-like planner

B. Runtime performance

The runtime performance analysis of our approach is presented in Table I to showcase that, after the policy is trained, it can be used to provide response times that are compatible with real-world requirements.

Policy type	Avg. 1-step (s)	Avg. n-steps (s)
Online	0.100489s	3.701310s
Offline	0.000479s	0.016419s

TABLE I: Runtime performance comparison with an online policy, that updates its model while taking actions, and also an offline policy, which only applies learned behavior.

VI. Discussion and Conclusion

This work presented an approach to allow a robot to share the effort required to avoid collision with a person by learning a policy that encodes stereotypical behaviors from persons during collision avoidance. The results observed during experimental evaluation show that the robot is capable of sharing effort with angles 0°, 45°, 90° without simply yielding to the person.

To our knowledge, this is the first work that approximates the human asymmetrical effort sharing during collision avoidance in 90° crossing scenarios in different crossing orders. This can allow a robot to better represent human-like behavior, this is important as following stereotypical motions were shown in recent works to reduce planning effort for persons in the environment.

For the short term, our plan is improve our model as to allow effort sharing with multiple persons, as our current approach is limited to the one person scenario. This would allow observation of cases where avoiding collisions with someone could by consequence cause additional effort to somebody else. Our long term goal is to apply this model into a real robot that has to avoid collision with multiple persons in a real environment.

ACKNOWLEDGMENT

This work is supported by the Brazilian National Counsel of Technological and Scientific Development (CNPq).

References

[1] T. Fraichard and H. Asama, "Inevitable collision states—a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, 2004.

[2] F. Lindner and C. Eschenbach, "Towards a formalization of social spaces for socially aware robots," *Spatial Information Theory*, 2011.

[3] C. Lichtenthaler, T. Lorenzy, and A. Kirsch, "Influence of legibility on perceived safety in a virtual human-robot path crossing task," in *IEEE Int. Work. Robot Hum. Interact. Commun.*, Paris (FR), Sep. 2012.

[4] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *IEEE Int. Conf. Robot. Autom.*, Anchorage (UM), May 2010.

[5] M. Shiomi, F. Zanlungo, K. Hayashi, and T. Kanda, "Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model," *Int. J. Soc. Robot.*, vol. 6, no. 3, 2014.

[6] P. Basili, M. Saglam, T. Kruse, M. Huber, A. Kirsch, and S. Glasauer, "Strategies of locomotor collision avoidance," *Gait and Posture*, vol. 37, no. 3, 2013.

[7] D. Carton, W. Olszowy, and D. Wollherr, "Measuring the effectiveness of readability for mobile robot locomotion," *International Journal of Social Robotics*, 2016.

[8] G. Ferrer and A. Sanfeliu, "Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago (UM), Sep. 2014.

[9] B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, St. Louis (UM), Oct. 2009.

[10] A. Olivier, A. Marin, A. Crétual, A. Berthoz, and J. Pettré, "Collision avoidance between two walkers: Role-dependent strategies," *Gait and Posture*, vol. 38, no. 4, 2013.

[11] M. Huber, Y. Su, M. Kruger, K. Faschian, S. Glasauer, and J. Hermsdorfer, "Adjustments of speed and path when avoiding collisions with another pedestrian," *PLoS ONE*, vol. 9, no. 2, 2014.

[12] S. Jansen, A. Toet, and P. Werkhoven, "Human locomotion through a multiple obstacle environment: Strategy changes as a result of visual field limthrough," *Experimental Brain Research*, vol. 212, no. 3, 2011.

[13] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, 1995.

[14] J. van den Berg, S. Guy, M. Lin, and D. Manocha, *Reciprocal n-Body Collision Avoidance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[15] R. Sutton and A. Barto, *Reinforcement Learning: An introduction*, M. Press, Ed., 1998.

[16] T. Hester and P. Stone, "Texplore: real-time sample-efficient reinforcement learning for robots," *Machine Learning*, vol. 90, no. 3, 2013.

[17] Y. Morales, A. Watanabe, F. Ferreri, T. Even, J. Ikeda, K. Shinozawa, T. Miyashita, and N. Hagita, "Including human factors for planning comfortable paths," in *IEEE Int. Conf. Robot. Autom.*, Seattle (UM), May 2015.

[18] R. Narain, A. Golas, S. Curtis, and M. Lin, "Aggregate dynamics for dense crowd simulation," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009.

[19] A. Bera and D. Manocha, "Realtime multilevel crowd tracking using reciprocal velocity obstacles," in *Int. Conf. on Pattern Recognition (ICPR)*. IEEE, 2014.

[20] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[21] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion-an inverse optimal control approach," *Autonomous Robots*, vol. 28, no. 3, 2009.

[22] J. Cutting, P. Vishton, and P. Braren, "How we avoid collisions with stationary and moving obstacles," *American Psychological Association*, vol. 102, no. 4, 1995.

[23] D. Marigold, V. Weerdesteyn, A. Patla, and J. Duysens, "Keep looking ahead? re-direction of visual fixation does not always occur during an unpredictable obstacle avoidance task," *Experimental Brain Research*, vol. 176, no. 1, 2007.

[24] J. Sardegna, S. Shelly, and S. Steidl, *The encyclopedia of blindness and vision impairment*. Infobase Publishing, 2002.

[25] P. Dierckx, *Curve and surface fitting with splines*. Oxford University Press, 1995.

On Multi-robot Search for a Stationary Object

Miroslav Kulich[1], Libor Přeučil[1] and Juan José Miranda Bront[2]

***Abstract*— Two variants of multi-robot search for a stationary object in a priori known environment represented by a graph are studied in the paper. The first one is generalization of the Traveling Deliveryman Problem where more than one deliveryman is allowed to be used in a solution. Similarly, the second variant is generalization of the Graph Search Problem. A novel heuristics suitable for both problems is proposed which is furthermore integrated into a cluster-first route second approach. A set of computational experiments was conducted over the benchmark instances derived from the TSPLIB library. The results obtained show that even a standalone heuristics significantly outperforms the standard solution based on k-means clustering in quality of results as well as computational time. The integrated approach furthermore improves solutions found by a standalone heuristics by up to 15% at the expense of higher computational complexity.**

I. Introduction

Assume a mobile robot autonomously operating in a priori known environment, in which a stationary object of interest is randomly placed. The objective is to find the object, whose position is not known in advance, in a minimal time. This problem is formulated as the Traveling Deliveryman Problem (TDP) provided that the environment is represented by a graph and probability of appearance of the object is the same for all vertices in the graph. Although TDP is similar to the Traveling Salesman Problem (TSP), objectives of these problems differ and thus an optimal solution for one problem is not necessarily optimal for the second problem [1].

TDP, which is NP-hard [2], has been studied from various perspectives during the last few years. Besides exact algorithms introduced by several authors [3], [4], Integer Linear Programming with Branch-and-Cut and Branch-Cut-and-Price approaches were proposed [5], [6], [7] for time-dependent TSP which generalizes TDP. The best exact algorithm [5], nevertheless, is able to solve instances with up to 107 vertices to optimality in several hours.

More useful are heuristics and meta-heuristics which provide good quality solutions with much lower computational effort. These rely particularly on Greedy Randomized Adaptive Search Procedure (GRASP), introduced originally by Feo and Resende [8], and Variable Neighborhood Search (VNS), proposed by Hansen and Mladenovic [9]. Salehipour et al. [10] employ a GRASP for TDP and compare the impact of VNS procedure as a local search phase with Variable Neighborhood Descent. Mladenovic et al. [11] propose a General VNS (GVNS), which improves Salehipour's results. Further improvements were achieved by Silva et al. [12] who propose a simple multi-start heuristic combined with an Iterated Local Search procedure. To the best of our knowledge, the approach by Silva is thus the one producing the best results, nevertheless, time needed to compute problems containing 100 vertices is more than ten seconds and instances with 200 vertices are computed in approximately one minute.

Approaches used by the robotics community are simpler. Sarmiento et al. [13] propose a modification of breadth-first algorithm which iteratively constructs all possible routes of the defined length, fixes the most promising one and starts the next search from this route as a prefix. Finally, a modified depth-first search algorithm with pruning and limited branching was introduced in our previous work [1].

The Graph Search Problem (GSP), introduced in Kutsoupias et al. [14], is formulated under the same settings as TDP, with the only difference that each vertex has assigned probability of finding the object when visiting the vertex and probabilities of the vertices differ in general. Besides some theoretical results regarding approximation schemes presented in Ausiello et al. [15], no further developments are present in the related literature. The only exception is our recent work [16], in which a tailored GRASP meta-heuristic for the GSP is introduced which is able to find near-optimal solutions for TDP and GSP problems up to 107 vertices in about one second of computing time.

Little attention has been paid to multi-robot variants of TDP and GSP. On the other hand, some inspiration can be found in approaches to the Multiple Traveling Salesman Problem (mTSP) or other routing problems. Besides genetic algorithms, neural networks, or ant colony optimization, the cluster-first route-second approach plays an important role. The key idea of this approach is to split all vertices into M clusters based on their location in space (M is the number of salesmen) and solve the traditional Traveling Salesman Problem for each cluster separately. For example, Sathyan et al. [17] use k-means clustering for the first phase followed by application of a genetic algorithm or 2-opt heuristic. Boone et al. [18] employ an initial k-means clustering and modify it by taking points from the cluster with the largest tour distance and adding them to one of the smaller clusters. Convex hulls, fuzzy logic, and the TSP solution are used to determine which points to switch. Geetha et al. [19] improved k-means algorithm for the Capacitated Clustering Problem by incorporating a priority measure to the criterion on which are vertices assigned to clusters. We use k-

[1] Miroslav Kulich, Libor Přeučil are with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague,Zikova 1903/4, 166 36 Prague, Czech Republic {miroslav.kulich, libor.preucil}@cvut.cz

[2] Juan José Miranda Bront is with Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina jmiranda@dc.uba.ar

means clustering followed by solution of Traveling Salesman Problem for each cluster by Chained Lin-Kernighan as a base of a goal assignment strategy in multi-robot exploration [20]. The presented experimental results indicate that this method provides more efficient assignment than former approaches.

In this paper we build on experience from the works mentioned above and present a cluster-first route-second approach for the Multiple Traveling Deliveryman Problem (mTDP) and the multi-vehicle case of GSP (mGSP). The approach extends our GRASP-based meta-heuristic for the single-vehicle problems [16] by incorporating the clustering phase. The key contribution thus lies in design of a novel clustering method which respects special aspects of mTDP and mGSP. The proposed solution is evaluated computationally and compared with an algorithm based on k-means. The experimental results show that our solution has potential to be applied in practice as it provides better results than the k-means based approach in almost all problem instances.

The rest of the paper is organized as follows. The problem is defined in Section II, the proposed clustering is presented in Section III, while the key ideas of the tailored GRASP-approach for TDP and GSP are summarized in Section IV. Computational results including instances of both mTDP and mGSP are presented and discussed in V. Finally, Section VI is dedicated to concluding remarks and future directions.

II. Problem Formulation

That is, formally, given

- a complete undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ stands for a finite set of vertices and E is a set of edges between these vertices: $E = \{e_1, e_2, \ldots e_{n^2}\}$, $e_{ij} = (v_i, v_j)$, $v_i, v_j \in V$, $i \neq j$,
- $t: E \to \mathbb{R}$: a cost t_{ij} associated with each edge e_{ij} representing time needed to traverse the shortest path from i to j,
- $p: V \to \langle 0, 1\rangle$: a weight for each vertex representing probability of presence of the searched object at the vertex,
- the number of the vehicles M, and
- $s_i \in V, i \in \langle 1, M\rangle$: starting vertices of the vehicles (note that several vehicles can start from the same vertex in general).

Define a *walk* $\boldsymbol{\omega} = \langle \omega_1, \omega_2, \ldots \omega_k\rangle$ as a sequence of vertices of G, i.e. $\omega_i \in V$ for $i \in \langle 1, k\rangle$. The overall objective is then to find a tuple of M walks $\Omega = \langle \boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \ldots, \boldsymbol{\omega}^M\rangle$ that visits all vertices of V at least once (i.e. $\forall v \in V\ \exists \omega^i_j \in \boldsymbol{\omega}^i:\ \omega^i_j = v$) and which minimizes the expected time to find the object:

$$\Omega = \arg\min_{W \in \Theta} \mathbb{E}(T|W) = \sum_{i=1}^{M} \sum_{j=1}^{|W^i|} \tau(w^i, j) p(w^i_j), \qquad (1)$$

where

$$\tau(w^i, j) = \sum_{\iota=1}^{j-1} t(w_\iota w^i_{\iota+1})$$

is the time when the vertex w^i_j, the j-th vertex of the walk $W^i \in W$ is visited and Θ is a set of all possible sets of walks in G. Moreover, all the walks have to start in the given starting vertices, i.e. $w^i_1 = s_i$ for $i = 1 \ldots M$. The minimal expected time is then

$$T_{exp} = \mathbb{E}(T|\boldsymbol{\omega}) = \sum_{i=1}^{M} \sum_{j=1}^{|\omega^i|} \tau(\omega^i, j) p(\omega^i_j). \qquad (2)$$

In summary, the aim is to minimize the average time the vertices are visited weighted by probabilities assigned to the vertices.

The multi-vehicle Traveling Deliveryman Problem is a special variant of mGSP with the only difference that the probability of finding the object is the same for all the vertices in the graph. These probabilities can be omitted from the equations for this case.

III. Proposed Clustering Approach

The proposed method for solving both mTDP and mGSP follows the cluster-first route-second schema. This means that all vertices of the graph are divided into M clusters assigned to the vehicles in the first phase. A TDP/GSP solver is then run for each cluster separately to optimize the order in which the vertices in the clusters are visited.

The clustering approach whose scheme is depicted in Algorithm 1 has a greedy nature. The algorithm starts with initialization of clusters: each cluster contains the starting vertex of the vehicle it is assigned to and current time needed to traverse each route is set to zero (lines 1–3). Moreover, the set of all not assigned graph vertices V_{REST} is set (line 4).

The main part of the algorithm is a loop between lines 5 and 18 which is processed until V_{REST} is empty. In each iteration of the loop, each not yet assigned vertex is examined to attach to each of the routes. Time in which the vertex is visited is computed first (line 9, followed by evaluation of a penalty function (line 10). This function was designed to prefer vertices that are visited earlier and with higher probability of the object of interest at them.

The pair $\langle v^{min}, \omega^{min}\rangle$ with the lowest value of the penalty function is chosen and the vertex v^{min} is attached to the end of ω^{min} (line 13) and time needed to traverse the ω^{min} is updated accordingly (line 14). Finally, v is removed from V_{REST} (line 15). Note that the algorithm does not only return partition of vertices into clusters, but also their orders within the clusters (line 19). The result can be thus used as a solution of mTDP/mGSP. An example of a solution found by the algorithm is shown in Fig. 1.

IV. Meta-heuristic for a single-vehicle case

Having the vertices partitioned into clusters, the second step is to find the best order for each cluster. To do that, we employ the greedy randomized adaptive search procedure (GRASP) meta-heuristic tailored for TDP and GSP [16]. The general scheme of the GRASP is shown in Algorithm 2.

The value of the best already found solution is initially set to a high number in the algorithm (line 1). The meta-heuristic then consecutively constructs initial solutions using some simple, typically greedy, heuristic (line 4). The obtained

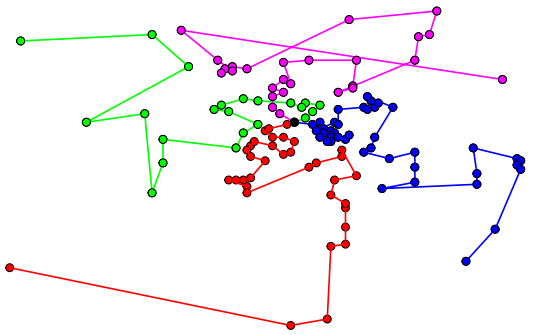


Fig. 1: The result of the proposed clustering method for the bier127 problem [21] and 4 vehicles.

solution is improved by a sequence of local search steps (line 5) and its cost is updated accordingly (line 6). If the improved solution is better than the current best solution, the best solution and its cost are updated (lines 7–9). After all initial solutions are processed, the best solution found is returned (line 10). More detailed description of the particular steps of the GRASP follows in the next paragraphs.

We consider two heuristics for initial construction of routes, both using the same general greedy scheme as shown in Algorithm 3. The algorithm starts with a route containing only the start vertex (line 1), consecutively finds the best vertex from vertices not yet connected according to the penalty function f. The found vertex is then appended at the end of the route (line 4) and removed from the set of not connected vertices (line 5).

The difference between the two heuristics lies in the definition of f, which affects the selection of the next vertex to be visited (line 4). We consider the standard distance-based function, $f_{\text{dist}}(u, v) = t(u, v)$. In order to incorporate the weights for each candidate, we further consider the function $f_{\text{ratio}}(u, v) = t(uv)/(1 + p(v))$. As both heuristics are deterministic, we employ Monte Carlo randomization to generate various initial solutions – the lower the penalty function value for the vertex, the higher probability that the vertex is selected.

After obtaining an initial feasible solution, an improvement phase is performed. In our case, a Variable Neighborhood Descent (VND) [10] is used. The idea of VND is simple: a neighborhood of the current solution is completely searched and the best neigbor replaces the solution. This process is repeated until no improvement is possible. A neighborhood of the route ω defined by an operator is a set of all routes which are formed from ω by application of this operator. Two different neighborhoods are considered which are formed by two local search operators:

- **Swap**: Select two vertices in the tour and swap them.
- **2-opt**: Select two non-adjacent arcs and replace them by two new arcs, obtaining a new route as a result.

When the result obtained by VND with Swap and 2-opt is promising, i.e. its cost is less than 10% of the current best solution, the LK-op operator is applied to this result. LK-op is an adaptation of Lin-Kernighan operator [22] designed for TSP. It starts with a feasible solution and considers each edge sequentially as a seed for an improvement procedure.

Algorithm 1: Proposed clustering algorithm.

Input: M – the number of vehicles
$G = (V, E)$ – a graph
t_{ij} – costs of edges
$p(i)$ – probabilities associated with vertices
$s_i \in V, i \in \langle 1, M \rangle$ – start vertices of vehicles
Output: $\boldsymbol{\omega} = \langle \omega_1, \omega_2, \ldots \omega_M \rangle$ – a tuple of sequences representing clusters

1 **for** $i \leftarrow 1$ **to** M **do**
2 $\omega^i \leftarrow \langle s_i \rangle$
3 $\tau_{\omega^i} = 0$
4 $V_{REST} \leftarrow V \setminus \{s_1, s_2, \ldots, s_M\}$
5 **while** $V_{REST} \neq \emptyset$ **do**
6 $min \leftarrow \infty$
7 **foreach** $\omega \in \boldsymbol{\omega}$ **do**
8 **foreach** $v \in V_{REST}$ **do**
9 $d = \tau_\omega + t(idx(v), idx(last(\omega)))$, where $idx(v)$ is the index of v and $last(\omega)$ is the last vertex of the route ω
10 $c = \frac{d}{1+p(idx(v))}$
11 **if** $c < min$ **then**
12 $min \leftarrow c$
13 $d^{min} \leftarrow d$
14 $v^{min} \leftarrow v$
15 $\omega^{min} \leftarrow \omega$
16 $\omega^{min} \leftarrow \omega^{min} + v$
17 $\tau_{\omega^{min}} \leftarrow d^{min}$
18 $V_{REST} \leftarrow V_{REST} \setminus \{v\}$
19 **return** $\boldsymbol{\omega} = \langle \omega_1, \omega_2, \ldots \omega_M \rangle$

The procedure attempts to obtain an improved route by application of a sequence of 2-opt moves, not necessarily improving the current solution, with a limited backtracking. If such a better path is found, it is accepted as the new initial solution and the procedure is restarted again from the first edge. Otherwise, the algorithm moves to the next edge, until all edges have been considered as a seed.

As LK-op is computationally demanding, the search space is reduced in two ways. First, a length of a sequence of performed 2-opt moves is limited to a maximum depth α. Moreover, not all edges are considered for each vertex, only limited number of shortest edges β is evaluated instead. Detailed description of the method as well as discussion about complexity of the particular steps can be found in [16].

V. Results

Performance of the proposed approach has been evaluated for both mTDP and mGSP. The experiments for mTDP were run on a set of standard instances from TSPLIB [21] with sizes between 52 and 1002. As there are no benchmark instances for mGSP, instances from TSPLIB were also used for which probabilities of vertices were generated randomly. To ensure repeatability of experiments, a generator from `random.org` was utilized for generating 10000 normally

Algorithm 2: GRASP scheme.

```
z_best ← ∞.
foreach h ∈ H do
    for k = 1, ..., N_it do
        Obtain a feasible route ω using h.
        Improve route ω by applying a local search step
        Update cost z of ω.
        if z < z_best then
            ω_best ← P
            z_best ← z
return ω_best
```

Algorithm 3: General greedy scheme.

Input: $G = (V, E)$ – a graph
t_{ij} – costs of edges
$p(i)$ – probabilities associated with vertices
s – starting vertex
Output: $\omega = \langle \omega_1, \omega_2, \dots \omega_K \rangle$ – a route

```
P ← s
V_REST ← V \ {s}
while V_REST ≠ ∅ do
    v ← arg min_{u∈V_REST} f(last(ω), u)
    ω ← ω + v
Return path P.
```

distributed random numbers between 1 and 10 and the string "2016-09-11" was set as a seed[1]. Random numbers were assigned to vertices respecting the order, i.e. i-th vertex of an TSPLIB instance has assigned i-th random number. Moreover, the numbers were normalized so that the sum of probabilities of all vertices for an instance is 1.

Regarding the parameters of the method, we set $\alpha = 20$, limiting the size of sequences of 2-opt moves, $\beta = 5$ for first 4 depths of backtracking, and $\beta = 1$ for the rest. N_{it} was set to 50, generating 100 initial solutions in total. Start positions of all vehicles were set to the first vertex of the instance.

The proposed cluster-first route-second approach was compared with pure clustering as proposed in Section III and with an approach combining k-means clustering and our GRASP meta-heuristic. More specifically, we employed k-means++ – an augmentation of k-means which significantly improves both the speed and the accuracy of k-means [23]. We refer to the methods as `proposed`, `clustering`, and `k-means`.

All experiments were performed within the same computational environment: a workstation with the Intel®Core i7-3770 CPU at 3.4 Ghz running Linux with the kernel 4.4.0. The algorithms have been implemented in C++ and compiled by clang 3.8.1. with "-O2" flag. 50 runs were run for each setup consisting of an instance, a number of vehicles, and a method to provide statistically significant results.

The results for mTDP are presented in Table I. Meaning of the symbols is following: M stands for the number of vehicles, BKS for the best known solution (to the best of our knowledge, there is no other mTDP solver and BKS is thus the best solution found by one of the evaluated methods), SD is standard deviation, and T is execution time. PDB is the percent deviation to BKS of the best solution values found by the algorithm (denoted as $best$), i.e. PDB=($best$-BKS)/BKS. Similarly, PDM is the percent deviation of the mean solution value to BKS. PDBs for the method which found the best solution are highlighted in bold. Note that `clustering` is a deterministic algorithm, therefore SD is always zero and PDM=PDB and are thus omitted.

The results show that `proposed` outperforms `k-means` in all cases except six, sometimes by more than 20% in PDB and by more than 30% in PDM. On the other hand, `k-means` produces better results for small numbers of robots, where the highest difference of PDM is less than 6%. `clustering` lies between these two. Its benefit is much lower computational complexity in comparision to the other methods, with no large gap of produced results to the best found solution. Difference of computational burden between `k-means` and the other methods increases with an increasing number of vertices as the k-means clustering is much slower that the proposed one. This applies especially for higher Ms, for which the influence of complexity of GRASP is reduced in comparison to `clustering` and `k-means` is thus more than 2 times slower than `proposed`.

The results for mGSP are presented in Table II in the same way as for mTDP. The running time of all methods slightly increased in comparison to mTDP, but the situation remains the same regarding quality of solutions. `proposed` produces best results in the majority of cases, followed by `clustering`. Again, `k-means` found best solutions for several instances and $M = 2$, but the gap does not exceed 6%.

VI. Conclusion

We formulated mutli-vehicle variants of two routing problems – the Traveling Deliveryman Problem and the Graph Search Problem and introduced a novel clustering approach which is designed especially for these two problems. The proposed clustering was then used together with the GRASP metaheuristics in the cluster-first route-second scheme as an integrated approach to the problems. The proposed integrated approach is, based on the performed experimental results, suitable to solve both problems as it outperforms the standard approach based on k-means clustering in quality of found solutions in the majority of cases with lower computational burden. Moreover, the proposed clustering can be used as a standalone solver as it produces solutions slightly worse than the integrated approach, but substantially faster.

Future research will go in two directions. We would like to use the proposed clustering in GRASP as a constructive heuristic and together with design of neighborhoods for exchanging vertices between routes extend pure GRASP for mTDP and mGSP. The second stream will focus on application of the proposed approach in other robotic applications. Similarly to using a mTSP solver for the exploration

[1]We are ready to publish the generated sequence if the paper is accepted.

Problem	M	BKS	clustering		proposed				k-means			
			PDB	T [ms]	PDB	PDM	SD	T [ms]	PDB	PDM	SD	T [ms]
berlin52	2	70235	3.98	0.24	**0**	0	0.79	42.67	6.82	15.76	6070.62	64.03
	4	39746	0.78	0.25	**0**	0	0	19.68	8.84	12.77	1052.73	31.31
	6	30563	0.03	0.27	**0**	0	0	8.75	14.24	18.07	824.95	17.48
	8	25470	1.34	0.30	**0**	0	0	6.61	13.53	21.11	1137.95	12.69
	10	23919	1.01	0.35	**0**	0	0	5.14	14.83	19.05	742.42	8.96
bier127	2	2354332	10.03	0.78	1.59	2.74	8388.61	396.88	**0**	7.66	142517.02	540.77
	4	1228367	13.10	0.64	5.84	5.84	0	116.97	**0**	16.83	80731.98	251.61
	6	879448	4.95	0.65	**0**	0	0	82.52	17.52	24.22	47044.46	183.57
	8	713125	0.97	0.66	**0**	0	0	52.17	14.96	23.25	27229.50	122.54
	10	612336	0.35	0.73	**0**	0	0	38.65	19.48	26.01	18738.51	99.08
gil262	2	153716	8.66	3.23	**0**	0.98	601.31	3055.23	0.06	4.32	3130.36	4342.95
	4	87114	6.96	2.16	**0**	0.21	81.67	1019.36	12.65	13.30	209.34	2187.43
	6	65428	3.10	2.07	**0**	0.04	22.76	599.17	13.56	18.98	2124.00	1168.37
	8	55306	1.90	2.10	**0**	0.01	8.14	447.16	16.25	21.84	1566.33	776.67
	10	50292	0.75	2.18	**0**	0	2.63	329.39	19.43	23.64	1417.02	617.23
lin318	2	3140312	15.07	4.29	4.55	5.75	15004.90	5855.93	**0**	2.92	32535.88	5931.95
	4	1811206	6.46	2.92	**0**	0.24	2817.56	1644.35	1.80	7.44	36532.68	2239.29
	6	1431946	4.18	2.86	**0**	0.08	1004.02	940.59	2.35	5.92	31411.96	1371.37
	8	1214427	3.24	2.85	**0**	0.03	292.51	563.80	4.56	8.13	44304.42	976.14
	10	1129348	3.93	3.03	**0**	0.01	256.28	565.05	3.55	6.49	27180.85	760.35
pcb442	2	5292831	5.07	9.13	**0**	0.56	17515.61	14006.46	5.56	6.46	26968.07	24343.18
	4	2849704	4.96	6.08	**0**	0.44	7294.97	3758.14	14.27	14.61	6929.25	8974.81
	6	2055340	2.07	5.75	**0**	0.14	2272.75	1641.71	17.98	21.26	33959.70	4861.82
	8	1607307	1.65	5.73	**0**	0.11	1234.00	1251.63	26.51	30.42	30783.56	3115.88
	10	1402893	2.74	5.84	**0**	0.05	898.36	988.54	29.37	33.29	24376.01	2261.04
rat575	2	994166	7.01	16.96	**0**	0.96	3631.90	37961.41	2.20	3.10	3053.47	66531.58
	4	524957	6.65	10.47	**0**	0.50	1110.60	9644.29	9.78	13.40	12210.99	23472.77
	6	375842	6.60	10.20	**0**	0.22	375.69	4100.55	11.89	13.89	8107.88	11401.33
	8	303082	3.46	10.10	**0**	0.06	137.15	2312.13	15.63	17.79	5449.48	7426.09
	10	260937	1.37	10.47	**0**	0.05	101.79	1406.92	18.82	22.19	4480.78	5577.58
u724	2	7479059	8.28	26.90	**0**	1.31	39681.60	57579.28	2.86	3.47	24549.00	79037.02
	4	3904506	7.77	17.47	**0**	0.52	8792.00	16542.24	9.23	12.04	59939.66	26556.46
	6	2926359	5.31	16.85	**0**	0.38	4477.46	8135.85	10.39	11.53	22159.80	15149.82
	8	2427262	4.49	17.03	**0**	0.11	1496.06	6363.60	11.95	13.89	28998.38	10828.30
	10	2111492	2.46	17.42	**0**	0.12	1014.74	3725.07	14.90	16.67	34886.45	8564.65
pr1002	2	65541078	5.99	52.28	0.35	0.99	215703.18	183435.59	**0**	0.63	227427.16	279021.00
	4	36553749	13.90	35.78	1.69	2.20	85095.75	53400.13	**0**	0.75	192716.60	81455.54
	6	26676866	9.48	36.38	**0**	0.28	33816.36	25943.29	1.17	2.10	318901.94	43332.34
	8	20946133	7.57	36.03	**0**	0.18	20700.88	14389.91	8.59	10.53	544729.91	30745.18
	10	19540052	6.21	37.82	0.96	1.14	17698.56	9307.68	**0**	3.34	524410.71	22968.16

TABLE I: Comparison of the algorithms for mTDP.

task [20] and a GSP solver for search in a priori unknown environment [1], [16], we would like to study properties of mGSP for multi-robot search in an unknown space.

ACKNOWLEDGMENTS

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688117 and the Technology Agency of the Czech Republic under the project no. TE01020197 "Centre for Applied Cybernetics".

REFERENCES

[1] M. Kulich, L. Přeučil, and J. J. Miranda-Bront, "Single Robot Search for a Stationary Object in an Unknown Environment," in *Robotics and Automation (ICRA), IEEE Int. Conf. on*, 2014, pp. 5830–5835.

[2] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, "The complexity of the travelling repairman problem," *Theoretical Informatics and Applications*, 1986.

[3] L. Gouveia and S. Voß, "A classification of formulations for the (time-dependent) traveling salesman problem," *European Journal of Operational Research*, vol. 2217, no. 93, 1995.

[4] I. Méndez-Díaz, P. Zabala, and A. Lucena, "A new formulation for the traveling deliveryman problem," *Discrete Appl. Math.*, vol. 156, no. 17, pp. 3223–3237, Oct. 2008.

[5] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa, "The time dependent traveling salesman problem: polyhedra and algorithm," *Mathematical Programming Computation*, vol. 5, no. 1, pp. 27–55, Sept. 2012.

[6] J. J. Miranda-Bront, I. Méndez-Díaz, and P. Zabala, "Facets and valid inequalities for the time-dependent travelling salesman problem," *European Journal of Operational Research*, May 2013.

[7] M. T. Godinho, L. Gouveia, and P. Pesneau, "Natural and extended formulations for the Time-Dependent Traveling Salesman Problem," *Discrete Applied Mathematics*, vol. 164, pp. 138–153, Feb. 2014.

[8] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[9] P. Hansen and N. Mladenović, "Variable Neighborhood Search," *Computers & Operations Research*, vol. 24, no. 1, pp. 1097–1100, 1997.

[10] A. Salehipour, K. Sörensen, P. Goos, and O. Bräysy, "Efficient

Problem	M	BKS	clustering		proposed				k-means			
			PDB	T [ms]	PDB	PDM	SD	T [ms]	PDB	PDM	SD	T [ms]
berlin52	2	1289.3665	15.49	0.29	2.55	2.55	0.06	55.73	**0**	7.67	101.55	65.39
	4	711.9565	8.92	0.32	**0**	0	0	24.22	5.10	9.70	23.96	40.98
	6	569.3354	1.28	0.33	**0**	0	0	11.74	8.06	11.66	12.52	21.95
	8	510.9161	2.82	0.36	**0**	0	0	8.35	4.91	9.25	14.94	14.45
	10	462.7143	0	0.38	**0**	0	0	4.53	9.43	13.30	16.38	10.43
bier127	2	16938.3426	16.94	0.92	**0**	0.58	59.14	465.85	1.19	9.68	1212.99	613.18
	4	9015.8078	10.15	0.78	**0**	0.11	7.17	179.04	6.09	21.01	774.75	301.05
	6	6439.1727	5.06	0.81	**0**	0	0.25	112.68	14.08	23.26	368.25	221.56
	8	5297.5682	2.33	0.86	**0**	0	0	63.20	10.91	22.10	280.29	153.78
	10	4517.5042	1.83	0.94	**0**	0	0	42.99	15.53	24.25	161.22	120.06
gil262	2	557.1953	12.84	3.46	**0**	1.03	2.38	3465.02	0.39	3.18	11.40	4415.25
	4	324.9695	8.78	2.51	**0**	0.30	0.49	1028.91	9.60	10.06	1.77	2628.16
	6	241.1403	5.21	2.49	**0**	0.12	0.16	604.07	13.62	17.69	6.33	1410.60
	8	208.0251	4.62	2.57	**0**	0.03	0.06	378.99	15.12	20.39	6.27	1009.05
	10	195.8603	1.75	2.89	**0**	0	0	315.60	13.94	18.16	4.62	788.35
lin318	2	9729.8214	17.30	4.85	5.48	6.23	40.37	6412.44	**0**	1.41	46.65	6757.29
	4	5614.8969	9.26	3.67	**0**	0.40	8.24	1769.25	1.00	5.74	112.89	2510.83
	6	4390.3071	4.68	3.52	**0**	0.16	3.95	878.58	2.62	7.51	118.88	1773.18
	8	3728.5347	4.50	3.77	**0**	0.02	0.75	606.11	5.42	9.02	135.38	1315.97
	10	3504.8153	4.69	4.13	**0**	0.01	0.43	547.21	2.99	5.94	96.50	1001.21
pcb442	2	11670.8481	11.20	10.06	3.23	4.25	45.68	13903.57	**0**	3.06	155.85	19525.87
	4	6076.4795	5.80	7.07	**0**	0.39	10.10	3348.96	14.86	15.81	22.50	8445.16
	6	4438.6460	4.89	6.70	**0**	0.19	3.57	1725.01	17.89	21.13	70.81	5055.13
	8	3634.2044	3.33	7.01	**0**	0.05	1.21	1113.55	22.60	24.91	61.11	3643.70
	10	3197.8358	3.34	7.64	**0**	0.01	0.39	733.25	24.93	28.05	47.30	2869.68
rat575	2	1624.6539	12.92	17.25	**0**	1.14	8.45	29891.17	0.50	1.17	5.19	50161.28
	4	885.9464	12.11	11.67	**0**	0.46	2.41	8358.31	4.77	8.81	20.46	20297.80
	6	647.9221	5.97	11.25	**0**	0.33	1.11	4596.77	5.93	8.06	14.37	11989.79
	8	527.9039	5.70	11.55	**0**	0.14	0.32	2678.61	9.93	11.43	7.55	8496.11
	10	444.0691	3.56	12.76	**0**	0.10	0.21	1842.64	15.72	19.20	7.37	6800.49
u724	2	10053.0689	20.39	27.42	1.63	3.08	47.51	64240.43	**0**	0.85	31.56	87189.09
	4	5492.4174	12.34	18.58	**0**	0.52	14.49	16706.83	2.46	5.66	95.87	31098.84
	6	3967.1502	9.06	17.38	**0**	0.35	4.86	10488.02	7.99	9.06	39.43	18807.40
	8	3377.1366	6.68	18.08	**0**	0.14	2.61	6682.94	8.28	9.73	24.42	13853.41
	10	2805.0308	4.43	20.14	**0**	0.10	1.27	4474.89	16.32	17.96	30.07	10550.73
pr1002	2	60325.4178	17.21	54.04	**0**	1.22	266.22	242200.87	1.78	3.06	289.47	271763.71
	4	34423.7763	20.83	36.08	5.10	5.62	98.06	52536.92	**0**	0.99	137.73	77147.53
	6	25056.6643	12.34	34.25	**0**	0.32	39.35	33415.13	3.26	4.06	236.43	49881.61
	8	20813.4050	9.51	34.61	**0**	0.24	23.37	19364.56	4.46	6.70	456.05	35652.01
	10	17750.8518	9.21	38.10	**0**	0.20	15.70	11952.95	5.79	8.58	491.02	25752.29

TABLE II: Comparison of the algorithms for mGSP.

GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem," *4OR*, vol. 9, pp. 189–209, 2011.

[11] N. Mladenović, D. Urošević, and S. Hanafi, "Variable neighborhood search for the travelling deliveryman problem," *4OR*, pp. 1–17, 2012.

[12] M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi, "A simple and effective metaheuristic for the Minimum Latency Problem," *European Journal of Operational Research*, vol. 221, no. 3, pp. 513–520, 2012.

[13] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson, "A multi-robot strategy for rapidly searching a polygonal environment," in *Advances in Artificial Intelligence - IBERAMIA 2004*, ser. Lecture Notes in Computer Science, C. Lematre, C. A. Reyes, and J. A. Gonzlez, Eds., vol. 3315. Springer, 2004, pp. 484–493.

[14] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis, "Searching a fixed graph," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, F. Meyer and B. Monien, Eds. Springer Berlin Heidelberg, 1996, vol. 1099, pp. 280–289.

[15] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela, "On salesmen, repairmen, spiders, and other traveling agents," in *Algorithms and Complexity*, ser. Lecture Notes in Computer Science, G. Bongiovanni, R. Petreschi, and G. Gambosi, Eds. Springer Berlin Heidelberg, 2000, vol. 1767, pp. 1–16.

[16] M. Kulich, J. J. Miranda-Bront, and L. Přeučil, "A meta-heuristic based goal-selection strategy for mobile robot search in an unknown environment," *Computers & Operations Research*, 2016, in press.

[17] A. Sathyan, N. Boone, and K. Cohen, "Comparison of approximate approaches to solving the travelling salesman problem and its application to uav swarming," *International Journal of Unmanned Systems Engineering*, vol. 3, no. 1, pp. 1–16, 2015.

[18] N. Boone, A. Sathyan, and K. Cohen, "Enhanced approaches to solving the multiple traveling salesman problem," in *AIAA Infotech@ Aerospace*. American Institute of Aeronautics and Astronautics, 2015.

[19] S. Geetha, G. Poonthalir, and P. Vanathi, "Improved k-means algorithm for capacitated clustering problem," *INFOCOMP Journal of Computer Science*, vol. 8, no. 4, pp. 52–59, 2009.

[20] J. Faigl, M. Kulich, and L. Přeučil, "Goal assignment using distance cost in multi-robot exploration," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ Int. Conf. on*, oct. 2012, pp. 3741 –3746.

[21] G. Reinelt, "Tsplib - a traveling salesman problem library." *INFORMS Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[22] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, 1973.

[23] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

Semantic Visual SLAM in Populated Environments

L. Riazuelo, L. Montano and J. M. M. Montiel

***Abstract*— We propose a visual SLAM (Simultaneous Localization And Mapping) system able to perform robustly in populated environments. The image stream from a moving RGB-D camera is the only input to the system. The computed map in real-time is composed of two layers: 1) The unpopulated geometrical layer, which describes the geometry of the bare scene as an occupancy grid where pieces of information corresponding to people have been removed; 2) A semantic human activity layer, which describes the trajectory of each person with respect to the unpopulated map, labelling an area as "traversable" or "occupied".**

Our proposal is to embed a real-time human tracker into the system. The purpose is twofold. First, to mask out of the rigid SLAM pipeline the image regions occupied by people, which boosts the robustness, the relocation, the accuracy and the reusability of the geometrical map in populated scenes. Secondly, to estimate the full trajectory of each detected person with respect to the scene map, irrespective of the location of the moving camera when the person was imaged. The proposal is tested with two popular visual SLAM systems, C2TAM and ORBSLAM2, proving its generality. The experiments process a benchmark of RGB-D sequences from camera onboard a mobile robot. They prove the robustness, accuracy and reuse capabilities of the two layer map for populated scenes.

I. Introduction

State of the art visual SLAM (Simultaneous Localization And Mapping) algorithms rely heavily on the rigidity prior, which assumes that each image is filled with a that is mostly rigid and persistent. To exploit the rigidity prior, the algorithms include a RANSAC-like voting stage to blindly detect and remove any dynamic element from the mapping process. This delivers a nice robust performance in mostly rigid scenes. Unfortunately, the observation of populated environments implies non-persistent scene changes and scene regions with severely non-rigid motions. The resulting images might be filled with non-rigid and non-persistent elements where the voting algorithms will fail. Our goal is to develop a VSLAM (Visual SLAM) system able to map populated scenes.

Our proposal is to detect and track people in each frame of a video stream (See Fig. 1), Our first contribution is to mask out of the SLAM processing those image regions corresponding to human activity. Hence, we restore the validity of the rigidity prior for the non-masked image regions that can now be processed successfully with the standard VSLAM algorithms. The masked frames are fused in a map that only represents the geometry of the non-human rigid elements in the scene. We refer to this as the *unpopulated map*. In our proposal, the unpopulated map is a dense occupancy grid computed as an Octomap [1] resulting from the fusion of a selected set of RGB-D keyframes which are accurately located in a common reference after the Bundle Adjustment of a sparse point feature matches [2], [3].

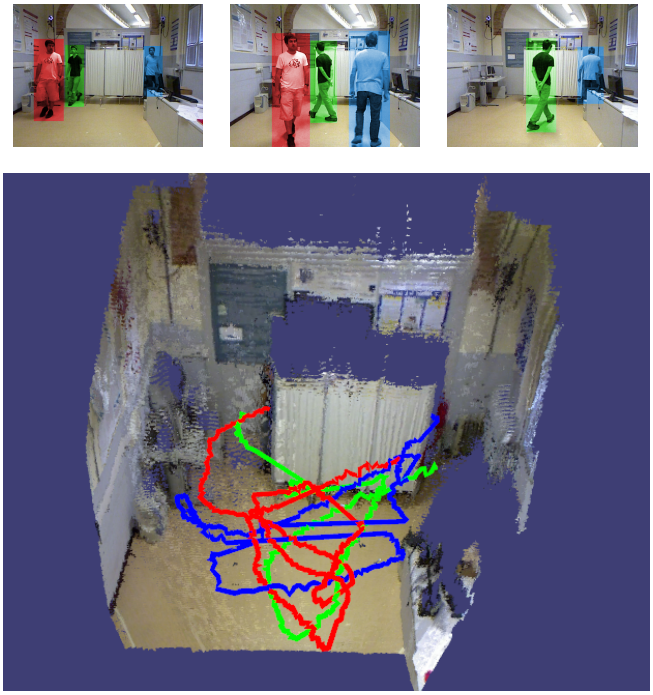


Fig. 1: (Top row) Typical frames of a populated scene. The human tracker detections are overlaid in colours identifiying the person. (Bottom row) the two layers map. The geometrical unpopulated map is a dense occupancy map after removing people populating the scene. The human activity semantic layer is the set of trajectories of the detected persons with respect to the unpopulated map.

Our second contribution is to exploit the human detection and tracking in each frame of the video stream to build a *semantic human activity layer* that registers the trajectory of each person entering within the limits of the unpopulated map. This is the result of combining the accurate camera location that the unpopulated map provides with the human detection in real-time at frame rate. The scene is observed with a mobile RGB-D camera, but the people trajectories are estimated on the map, irrespective of where the camera was when the person was imaged (See Fig. 1.) The semantic layer can be exploited for autonomous robot navigation and

*This research has been funded by Spanish Government grants DPI2016-76676_R_AEI/FEDER-UE, DPI2015-67275-P and the Aragón regional grant DGA-FSE (grupo T04). The authors are with the Robotics, Perception and Real-Time Group, Aragón Institute of Engineering Research (I3A), Universidad de Zaragoza, 50018 Zaragoza, Spain. {riazuelo, montano, josemari}@unizar.es

planning in populated environments.

The remainder of the paper is organized as follows: Section II discusses related work. Section III describes our system. Section IV presents the experimental results and Section V summarizes the main conclusions.

II. Related work

Most semantic mapping approaches are focused on static objects instead of moving objects. In [4] and [5] a 3D laser scan is used for enhancing the map with semantic information. Using 3D points from a stereo camera, [6] builds a map and enriches it by inserting CAD models corresponding to the detected objects. Combining object recognition and visual SLAM in order to produce semantic maps has been extensively studied in recent years. [7] and [8] merged a monocular SLAM system and object recognition for enriching the map. In [9] a combined approach is also presented, the authors not only compute the position of the objects in the map, but also add the objects to the optimization process. A recent work [10] incorporates the use of depth information for mapping and object recognition of object instances. [11] also uses RGB-D sequences, but goes further because the goal of the recognition is not object classes but categories. They create consistent 3D semantic reconstruction of indoor scenes and categorize each voxel in real-time, assigning a semantic label to the voxel according to a specified category. However, removing dynamic objects and annotating them is not a problem which has been studied in depth. OctoMap [1] blindly removes moving objects due to its probabilistic nature; it eventually filters out moving objects, but it does not contain semantic information about the removed objects. In contrast, our approach first identifies objects by categories and then removes them. It is even able to deal with stationary people. [12] also deals with moving objects using a volumetric representation, but they do not add semantic information about these objects to the map. A preliminary work to identify and remove moving objects finding corresponding in two views is presented in [13].

Navigating in the presence of humans requires knowledge of people's movements. In [14] a collection of approaches to human-aware navigation are presented. [15] presents real-time people perception framework, using detectors based on laser and RGB-D data and a tracking approach able to fuse multiple detectors.

The study of human motion patterns has become increasingly significant in recent decades. [16] presents a method for classifying regions for human movements and [17] proposes a method to recognize and predict people's path using a pre-trained SVM as a classifier. In [18] an approach for modeling the dynamics of human movements with a grid-based representation is presented. However on these methods the sensor remains always static in the scene in contrast to our approach.

Human detection and pose estimation in populated environments is a problem studied in considerable depth in the literature [19], [20], [21]. On the subject of articulation, [22] proposes a fully connected graphical model for representing articulated models. [23] describes a general method for human pose estimation in static images based on a representation of part models. A generic approach based on the pictorial structures framework for articulated pose estimation is presented in [24].

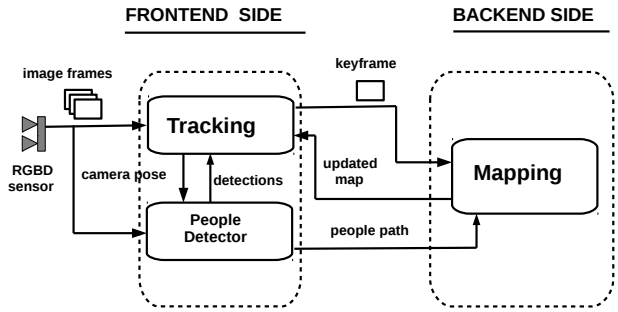


Fig. 2: Typical VLSAM parallel architecture and people detector integration.

[25] proposes a mathematical framework to integrate SLAM and moving objects whith the use of depth sensors. [26] presents an approach close to ours as they mask detected people from the geometrical processing. However instead of building a full SLAM map they only compute visual odometry. Furthermore, additionally their focus is on people tracking while neglecting the scene unpopulated map.

Our proposal is a system paper that builds strongly on state of the art visual SLAM systems processing RGB-D images, C2TAM [2], and ORBSLAM2 [27]. Regarding people detection we integrate [28] because of its real-time RGB-D detection and tracking for mobile robots and head-worn cameras.

III. System Description

Klein and Murray proposed the ground-breaking PTAM architecture for purely RGB monocular sequences in [29]. This is still the basis of state-of-the-art VSLAM systems also monocular [3] or RGB-D C2TAM [2] or ORBSLAM2 [27]. Our proposal also builds on top the RGB-D ones. Although our approach is agnostic to the VSLAM approach, for the aim of simplicity we focus the system description on ORBSLAM2. The architecture is based on two intertwined processes running in parallel. The two processes are generically named as *frontend* and *backend*, see Fig. 2. The frontend is focused on a minimal set of operations to provide the camera location in real-time, it assumes that the backend has provided a map of the scene. The backend concentrates the expensive mapping operations to estimate the scene map by means of non-linear optimization, it iterates at low frequency providing fast –but not real-time– updates of the estimated map.

The real-time camera pose is a tracking algorithm which assumes that the camera trajectory is smooth. If it gets lost, for example due to camera occlusion or motion blur, then the camera location has to be located from scratch before resuming tracking. This recovery stage is called *relocation* in the VLSAM literature, and it is also a standard component

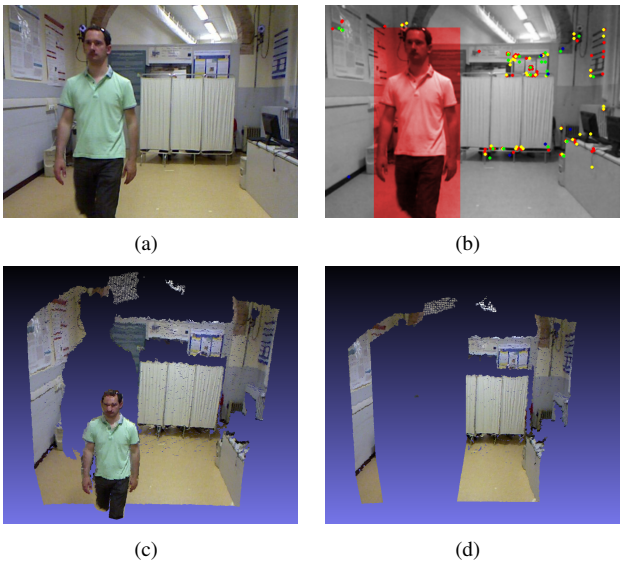


Fig. 3: (a) Raw RGB. (b) Interest point detection and human activity masking. (c) Raw RGB-D depth channel. (d) RGB-D point depth channel after removing people depth points.

of contemporary SLAM systems.

Our proposal integrates a real-time people detection and tracking stage [28] in the system to enable its operation in populated environments. The people detection interacts with all the components and stages of the visual SLAM. We describe below the modified frontend, backend, relocation and human activity detection layer of the map.

A. Frontend process

The frontend processes each RGB-D camera frame in real-time to provide the camera location. Its main component is the *camera tracking*, which assumes that both an accurate sparse 3D point map of the scene and an estimated camera position, from which putative point matches between the current frame and the map are estimated. Then, the camera pose is estimated by non-linear minimization of the map reprojection error. The optimization is cheap because only the six d.o.f of the camera pose are optimized. The map points are not optimized because their estimate locations, provided by the backend, are assumed to be perfect.

The reprojection error includes a robust influence function [30] that marks as an outlier any match with a big reprojection error. If the outlier rate exceeds 50% of the computed matches, the camera location estimate is very likely to break down. Outliers correspond to mismatched points or to matches not following the rigidity prior. Hence, human activity makes the camera estimation more likely to fail because it generates outlier matches that increase the outlier fraction.

We propose to identify the frame regions which image human activity in order to mask them out of the camera tracking processing. The first step of the tracking is to detect interest points in the current frame, typically FAST [31]. We additionally process the frame with a real-time people tracker [28] that yields a bounding box per each detected person. All the points of interest within the bounding boxes are masked out for the subsequent matching stage (Fig. 3 (b)). By masking out the FAST points on humans, we remove points that are probably outliers, reducing the proportion of outliers and hence increasing the probability of a successful camera location estimate.

The people tracker provides a 3D position of each detected person with respect to the RGB-D camera frame simply by reading the depth channel. Additionally, we have ready available each person trajectory referred to the static scene map frame, by just composing the person position with respect to the camera with the camera pose provided by the camera tracking. This mechanism to compute the people tracks referred to the map is the main ingredient of the semantic people activity layer of the map.

B. Backend process

The backend concentrates all the compute-intensive or non-time critical operations derived from the mapping estimation. Its main component is a non-linear iterative optimization, after each optimization step, the map provided to the frontend is updated. It is only possible to achieve an update rate significantly lower than the frame rate, but fortunately it is acceptable.

The backend exploits the fact, well-known since the early times of photogrammetry, that given the points correspondences along a monocular image sequence both the camera poses and the 3D point positions can be estimated up to scale factor, for a rigid scene. The gold-standard algorithm is non-linear minimization of the reprojection error, known in the literature as Bundle Adjustment (BA) more specifically, the iterative optimization algorithm used is the Levenberg-Marquardt. We use an extension in the case of stereo cameras [27].

PTAM-like algorithms exploit the fact that not all the frames in the sequence have to be included in the BA, but only a small fraction of them, known as keyframes. Different heuristics have been proposed for keyframe selection, for example in [29], [3], [2]. The heuristics application results in a reduced set of keyframes that cover the imaged scene while having enough overlapping to ensure that all the points in the map are imaged from several images with a significantly different point of view in order to render parallax, ensuring a good geometrical conditioning for BA. There is overwhelming experimental evidence of the efficacy of these heuristics in the keyframe selection. The keyframe selection is made in the frontend.

The main challenges that BA has to face are the outliers, the local minima, and the high number of iterations to converge to the minimum. All these drawbacks can be overcome if an initial guess close to the solution is available. The intertwining between the frontend and the backend is responsible for mutually providing the necessary initial guesses for the non-linear optimization. When the camera explores new scene areas not yet included in the map, the frontend sends a new keyframe to the backend, with

	Translation Sequence				Arc Sequence			
	ORBSLAM2		C2TAM		ORBSLAM2		C2TAM	
	ATE (mm)	% Frames Tracked	ATE (mm)	% Frames Tracked	ATE(mm)	% Frames Tracked	ATE(mm)	% Frames Tracked
No Masking	53.3	92.3	42.6	95.9	39.2	97.0	41.7	93.5
Ground-Truth Masking	15.9	90.7	16.7	92.3	19.6	98.6	28.6	94.5
People Detector Masking	22.6	92.5	20.2	94.3	21.2	99.2	29.1	87.5

TABLE I: System performance over KTP dataset sequences using ORBSLAM2 and C2TAM.

an initial guess for its location. A low spurious rate set of matches between the new keyframe and all the other keyframes already in the map is also available. From this initial information, matches for new points to expand the map can be robustly found exploiting the scene rigidity. Again, masking out the keyframe regions corresponding to human activity helps to reduce the outlier rate in the new matches, and hence increases the robustness. Given the new camera pose estimate, initial guesses for the points newly added to the map can be also estimated. Thanks to the initial guesses the convergence for the newly added camera and map points is fast.

The sparse map M is the result of the BA. It is composed of a set of l keyframes $\{K_1, K_2, \ldots, K_l\}$ and n 3D sparse feature points $\{P_1, P_2, \ldots, P_n\}$. Per each keyframe, it estimates the camera position coded as the rigid transformation T_{WK_j} referred with respect to world frame W. The keyframe also contains a the camera depth channel of the image, which in contrast to the sparse points of the maps can provide dense depth information of the scene. Once we have accurate keyframe poses, T_{WK_j} from the BA, we use the Octomap algorithm [1] to fuse all the depth maps into a single occupancy map. Prior to the fusion, the depth regions corresponding to human activity are also masked out from the keyframes, so that the generated Octomap does not contain the people populating the scene. This people-removal process does not have to be in hard real-time, so it is performed in the backend and only on the keyframes. The unpopulated map could be easily reused in future reobservations of the same scene, irrespective of the present and future the human activity.

C. Camera Relocation

As mentioned above, relocation is mandatory after the loss of camera tracking, and it is necessary to be able to reuse a previously available map. The ORBSLAM2 relocation algorithm is currently the state of the art. Its first step is an indexed search for matches known as DBoW [32]. The search for matches between the current image and the available map mimics a query in a database, where the current image is the query, and the dabatase is holds the map points. In the second stage the camera pose is recovered using a RANSAC-like approach. In our proposal, during the relocation step, we mask out the detected people from the current image before querying the DBoW database. It has to be considered that in building the database, the regions corresponding to human activity were also masked out from the map, hence also from the database. We can conclude that human activity detection and removal extends the lifespan of map reuse, because the number of spurious matches is reduced both from the database and from the queries.

D. People detection and human activity layer

To deal with human activity we propose to integrate a people detection and tracking stage, in our case we use [28]. We process all the images coming from the RGB-D camera. Each detected person in an image is coded as an image bounding box and a label is created that uniquely identifies the person. The 3D position of the person with respect to the camera is computed as the position of the center of the bounding box, and the depth channel. All these pieces of information are computed and stored in the frontend.

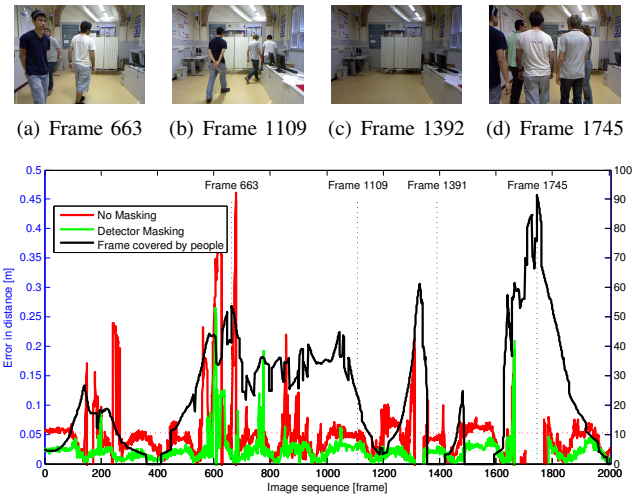
When the frontend decides to create a new keyframe, it sends the keyframe to the backend. Additionally all the information registered for all the frames since the last keyframe is also sent to the backend. The pieces of information included are the frame poses and all the instances of people detections.

The backend generates the human activity semantic layer, composed paths of each person with respect to the unpopulated static map, irrespective of where the camera was when the people were detected. These trajectories are computed by the composition of the 3D position of each person with respect to the camera with the 3D camera pose computed by the frontend. After each iteration of the backend, the position of each keyframe is updated. We keep the connection of the intermediate frames with the keyframes to propagate the position updates of the keyframes to the intermediate frames, and from there to each person's trajectory, which is updated at same rate as the unpopulated map. This information adds a semantic label to activity layer of the map as a "occupied area" by people. This semantic information will be use by a mobile robot for navigating in the same environment.

IV. Experiments

In this section, we present the experiments performed for validating our approach. The system is implemented in C++ using Robot Operating System (ROS) [33]. An Intel Core i7 @2.67GHz processor was used for running the VSLAM system and the people detector at a 24 fps rate. For validating the approach we have used the Kinect Tracking Precision Dataset (KTP) presented in [34]. This contains 5 different sequences acquired with a Microsoft Kinect on board a mobile platform. We have selected this dataset because it provides RGB-D images in a context where a robot makes a trajectory and also includes a ground-truth of both the position of the robot and the detections of the people. Regarding the VSLAM system, we have tested our approach using two different PTAM-like algorithms adapted for RGB-D, C2TAM [2] and ORBSLAM2 [27]. Both of them use [28] for human activity detection.

Table I presents the results of both VSLAM algorithms running over two of the KTP sequences: "Translation" and



(a) Frame 663 (b) Frame 1109 (c) Frame 1392 (d) Frame 1745

Fig. 4: Effect of human activity masking per each frame of the sequence "Translation" processed by ORBSLAM2. Top row displays a representative case.

"Arc". The metric selected for measuring the performance of the system is the Absolute Trajectory Error (ATE) after aligning the computed trajectories with the ground truth by means of a rotation and translation. We also include the rate of frames successfully located, named "% Frames Tracked". We run the sequences with three different configurations for human activity detection: *No masking*, in which human activity detection and masking is deactivated; *Ground-Truth Masking*, where we use the ground truth bounding boxes detections provided by KTP, in order to find an upper performance limit; and *People Detector Masking* by which the system is fed with the real people detection provided by [28].

First of all we can see how the frame tracked percentage is very similar for all the configurations. However the ATE error is reduced by more than half if human activity masking is applied to the images, which proves the benefit of masking. Also we can see a small increase in error depending on whether we use the ground-truth detections or the people detector. This increase in the error is minimal, proving that we are close to the upper performance limit.

Figure 4 presents in detail the evolution of the error during the whole sequence for the "Translation" and ORBSLAM2 case. The plot displays the error with people detector masking (green line) and with no masking (red line). The area of the image covered by people is also plotted, as an index of the human activity. The camera location error is highly correlated with the level of human activity. Four representative frames are selected in order to visualize this correlation. In the frames 4(a) (#663) and 4(d) (#1745) we can see how the error increases due to the increment in human activity. The higher the occlusion level of the image caused by the people in the scene, the lower the system accuracy. There are even situations in which the occlusion level is so high (Frame 4(d) (#1745)) that the system cannot be located and the camera is lost. In addition we can see in frame 4(b) (#1109) how

	Translation Sequence		Arc Sequence	
	ATE (mm)	% Frames Relocated	ATE (mm)	% Frames Relocated
NoMasking	51.2	87.5	32.7	78.8
Ground-Truth Masking	16.2	76.8	21.6	56.7
People Detector Masking	17.8	75.2	22.5	37.2

TABLE II: Camera relocation performance over KTP dataset sequences using ORBSLAM2.

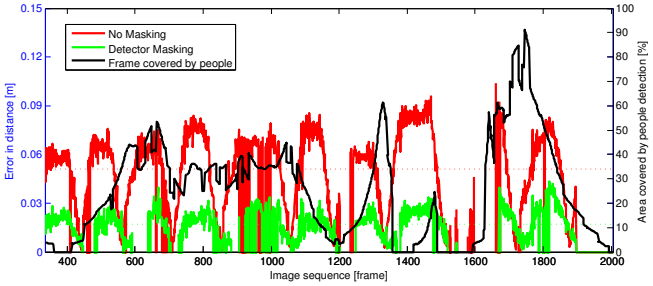


Fig. 5: Effect of human activity masking on relocation. ATE after relocation per each frame of the "Translation" sequence processed by ORBSLAM2.

the human activity level decreases and hence the error also decreases. When there are no people in the image (Frame 4(c) (#1391)) we can see in the figure how the error is maintained or even decreases.

For validating the camera relocation performance we have run KTP dataset "Translation" and "Arc" sequences over ORBSLAM2 because its relocation algorithm based on bag of words is on top of the state of the art. We have run the VSLAM system on the first part of the scene (about 300 frames) during the scene exploration stage. Afterwards we close the map and force a camera relocation for each frame in the rest of the sequence. Table II presents the results of the camera relocation performance for different options of masking, applied both to the map creation and the subsequent relocation. We can see the improvement in the ATE metric when human activity is masked. Regarding the use of the ground-truth for masking the people, the ATE metric is quite similar to that achieved by the implemented detector. We can also see how the percentage of relocated frames drop with people masking, this is due to with no masking some frames are not relocated correctly and the number of relocated frames increases.

Figure 5 presents the evolution of the relocation error with (green line) and without (red line) masking people. In this case we can also see the correlation between the error and the area of the image covered by people. In the frames in which the image is covered by people, the error is greater than in which are without people. In addition, depending on the level of occlusion, the relocation algorithm is unable to provide a position. We can conclude that using a masking technique improve the camera relocation performance.

Regarding the geometrical layer, figure 6 displays the improvement provided by our C2TAM approach after the initial exploration of the scene ($\approx$ 350 first frames). Due to

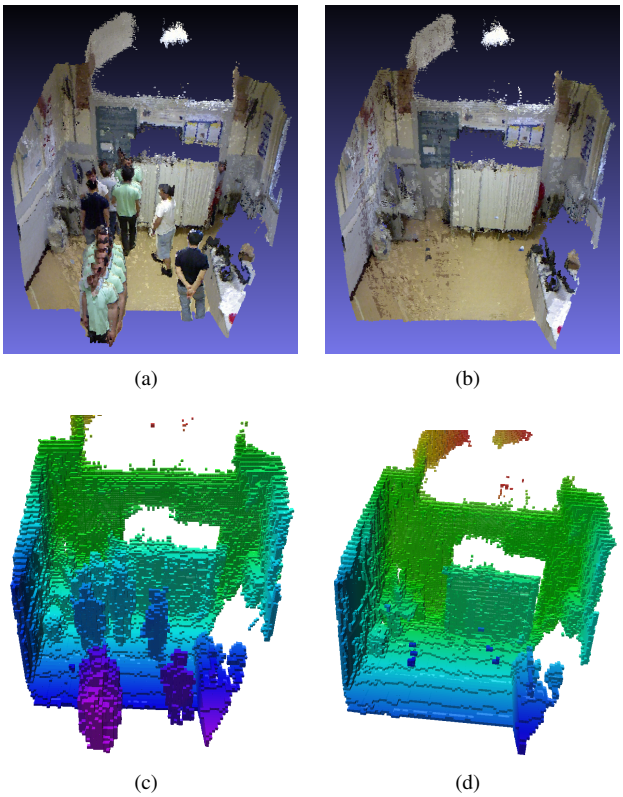


Fig. 6: 3D reconstruction and Octomap using unmasked 3D point cloud (a,c), and using the human activity masked data (b,d).

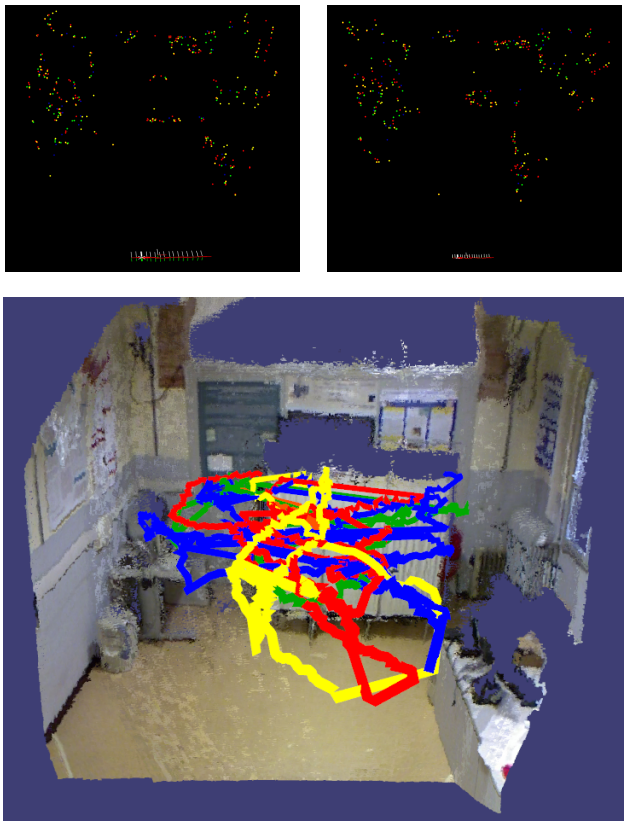


Fig. 7: Sparse map and camera trajectory (top), and geometrical and human activity layer (bottom).

the information provided by the people detector about the estimated position of the people in the images, a process of annotation and removing the people from the 3D point cloud is performed. Figure 6(b) shows the result of this process. The number of points in the scene is reduced in a 30%, and there is not any evidence of people in the scene, in contrast to raw the reconstruction shown in figure 6(a) where artifacts due to people are noticeable.

Comparing the initial texture alignment with the octomap approach, we can see in the figure the maps obtained. In figure 6(c), an octomap is built using the raw data from the 3D point cloud associated to each keyframe witout masking. In contrast, using our approach (figure 6(c)), the octomap built masking the people information does not even include static persons.

Figure 7 summarizes the results obtained by the VSLAM system proposed. Top images show the sparse map (front and top view) generated by C2TAM after masking. The map is used for locating the RGBD camera in the scene. We can see the feature points extracted from the rigid scene and the camera trajectory followed by the robot during the experiment. In the bottom image, we can also see the unpopulated map that describes the 3D geometry where information corresponding to the people has been removed. In addition, the people trajectories that define the semantic layer are also displayed. A full video of the validation can be found in [1].

V. Conclusions

Thanks to the embedding of human detection in a rigid visual SLAM system, it is possible to build precise VLSAM maps in populated environments.

Our system integration approach can effectively remove human activity from an unpopulated map simply by removing the people from the input sequence. People detection and removal not only benefits the scene description, it also benefits the camera tracking and relocation, significantly improving their performance in populated environments, and reducing the ATE error by more than half. Additionally it is possible to generate a semantic layer that exhaustively registers human activity in the mapped area, irrespective of the camera motion during acquisition. We can conclude that both the mapping and the people tracking benefits from the combination. In addition, the system is able to deal with the false positives provided by the person detector, since the detection is performed in all frames of the sequence.

We have tested the embedding in two popular rigid VSLAM systems, with a similar gain in performance. We can therefore conclude that our proposal is agnostic to the VLSAM method, and hence can potentially improve any SLAM system. In the same way, we can say that the combination of

[1] http://robots.unizar.es/data/videos/ecmrRiazuelo17

this method map building with the recognition of people is agnostic to the algorithms used, since any detector that give us information of the person in the scene can be included in the proposed system.

People detection can be considered as an instance of class category recognition and tracking. A similar benefit in terms of an increase in the robustness, map reuse and tracking of the instances irrespective of the camera motions caould be achieved in other environments, for example vehicle detection for automated driving, or a tool detector for endoscope-guided minimally invasive surgery.

Mapping populated environments where human movements are prevalent is a key capability for service robots. As future work we plan to develop robot motion planners in such environments, profiting from the human activity mapped. Planning and navigation using this information can be improved by the knowledge of the patterns of human movement.

REFERENCES

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[2] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C^{2}tam: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 915–926, Nov. 2008.

[5] N. Goerke and S. Braun, "Building semantic annotated maps by mobile robots tracking," *Towards Autonomous Robotic Systems*, pp. 149–156, 2009.

[6] M. Günther, T. Wiemann, S. Albrecht, and J. Hertzberg, "Building semantic object maps from sparse and noisy 3d data." in *IROS*. IEEE, 2013, pp. 2228–2233.

[7] R. Castle, G. Klein, and D. Murray, "Combining monoSLAM with object recognition for scene augmentation using a wearable camera," *Image and Vision Computing*, vol. 28, no. 11, pp. 1548–1556, 2010.

[8] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic slam using a monocular camera," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, sept. 2011, pp. 1277 –1284.

[9] D. Gálvez-López, M. Salas, J. D. Tardós, and J. M. M. Montiel, "Real time monocular object slam," *Robot. Auton. Syst.*, Accepted for publication, 2015.

[10] R. Salas-Moreno, F. R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *IEEE Proc. Computer Vision and Pattern Recognition (CVPR)*, 2013.

[11] A. Hermans, G. Floros, and B. Leibe, "Dense 3d semantic mapping of indoor scenes from RGB-D images," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 2631–2638.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136.

[13] K. Litomisky and B. Bhanu, "Removing moving objects from point cloud scenes." ser. Lecture Notes in Computer Science, X. Jiang, O. R. P. Bellon, D. B. Goldgof, and T. Oishi, Eds., vol. 7854. Springer, 2012, pp. 50–58.

[14] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1726–1743, Dec. 2013.

[15] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide, "Real-time multisensor people tracking for human-robot spatial interaction," in *Workshop on Machine Learning for Social Robotics at International Conference on Robotics and Automation (ICRA)*. ICRA/IEEE, 2015.

[16] Z. Wang, P. Jensfelt, and J. Folkesson, "Building a human behavior map from local observations," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2016, pp. 64–70.

[17] S. Xiao, Z. Wang, and J. Folkesson, "Unsupervised robot learning to predict person motion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 691–696.

[18] Z. Wang, P. Jensfelt, and J. Folkesson, "Multi-scale conditional transition map: Modeling spatial-temporal dynamics of human movements with local and long-term correlations," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 6244–6251.

[19] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *Int. J. Comput. Vision*, vol. 63, no. 2, pp. 153–161, Jul. 2005.

[20] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 878–885.

[21] K. Mikolajczyk, B. Leibe, and B. Schiele, "Multiple object class detection with a generative model," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, June 2006, pp. 26–36.

[22] M. Bergtholdt, J. Kappes, S. Schmidt, and C. Schnörr, "A study of parts-based object class detection using complete graphs," *Int. J. Comput. Vision*, vol. 87, no. 1-2, pp. 93–117, Mar. 2010.

[23] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, 2011, pp. 1385–1392.

[24] M. Andriluka, S. Roth, and B. Schiele, "Pictorial Structures Revisited: People Detection and Articulated Pose Estimation," 2009.

[25] C. Wang, C. E. Thorpe, S. Thrun, M. Hebert, and H. F. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *I. J. Robotic Res.*, vol. 26, no. 9, pp. 889–916, 2007.

[26] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "Robust multiperson tracking from a mobile platform," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 10, pp. 1831–1846, 2009.

[27] R. Mur-Artal and J. D. Tardos, "Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras," *arXiv preprint arXiv:1610.06475. To appear in IEEE Trans. on Robotics.*, 2016.

[28] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 5636–5643.

[29] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.

[30] P. J. Huber, *Robust Statistics*. Wiley-Interscience, 1981.

[31] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer vision–ECCV 2006*, pp. 430–443, 2006.

[32] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[34] M. Munaro, F. Basso, and E. Menegatti, "Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks," *Robot. Auton. Syst.*, vol. 75, no. PB, pp. 525–538, Jan. 2016. [Online]. Available: https://doi.org/10.1016/j.robot.2015.10.004

Mapping likelihood of encountering humans: application to path planning in crowded environment

Fabrice Jumel[1,2] and Jacques Saraydaryan[1,2] and Olivier Simonin[1,3]

***Abstract*— An important challenge for autonomous robots is to navigate efficiently and safely in human populated environments. It requires that the robots perceive human motions and take into account human flows to plan and navigate. In this context we address the problem of modeling human flows from the perception of the robots, by defining a grid of the human motion likelihood over the environment, called flow grid. We define the computation of this grid as a counting based mapping. Then we define a path planning taking into account the risk of encountering humans in opposite direction. We first evaluate the approach in simulation by considering different navigation tasks in a crowded environment. For this purpose, we compare three A*-based path planning models using different levels of information about human presence. Simulations involving 200 moving persons and 4 collaborative robots allow to evaluate simultaneously the flow mapping and the related path planning efficiency. Finally we experiment the model with a real robot that maps human displacements in its environment.**

I. Introduction

Deploying robots in human environments has become a central challenge in robotics. Many applications require to define robot behaviors that are able to take into account the presence of humans and to interact with them. Robot have to perform their task while being safe for humans and aware of their social rules [12].

In this paper, we focus on robotic path planning and navigation in environment where many people are moving. We are interested to deal with the problem of identifying where humans regularly move/appear, in order to optimize their paths among humans. In the robot workspace, the human activity makes the environment very dynamic, it is then essential to identify the recurring displacements (the regularities). We consider that robots know the static part of the environment (e.g., a metric map) and are able to localize themself. SLAM techniques allow to make such an assumption [14].

By detecting displacements of humans, we aim at mapping human flows in dense populated environments. This requires to be able to detect human motions around the robots, by considering only the data coming from the embedded sensors. Current sensors, camera or LIDAR, allow to detect human's location and velocity. We do not consider that the environment is equipped with external fixed cameras or sensors. Therefore, robots have to move in order to cover the environment and to update their knowledge of the human activity. We propose to incrementally compute a grid, called *flow grid*, where each cell will contain a set of likelihood that represents represent human motion in different directions. This differs from standard grid approaches that models only human presence likelihood, as in [15], or that generates a flow map after having collected data.

However, building a map of human flows from robot perceptions can require a long time of observation. This leads us to complete observations with motion patterns that allow to extend/anticipate the model of flows.

Computing a representation of human displacements and flows aims to optimize robot navigation in dense populated environment. We evaluate this ability by comparing three path planning models exploiting different maps of the likelihood of encountering humans.

The paper is organized as follows. Section II presents related work to the problem of modeling and predicting human motion and flows. In Section III we introduce the flow-grid model and the predictive flow grid. In Section IV we define three path planning models based on different representation of human presence in the environment. Section V presents the experimental scenarios and compare results with the different path planning models. Then Section VI gives the first results of computing the flow grid with a real robot in a human populated environment. Finally, in Section VII we conclude the paper and draw up the perspectives.

II. Related Work

Since two decades, a lot of work has been dedicated to the detection and the prediction of moving objects. This defines the problem of tracking moving targets. The most common approaches use techniques such as Kalman filters, HMM and more generally Bayesian models [11]. The problem is more challenging when the tracking is performed from a moving robot. We can mention the HSBOF model [3] which is able to detect and predict object and people motion from a camera embedded on a moving vehicle.

Among these works, some of them are specialized in tracking human motion, as [16]. For this purpose they can use models of human behavior, established by psycophysiologists, as in the work of [12].

Even recent works are concerned with tracking simultaneously several persons [10], few focus on modeling human flows. A recent review [4] presents social-aware navigation frameworks and compares them. As it is highlighted by authors, few work are focusing on global social-aware planning.

[1]Laboratoire CITI-Inria, Chroma team, 6 avenue des Arts, 69621 Villeurbanne

[2] Universite de Lyon, CPE Lyon, Domaine scientifique de la Doua, 69100, Villeurbanne, France firstname.name@cpe.fr

[3] Universite de Lyon, INSA Lyon, Inria, 20 Avenue Albert Einstein, 69100, Villeurbanne, France olivier.simonin@insa-lyon.fr

In this paper we focus on modeling flows of human, or recurring human displacements, which can be seen as a long-term task. Much work has been carried out to simulate crowds of people (see for instance [6]), but few has considered the problem of representing such flows from the perceptions of a robot (ie. with embedded sensors).

Recent works have considered the problem of learning human flows from one or several static cameras or LIDAR. In particular we can mention the work of *Tipaldi and Arras* [15] defining the spatial affordance map. The model learns the spatio-temporal distribution of events, from the observations in each cell representing the environment. The main limit of such an approach is to only model human presence, without information about motion of human flows. Then *Kucner et al.* introduced Conditional Transition Maps [9], which models the probability of human transition between neighbouring cells of a grid representing the environment. However, this approach requires to determine from which cell the observed person arrives, its current cell, and its next cell. This can be achieved with static cameras overhanging the environment, it is more difficult with embedded camera on moving robots. We note also that the combination of transitions requires to learn 64 parameters per cell. An extended approach [19] allows long-term spatial correlations but proposes to reduce the neighboring transitions to 4 directions (north, south, east, west).

In [17] and [18] *Wada et al.* have shown that human activity, in particular walking, can be observed and mapped from embedded sensors and with SLAM techniques. After collecting observations from each region of the environment they can generate a human motion grid, giving in each cell the statistics of walking direction in every 15 degree [18]. This approach does not consider an online computation of this flow model from the observation of moving robots, as we propose in this paper.

The method we propose can be seen as an extension of the affordance maps, where we add in each cell the modeling of human motion for a set of directions (8 in practice). This allows us to build incrementally a grid of human motion likelihood. This process is done simultaneously by several robots that share the computed grid.

III. Modeling human flows

A. Approach overview

We adopt a grid representation to discretize the flow and to limit the information to learn. The objective is to build a grid where each cell holds the likelihood of encountering humans.

We first define a two layer model of the human flows:

- a **flow grid** modeling in cells the likelihood to observe a human moving in a set of directions.
- a **prediction flow grid** that estimates the human flow from the flow grid.

The general idea is to update the likelihood values of the **flow grid** at each observation of a human presence/motion (counting-based approach). When humans are observed, we also estimate their short-term motion in the **prediction flow grid**. These two grids are illustrated in the Figure 1.A. Now we define them formally.

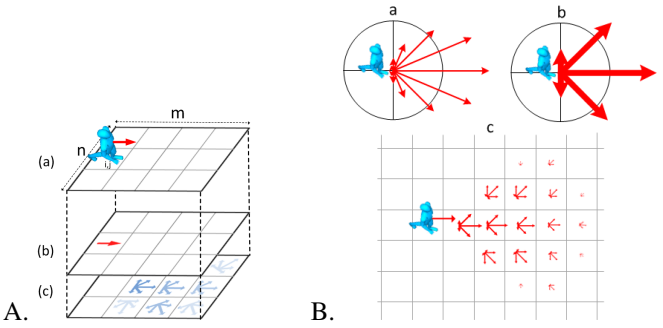


Fig. 1. A. Two layers grid model of the human flows (flow grid and prediction) B. Possible motions from Von Mises Gaussian circle probability distribution: (a) sample of Von Mises probability distribution with $\mathtt{C} = 2$, (b) normalized distribution with $|K| = 8$, (c) prediction direction grid according a given observation.

B. Flow grid layer

In each cell $c_{x,y}$, we discretize the possible flow directions by a set of K directions, and we note $k_i \in K$ each direction. For instance, we can set $K = \{North, East, South, West\}$.

We note Z^t the set of observations performed by all the robots up to time t. **An observation**, in a cell $c_{x,y}$, consists in identifying a human direction, eventually none, and its duration. By hypothesis, only one human can occupy a cell at a given time. In practice, we consider cell sizes from $0.25m^2$ to $1m^2$.

Let note $R = \{r_1, r_2, .., r_n\}$ the set of robots.

We note $t_{c_{x,y}}(r)$ the sum of the durations (in seconds) of all observations performed by the robot r on cell $c_{x,y}$. $t_{c_{x,y},k}(r)$ is the sum of durations of the observation of a human moving in direction k in cell $c_{x,y}$.

We set $M_{flow} = \forall x \forall y \forall k (M_{c_{x,y},k})$ the grid of human motion likelihood in every direction k of each cell $c_{x,y}$. By considering that human flows are stationary processes, each cell value (in $[0, 1]$) is computed (updated) as follows:

$$M_{c_{x,y},k}(Z^t) = \frac{\sum_{n=1}^{|R|} t_{c_{x,y},k}(r_n)}{\sum_{n=1}^{|R|} t_{c_{x,y}}(r_n)} \quad (1)$$

To deal with non stationary processes a forgetting factor could be added (not developed in this article).

As the matrix M_{flow} holds the human motion likelihood in observed cells we can deduce the human presence likelihood, noted M_{pres}, that corresponds to the affordance-map proposed in [15] :

$$M_{pres_{c_{x,y}}}(Z^t) = \sum_{k \in K} M_{x,y,k}(Z^t) \quad (2)$$

C. Prediction flow grid layer

Building the flow grid from local robot observations leads naturally to partial information over the whole environment.

In order to accelerate the flow learning, we add a human motion pattern around the last observations.

Many approaches have been proposed to predict human motion. Generally, it is computed from current velocity and direction observation in addition to the beliefs of the past robot observations, as done in [20]. Moreover, it has been shown that predicting the motion of a pedestrian by a velocity-based linear projection [7] is known to be a reasonable approximation for short term behavior [13].

As we model human direction in cells, the short term prediction is computed by the use of the Von Mises angular probability distribution [8]:

$$\phi(\theta|\mu,\sigma) = \frac{e^{\complement cos(\theta-\mu)})}{2\pi I_0(\complement)} \qquad \frac{1}{\complement} = \sigma^2 \tag{3}$$

where μ is the measure of location, C is the measure of concentration ($\frac{1}{\complement} = \sigma^2$), I_0 is the modified Bessel function of order 0. As it was highlighted in [13], moving object prediction deals with variation of speed and translation/rotation moves. We consider that a human can take a maximum rotation speed of $90°.s^{-1}$ with an average linear speed of $1m.s^{-1}$. Considering these properties and the fact that current cell dimension is $1m$, we use Von Mises equation with a parameter of $\complement = 2$ (see the Figure 1.B).

When a robot observes a human direction at a given location c, the flow grid is updated and the predicted flow grid on this cell is cleared (for each $k \in K$, $M^{pred}_{c,k} = 0$).

The Von Mises angular Gaussian distribution is applied on the neighbor cells and result in a set of possible directions. First step consists to create an initial predicted cell from the last observation and next to apply a Von Mises Pattern computation on the neighborhood of this cell.

The main idea of the Algorithm 1 is to propagate the pattern from cells with maximum probablity presence towards direction with maximum probability (line 6). Therefore, we consider that the number of possible directions is no more higher than the grid cell connectivity (e.g 8 or 4). The propagation stops when computed probability is lower than a threshold (min_value line 16).

The predicted flowgrid of cell c_{ni} in each k_{ni} direction is computed as the probability associated to angle θ_{ni} of a normalized Von Mises distribution times the probabilities of human direction observation $M_{c,k}$:

$$M^{pred}_{c_{ni},k_{ni}}(Z^t) = \phi^{'}_K(\theta_{ni}|\theta_k,\sigma) \times M_{c,k}(Z^t) \tag{4}$$

$$\phi^{'}_K(\theta_{ni}|\theta_k,\sigma) = \frac{\phi(\theta_{ni}|\theta_k,\sigma)}{\sum_{k' \in K} \phi(\theta_{ni}|\theta_{k'},\sigma)} \tag{5}$$

where the Von Mises function is centered on the angle associated to the observed direction $\mu = \theta_k$ with a dispersion $\sigma = \frac{1}{\sqrt{2}}$, and $M_{c,k}(Z^t)$ is the probability of the observation of the direction k on the cell c.

The Figure 1.B(c) shows the result of a Von Mises pattern computation. This distribution is normalized according to the number of possible directions $|K|$ ($K = 8$ on the Figure 1.B(b)). Then the resulting predicted directions are shown on the Figure 1.B(c) where arrow length represents the probability value.

Algorithm 1: Von Mises pattern computation

Input: c_{ni},M_{flow}, M^{pred}_{flow}, min_value
Output: updated M^{pred}_{flow}

1 **Process:**
2 $CellToProcess$ = c_{ni}
3 $VonMisesComputedCell$ = c_{ni}
4 **while** $CellToProcess \neq \varnothing$ **do**
5 c = first element of $CellToProcess$
6 **for** *each* k_i *ordered by decreasing value of* M^{pred}_{c,k_i} **do**
7 c_n = neighbor cell of c in direction k_i
8 **if** $c_n \notin VonMisesComputedCell$ **then**
9 add c_n to $VonMisesComputedCell$
10 **if** $M_{pres_{c_n}}$ *not defined* **then**
11 // Precompute Von Mises on c_n :
12 **for** *each* $k_{ni} \in K$ **do**
13 $M^{precompute,pred}_{c_n,k_{ni}} = \phi^{'}(\theta_{ni}|\theta_{k_i},\sigma) \times M^{pred}_{c,k_i}$
14 **if** $M^{precompute,pred}_{pres_{c_n}} > M^{pred}_{pres_{c_n}}$ **then**
15 $M^{pred}_{pres_{c_n}} = M^{precompute,pred}_{pres_{c_n}}$ //update
16 **if** $M^{pred}_{pres_{c_n}} > min_value$ **then**
17 add c_n to $CellToProcess$
18 sort $CellToProcess$ by decreasing $M^{pred}_{pres_{c_n}}$
19 Remove c from $CellToProcess$

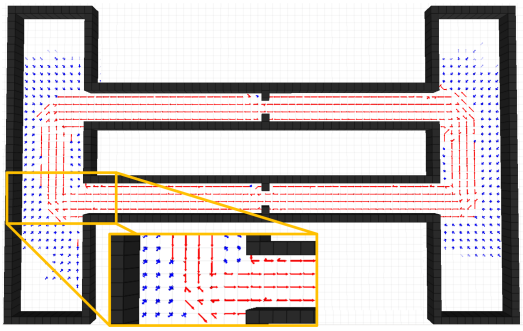


Fig. 2. Flow grid (red) and Prediction flow grid (blue) at 30s

D. Illustration of the flow grid buiding

We use the simulator presented in Section V to illustrate the proposed model. The Figure 2 shows the flow grid built by 4 robots moving in 4 connected corridors where people are walking in the anti-clockwize direction (see Scenario in Section V-B). It appears that the estimated flow corresponds to the motion directions induced by the human trajectories. One can see that predicted motions (in blue) can coherently complete the flow built from the observations (in red).

IV. Path planning in dynamic environment

In this section we show how to adapt the A* path planning computation to take into account different models of human presence in the environment.

A. Cost estimation in human populated environment

We consider navigation techniques based on computing optimal path in "reachability graphs". We use the A* algorithm which is standard for robot path planning.

A* is using an heuristic to control the order of exploration of the cell in the decision process. At each iteration of its main loop, A* determines which of its partial paths to expand into one or more longer paths. This is based on an estimate of the cost still to go to the goal node. A* selects the path that minimizes the function $f(n)$. $f(n) = g(n)+h(n)$ where n is the last node on the partial path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic that estimates the cost of path from n to the goal without knowledge. The heuristic used in this paper, $h(n)$, is the classical euclidean distance from n to the goal without obstacle.

To take into account the presence of humans, we consider three different map-distances to compute the cost function $g(n)$ in A* planning :

- an equiprobable cost, that is the non informed model,
- a human presence estimation, ie. the affordance-map computed as a spatial Poisson process [15],
- a human motion estimation, ie. the flow-grid map introduced in this paper.

1) A algorithm with equiprobable presence:* The cost between 2 neighbors cells n and $(n-1)$ is defined as:

$$g(n) = \|n,(n-1)\| \times (1+F) + g(n-1) \qquad (6)$$

where $\|n,(n-1)\|$ is the distance cost between n and $(n-1)$ and F is a positive factor to represent human presence. As the disturbance due to humans is not know it is considered as identical in all cells. Then F is set as a constant (1 in the experiments).

2) A algorithm with affordance-map:* An affordance map estimates the human presence probability in each cell as a spatial Poisson process [15]. In each cell n, the human presence estimation, noted $Poisson(n)$, is computed as follow: $Poisson(n) = M_{pres_n}$ (see eq. (2) Section III-B)

This allows to define the following cost function :

$$g(n) = \|n,(n-1)\| \times (1+F.Poisson(n)) + g(n-1) \quad (7)$$

We add to the distance a cost directly linked to the probability of crossing a human when moving to cell n, i.e. $Poisson(n)$. The factor F allows to tune the weight of the human disturbance in the cost function.

3) A algorithm with Human flow estimation:* The human flow grid and the prediction flow grid are used to determine the navigation cost. Let $flow(n)$ be the human flow cost in n, k the direction from $(n-1)$ to n and $\bar{k}$ the opposite direction of k. We define the cost of moving to cell n as directly linked to the probability of having a human flow in the opposite direction : $flow(n) = M_{n,\bar{k}}(Zt)$. In the case where $M_{n,\bar{k}}$ is not defined, the predicted flow grid $M^{pred}_{n,\bar{k}}$ is used.

This leads to the following cost function :

$$g(n) = \|n,(n-1)\| \times (1+F.flow(n)) + g(n-1) \quad (8)$$

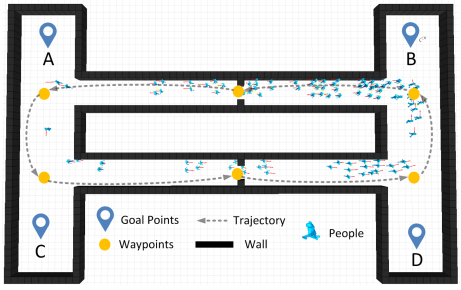


Fig. 3. Illustration of the simulated scenario

The factor F allows to tune the weight of the perturbation caused by an opposite flow of humans. We study the factor F parameter in the following section.

V. Experiments

A. The simulator

To evaluate the different A* based algorithms we developed a simulator based on PedSim, the 2D simulator of pedestrian crowd proposed by [2] [5]. In this simulator each pedestrian (agent) follows a predefined path by moving thanks to a local planner. This planner computes repulsive forces from obstacles and attractive forces from social link and attractive area, according to the pedestrian model defined in [6]. It provides trajectories avoiding collision with dynamic obstacles, i.e. people and other robots.

We extended this simulator to provide autonomous decision and interactive abilities to robots. They can communicate to others information about the pedestrians and robots they detect in their perception range. Robots can navigate along waypoints, computed with one of the A* based algorithms (listed in section IV-A).

An additional module is in charge of computing the flow grid and the human prediction flow grid. This module uses the direction information of humans detected by the robots. The Figure 2 illustrates the two grids, merged, after 10 minutes of the scenario presented in the following section.

B. Scenarios

The test environment is composed of two rooms connected by two long corridors (see Figure 3). Moreover, two sets of people (a total of 200 individuals) move in the environment by following a predefined circuit that consists in crossing the corridors in the anti-clockwise direction.

Each group of persons is composed of random socially linked people (people that stays close together during their navigation) and of individuals. Each person is set with a random walk speed from $0.5m.s^{-1}$ to $1m.s^{-1}$. At the beginning, the two groups are placed in opposite location (near point C and point B in Figure 3). After few minutes, people will be dispersed all along the corridors. In order to be more realistic, the simulator adds some random moves to the social forces model and prevents robots to try to pass through human groups if the space is not large enough (social rule).

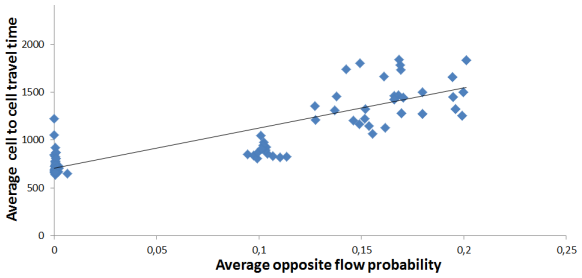


Fig. 4. Evolution of cell to cell time travel when the opposite likelihood flow grows.

The robots move with a speed of $1m.s^{-1}$ and have an omnidirection perception of radius $5m$.

The first scenario focuses on the navigation of one robot. This robot moves from point A to point B then from B to A and repeats this task until the end of the experiment. The path from A to B is particularly interesting because the shortest path crosses the human flow in opposite direction. We conduced numerous experiments to evaluate the performances of the different path planning algorithms.

The second scenario involves 4 robots that start respectively on position A, B, C and D (see Figure 3). The robots are asked to continuously navigate between two fixed positions until the end of the experiment. Their round trips are respectively between A-B, B-A, C-B and D-A. This allows to test different start-destination objectives within the same human flow scenario. The difference between A-B and B-A appears at the beginning of the experiment, as the first travel is not symmetric considering humans location. In this scenario, **robots cooperate to build a common flow grid and a common affordance map**. These shared grids are used by the robots to compute their path each time they reach a start/destination point.

We analyze in the next sections the time performance obtained with the three A*-based path planning.

C. Setting the human disturbance factor F

Before measuring performances of the A*-based path planning algorithms we need to determine their respective human disturbance factor F (eq. 6, 7, 8). This factor evaluates how humans can perturbate the travel from cell to cell, regarding the map information of each model.

In the non-informed model (eq. 6), human presence is equiprobable. The value of the parameter F has no influence. We can only consider that disturbance is homogeneous, leading to use **F=1** .

In cases of affordance-map and flow grid models, the value of F influences directly the resuting path. If we underestimate the F value we could compute paths that will be costly in time, due to the risk of meeting more people than expected. If we overestimate this value, we could compute longer paths, that risk to avoid unnecessary low human presence areas, leading finally to take more time.

In order to estimate this factor, we considered different human densities in a scenario where a robot move continuously from A to B and conversely. For each travel, we considered all the cells of the followed path and we computed an average cell-to-cell time and an average opposite flow likelihood ($flow$ in equation 8). Each result corresponds to one point in the figure 4.

The obtained graph shows the correlation between this two values. When the average opposite flow grows the average travel time too. We can see that the absence of opposite human flow (case of B to A travel) gives a null opposite flow for a shortest cell to cell travel time. A simple regression analysis gives a linear model of the form $minimum_travel_time.(1 + F.flow)$. The minimum travel time per cell have no influence in the path F calculus, only F factor have to be considered. The factor obtained in this characterization is **F=7.55**.

For the affordance-map based model, a similar analysis gives **F=3.5**. In the rest of the paper, we will use the three obtained F factors.

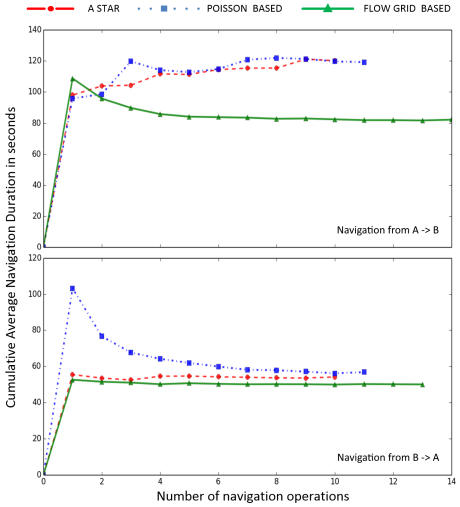


Fig. 5. Evolution of time performance on $A \Leftrightarrow B$ path for each model

D. Performance analysis on $A \Leftrightarrow B$

Measures of performance of each model on the $A \Leftrightarrow B$ path are presented in Figure 5. The average duration of each approach is presented for both $A \rightarrow B$ and $B \rightarrow A$ navigation. The x-axis is the number of round-trip performed by the robot during an experiment of 30 min. The y-axis gives the cumulative average duration of the travel $A - B$. Each average time of each algorithms was computed over 10 experiments of 30 min.

The red line (circle marker) plots the results of the A* model, the blue dotted line (square marker) plots the A* Poisson based model and the green line (triangle marker) plots the A* Flow grid model. We can notice that the approach using the Flow grid performs more round trips (14) than other approaches, confirming that its average travel duration is better.

In the $A \rightarrow B$ path, the A* Flow grid based model is quickly better than the other approaches. Morover, one can see the duration to travel from A to B decreases over the time.

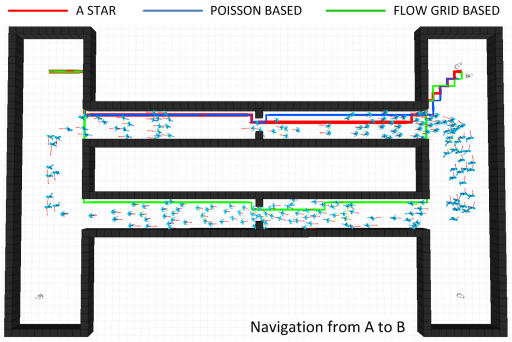


Fig. 6. A− >B paths selection after 10 minutes of experimentation

The classical A* model (non-informed) and the Poisson based model show a slight increases of the $A \rightarrow B$ duration. This is the consequence of the initial two groups of persons that spread along the corridors and increase their obstruction to robots over the time.

It is clear that going from A to B is more easy through the bottom corridor (same direction as the human flow) and going from B to A is more easy through the top corridor (for the same reason). The A* Flow grid model is the only one to be able to choose the bottom corridor to go from A to B as soon as it has learnt enough information about the human flow. This is illustrated in Figure 6 where computed paths are plotted at time 10 min.

Concerning the duration of the $B \rightarrow A$ path, all curves converge to a same average duration (Figure 5). The Poisson Based model gets a high first step duration value. The model computes the cost of the top corridor, where humans are observed, while it estimates that the other corridor is a better solution as it has been not yet observed by the robots. As a consequence the Poisson based model chooses the non observed corridor where the human flow is in the opposite direction.

The shortest path, which passes by the top corridor, has a common direction with the human flow. Then, in scenario $B \rightarrow A$, the three models find quickly the identical solution.

In the beginning of the experiment, one can note that the Flow grid model is the fastest to adapt its choice when meeting group of people (see Fig. 5 $B \rightarrow A$). The Poisson based model, modeling only human presence, needs more time to estimate the best path before people spread along the corridors.

E. Average performance on different paths

Table I presents the average duration of round trip between each start destination positions (Figure 3), after 30 minutes of experiment. The three A*-based models are compared.

Regarding our scenario, the A* Flow grid based approach is more efficient than the standard A* and the A* Poisson based model for most of robot's paths. For paths $A \Leftrightarrow B$, $C \Leftrightarrow B$ and $B \Leftrightarrow A$, the A* Flow grid model reaches average time much lower than others (best results are displayed in grey cells).

Path	A*	A* Poisson Based	A* Flow Grid Based
$A \Leftrightarrow B$	151.0 s	142.7 s	103.7 s
$C \Leftrightarrow B$	317.3 s	249.6 s	146.7 s
$B \Leftrightarrow A$	167.3 s	166.4 s	118.1 s
$D \Leftrightarrow A$	134.8 s	220.0 s	136.1 s

TABLE I

AVERAGE ROBOT TRAVEL DURATION FOR EACH A* BASED MODEL

Regarding the $D \Leftrightarrow A$ path, the standard A* model obtains better results. In the early steps of the experiment, the A* Flow grid model selects a costly trajectory due to a partial knowledge about the human flow. The first selected path crosses the human flow in the opposite direction (humans have not yet been oberved by robots in the bottom corridor). The A* Flow grid model will compute a path passing by the top corridor as soon as the human flow will be sufficiently observed in the bottom corridor.

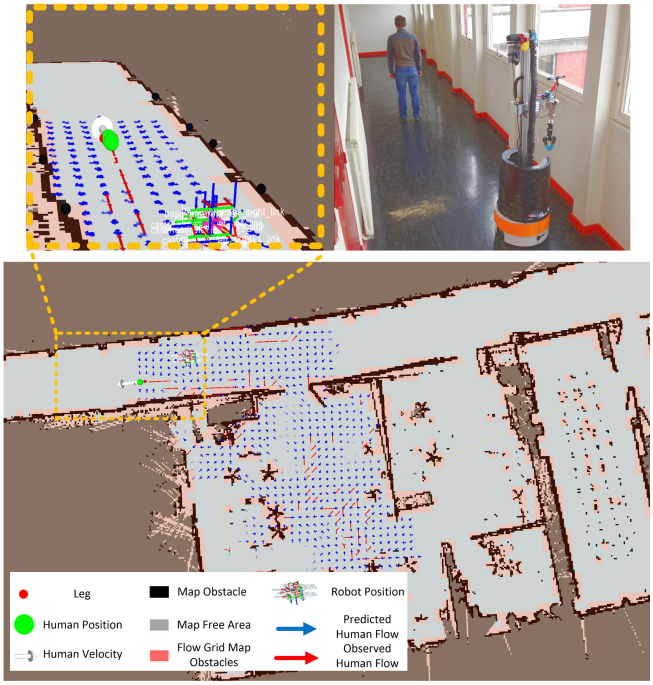


Fig. 7. Example of Flow grid (red) and Prediction Flow grid (blue) obtained with a a real robot (using a PMB-2 base and a Sick 2D laser)

VI. EXPERIMENTATION WITH A REAL ROBOT

Robotic experiments are necessary to evalute the proposed human flow mapping in real environments. We propose a first experiment with one robot.

The robot is a PMB-2 base from Pal Robotics equipped with a Sick TiM561 2D Laser (resolution of 1° angular, aperture angle of 270° and scanning range of 10m), see Fig. 7. It has been developped to participate to the Robocup@Home international competition (*http://www.robocupathome.org*).

Standard libraries of ROS are used for the robot localization and for the detection and localization of humans

around the robot (legs detector function). This data are used to compute online the flow grid. The figure 7 shows an example of such a grid built by the robot evolving in the test environment during 5 min. This area is a largely open space of 500 m2 including a corridor, a meeting room and desktop rooms. The result is extracted from the ROS visualization tools. A video of the mapping is available at [1] [1] and shows the incremental building of the flow grid while the robot meets or observes people walking around it.

Despite the current implementation limitations (e.g. delays due to large amount of markers to display) and the people detection accuracy (based on legs detection with Lidars), this experimentation shows clearly that real robots can build a representation of the recurring displacements of humans. We will now experiment in more dense populated environments. Future work will also focus on evaluating path planning computed with the flow grid information, and for different real scenarios and areas.

VII. Conclusion

In this paper, we have addressed the problem of mapping human flows with mobile robots evolving in indoor environments. For this purpose, we introduced the *Flow grid model*, which computes in each cell the likelihood of having a person moving in a certain direction. The approach relies on a statistical counting process from the observations of the robots. We completed the model by a *Predicted flow grid* based on a Von Misses Pattern of human motion, allowing to anticipate the flow in non-observed areas.

We evaluated this model by considering a task of path planning and navigation in a human populated environment. We compared three A*-based models, where the cost function depends on information about human presence in cells, that is i) non informed model (standard A*) ii) informed of human presence likelihood [15], iii) informed of human presence and motion direction likelihood, i.e. the flow grid model. Experimental results, in simulated crowded environments, show that the flow grid based path planning provides better results, in term of time to travel the environment, than the two other models.

Finally, we experimented the flow grid mapping with a real mobile robot (a PMB-2 base equipped with a Sick 2D Laser). While moving few minutes in the environment and encountering some people walking, the robot generated incrementally a map modeling the main repetitive displacements, see Fig. 7.

We plan to continue the work in two main directions. The first one is to consider new scenarios in simulation, including various and crossing human trajectories, to complicate the flows to model and to evaluate the robustness of the approach. The second perspective is to go further with real robot experiments, in particular by considering several robots cooperating to build the flow grid in dense crowded environments.

[1] https://youtu.be/Z8qDJH2CkLc.

References

[1] Human flow mapping in real environment with the pmb2 mobile base, https://www.youtube.com/watch?v=z8qdjh2cklc.

[2] Microscopic pedestrian crowd simulation system, pedsim, http://pedsim.silmaril.org/.

[3] Q. Baig, M. Perrollaz, and C. Laugier. A robust motion detection technique for dynamic environment monitoring: A framework for grid-based monitoring of the dynamic environment. *Robotics Automation Magazine*, 2014.

[4] S. Chik, C. Yeong, E. Su, T. Lim, Y. Subramaniam, and P. Chin. A review of social-aware navigation frameworks for service robot in dynamic human environments. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(11):41–50, 2016.

[5] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In P. M. A. Sloot, B. Chopard, and A. G. Hoekstra, editors, *ACRI*, volume 3305 of *Lecture Notes in Computer Science*, pages 581–590. Springer, 2004.

[6] D. Helbing and P. Molnr. Social force model for pedestrian dynamics. *Physical Review E*, pages 4282–4286, 1995.

[7] T. Ikeda, Y. Chigodo, D. Rea, F. Zanlungo, M. Shiomi, and T. Kanda. Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Robotics: Science and Systems*, 2012.

[8] P. E. J. K. V. Mardia. *Directional Statistics*. John Wiley and Sonsons Inc., 2000.

[9] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal. onditional transition maps: Learning motion patterns in dynamic environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1196–1201, 2013.

[10] T. Linder and K. O. Arras. Multi-model hypothesis tracking of groups of people in RGB-D data. In *17th International Conference on Information Fusion, FUSION 2014, Salamanca, Spain, July 7-10, 2014*, pages 1–7, 2014.

[11] P. Papadakis, A. Spalanzani, and C. Laugier. Social mapping of human-populated environments by implicit function learning. In *IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013.

[12] J. Rios-Martinez, A. Spalanzani, and C. Laugier. Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2014–2019, 2011.

[13] M. Seder and I. Petrović. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation - ICRA 2007, Roma, Italy, 10-14 April 2007*, pages 1986–1991, 2007.

[14] C. Stachniss. *Robotic Mapping and Exploration*, chapter Coordinated Multi-Robot Exploration, pages 43–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[15] G. D. Tipaldi and K. O. Arras. I want my coffee hot! learning to find people under spatio-temporal constraints. In *ICRA*, pages 1217–1222. IEEE, 2011.

[16] M. Volkhardt, C. Weinrich, and H.-M. Gross. Multi-modal people tracking on a mobile companion robot. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 288–293, Sept 2013.

[17] T. Wada, Z. Wang, T. Matsuo, Y. Ogawa, Y. Hayashibara, Y. Hirata, and K. Kosuge. Building human motion map for mobile robot in the indoor dynamic environment. In *Proc. of the 2010 IEEE International Conference on Robotics and Biomimetics*, pages 543–548, 2010.

[18] T. Wada, Z. Wang, Y. Ogawa, Y. Hirata, and K. Kosuge. Incremental human motion map system and human walking behavior representation in indoor environment. In *Proc. of the 2012 IEEE International Conference on Robotics and Biomimetics*, pages 747–752, 2012.

[19] Z. Wang, P. Jensfelt, and J. Folkesson. Multi-scale conditional transition map: Modeling spatial-temporal dynamics of human movements with local and long-term correlations. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6244–6251, 2015.

[20] C. Weinrich, M. Volkhardt, E. Einhorn, and H.-M. Gross. Prediction of human collision avoidance behavior by lifelong learning for socially compliant robot navigation. In *ICRA*, pages 376–381. IEEE, 2013.

Online Switch of Communication Modalities for Efficient Multirobot Exploration

Francesco Amigoni, Jacopo Banfi, Alessandro Longoni, and Matteo Luperto

Abstract— **Exploration of unknown environments with multirobot systems subject to communication constraints is a task involved in several applications, like search and rescue and monitoring. The approaches proposed so far to address this problem are usually based on the use of a single *communication modality*, for instance, either multi-hop (MH) or rendezvous (RV), during the whole mission. However, it has been conjectured that online switching between communication modalities could be beneficial. In this work, we empirically investigate this hypothesis by presenting an exploring multirobot system that can originally switch between communication modalities during an exploration mission.**

I. Introduction

Exploration is the task of building a complete *map* representing an initially unknown environment. When employing a multirobot system for building such a map, it often happens that robots are not able to always communicate with each other throughout the whole mission, because their communication ranges are limited. Moreover, it is also common to require the robots to provide reports to a Base Station (BS), so that human operators can supervise the evolution of the mission (and, possibly, receive some other useful data, such as a video feed).

While the unlimited communication setting has been widely studied (see, e.g., [1]), the case of multirobot exploration in presence of limited communication still presents challenging research questions. The approaches presented in the literature can be roughly classified according to the strength of the constraints they impose on connectivity between robots. On the one hand, there are approaches that enforce a "strict" form of connectivity. In this case, the robots and the BS are connected with each other at all time instants [2], [3] or they are allowed to shortly disconnect only while traveling between connected configurations [4], [5]. When robots are connected, they form a relay chain to send gathered data to the BS, for this reason we call this communication modality *multi-hop* (*MH*). On the other hand, there are approaches using a "soft" form of connectivity, in which connections between robots and the BS are not planned but emerge from the behavior of the robots. In this case, robots freely explore the environment and return to the BS when some condition is met (e.g., a threshold on explored area is reached) [6], [7]. Since, when returning to the BS, robots can opportunistically meet and exchange collected information, we call this communication modality *rendez-vous* (*RV*).

The two communication modalities have complementary positive and negative sides. In the MH communication modality, robots can quickly deliver new information to the BS, but changing the robots' deployment may result into long traveled distances that increase exploration times. On the other hand, the RV communication modality often requires less time and traveled distance for exploring an environment, but at the expenses of a lower situational awareness at the BS. To the best of our knowledge, exploring multirobot systems presented in the literature adopt a single communication modality (either MH or RV). However, it has been observed that online switching between communication modalities could be beneficial to exploration missions by balancing exploration efficiency and awareness at the BS. In particular, in [8], a conjecture is advanced that the MH modality is well suited for initial and final stages of an exploration mission, while the RV modality is more convenient in the middle of the mission.

In this paper, we investigate the possibility of combining the best features of the two described modalities through the study of a multirobot system that can originally switch between communication modalities during an exploration mission. The proposed switching policy selects a communication modality according to the estimated percentage of explored area. Experimental results (obtained in simulation) suggest that switching of communication modalities leads to a balanced performance in terms of exploration time, traveled distance, and situational awareness at the BS.

II. Multi-hop and Rendez-vous Modalities

In this section we provide an overview of two types of communication modalities, namely MH and RV, which are usually employed in multirobot exploration.

The MH communication modality can be thought as enforcing a form of "strict" connectivity, informing the BS each time new data are acquired. In particular, robots are required to form a relay chain connecting the frontiers of the environment (i.e., the boundaries between mapped and unknown space) with the BS. This modality provides up-to-date situational awareness to the BS but constrains the robots in their movements. An approach following the MH modality is introduced in [4] and generalized in [5]. In particular, while the strategy proposed in [4] computes a new deployment of robots once they have all reached their currently assigned locations, [5] suggests to compute new

Authors are with the Dipartimento di Elettronica, Informazione and Bioingegneria, Politecnico di Milano, Milano, Italy {francesco.amigoni, jacopo.banfi, matteo.luperto}@polimi.it, alessandro3.longoni@mail.polimi.it A preliminary extended abstract of this paper has been accepted at the ICRA2017 Workshop on Multi-robot Perception-Driven Control and Planning.

deployments for subsets of robots, provided that they have accomplished their assigned tasks.

Two relevant examples of strategies embracing the RV communication modality are presented in [6], [7]. In particular, [6] presents a role-based exploration strategy where robots are preassigned with the role of *explorer* or *relay*, and information is brought to the BS by the relays after having arranged a meeting with the explorers at rendezvous locations. In [7], instead, the behavior of the robots is regulated by a local utility function which considers the amount of information a robot has not yet delivered to the BS and the estimated amount of information known by the BS. As soon as an exploring robot notices that it is carrying "too much" information still unknown at the BS, it goes back to report it: if two robots meet while returning, only one of them is left in charge of going to the BS, leaving the other free to explore. Thus, this strategy gives rise to a form of "implicit rendez-vous" between robots.

In [8], variants of the above MH and RV communication modalities are implemented as a testbed for a routing protocol to exchange information between exploring robots. From the analysis of obtained results, a conjecture is advanced that a MH modality is well suited for initial and final stages of an exploration mission, while a RV modality is more convenient in the middle of the mission. The idea is that sharing initially acquired information promotes an effective deployment of robots (e.g., a balanced spreading), which can then explore freely, before eventually reconnecting to quickly convey to the BS the last perceived data.

In this paper, we investigate the above hypothesis by presenting an exploring multirobot system that, originally, can switch between the MH modality of [5] and the RV modality of [7].

III. System Overview

We consider a two-dimensional indoor environment, which is initially unknown. We consider a system composed of n homogeneous robots $R = \{r_1, r_2, \ldots, r_n\}$ and of a BS. The information collected by the robots in the environment should be eventually delivered to the BS, where human operators supervise the exploration mission. Each robot can move in the environment and perceive it with a laser range scanner. Robots can also communicate with each other (and with the BS) within a range R_c, which is not constant but varies according to some factors, such as the presence of walls (see Section V-A). We assume that, when two robots can communicate, the available bandwidth provided by their communication devices is unlimited (for the purposes of the mission). For simplicity, we assume that all the robots are equal. Each robot is able to build a local occupancy grid map starting from the scans it collects (e.g., using gmapping [9]), while the BS is able to merge the local maps received from the robots in a global map M_{BS} of the environment (e.g., using a method like that in [10]). On both the local maps and the global map, *frontiers* are calculated as the clusters of known free cells that are adjacent to unknown cells. The goal of the mission is to build a complete global map of the environment, namely a map without frontiers, trading-off the minimizations of exploration time, of distance traveled, and of time during which robots are disconnected from the BS.

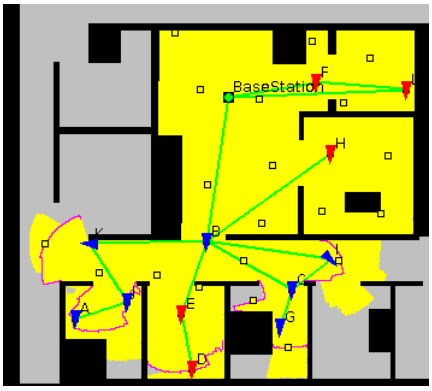


Fig. 1. MH modality, exploration snapshot. Blue and red: not ready and ready robots, respectively. Black-edged squares: vertices of G. Green: current communication links. Purple: frontiers of the last issued plan. Image from [5].

As introduced above, we consider a multirobot system that can operate under two different communication modalities, MH [5] and RV [7]. In the following, we provide a brief overview of the two modalities, referring the reader to the original papers for full details. Note that each communication modality is associated to a different coordination strategy for the robots.

A. A MH modality (Banfi et al. [5])

When adopting MH as communication modality, robots are required to form relay chains connecting the frontiers with the BS (Fig. 1). To cast this process into a rigorous optimization framework, a graph $G = (V, C)$ is constructed upon the occupancy grid map. The vertex set V represents candidate robots' locations, while the edge set C represents the possibility of communicating between two vertices in V under a given *conservative* communication model (like, for instance, one based on limited-distance line-of-sight). At any time instant, a robot can be *ready* or not. Informally, a robot is ready if it has acquired and/or relayed the information from the assigned frontier(s) to the BS. Note that, in an efficient deployment, a robot not placed on a frontier may serve as data relay for other robots placed on frontiers (like in a tree structure).

At the beginning, all the robots are ready and connected to the BS. Deployments of subsets of robots in the environment are then iteratively issued by the BS (which acts as a central coordinator) as soon as a given number θ of robots becomes ready (in our experiments, we set $\theta = 1$ to avoid idle robots). To simplify the deployment process, the problem is decomposed in two sub-problems. First, find the "best" set of connected robots' positions (i.e., vertices in V all connected to the BS by edges in C) according to a utility function that considers the expected information gain at those positions (regardless of which robot occupies which position). Second, allocate the robots on such positions to

minimize their traveling cost. The first problem is solved by means of an approximated algorithm based on finding solutions to a sequence of Steiner tree problems. The second problem is then easily solved by means of the Hungarian algorithm.

B. A RV modality (Spirin et al. [7])

In this modality, the robots explore independently the environment, moving toward the most promising frontiers they know (according to a utility function which takes into account both the estimated information gain and the travel distance), and return to the BS when they have gathered enough information. Specifically, a robot r_i goes back to the BS when:

$$\frac{|M_{BS}|}{|M_{BS}| + |M_i|} < th,$$

where $|M_{BS}|$ is the amount of information (e.g., number of known cells) contained in the global map M_{BS} known at the BS (according to what robot r_i knows), $|M_i|$ is the amount of novel information contained in the local map of robot r_i that is not in M_{BS}, and $th \in [0, 1]$ is a fixed threshold. The larger the value of th, the more frequently a robot returns to the BS (in our experiments, we set $th = 0.8$ to ensure a good situation awareness at the BS). The coordination between robots is episodic and opportunistic. If two robots meet (i.e., if they are within R_c), they first merge their local maps and share their believes about M_{BS}. Then, the robot nearest to the BS will be in charge of going back to the BS, while the other one will be free to going back to explore.

The robots can be basically involved in two behaviors: exploring or returning to the BS to deliver information. In [7], it is observed that for small values of th and for a large number of robots n, some robots start to act as relays, simply moving back and forth between the BS and the other exploring robots. This happens because robots frequently meet while returning to the BS to deliver their information. When they are within communication range, they exchange information and, having now the same knowledge of the environment, the robot nearest to the BS assumes the responsibility of delivering back to the BS the combined new information. When this relay robot tries to move again toward some frontier, it is likely that it meets another returning robot, and the process repeats.

IV. Switching between Modalities

In order to switch between the MH and the RV communication modalities (and associated coordination strategies), we introduce a *switching policy*, which returns a communication modality according to the current state of the mission. In particular:

$$\mathit{SwitchingPolicy}(M_{BS}, m^{t-1}) = m^t,$$

where M_{BS} is the map currently known at the BS, m^{t-1} is the current communication modality, and m^t is the new communication modality. In our system, $m^t \in \{\text{MH}, \text{RV}\}$ for all time steps t. If $m^t \neq m^{t-1}$, then there is a switch in the communication modality.

In principle, there can be several ways in which the *SwitchingPolicy* can be implemented. Following the conjecture reported of [8], we implement a simple *SwitchingPolicy* that performs a switch from the starting MH modality to the RV modality after the initial stage of the exploration mission and a switch back from RV to MH in the final stage of the exploration mission. The communication modality is thus changed according to:

$$\mathit{SwitchingPolicy}(M_{BS}, m^{t-1}) = \begin{cases} \text{RV} & \text{if } \alpha \leq \frac{\mathit{area}(M_{BS})}{\hat{A}} < \beta \\ \text{MH} & \text{otherwise} \end{cases}$$

where, $\mathit{area}(M_{BS})$ is the extent of the area of the known free space of the map M_{BS} and $\hat{A}$ is an estimate of the area of the free space of the whole environment. The two parameters α and β (both in $[0, 1]$ and such that $\alpha \leq \beta$) control the switch from MH to RV and from RV to MH, respectively.

Calculating $\hat{A}$ amounts to estimate the entire area of an environment that is only partially known by the multirobot system. We propose three different ways to perform such estimation:

- $\hat{A} = A$, the real area of the environment. This is an often unrealistic situation, but it is useful for comparison.
- $\hat{A} = \hat{A}_{\text{BB}}$, the area of the bounding box of the environment. In realistic situations, it is often possible to know the outer perimeter of an indoor environment (e.g., from satellite images or from floor plans). From the outer perimeter, the bounding box and its area can be easily calculated.
- $\hat{A} = \hat{A}_{\text{AV}}$, the average area of environments of the same type of the explored one. Human-made environments, like buildings, are characterized by strong regularities, not only within the same building, but also between buildings of the same type (like schools, offices, hospitals, ...). In previous work, we have modeled these regularities according to the concept of *building type* [11], [12]. Assuming to know the type of the environment the robots are exploring, we calculate the average area of the environments of the same type present in the data sets of [11], [12]. For instance, Fig. 2 shows the histogram distribution of the areas of 43 school buildings and the average area.

All three methods provide an estimated area for the environment being explored, which is calculated at the beginning of the mission.

The decision of switching between different modalities is taken by the BS and considering $\mathit{area}(M_{BS})$, the total amount of area known to the BS at each time step t. After each update of M_{BS} (triggered by reception of data from robots), the BS compares the two parameters, α and β, with the current exploration progress $\frac{\mathit{area}(M_{BS})}{\hat{A}}$. If the conditions for a strategy switch are met, the BS sends its decision to the connected robots. In the case of a switch from MH to

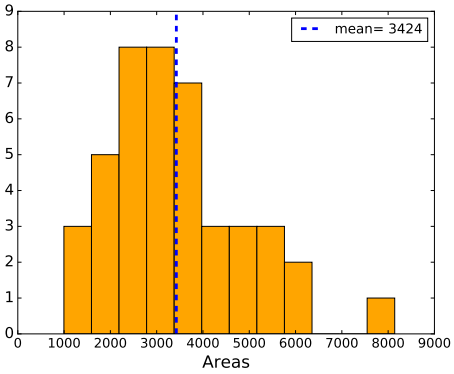


Fig. 2. Areas of 43 school buildings (in m^2) and average area.

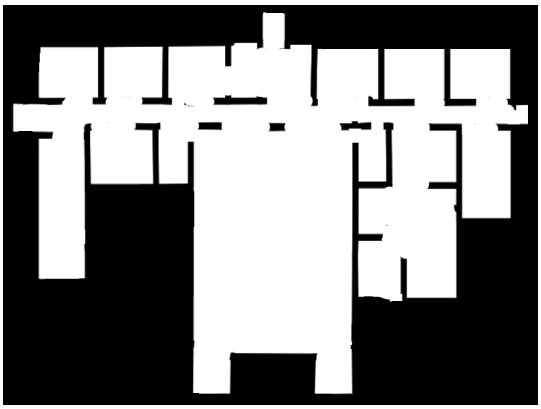


Fig. 3. One of the simulated environments representing a real-world school building.

RV strategy, robots are usually already connected to the BS (or they will be, as soon as they reach their current goal positions) and the policy switch happens immediately. In the case of switch from RV to MH, instead, we study the two following scenarios.

Basic scenario: we assume that the BS is endowed with the same communication device (e.g., a Radio-Frequency (RF) transceiver) of the robots, characterized by a (relatively) low range R_c and by a high bandwidth. Under this assumption, the BS alerts all the robots that gradually enter the BS's communication range to stop their current tasks. When all the robots eventually can communicate to the BS, the robots restart exploring using the MH modality. Note that this process might require some time, since robots moving according to RV modality restore the connection with the BS only when coming back to report new data (which can happen at unpredictable times).

Additional long-range low-bandwidth channel: we assume that the BS is endowed with a second communication channel with low bandwidth and long range, able to cover the whole environment. This channel, similar to an AM radio frequency, is mono-directional: the BS is the only transmitter and the robots can only receive. Using this additional communication channel, the BS signals to the robots the need to switch from RV to MH modality, avoiding to wait until all robots are connected. After the signal is received, all robots immediately return to the BS and switch from RV to MH. Note that limited bandwidth is not a problem, since a single-bit information is enough to communicate the switching command. This second channel is not used to communicate any other information about the environment (partial maps, video feeds, ...).

V. Experimental Results

We run simulated experiments in MRESim [13], which is an easy-to-use simulator for multirobot exploration with communication constraints. In the experiments reported here, we consider teams of 6 robots plus the BS, exploring 12 school buildings. Our main aim is to validate the conjecture of [8], namely that alternating MH and RV yields better exploration performance than using a single communication modality.

Given an environment, we consider three metrics:

1) the *time taken* by the robots to completely explore the environment,
2) the *total distance traveled* by the robots to completely explore the environment (which can be easily related to energy consumption),
3) the *disconnected time*, namely the time robots have not been in communication with the BS.

MH and RV modalities perform very differently if we consider these three metrics: RV is generally better than MH in the time taken and the total traveled distance, while MH is generally better than RV for what concerns the disconnected time. In the following, results are presented using heatmaps. For each combination of values of α and β of our switching policy (from 0 to 1 with step 0.1), the heatmap reports a cell with a color: green (■) means good performance, red (■) means bad performance, and yellow (■) means average performance.

A. Simulation settings

Maps in MRESim are 800×600 px, which correspond to environments of approximatively 40×30 meters. One example of one of the environments is shown in Fig. 3. Robots are equipped with a simulated laser range scanner, whose data are fed to a simulated SLAM module. We set a sensing range of ~ 5 meters, with a field of view of 120 degrees. Robots are also equipped with simulated short-range RF transceivers (which simulates, for instance, a WiFi signal), adopting the signal strength model of [14]. According to this model, the signal strength at distance d_m from the emitting source is computed as:

$$S = P_{d_0} - 10 \times N \times \log_{10} \frac{d_m}{d_0} - \min(nW, C) \times \mathit{WAF}, \quad (1)$$

where P_{d_0} is the signal strength at the reference distance d_0, N is the rate of the path loss, nW is the number of obstructing walls, *WAF* is the wall attenuation factor, and C is the maximum number of walls where the attenuation

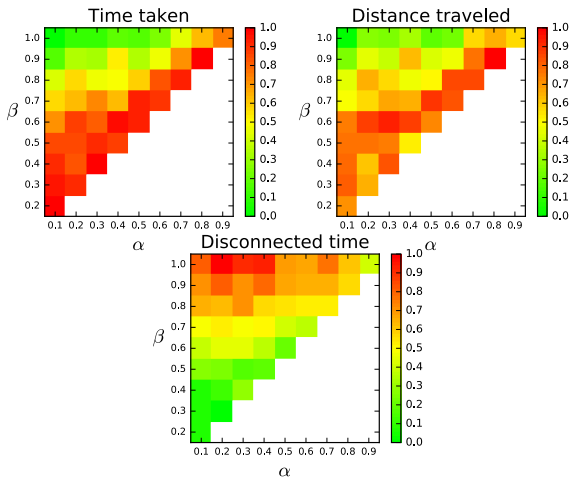


Fig. 4. Average results with the perfect estimator $\hat{A} = A$.

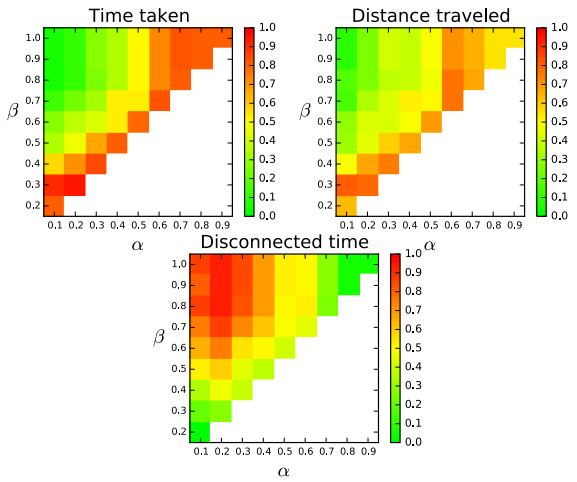


Fig. 5. Average results with the bounding box estimator $\hat{A} = \hat{A}_{\text{BB}}$.

factor needs to be considered. In our experiments, we set $d_0 = 180$ px ($\sim 1/5$ of the map diagonal, approximatively 9 meters), $P_{d_0} = -92$ dBm, $N = 2$, $C = 4$, $WAF = 3$. We set the cutoff value to -92 dBm. Therefore, d_0 is the highest possible value for R_c. For what concerns the conservative communication model used to build the graph G used by the MH modality, we assume that two robots will always be able to communicate when within distance d_0 *and* in line-of-sight. The second communication channel is modeled in a similar way, but allowing communication across the whole environment.

B. Results

Fig. 4 shows the average results obtained over the 12 school buildings considering $\hat{A} = A$, namely a perfect estimator of the area of the environment. For each metric, the performance is normalized over the environments and over the runs with different combinations of α and β. From the figure, it is clear that the three metrics are conflicting. Shortest time taken and distance traveled are obtained when β is close to 1 (namely, when the final switch from RV to MH is performed very late during the mission). Conversely, shortest disconnected times are obtained with small values of β (namely, limiting the time during which RV is used).

Fig. 5 shows the results for the bounding box estimator ($\hat{A} = \hat{A}_{\text{BB}}$). By definition, this estimator overestimates the area of the environment, and this is particularly true for environments that do not have a box-like shape (for which the estimation error is $\sim 30\%$), like T-shaped (estimation error of $\sim 100\%$) and H-shaped buildings (estimation error $\sim 50\%$). (In our experiments, we use 8 box-shaped buildings, 3 T-shaped ones, and one H-shaped). Note that the bounding box model overestimates the area even for box-shaped buildings, because of gaps between rooms, as those caused by walls or by closed/service rooms (like elevator shafts). The consequence of overestimating the area is that the ranges of values for α and β that produce good performance relative to time taken and distance traveled are larger (visually, the green areas are larger). However, at the same time, it is harder to find a good combination of values for α and β that minimize the disconnected time. This is the main limit of the bounding box estimator, which is often unable to guarantee good situation awareness at the BS. For this model, the combinations of values for α and β that yield good results (namely, the green area) seem to be larger with respect to the perfect estimate model. Also, this green area is slightly moved from values where the RV modality is predominant (namely, with low α and $\beta = 1$) to values where $\alpha < 0.5$ and $\beta \geq 0.6$. Using the bounding box model, when an exploration run is approaching to discover the last remaining parts of the environment, the estimator returns that the percentage of the environment explored is much smaller than it actually is. This leads to never employ the second switch to a MH modality, because the threshold β is never met.

In Fig. 6, we report the results for the average area estimator ($\hat{A} = \hat{A}_{\text{AV}}$). In this case, the performance is similar to that obtained with the perfect estimator (visually compare Figs. 4 and 6), with the presence of ranges of values for α and β that provide a good trade-off between the effort needed to explore the environment and the maintenance of a good situational awareness at the BS. However, being based on an average value, this estimator shows some weakness when the area of the environment being explored significantly deviates from the average of the areas of the environments of the same type. Values around $\alpha = 0.5$ and $\beta = 0.8$ produce a good balancing between the benefit of a MH modality and those of a RV one when using the average area estimator, with good performance for all the three metrics combined.

Overall, the conjecture of [8] seems to be supported by our empirical evidence: online switching from MH to RV and from RV back to MH leads to a balanced exploration performance. This balancing is not obtained when using combinations of values close to those corresponding to single

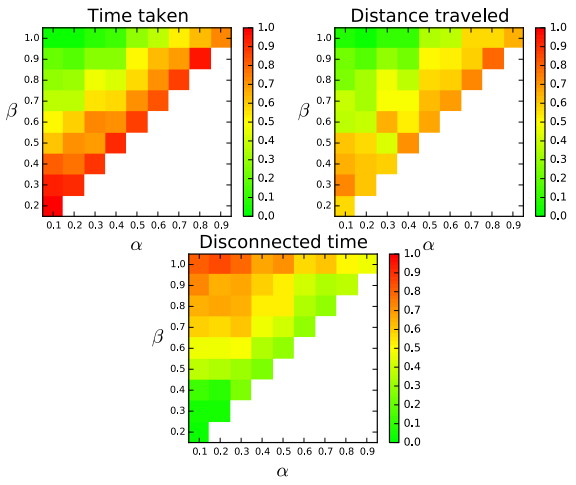


Fig. 6. Average results with the average area estimator $\hat{A} = \hat{A}_{\text{AV}}$.

communication modalities, namely $\alpha = \beta$ for MH and $\alpha = 0$, $\beta = 1$ for RV.

Finally, we evaluate the effect of using the low-bandwidth communication channel for switching from RV to MH. Fig. 7 shows that the use of the second communication channel does not substantially change the overall performance of our exploring multirobot system. The only improvement that can be found is a slight reduction of the disconnected time.

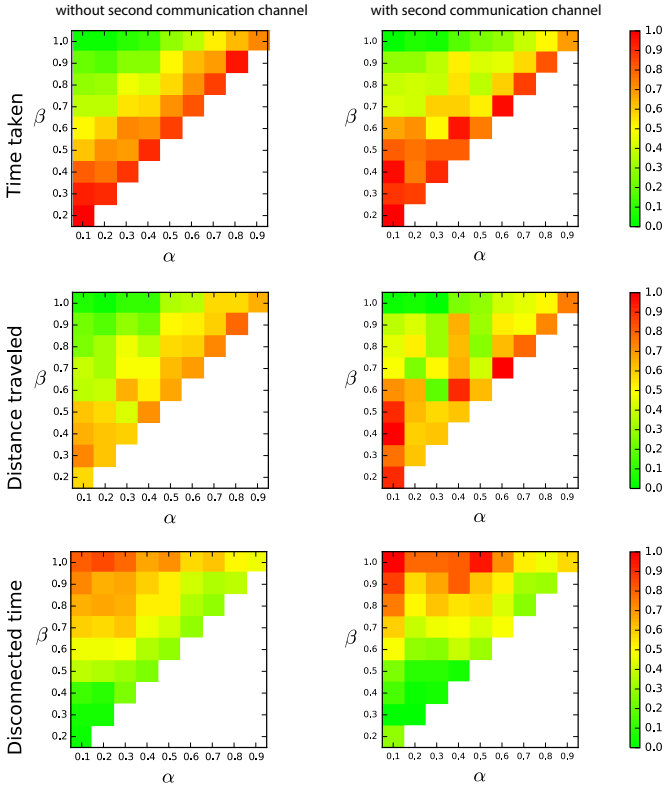


Fig. 7. Average results with the average estimator when not using the second communication channel (left) and when using it (right).

VI. Conclusions

We have presented a novel multirobot system for exploration that is able to dynamically switch communication modality during the mission according to different estimates of the whole area of the (unknown) environment. We have focused on switching between two communication modalities, namely MH and RV, which have strict and soft connectivity constraints, respectively. The system has been evaluated by considering the exploration time, the distance traveled, and the time during which robots are disconnected with the BS. Results show that alternating the MH and the RV modalities nicely balances the benefits of the two approaches.

Future works include testing different switching policies, possibly involving different criteria beyond the one based on the estimated area, and testing more complex combinations of different communication modalities, also measuring the costs of communication and the redundancy of information transmitted to the BS. Moreover, the impact of the number of robots and of their locomotion and perception capabilities could be better assessed. Finally, implementation on real robots is required to confirm the findings of this paper.

References

[1] M. Julia, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Auton Robot*, vol. 33, pp. 427–444, 2012.

[2] M. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Eng Pract*, vol. 15, no. 4, pp. 435–445, 2007.

[3] P. Mukhija, K. Krishna, and V. Krishna, "A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint," in *Proc. IROS*, pp. 4806–4811, 2010.

[4] Y. Pei, M. Mutka, and N. Xi, "Connectivity and bandwidth-aware real-time exploration in mobile robot networks," *Wirel Commun Mob Comput*, vol. 13, no. 9, pp. 847–863, 2013.

[5] J. Banfi, A. Quattrini Li, N. Basilico, I. Rekleitis, and F. Amigoni, "Asynchronous multirobot exploration under recurrent connectivity constraints," in *Proc. ICRA*, pp. 5491–5498, 2016.

[6] J. de Hoog, *Role-based multirobot exploration*. PhD thesis, University of Oxford, Oxford, 2011.

[7] V. Spirin, S. Cameron, and J. de Hoog, "Time preference for information in multiagent exploration with limited communication," in *Proc. TAROS*, pp. 34–45, 2013.

[8] T. Andre, *Autonomous exploration by robot teams: coordination, communication, and collaboration*. PhD thesis, Alpen-Adria-University, Klagenfurt, 2015.

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE T Robot*, vol. 23, pp. 34–46, 2007.

[10] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *P IEEE*, vol. 94, no. 7, pp. 1384–1397, 2006.

[11] M. Luperto and F. Amigoni, "Exploiting structural properties of buildings towards general semantic mapping systems," in *Proc. IAS*, pp. 375–387, 2014.

[12] M. Luperto, A. Quattrini Li, and F. Amigoni, "A system for building semantic maps of indoor environments exploiting the concept of building typology," in *Proc. RoboCup*, pp. 504–515, 2013.

[13] V. Spirin, J. De Hoog, A. Visser, and S. Cameron, "MRESim: A multi-robot exploration simulator for the rescue simulation league," in *Proc. RoboCup*, pp. 106–117, 2014.

[14] V. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. INFOCOM*, pp. 775–784, 2000.

Autonomous Landing On A Moving Car With Unmanned Aerial Vehicle

Tomas Baca[1], Petr Stepan[1] and Martin Saska[1]

***Abstract*— This paper presents an implementation of a system that is autonomously able to find, follow and land on a car moving at $15\ \mathrm{km/h}$. Our solution consists of two parts, the image processing for fast onboard detection of landing platform and the Model Predictive Control tracker for trajectory planning and control. This approach is fully autonomous using only the onboard computer and onboard sensors with differential GPS. Besides the description of the solution, we also present experimental results obtained at MBZIRC 2017 international competition.**

MULTIMEDIA MATERIAL

A video attachment to this work is available at [11].

I. INTRODUCTION

Multirotor helicopters, in robotic literature called Unmanned Aerial Vehicles (UAVs) or Micro Aerial Vehicles (MAVs), due to their possible very small size, have become widely popular within the scientific community for their well studied dynamic properties and high applicability. Ability to hover in one place has advocated their use as a sensor carrying platform and for testing various approaches of fully autonomous flying, even without any human intervention during the mission in future. Three tasks need to be solved in such a deployment of autonomous MAV systems: take off, trajectory following possibly with an environment interaction, and precise landing. Robust and safe autonomous landing seems to be the most challenging part of the overall MAV mission mainly if a dynamic target of unknown position and windy outdoor conditions are considered. In this paper, a system designed to solve this task is introduced together with breathtaking results of repeated fully autonomous landing on a $15\ \mathrm{km/h}$ moving platform in an outdoor arena with the wind reaching $10 - 20\ \mathrm{km/h}$ and achieving a precision in tens of centimeters.

A. MBZIRC 2017 competition

The research achievements presented in this paper were motivated and results obtained within the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 competition organized by the Khalifa University of Science in Abu Dhabi. The MBZIRC competition brought a significant impact to the robotic community, and mainly to the field of MAVs, due to the ambitiously selected robotic challenges on the edge of current state-of-the-art chosen by a board of top scientists in the field. One of these challenges motivated by current needs of industry was autonomous landing on a moving vehicle discussed in this paper.

[1]Authors are with the Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, Prague 6, Email: tomas.baca@fel.cvut.cz.

The MBZIRC challenges were designed to provide a spectacular performance and to attract a broad general audience. The most importantly, the MBZIRC competition could be considered as a relevant and objective benchmark of the tasks, which are currently solved by the robotic community since the successful teams had to achieve the given goal after only a few minutes of preparation in one trial. No repeated tests, which is the standard practice in most of the laboratory experiments, were allowed and moreover the system robustness was exhibited in current environment (light and windy) conditions (teams could not influence the start of their run). Out of 140 registered teams from almost all best robotic groups worldwide, 25 top teams were selected after several preliminary rounds to compete in the final competition in Abu Dhabi in March 2017[1]. The solution, described in this paper, presents the best reliability and robustness (only two teams landed precisely in both final trials) and the fastest performance from all competition trials (the fastest time of landing in the entire competition was achieved by the proposed system during the grand challenge, where together three MAVs were deployed simultaneously). The only competitive solution was presented by team Fly Eagle from Beijing Institute of Technology. They also landed in both trials in a similar time, but their system is not published yet to be able to compare both systems and to highlight differences. Nevertheless, both solutions can be considered as a valuable contribution to the robotic field since according to our knowledge no other system does exist to be able to solve this very demanding and complex challenge in these outdoor conditions (which was also the reason, why this task was selected by respected robotic leaders for the competition).

Before the competition, few solutions of autonomous landing with visual feedback were described in literature [5], [6], but most of these systems are capable of landing only on a static or slowly moving pattern [8], [9], [10] and only in laboratory conditions, without the presence of wind and with stable light conditions.

To sum up the contribution of this paper, the proposed method enables to detect a landing pattern, to estimate its relative position and velocity, to predict motion of both systems (the MAV and the landing platform) and based on these states estimation it designs sequence of optimal control inputs taking into account external disturbances such as wind and imprecisely identified MAV model in MPC fashion using only onboard computational resources. This

[1]Results of our team from this qualification process can be found at http://mrs.felk.cvut.cz/projects/mbzirc

approach provides high robustness and precision in the landing task, which is a crucial element for fully autonomous missions (such as periodical surveillance, reconnaissance, object carrying, and monitoring) in which MAVs are especially appealing.

B. Contributions

We present a solution to the challenge 1 of MBZIRC 2017 competition. The MAV can follow a car moving at $15\ \mathrm{km/h}$ autonomously. While following, it lands on the car roof and attaches itself using magnetic legs. The landing on a moving car is robust on very challenging outdoor condition with wind speed up to $10\ \mathrm{m/s}$.

II. The problem definition

The task consists of an autonomous landing of an MAV on a moving car. The car is equipped with a graphical black pattern on a white metal board. The landing pattern consists of a single circle with a cross drawn in its center. The autonomous landing is conducted with a multirotor aircraft equipped with cameras, differential GPS receiver, and other sensors. Although the GPS is not a necessity, long-term and robust autonomous flight proves to be simpler than using, e.g., only visual odometry for localization. The speed profile of the car is known, starting at $15\ \mathrm{km/h}$ and slowing down, as well as its trajectory, except its initial conditions – position and direction. The track, in which the car will drive, can be measured before experiments to improve the search or tracking.

We assume our MAV is equipped with a down-facing camera, differential GPS and laser rangefinder for measuring its altitude above the ground.

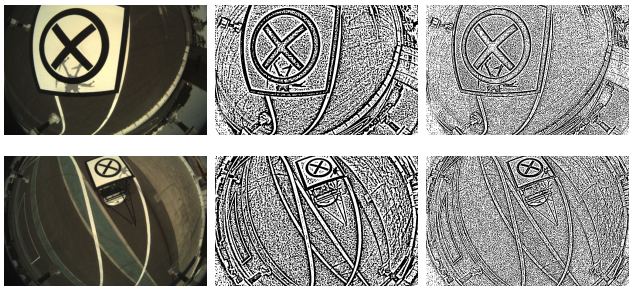
III. Computer vision

The goal of pattern detection is to detect the landing pattern robustly. The image processing was computed on Intel NUC embedded PC with Intel Core i7 5557U processor. A single mvBlueFOX-MLC200w color camera was used. This camera has a global shutter and therefore is suitable for aerial robotic purposes. The camera can provide 93 images per second with resolution 752×480. A miniature SuperFisheye lens Sunex DSL215, together with with 1/3" camera sensor, created an image with horizontal FOV of $185°$. Our detection algorithm can detect the landing pattern in one image using a single thread in $15\ \mathrm{ms}$. The final detection rate is $50\ \mathrm{Hz}$.

A. Algorithm overview

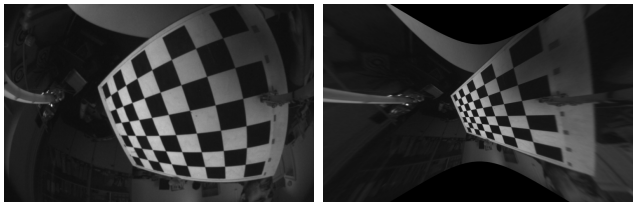
Landing pattern detection is based on standard computer vision approaches using OpenCV tool. The landing pattern (see Fig. 1a) contains circle and cross. The detection algorithm is based on circle detection combined with cross detection inside the circle. The detection has to be robust to various weather conditions, changes of light intensity, and direct sunshine with shadows cast by the aircraft.

The first step of pattern detection is a method of adaptive thresholding. The result of the thresholding with box size 5 pixels and 11 pixels can be seen in Fig. 1b resp. 1c. The adaptive threshold is robust to light intensity and is used to segment the border of the circle in the image. The size of the block for adaptive threshold depends on expected pattern size, which depends on drone altitude.



(a) (b) (c)

Fig. 1: a) original image from the camera, b) adaptive threshold with box size 11 pixels, c) adaptive threshold with box size 5 pixels.



(a) (b)

Fig. 2: a) original image from the camera, b) undistorted image.

A variant of the detection algorithm is selected based on the current altitude of the MAV:

- altitude more than $5\ \mathrm{m}$, the pattern is small, the size of circle in the image ranges from 10 to 20 pixels
- altitude between $1.5\ \mathrm{m}$ and $6\ \mathrm{m}$, the whole circle can be detected
- altitude less than $1.5\ \mathrm{m}$, only a part of circle is detected, detection is based only on cross detection

The circles and lines are detected in the segmented image from the adaptive threshold. The object detection is done in image coordinates, in contrast with the classification, which is applied on undistorted images. Undistorting images before further computations is crucial for further processing.

B. Fish-eye lens

The identification of the lens was performed using OpenCV 3.2 and its fish-eye model. The original image from the drone can be seen in Fig. 2a, the corrected image in Fig. 2b.

Due to the slow implementation of the fish-eye model in OpenCV, a custom version of the undistort method was implemented. This new undistort function is even faster than the conventional camera model. The original fish-eye undistort function computes transformation from image coordinates (x, y) to undistorted coordinates by following steps:

1) compute world point

$$(w_x, w_y) = \big((x - c_x)/f_x, (y - c_y)/f_y\big), \quad (1)$$

where (c_x, c_y) is the image center and (f_x, f_y) focus in axis x and y.

2) compute distance from center as

$$\theta_0 = \sqrt{w_x^2 + w_y^2}, \quad (2)$$

3) iteratively compute undistortion coefficient (10 times)

$$\theta_{i+1} = \frac{\theta_0}{1 + k_0 \cdot \theta_i^2 + k_1 \cdot \theta_i^4 + k_2 \cdot \theta_i^6 + k_3 \cdot \theta_i^8}, \quad (3)$$

where k_0, k_1, k_2, k_3 are distortion coefficients detected by fish-eye calibration.

4) using θ_{10}, compute $scale = \frac{\tan(\theta_{10})}{\theta_0}$ and undistorted coordinates $(ud_x, ud_y) = (w_x * scale, w_y * scale)$

This approach is very slow, because iterative computation of θ_{10} needs 80 float multiplication and 17 float divisions, so together it needs 97 float operations.

Because the distortion coefficients are known in advance, we can prepare results of scale for all values of θ_0 with sufficient precision. For our image resolution 752×480, we are using array $scale_k$ with 1000 of values in $\langle \theta_0, \theta_{max} \rangle$, where $\theta_0 = 0$, $\theta_{max} = \frac{\Pi}{2}$. Additionally, we can omit square root function, because we can prepare scale values for the square distance from image origin. Finally, the computation of undistorted coordinates for image coordinates (x, y) takes following steps:

1) compute world point

$$(w_x, w_y) = ((x - c_x)/f_x, (y - c_y)/f_y), \quad (4)$$

where (c_x, c_y) is the image center and (f_x, f_y) focus in axis x and y,

2) compute square distance from center as

$$\theta_0 = w_x^2 + w_y^2. \quad (5)$$

3) using precomputed array $scale_k$,

$$scale = scale_k \left(\frac{\theta_0}{\theta_{max}} \cdot 1000 \right) \quad (6)$$

and compute undistorted coordinates

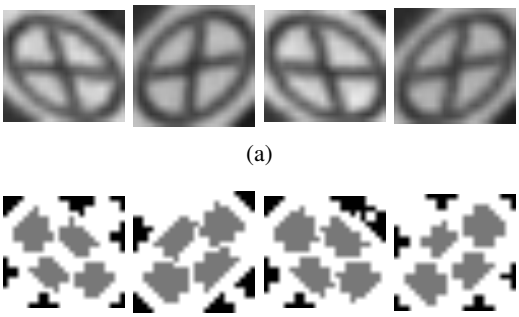
$$(ud_x, ud_y) = (w_x * scale, w_y * scale) \quad (7)$$

This approach is using only 7 float multiplications and is approx. 15 times faster than the conventional one.

C. Robust detection

The robust detection of the landing pattern is based on circle detection on images from adaptive threshold procedure. To eliminate false positive detections, the circle must contain a cross. The detection of the cross relies on three algorithms, depending on circle size.

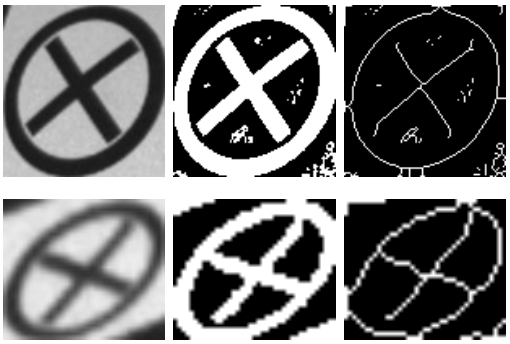
If one of the axis of the ellipse (circle) is shorter than 30 pixels, then a line, making a part of the cross, cannot be detected reliably. In such case, the cross is detected by searching for four areas with similar size. Mathematical morphology operation *closing* is applied, and the number of closed areas is computed. The cross is detected if there are four closed areas, having similar size. Figure 3a shows original images, wheres Fig. 3b shows detected areas depicted by gray color.



(a)

(b)

Fig. 3: a) Original images from camera, b) result of operation morphological closing.



(a) (b) (c)

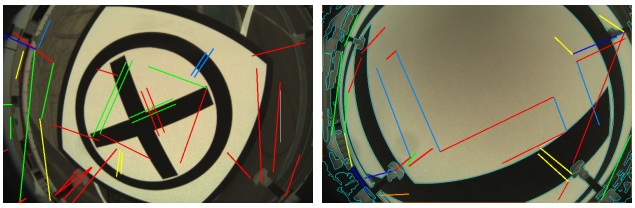
Fig. 4: a) original image from the camera, b) adaptive threshold with box size 11 pixels, c) result of Guo Hall thinning.

Cross of a size between 30 and 150 pixels can be detected by Guo Hall thinning [2]. This algorithm cannot be used for larger circles because of its computational demands. The Guo Hall thinning enables very robustly detect correct lines inside the circle. The cross inside the circle is detected if a crossing point of two the biggest lines is near the center of the ellipse. Figure 4 shows the original images from the camera in comparison to images after applying adaptive thresholding and the results of Guo Hall thinning algorithm.

For circles larger than 150 pixels, the cross is detected by finding two pairs of parallel lines (see fig. 5a red and green lines) forming a border of the cross. At least two pairs of parallel lines with correct size and thickness are required, to positively detect the cross and its center. This method provides detection of the landing pattern if the whole circle cannot be seen. Figure 5b depicts the pattern reconstructed only from two visible lines of the cross.

D. Global pattern position

The last step is computing the position of the landing pattern in global world coordinate system. Suppose that we



(a) (b)

Fig. 5: Line detection for a) cross inside of the circle and b) when only a part of the landing pattern is detected. Only boundaries are shown, as would be in undistorted images.

know the camera position $cam = (cam_x, cam_y, cam_z)$ and its orientation. Using position of the center of the landing pattern $(ud_x, ud_y, 1)$ in undistorted camera coordinate system we can compute a vector $d = (d_x, d_y, d_z)$ from camera to center to the landing pattern in world coordinate system. Knowing the altitude alt of the camera from ground and height of the landing vehicle we can compute the position of the landing pattern lan as:

$$lan = cam + d * \frac{alt - high}{d_z}. \tag{8}$$

A precise estimate of the MAV altitude is required. Experiments showed that the altitude error could be up to $1\ \mathrm{m}$ when relying on a laser rangefinder or other available sensors. Therefore, for purposes of the pattern detection, we compute the distance to the pattern from the apparent size of the ellipse.

IV. Camera calibration

The successful landing depends on precise localization of the landing pattern in world coordinates. Synchronization of MAV position to the time of image acquisition is important to correctly compute the global coordinates of the car. Calibration of the camera coordinate system with MAV coordinate system has the same importance.

Following steps describe the parameters taken into account when calculating the global position of the car:

- position translation of the camera origin relative to the MAV body
- angular shift of a camera mount to the MAV body
- time shift of camera data to the time of known position and orientation of the MAV

The position shift of the camera coordinate system to the MAV coordinate system was measured on a bench during the construction of the MAV. The experiments show that the best way to estimate the angular shift is to use real-world data. The time shift of camera data has to be estimated by experiment too because it depends on many hidden parameters of ROS and OS system. It varies with camera type, camera interface and real-time features of the underlying operating system.

We found it difficult to set all parameters at once, due to the time shift and angular shift being strongly connected. We designed two flight scenarios to automate the estimation.

Both scenarios rely on differential GPS up-to $5\ \mathrm{cm}$ localization precision. The MAV takes off from the center of the landing pattern, and therefore the target's position in world coordinate system can be automatically detected. The first scenario was designed to detect the angular shift of the camera, and therefore the MAV flies as stable along a predefined trajectory around the landing pattern. The second scenario contains aggressive maneuvers with a high angular rate of the MAV.

Finding the angular shift is split into two parts. First, the yaw angle α is estimated as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{9}$$

Then roll and pith angles are found separately using x and y global coordinates respectively, all obtained from designed flight. For example, to estimate the angular correction using angle $\gamma = \tan\frac{d_x}{f_x}$ we apply

$$x' = \frac{(x - d_x) \cdot f_x^2}{f_x^2 + d_x \cdot x}. \tag{10}$$

The search for the angular shift can be defined as an optimization task with variables α, d_x, d_y. The task was solved by finding a minimal value for all combinations of values α from interval $\langle -5°, 5° \rangle$ with step $0.5°$, values d_x, d_y from $\langle -50, 50 \rangle$ with step 2 using focal length $f_x = f_y = 261$ pixels. The task was solved offline using recorded flight data.

V. Guidance law

The guidance law we present is a modular pipeline consisting of five components. Following paragraph gives a shortlist of those components, which are subsequently describe in following sessions V-.1 to V-.4.

The first component is the **Cross detector** (presented in Chapter III), which provides measurements of car position in the world frame coordinate system. The measurements are processed by **Car state estimator**, which outputs an estimate of the car states in the means of nonholonomic, car-like model. The estimate is then conveyed to the **Car state predictor**, whose output is a future trajectory of the car. The trajectory is then taken as reference by the **MPC tracker**, which minimizes a quadratic error of MAV future states over the predicted future, to fly above the car. As a result, the MPC tracker outputs desired states (position, velocity, and acceleration) to the **State feedback controller**. The state feedback controller, being the last part of the pipeline, produces commands for the Pixhawk flight controller.

1) Car state estimator: The car state estimator was designed using the Unscented Kalman Filter (UKF) [4]. UKF allows estimation of system states using a nonlinear transfer function. A discrete, nonholonomic, car-like model with state box constraints was used to further allow precise prediction

of vehicle future movement:

$$\begin{aligned}
\mathbf{x}_{[n+1]} &= \mathbf{x}_{[n]} + \dot{\mathbf{x}}_{[n]}\Delta t, \\
\dot{\mathbf{x}}_{[n+1]} &= \begin{pmatrix} \cos\phi_{[n+1]} \\ \sin\phi_{[n+1]} \end{pmatrix} v_{[n+1]} \\
\phi_{[n+1]} &= \phi_{[n]} + \dot{\phi}_{[n]}\Delta t, \\
\dot{\phi}_{[n+1]} &= K_{[n]} v, \\
v_{[n+1]} &= v_{[n]} + a_{[n]}\Delta t, \\
K_{[n+1]} &= K_{[n]} + \dot{K_{[n]}}\Delta t,
\end{aligned} \tag{11}$$

where $\mathbf{x} = (x_x, x_y)^T$ is position of the car, ϕ is its heading, K is curvature of its turn, v is car scalar velocity, a is car scalar acceleration and Δt is sampling time difference.

State estimation served two purposes within our pipeline. First, it acted as a filter for incoming measurements from the *cross detector*, with measurements variance being adjusted with respect to the altitude. Second, it allowed estimating unmeasured states, later important for predicting car future movement. The resulting estimate is outputted at 100 Hz.

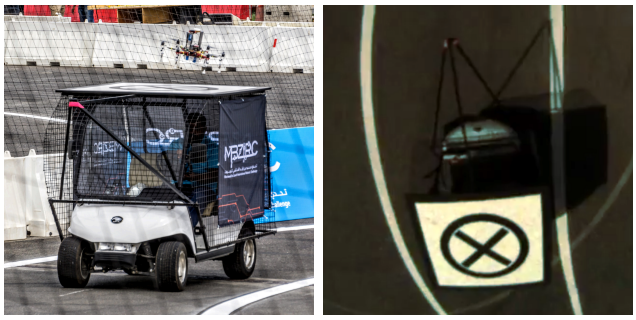
2) Car state predictor: Using the information from the *car state estimator*, we predicted its future trajectory using the same dynamic model. In the case of a general car trajectory, tracking with such prediction worked reliably up to 20 km/h. For purposes of the competition, we decided to use a prior knowledge about the competition arena. The curvature of the predicted trajectory was biased using a known map of the arena and the track on which the car was driving. Based on the observations during rehearsals, the day before the competition, this technique was further extended to bias predicted points by projecting them directly in the driving track. The predicted trajectory was outputted at 30 Hz.

3) MPC tracker: In our pipeline, a trajectory tracker is responsible for generating a set of desired states of the MAV (position, velocity, and acceleration) to follow a set trajectory (generated by the *car state estimator*). It uses a decoupled, 3-rd order translational dynamics to simulate a virtual MAV at 100 Hz. The virtual MAV is then controlled by Model Predictive Control with 8 s prediction horizon, also at 100 Hz. States of the virtual MAV are sampled and handed out to the *state feedback controlled* as a reference. Thanks to the MPC, the tracker provides necessary feed-forward action to follow the known future path. The particular MPC control approach is based on previous work presented in [3], further extended to support state constraints in velocity and acceleration.

4) State feedback controller: The final part of the pipeline is SO3 state feedback controller presented in [7]. SO3 controller provides 100 Hz feedback on the desired position, velocity, and acceleration of the MAV. Controller output is desired tilt angles, yaw rate and total thrust which are sent to Pixhawk embedded controller as a reference.

A. Approach and landing strategy

The whole mechanism of approaching and landing was realized as deterministic finite state automata. After the sequence was initiated, the MAV took off and moved to the center of the workspace, where it waited for the car to appear in the field of view of the camera. Such strategy minimizes the mean time of the search given the constraints of our MAV and the unknown (random) initial condition of car position and direction. After the car was spotted and the covariance of its state estimate surpassed a given threshold, the MAV started to follow the car while it maintained current altitude. In this case, the approach trajectory, produced by *Car state predictor*, was designed as minimum-snap with orthogonality constraint – the MAV first moved orthogonally to a point, where it would meet the car. Such approach mitigated unwanted direct movement towards the car, which would tilt the MAV in car direction and possibly lost it from the line of sight. After the MAV had been aligned with the car, a first descending stage was initiated. When descended to 4 m, the MAV was waiting for a precise alignment, after which a rapid landing stage started. During the rapid landing stage, a laser rangefinder provided a trigger for motor shut-off. In the case of lost alignment or lost tracking, the respective states were stepped back, or the whole automata was reset.



(a) (b)

Fig. 6: a) Our MAV landing on the car. b) The landing platform from MAV camera point of view.

VI. Experimental results

The challenge took place on a flat, rectangular area with dimensions 90×60 m. The area contained a marked track in a shape of figure 8 (see Fig. 8). The car was driving within the marked track, starting on a random place and heading in a random direction. A decreasing speed profile was defined, starting at 15 km/h and later slowing down to 5 km/h. The car was equipped with a 1.5×1.5 m landing platform, bearing a landing pattern (see Fig. 6).

Before the round, the competing team would place the MAV at a starting location and wait for a mark to initiate the flight. The team whose MAV would land the fastest wins. Penalization was issued if parts of the drone would fell off during the landing or for any human intervention after it took off.

A. Experimental platform

The MAV was assembled mostly from off-the-shelf components, except a few 3D-printed parts. Common DJI F550 hexacopter frame (see Fig. 7) was equipped with Pixhawk

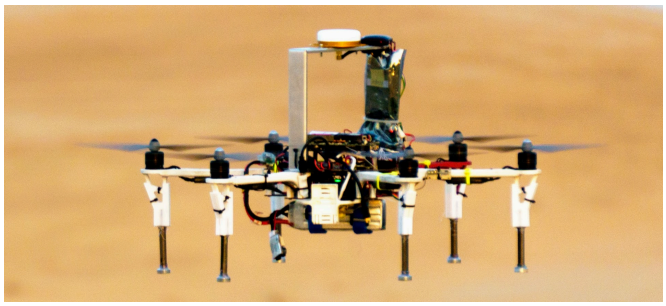


Fig. 7: MAV used in the experiments was based on DJI F550 hexacopter frame, Pixhawk stabilization board, and Intel NUC computer.

autopilot system, flashed with stock PX4 software. Our computer vision and guidance software were executed on Intel NUC with Core i7 processor. The MAV utilized a single down-facing MatrixVision Bluefox camera with fisheye lens. Drone's landing gear was equipped with neodymium permanent magnets to fix it after the landing. All custom software was build using ROS (Robot Operating System), running upon Ubuntu operating system.

B. Trial results

Our platform was tested in 4 trials over two days of the competition. First two were part of a separate challenge solely focused on landing on a moving vehicle, wheres the other two were part of a *Grand challenge*, where other robotic tasks were performed simultaneously in the same arena (Cooperative collecting of objects by multiple MAVs and a ground robot task). First two trials were completed successfully with the flight time of 1 min, 44 sec, and 0 min, 1 min, 28 sec. The third trial was also successful while having the best score of all teams – 0 min, 25 sec of flight time. The fourth trial was the only unsuccessful one due to the MAV misalignment in the final stage of the landing, which was falsely classified. Fig. 8 plots all three successful trials in a top-down view, as recorded by the MAV during the flight. Videos, capturing the trials, can be found at [11].

VII. Conclusion

We showed that platform presented in this paper is capable of landing on a car driving at speed 15 km/h, using only onboard computational resources. The platform was tested in MBZIRC 2017 competition, where it performed with the fastest landing time among all competing teams.

VIII. Acknowledgements

The work has been supported by CTU grant no. SGS17/187/OHK3/3T/13, the Grant Agency of the Czech Republic under grant no. 17-16900Y and by Khalifa University.

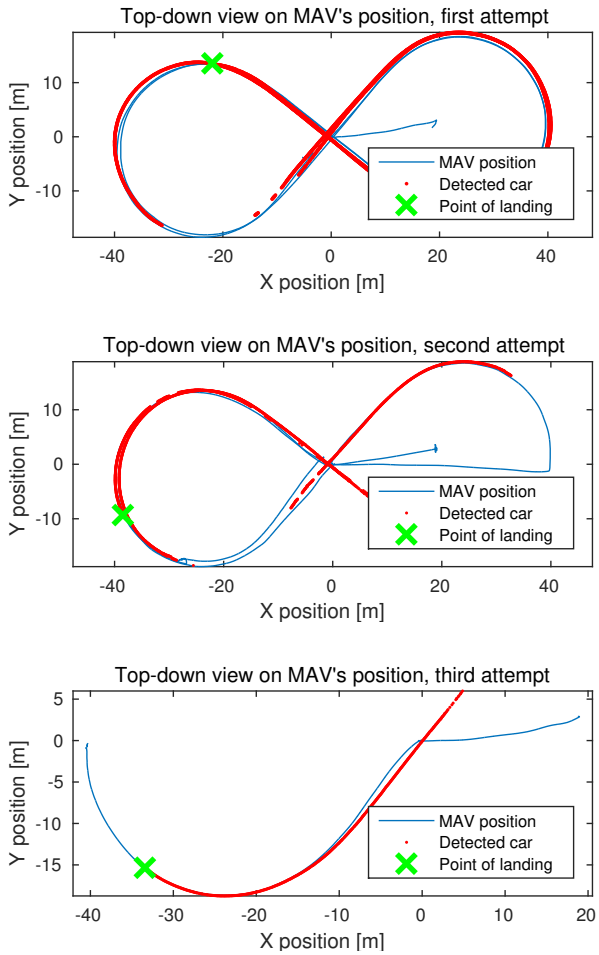


Fig. 8: Top-down view of all three successful trials: 1:44 min flight (top), 1:28 min flight (middle) and 0:25 min flight (bottom).

References

[1] J. Kannala and S. S. Brandt, A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 8, August 2006.

[2] Z. Guo and R. W. Hall, Parallel Thinning with Two-Subiteration Algorithms, Communications of ACM, vol. 32(3), March 1989, pp. 359-373.

[3] T. Baca, G. Loianno and M. Saska, Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles, in IEEE MMAR 2016.

[4] E. Wan and R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, IEEE AS-SPCC 2000.

[5] M. Saska, T. Krajnik and L. Preucil, Cooperative UAV-UGV autonomous indoor surveillance, IEEE SSD 2012.

[6] S. Lange, N. Sunderhauf, and P. Protzel, A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments, IEEE ICAR 2009.

[7] T. Lee, M. Leok, and N. McClamroch, Nonlinear robust tracking control of a quadrotor UAV on SE(3), Asian Journal of Control 15.2 (2013): 391-408.

[8] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, Visually guided landing of an unmanned aerial vehicle, IEEE Transactions on Robotics and Automation, vol. 19, no. 3, pp. 371380, Jun. 2003.

[9] M. Fu, K. Zhang, Y. Yi, Ch. Shi, Autonomous landing of a quadrotor on an UGV, IEEE ICMA 2016.

[10] J. Kim, Y. Jung, D. Lee, D. H. Shim, Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera, IEEE ICUAS 2014.

[11] Supplementary multimedia `http://mrs.felk.cvut.cz/ecmr2017landing`

Improving Sonar Image Patch Matching via Deep Learning

Matias Valdenegro-Toro[1]

***Abstract*— Matching sonar images with high accuracy has been a problem for a long time, as sonar images are inherently hard to model due to reflections, noise and viewpoint dependence. Autonomous Underwater Vehicles require good sonar image matching capabilities for tasks such as tracking, simultaneous localization and mapping (SLAM) and some cases of object detection/recognition. We propose the use of Convolutional Neural Networks (CNN) to learn a matching function that can be trained from labeled sonar data, after pre-processing to generate matching and non-matching pairs. In a dataset of 39K training pairs, we obtain 0.91 Area under the ROC Curve (AUC) for a CNN that outputs a binary classification matching decision, and 0.89 AUC for another CNN that outputs a matching score. In comparison, classical keypoint matching methods like SIFT, SURF, ORB and AKAZE obtain AUC 0.61 to 0.68. Alternative learning methods obtain similar results, with a Random Forest Classifier obtaining AUC 0.79, and a Support Vector Machine resulting in AUC 0.66.**

I. Introduction

One of the basic problems in robotics is data association, where sensor readings have to be associated with previous measurements, as the combination of sensor data reduces noise and improves robot understanding of the world. Autonomous Underwater Vehicles (AUVs) constantly struggle with data association, as the underwater environment is very hostile for sensing. Some common robot tasks that require data association are tracking and simultaneous localization and mapping (SLAM). Object detection/recognition can also benefit from data association in the form of matching images if the task is to locate an object and only a single training sample is available.

Acoustic sensing (Sonar) is used in underwater environments as sound can travel large distances on water with little attenuation. Optical cameras are not an option as light is attenuated and absorbed by particles in the water column. Interpretation of acoustic images is not trivial as unwanted reflections, noise, and low signal-to-noise ratio (SNR) degrades the amount of information that the AUV can gather.

For sonar, matching image patches to known objects or landmarks in the environment is an important problem. Matching can also be formulated for other tasks such as mosaicing [1], where sonar images must be registered before being combined to improve SNR ratio and image resolution. Matching sonar images is difficult due to viewpoint dependence.

In this work, we propose the use of Convolutional Neural Networks (CNN) to learn a matcher for sonar images. Our objective is to produce a function that takes two sonar image patches and makes a binary decision: both images correspond to different views of the same object, or not. Matching should be possible even as insonification[1] varies due to AUV or sensor movement, different views, and object rotation or translation.

CNNs have obtained very good results [2] in different tasks that use optical color images, such as object recognition [3] and transfer learning [4]. We have previously evaluated CNNs for object recognition in sonar images and found that they also improve the state of the art [5]. CNNs have also been used to match patches from color images [6] with high accuracy. These results motivate us to use CNNs for sonar data, as the trained network can learn sonar-specific information directly from the data.

We show that we can build and train a CNN that matches Forward-Looking Sonar (FLS) image patches with high accuracy (AUC[2] 0.91), surpassing the state of the art keypoint matchers such as SIFT and SURF (with AUC in the range 0.61 - 0.68).

Our contributions are: we propose an algorithm to generate matching pairs from labeled objects for training, we learn the matching function directly from labeled data, without any manual feature engineering, we show that it is possible to match sonar images with relatively high accuracy.

II. Related Work

Matching sonar images with high accuracy has been a unsolved problem for a long time [7] [1] [8]. This is due to specifics issues in sonar imaging, such as viewpoint dependence, non-uniform insonification, low signal-to-noise ratio, low resolution, and low feature repeatability [9]. Most methods that are used for different kinds of matching in sonar imagery are not specifically designed for sonar (originally developed for optical images), and do not consider sonar-specific information.

Kim et al. [10] matches keypoints detected with the Harris corner detection to register general sonar images. Vandrish et al. [8] compares the use of SIFT [11] with different feature methods for sidescan sonar image registration, concluding that for this task SIFT performs best. Hurtos et al. [1] uses Fourier-based features for registration of Forward-Looking Sonar images, with great success. These kind of features could be used to make a matching decision, but they only work appropriately when rotation/translation between frames are small. Pham et al. [12] uses block-matching guided by

[1]Matias Valdenegro-Toro is with Ocean Systems Laboratory, School of Engineering & Physical Sciences, Heriot-Watt University, EH14 4AS, Edinburgh, UK m.valdenegro@hw.ac.uk

[1]Amount of acoustic signal that "illuminates" the target area by the sonar sensor.

[2]Area under the ROC Curve

a segmented sonar image with a Self-Organizing Map for registration and mosaicing of sidescan sonar images.

A large portion of the research about matching sonar images is devoted to registration and mosaicing [10] [1]. Both processes require many assumptions on the kind of images and their content, specially when considering non-uniform insonification and simple transformations between images.

In comparison, CNNs [3] have been used to compare and match color image patches. Zagoruyko et al. [6] uses CNNs trained on a dataset of 500K matched patches with high accuracy in tasks such as stereo matching and descriptor evaluation. Raw matching performance is also good, but only possible due to the availability of large labeled datasets.

Zbontar and LeCun [13] also use CNNs for stereo matching, improving over the state of the art in several datasets. These recent results using CNNs motivate us to explore such algorithm for matching sonar images. CNNs have several advantages when applied to sonar imaging: they can learn sonar-specific information directly from raw data, they do not require feature engineering or specific data preprocessing, and they make little assumptions on input data.

III. Matching Sonar Image Patches with CNNs

A. Training Data Generation

Given a dataset containing labeled bounding boxes (including object classes), we generate matching and non-matching image pairs that are sampled from the dataset. We do this by using object class information to generate matching image pairs, and we also produce non-matching pairs that contain objects versus background. The dataset that we used for this purpose was originally designed for object detection. We generate the following kinds of pairs:

- **Object vs Object, Same class**. A matching pair is generated from two objects of the same class. We sample two random image crops of objects in the same class and generate one pair, typically both crops corresponds to different perspectives of the same object, or different insonification levels from the sensor.
- **Object vs Object, Different class**. A non-matching pair is generated from two objects from different classes. This makes the assumption that objects in the dataset are not similar across different classes.
- **Object vs Background**. A non-matching pair is generated by sampling one background patch that has IoU score lower than 0.1 with the ground truth and generating a pair with a random object image crop.

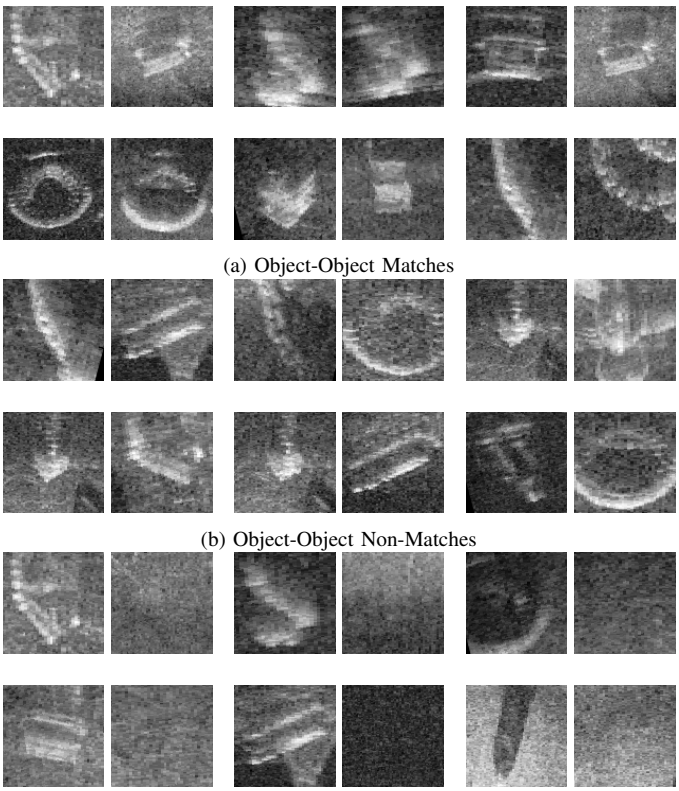
As the number of possible non-matching pairs is very large, we balance matches (positive) and non-matches (negative) samples to be $1:1$. This is done by sampling 10 matches per object, 5 non-matches between objects of different class, and 5 non-matches with background. The detailed algorithm is presented in Algorithm 1. We generate pairs of 96×96 image crops, as this is the most appropriate size for the objects in our dataset. A small sample of generated pairs is shown in Fig. 1.

Algorithm 1. Training Data Generation

Input: Labeled Image dataset I with bounding boxes B_i and class labels C_i, number of positive samples S_p, number of negative samples S_n.

Output: List of matching pairs L_m and list of non-matching pairs L_{nm}.

$L_m \leftarrow \emptyset, L_{nm} \leftarrow \emptyset$
for img $\in I$ **do**
  for object $o \in$ img **do**
    $OC \leftarrow$ crop B_o from img.
    for $i = 0$ to S_p **do**
      $MC \leftarrow$ sample random object p of class C_o and make an image crop.
      Append (OC, MC) to L_m
    end for
    for $i = 0$ to S_n **do**
      $NMC \leftarrow$ sample random object p of class $C_p \neq C_o$, and make an image crop.
      Append (OC, NMC) to L_{nm}
    end for
    for $i = 0$ to S_n **do**
      $BC \leftarrow$ sample random background patch and make an image crop.
      Append (OC, BC) to L_{nm}
    end for
  end for
end for



(a) Object-Object Matches

(b) Object-Object Non-Matches

(c) Object-Background Non-Matches

Fig. 1. A small sample sonar image patches labeled as matching or non-matching that were generated by our algorithm. These patches were captured with an ARIS Explorer 3000 Forward-Looking Sonar.

B. CNN Architecture

We base our architectural choices on the work of Zagoruyko et al. [6]. This paper introduced CNNs for matching image patches and propose three different architectures for that task: A siamese, pseudo-siamese and a two-channel architecture. We use the two-channel and siamese architectures. In the two-channel architecture, both input images (denoted as I_A and I_B) are merged to form a two-channel image, which is the input to our neural network.

The siamese architecture uses two branches that share weights, I_A is input to the left branch, while I_B is input to the right branch. The output of each branch is a feature vector of a fixed size. Both feature vectors from each branch are concatenated to form a single vector that is input to a decision network (formed only of fully connected layers). The idea of a siamese network is that both branches share weights and will learn patch invariant features that are useful for the decision network to produce a matching decision. We use two 96×96 input images to be matched.

We use the following notation for CNN layers: Conv(N_f, $F_w \times F_h$) is a convolutional layers with N_f filters of width F_w and height F_h. MP(P_w, P_h) is a max-pooling layer with sub-sampling size of $P_w \times P_h$, and FC(n) is a fully connected layers with n output neurons.

We designed four CNN architectures, two using a 2-Channel approach, and two using a Siamese architecture. We obtained these architectures by performing grid search over a defined set of variations, including depth, number of filters, and filter size. We now describe the 2-Channel architectures:

- **2-Channel CNN Class**. This network is designed to output a binary matching decision (match or non-match), with a two-element output probability distribution given by a softmax function. The full network architecture is Conv(16, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-Conv(16, 5×5)-MP(2, 2)-FC(64)-FC(32)-FC(2). The network is trained using a categorical cross-entropy loss function, as the matching decision is formulated as a classification problem.
- **2-Channel CNN Score**. This network outputs a matching score in the range $[0, 1]$ with a sigmoid function. The full network architecture is Conv(16, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-Conv(16, 5×5)-MP(2, 2)-FC(64)-FC(32)-FC(1). This network is trained using binary cross-entropy loss function, and the activation at the output is sigmoid.

The Siamese architectures are based on branches with configurations Conv(16, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-Conv(64, 5×5)-MP(2, 2)-Conv(32, 5×5)-MP(2, 2)-FC(96)-FC(96). The output feature vector contains 96 elements, and the output activation is sigmoid. From this branch architecture we derive the following architectures:

- **Siamese CNN Class**. Both branch outputs are concatenated to form a 192 element vector, that is passed through a decision network with configuration FC(64)-FC(2). The output activation in this case is softmax. This network is trained with a categorical cross-entropy loss.
- **Siamese CNN Score**. Same as the previous architecture, but the decision network has configuration FC(64)-FC(1), with a sigmoid output activation. This network is trained with a binary cross-entropy loss.

All four architectures were obtained by doing grid search over a varying number of layers, convolutional filters and fully connected neurons. The categorical and binary cross-entropy loss functions are the same for the case of binary classification, with the only difference being the application to a single output or to a two-element vector (see Eq 1).

$$L(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) = -y_0 \log(\hat{y}_0) - (1-y_0) \log(1-\hat{y}_0) \tag{1}$$

All architectures use ReLU activations, except at the output layers, and are trained on the same dataset, and learn to discriminate sonar image patches. Dropout [14] is used after every fully connected layer (except at outputs). We also evaluated the use of Batch Normalization [15] but Dropout is superior in achieving good generalization performance. Our original design was the class output networks, posed as a classification problem, which works well but interpretation of the output is not trivial as the scores are correlated with the classification outputs. This motivated us to explore the scoring architecture that outputs a score directly that can be separated to obtain a matching decision by a simple threshold . Note that [6] uses networks that only output a score, while we both evaluate continuous score and discrete classification outputs.

C. Training

Our networks are trained using stochastic gradient descent from random initialized weights. We train using mini-batch gradient descent using a batch size of $b = 128$ images. We adopt the ADAM optimizer [16] for accelerated training and learning rate decay. The initial learning rate is $\alpha = 0.1$. All networks are trained for 5 epochs. We tuned this value on a validation set with early stopping.

In order to prevent the network from learning patterns in the order images are presented, we augment the dataset and for each training pair (A, B), we add the pair (B, A) to the augmented training set. No other data augmentation was used.

IV. Experimental Evaluation

A. Data

We have captured a dataset of marine debris objects in our water tank with an ARIS Explorer 3000 (FLS). This dataset consists of 2072 images with ~ 2500 total object instances labeled in 9 classes (Metal Cans, Bottles, Drink Carton, Metal Chain, Propeller, Tire, Hook, Valve, Background). On this dataset we ran our matching pair generation algorithm (Algorithm 1). In order to evaluate generalization performance of our networks, we split the dataset according to object class, before obtaining train and test splits. This generated two datasets:

- **Dataset D**: In this dataset the train and testing sets are generated with different objects. Classes 0-5 are used to generate the training set, while classes 6-9 are used to generate the test set. All of our matching networks are trained on this training set.
- **Dataset S**: This dataset is generated using all classes, and split into a training and testing set. From this split we only use the testing set to evaluate performance.

The training set (from Dataset D) contains 39840 matching and non-matching pairs (50% each). Both testing sets (D and S) contain 7440 matching and non-matching pairs, also balanced. All reported metrics are evaluated on the test set.

B. Matching Performance

In this section we evaluate raw matching performance. We plot the ROC curve, and report the Area Under the Curve (AUC). We also obtained accuracy scores for the test set. Accuracy for the matching networks was obtained by considering the raw match probability p (second element of the softmax output, or the sigmoid output score) and taking the class with maximum probability (Eq 2).

$$c = \arg\max\{1 - p, p\} \tag{2}$$

We compare both our matching networks with the state of the art keypoint detectors and feature extractors, namely SIFT [11], SURF [17], ORB [18] and AKAZE [19]. SIFT and SURF represent the best keypoint detectors for optical images, while ORB was chosen to evaluate its binary features. While it is known [1] that these algorithms do not perform well in sonar, there is no other comparison point, as no keypoint detectors have been developed specifically for sonar images. We also compare with Machine Learning (ML) based methods, namely a Support Vector Machine (SVM) as classifier, a Support Vector Regressor (SVR) to regress a score, and Random Forest (RF) classifier and regressor.

Accuracy for keypoint algorithms is obtained by considering a positive match when the ratio test [11] gives at least 1 good match. If there are no good matches, then we output a negative match. This threshold is low on purpose to evaluate the best performance of a keypoint matching system.

As our dataset is generated using one type of matching pairs and two different types of non-matching pairs, we also compute the accuracy over each kind of match. This is reported as "Obj-Obj $+$" for Object-Object matching pairs, "Obj-Obj $-$" for Object-Object non-matching pairs, and "Obj-Bg $-$" for Object-Background non-matching pairs.

Table I displays the main results, only considering the best performing matching networks. Our methods have considerably higher AUC and mean accuracy, which shows that using neural networks for matching sonar images does have a considerable improvement over the state of the art. The 2-Chan CNN Class network has an advantage of 22.4 AUC percentage points over ORB, with a corresponding increase of 31.3 accuracy percentage points.

Classical keypoint detectors match sonar images with a chance that is slightly better than chance, with the best classical method being ORB with 0.682 AUC. Both our matching CNNs outperform classic matches by a considerable margin. Our class matcher also outperforms the scoring matcher. This is due to the fact that scoring matcher is considerably harder to train because the sigmoid output easily saturates. This also contrasts with the results from [6] as they use $\{+1, -1\}$ scoring with a hinge loss. Our results show that a softmax output with cross-entropy loss can outperform a saturating non-linearity.

Machine Learning methods also perform poorly, but better than keypoint matching. A RF classifier has the highest non-CNN AUC at 0.795, but their Obj-Obj positive accuracy is considerably poor than the alternatives. SVM and SVE have AUC that is close to keypoint matching.

Classic matchers have a higher Obj-Obj positive accuracy, and this can be explained by overconfident predictions that classify too many pairs as positive matches. This can easily produce high accuracy for positive matches, but will hurt the performance of negative matches. Our matching networks seem to be more balanced, but still their lowest accuracy is when they need to predict a positive match. Both results show the difficulty of matching sonar images. ML methods suffer from the opposite, where they are very accurate for negative matches, but suffer in accuracy for positive matches.

Fig. 2 shows the corresponding ROC curves for the classic matchers, ML-based methods, and our best performing networks. The positive class probability p of Class matching networks and SVM/RF-Class is used to construct the ROC curve, while for Score matching networks and SVR/RF-score the raw score is considered to produce the curve. For keypoint detectors we vary the minimum number of good keypoint matches to declare a positive match. Keypoint matchers have a curve that is very close to random chance, while our methods are closer to a perfect matcher. There is still a considerable room for improvement in the sonar image matching problem. All tested methods produce results that are better than random chance, and RF-based methods are superior when compared to keypoint matching, but still our matching networks are considerably better.

Tables III and II show a comparison of all our matching networks on the **S** and **D** datasets. Corresponding ROC curves are shown in Fig. 3a and Fig. 3b.

Looking at Fig. 3a, we can see that network 2-Chan CNN Class performs the best when compared to the scoring network, but this trend reverses when looking at Siamese networks (Fig. 3b), as the highest AUC is obtained by Siamese CNN Score. When comparing performance on **S** and **D** datasets, we show that performance slightly increases when evaluating on the same objects as the training set (Test set **S**). This is expected and shows a slight amount of overfit to the training set objects. But generalization performance to unseen objects (Test set **D**) is still good.

Finally, Table IV offers a breakdown of accuracy in our matching networks over the three different match pairs on both datasets **S** and **D**. When evaluated in different objects, all networks have decreased performance on Object to Object positive matches, while at the same time having adequate

Method	AUC	Mean Accuracy	Obj-Obj + Acc	Obj-Obj − Acc	Obj-Bg − Acc
SIFT	0.610	54.0%	74.5%	43.6%	44.0%
SURF	0.679	48.1%	89.9%	18.6%	35.9%
ORB	0.682	54.9%	72.3%	41.9%	60.5%
AKAZE	0.634	52.2%	95.1%	4.8%	56.8%
RF-Score	0.741	57.6%	22.5%	88.2%	97.2%
RF-Class	0.795	69.9%	12.5%	97.7%	99.7%
SVR-Score	0.663	70.5%	57.2%	66.6%	87.5%
SVM-Class	0.652	67.1%	54.4%	69.1%	90.5%
2-Chan CNN Class	0.910	86.2%	67.3%	95.2%	96.1%
2-Chan CNN Score	0.894	82.9%	68.0%	96.1%	84.5%

TABLE I. Comparison of classic keypoint algorithms for matching versus our two best performing matching networks. Area Under the ROC Curve (AUC), Accuracy at match threshold zero, and Accuracy for each match type is reported for Test Set **D**.

Network Type	Output	Test Objects	AUC	Mean Accuracy
2-Chan CNN	2 Class	Different	0.910	86.2%
2-Chan CNN	Score	Different	0.894	82.9%
Siamese CNN	2 Class	Different	0.855	82.9%
Siamese CNN	Score	Different	0.826	77.0%

TABLE II. Accuracy and Area Under the ROC Curve (AUC) metrics for Test Set **D**

Network Type	Output	Test Objects	AUC	Mean Accuracy
2-Chan CNN	2 Class	Same	0.944	86.7%
2-Chan CNN	Score	Same	0.934	85.4%
Siamese CNN	2 Class	Same	0.864	75.8%
Siamese CNN	Score	Same	0.895	80.6%

TABLE III. Accuracy and Area Under the ROC Curve (AUC) metrics for Test Set **S**

performance in both negative cases. For evaluation in the same objects as the training set, this trend reverses (as expected) and the largest accuracies are reported in the Object to Object positive matches. This indicates that the networks are slightly overfitting the objects in the training set, but still they do provide an improvement over keypoint matching and ML-based methods in unseen objects. Matching an object to a different view of the same object is also a hard problem, and discarding matches from different objects or to background is easier.

C. Discussion

Our 2-Channel CNN Class matching networks performs well, with an AUC of 0.91 when evaluated on unseen data, and AUC of 0.94 on a dataset that shares objects with the train set. We have not seen any previous work claiming to match sonars images with such performance. Classic methods (SIFT/SURF and ORB) are known to work poorly in sonar, but still they are being used for registration in a large part of the literature [10] [7] [8]. We have not seen quantitative results of keypoint matching in sonar images, and our extensive evaluation is useful for comparison.

One clear limitation of our evaluation is the small size of our datasets (39K for training and 7K for testing). We believe our networks could greatly benefit from a bigger dataset, containing more variability in objects as well as more object views. A dataset captured in real underwater conditions is ideal but hard to produce.

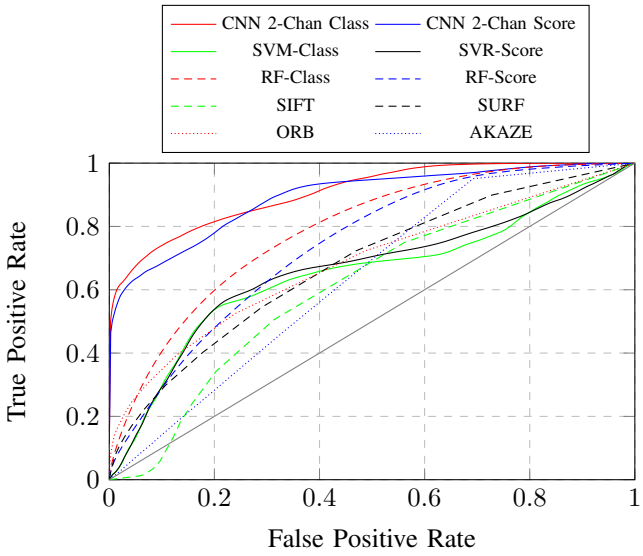


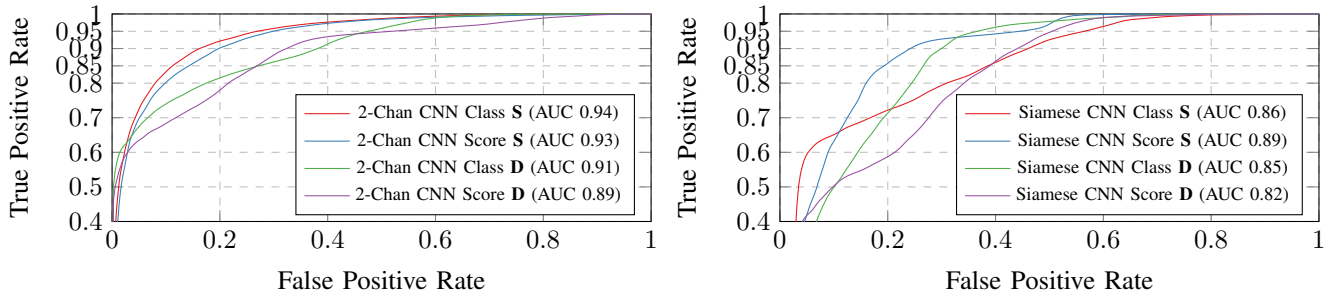
Fig. 2. ROC curve comparing our matching networks with ML-based methods (SVM/SVR and Random Forest, RF) and keypoint matching methods (SIFT, SURF, ORB and AKAZE). Our methods clearly outperform other methods when used to match sonar images. These curves were evaluated on the **D** dataset. The grey line represents random chance limit.

V. Conclusions

We have proposed the use of CNNs to learn a matching function for sonar images. Our results show that our 2-Chan CNN Class network can match FLS image patches with high accuracy (AUC 0.91), while classic keypoint matching methods can do it only with low accuracy, slightly better than random chance (AUC 0.61 to 0.68). ML-based methods (SVM and Random Forests) are also inferior (AUC 0.65 to 0.80).

We believe that our results are appropriate for the small (39K samples) training dataset that we possess, and they could improve if more data and object/background variability is available. Our dataset was captured under laboratory conditions in a small water tank containing household garbage objects, and our work can surely be improved with a dataset captured in real underwater scenes.

Our method is limited by available number of images and objects in them. Our network architecture could fail to generalize with other objects, specially if their shape is radically different. Background is also a concern, as the



(a) 2-Channel CNN

(b) Siamese CNN

Fig. 3. ROC curves comparing 2 Channel and Siamese matching networks on test sets **S** and **D**. This comparison shows the difference in performance Test set **S** was generated with the same objects as in the training set. Test set **D** was generated with different objects than in the training set.

		Different Objects			Same Objects		
Network Type	Output	Obj-Obj + Acc	Obj-Obj − Acc	Obj-Bg − Acc	Obj-Obj + Acc	Obj-Obj − Acc	Obj-Bg − Acc
2-Chan CNN	2 Class	67.3%	95.2%	96.1%	86.6%	75.7%	97.8%
2-Chan CNN	Score	68.0%	96.1%	84.5%	85.0%	77.5%	93.7%
Siamese CNN	2 Class	62.9%	89.9%	96.0%	92.2%	39.4%	95.8%
Siamese CNN	Score	49.2%	84.7%	97.0%	89.1%	55.3%	97.3%

TABLE IV. Accuracy by matching type, for the **S** and **D** datasets

background in our water tank is not representative of a typical underwater environment.

We expect that AUV perception will benefit from our work and open possibilities of advanced CNN-based matching and registration methods. Our work can be applied to any kind of sonar, as we did not make assumptions on the kind of image produced by the sonar.

As future work, we would like to build a similarity function for sonar images instead of making a binary decision and learn from image patches in a unsupervised way. The best case is to learn from a dataset that contains automatically matched scenes with another sensor (like depth in optical images) as [13] and [6] do.

Acknowledgments

This work has been partially supported by the FP7-PEOPLE-2013-ITN project ROBOCADEMY (Ref 608096) funded by the European Commission. The author would like to thank Leonard McLean for his help in capturing data used in this paper.

References

[1] N. Hurtós, Y. Petillot, J. Salvi *et al.*, "Fourier-based registrations for two-dimensional forward-looking sonar image mosaicing," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5298–5305.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[5] M. Valdenegro-Toro, "Object recognition in forward-looking sonar images with convolutional neural networks," in *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–6.

[6] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[7] S. Negahdaripour, M. Aykin, and S. Sinnarajah, "Dynamic scene analysis and mosaicing of benthic habitats by fs sonar imaging-issues and complexities," in *OCEANS'11 MTS/IEEE KONA*. IEEE, 2011, pp. 1–7.

[8] P. Vandrish, A. Vardy, D. Walker, and O. Dobre, "Side-scan sonar image registration for auv navigation," in *Underwater Technology (UT), 2011 IEEE Symposium on and 2011 Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC)*. IEEE, 2011, pp. 1–7.

[9] N. Hurtós, N. Palomeras, S. Nagappa, and J. Salvi, "Automatic detection of underwater chain links using a forward-looking sonar," in *OCEANS-Bergen, 2013 MTS/IEEE*. IEEE, 2013, pp. 1–7.

[10] K. Kim, N. Neretti, and N. Intrator, "Mosaicing of acoustic camera images," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 152, no. 4, pp. 263–270, 2005.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] M. T. Pham and D. Guériot, "Guided block-matching for sonar image registration using unsupervised kohonen neural networks," in *2013 OCEANS-San Diego*. IEEE, 2013, pp. 1–5.

[13] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. IEEE, 2011, pp. 2564–2571.

[19] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Mach. Intell*, vol. 34, no. 7, pp. 1281–1298, 2011.

Coherent swarming of unmanned micro aerial vehicles with minimum computational and communication requirements

Daniel Brandtner and Martin Saska

***Abstract*— An algorithm designed for stabilization and control of large groups of micro aerial vehicles (MAVs) - multirotor helicopters - without any explicit communication is proposed in this paper. The presented algorithm enables a swarm of MAVs to maintain its coherence and perform a compact motion in complex environments while avoiding obstacles with only very limited computational and sensory requirements. The method is very robust to incomplete sensory information, it enables a fully distributed applicability, and it is highly scalable. Increasing amount of MAVs even improves the required coherence behaviour. Numerous simulations in different environments were conducted to verify the algorithm, show its potential, and explore its various configurations.**

I. INTRODUCTION

Decreasing size and increasing robustness of micro aerial vehicles (MAVs) allow us to consider deployment of large multi-MAV groups instead of heavy and well-equipped unnamed aerial vehicles. Distribution of sensory power to small flexible units increases reliability of the system and enables applications where distributed sensing or acting is necessary and which would not be possible with a single vehicle. Substitution of a large vehicle by a team of light MAVs is especially important due to safety reasons in scenarios where the autonomous system may interact with humans and where the large platforms may cause injury and damage objects in their proximity. The applications with presence of people are often located in urban and indoor environment, where global navigation satellite systems (such as GPS) are not available or their precision is not sufficient for group stabilization, and onboard sensors are required for mutual localization of individuals in the group. The proposed swarm stabilizing algorithm is designed for using vision based relative localization [1] by onboard cameras, carried by MAVs, which was already successfully employed by our team for cooperative surveillance by a team of self-localized and stabilized MAVs and for compact formations flying [2], [3], [4]. Although we call the group in [2] as a swarm, the method requires centralized planning and MAV coordination. In this paper, we propose a novel method that allows to control the visually stabilized MAV groups in a fully decentralized way and without explicit communication, which is the main requirement of swarm robotics research (see Fig. 1 for motivation).

The authors are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague. martin.saska@fel.cvut.cz



Fig. 1. Motivation: A group of MAVs above the sand dunes following the sun-set direction.

A. Related Work

Swarm robotics, which is aimed to implement a collective behaviour without an explicit central control law, usually arises from principles of swarm intelligence dealing with decentralized, self-organized multi-agent systems [5]. Many of these swarming algorithms are inspired by behaviours observed on animal interactions. For example, the BOID model is inspired by bird flocking [6], algorithm in [7] by nesting and foraging habits of various species of insects, and approaches in [8], [9] by fish schooling.

The methods [7], [8], [9] as well as for example PSO (Particle Swarm Optimization)-based algorithms [10], [11] and methods [12], [13], [14] require global knowledge and communication unlike the approach discussed in this paper, which is strictly distributed. Also the potential field-based swarming method in [15] relies on explicit wireless communication between neighbour MAVs, while the cognitive-based algorithm in [16] uses global positioning systems and broadcasting of MAV positions. Frequently used BOID models [17], [18], similarly as the proposed approach, rely on mutual localization and stabilization of neighbouring particles, but require a full information on the relative position and even velocity of neighbours, whereas our method uses only a binary information on the presence of the neighbours in proximity of the particular MAV. This is a very important capability for control of swarms of light-weight and simple MAVs as it significantly reduces requirements on the onboard mutual localization system.

The proposed method, which is inspired by a study of coordination of ground robots in [19], enables the swarm of MAVs to perform a coherent motion towards a given target using only an onboard binary light sensor that can recognize whether the target is in sight or not. Although in [19], the group members share this information with neighbours to get a vague common idea of the target location, our method uses

individual on-board estimation of the target position without the need of the explicit communication.

To sum up the contribution of this paper, the presented algorithm enables to stabilize an MAV swarm in environments with obstacles without the need of global positioning systems and an explicit wireless communication, and with significantly limited computational and sensor requirements. Moreover, the important contribution of this paper is a comprehensive analysis of the achieved swarming behaviour aimed at confirmation/disapproval of a set of hypothesis assumed in swarm robotics literature or being compiled based on our experience with the BOID models used for MAV swarm control.

B. Problem Definition and MAV set-up

The task solved in this paper is stabilization of a swarm of M MAVs and its navigation into a desired location in environments with static and dynamic obstacles. For the swarm coherence, it is required that all swarm members are relatively stabilized with at least α neighbours in the group, which means that at least α helicopters are mutually localized by each MAV using onboard sensors. Therefore, we assume that all M MAVs in the swarm are equipped with the sensors enabling to discern the adjacent robots (called "neighbours"), localize the obstacles, and detect direction of the required target. We suppose that the neighbours detection (not necessarily full localization - only a binary information on a presence of a neighbour) is ensured within a range of detection with radius R. Let us denote the actual number of detected neighbours within radius R by variable N.

Also the position of the target does not need to be observed precisely. In the presented experiments, a set of 4 simple vision sensors is deployed around the robot providing information in which quadrant corresponding to the particular sensor the target is located (no precise information on the bearing and distance of the target is required). The obstacles are localized in a detection range Ra, where $Ra < R$, and again the knowledge of distance to obstacles is not required and also the bearing estimation error can be very high (30° error was considered in the experiments).

II. Swarming Model

The propoded method is inspired by principles of the swarming model introduced in [19] to control a group of ground robots. We adopted the basic swarming rules, modified them, and integrated them into a multi-MAV visually-stabilized system. For MAV control strictly without communication (in [19] a communication channel was used to share the information among the robots), constraints of the onboard visual relative localization mechanism have to be considered. Additionally, motion constraints of the low level onboard model predictive control technique [20], which is used to deal with the high dynamics of MAVs, are integrated into the swarming approach.

The proposed swarming model used for MAV-swarm stabilization is composed of three behavioural states: *forward* (default setting for all MAVs), *coherence* and *avoidance*. MAVs in the *forward state* fly straight with a constant velocity as long as they are forced to enter into another state. If the required connection with at least α neighbours is lost (i.e. $N < \alpha$) for an MAV, it enters into the *coherence state*, in which turns and flies back until the connection is restored. Once the MAV enters back into the localization range ($N >= \alpha$), it performs a random turn and returns back into the *forward state*. The coherence manoeuvre can be triggered only once per cf cycles to allow the MAV to regain its visual connection. The random turn is applied if only the swarm coherence is required and the swarm movement direction is not controlled. If a desired target is required to reach by the swarm members, the turning is influenced by the estimated position of the target as mathematically described in Algorithm 1. A proper setting of α is important mainly for large swarms, we are focused on, where keeping the localization constraints between all pairs of MAVs is not possible. Properly defined α enables a wider and better structured swarm, where (ideally) each robot has precisely α neighbours and the swarm forms a stable regular net, which is not precomputed and arises autonomously.

while *true* **do**
  foreach *MAV* m **do**
  if m *is in sensory range and visible* **then**
    $neighbors++$
  end
  ;
  if $state = Forward$ *AND* $neighbors < prevNeighbors$ *AND* $neighbors < alpha$ **then**
    $turnAngle = -\pi$
    $lostNeighbors = prevNeighbors$
    $counter = 0$
    $state = Coherence$
  else if $counter = cf$ *AND* $state = Coherence$ **then**
    if $visible(target)$ **then**
      $offset = activeSensor.direction$
    else
      $offset = 0$
    end
    $turnAngle = offset + rand(-\pi/2, \pi/2)$
    $state = Forward$
  else
    continue in same direction
  end
  $prevNeighbors = neighbors$
  $counter++$
end

Algorithm 1: Swarm coherence algorithm implemented onboard of each MAV in a decentralized way.

The *avoidance state* is used if an MAV gets closer to another MAV/MAVs or an obstacle than a given threshold and an evasive manoeuvre has to be performed. This manoeuvre (the same manoeuvre is applied for obstacle avoidance as well as for mutual collisions of swarm members avoidance)

controls the MAV in the opposite direction to avoid the collision (see sketch of this behaviour on Figure 2).

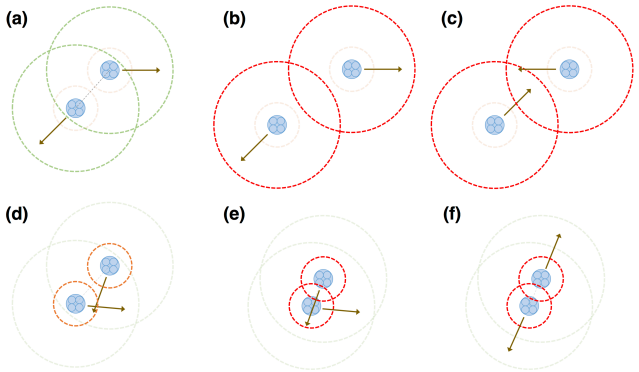


Fig. 2. Basic principle of the coherence (a-c) and avoidance abilities (d-f) of the algorithm. The larger circle indicates the neighbour detection range R and the smaller circle the obstacle detection range Ra. The MAVs fly initially straight in random directions in the *forward state* - (a). When the MAVs lose mutual localization, they enter the *coherence state* - (b), turn around and fly in the opposite direction - (c). When contact is renewed, the MAVs perform a random turn and fly straight in this direction in the *forward state* - (d). If the MAVs move too close to each other, an evasive manoeuver is triggered - (e). They enter the *avoidance state* and are mutually repulsed - (f).

III. MAV SWARMING BEHAVIOUR ANALYSIS

One of the main contributions of this paper is to analyse behaviour of such a minimalistic swarming algorithm, which should verify its usability for large swarms of micro aerial vehicles with very limited sensory equipment. The proposed method is designed in a way that numerous different behavioural patterns can be achieved using various sets of algorithm parameters. Numerous hypothesis from [5] and hypothesis motivated by studies of swarming behavior of ground robots in [19] and by behaviour of MAV swarms stabilized by frequently used BOID model in [17] have been formulated based on expected influence of the parameter setting on the algorithm performance. These hypothesis have been experimentally validated (approved or disproved) and optimal parameters setting was found for different required behavioural patterns. The simulations were run with the identified MAV model in a realistic robotic simulator V-REP. For each configuration of tested parameters values, 10 simulations of 10 minutes flight have been performed. The following three factors of the swarm behaviour have been studied in the analysis.

The swarm coherence is described by the ratio of the sum of time intervals where the swarm forms a connected graph to the total time of the experiment. This means the swarm is considered as "incoherent" from the first moment an MAV, or a sub-swarm of MAVs, becomes disconnected from the rest of the group. The coherence is the most important aspect of a coherent swarming mechanism and should be maximized.

The swarm spreading is defined as the standard deviation of the MAVs positions from the swarm center. The swarm spreading may be maximized to cover large areas with a minimum number of robots, but in some applications compact swarms are required (e.g. for motion in clustered environments).

The state distribution indicates time intervals that MAVs spend in the different states of the swarming model (the *forward*, *avoidance*, and *coherence* states). In case of the navigation towards a given goal, the time spent in the *forward state* is maximised. In this state, the MAV is most efficient at performing the given task, since it does not perform manoeuvres perturbing its flight.

A. Analyses of influence of the parameter α and the number of the robots

In this section, hypothesis that with the higher α (the required number of relatively localized neighbours of each MAV) the swarm coherence is increased, the swarm spreading is decreased, and the time spent in *avoidance* and *coherence* states is decreased and that using large swarms increases the swarm coherence, increases the swarm spreading, and increases the time spent in *avoidance* and *coherence* states are evaluated in simulations with 5, 10, and 20 robots. The data obtained in the simulations are displayed on graphs in Figures 3 and 4, showing the behaviour of different-sized swarms with different values of α.

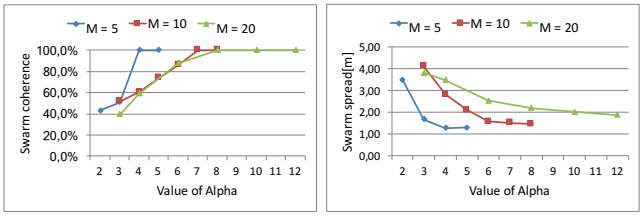


Fig. 3. Swarm coherence and swarm spreading for different number of robots and parameter α.

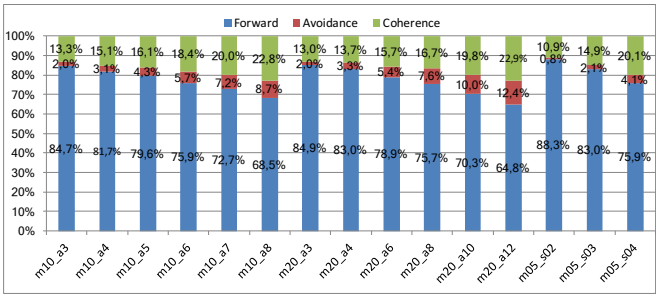


Fig. 4. State distribution during the experiments from Figure 3. The labels on the x-axis indicate the used configuration of the swarm (eg. m05_a04 represents a swarm of 5 MAVs using $\alpha = 4$).

Based on these data we can confirm that with higher value of α the swarm coherence is increased. Mainly for the bigger swarms, the probability that the group connectivity is broken decreases almost linearly with values of the parameter α, reaching probability 0% for $\alpha = 7$. With growing α, the swarm spreading decreases as expected, again this effect is observed mainly for smaller values of α. In contrast with the hypothesis, growing value of parameter α increases the time

spent in the *avoidance state* due to the increased compactness of the swarm and also in the *coherence state*, which is caused by an increased sensibility to neighbour loss.

More MAVs in the swarm does not increase the swarm coherence contrary to our hypothesis. The results indicate that there proper values of α (the proper values are 7 and 8 for our MAV model) to guarantee the coherence of the swarm. These values are not dependent on the number of robots, which is an important observation and it supports the scalability of the method. With larger swarm, swarm spreading is increased, but this increase is not proportional. Surprisingly, in larger swarms, the MAVs do not spend more time in the *avoidance* and *coherence* states. With constant α, the state distribution is similar for different swarm sizes. The behaviour of an individual MAV is influenced only by its local surrounding, not by the total number of robots, which is also an important observation for swarm scalability in comparison with the BOID model that suffers from increasing density of swarm members in the center of large groups even-though approaches using the BOID model require more precise sensors and bigger computational power than the proposed method. In the following simulations, 10 robots and $\alpha = 5$ will be used to test the impact of other parameters on the swarm coherence.

B. Influence of sensors capability on the swarming behaviour

Since the proposed approach is designed to allow coherence swarming using limited sensory information, the influence of the sensors capability on the swarming behaviour is important to study. Let us again postulate a set of hypotheses that are considered in swarm literature and based on our experience with BOID models.

Longer range of sensors used for relative localization of neighbours increases the time MAVs spent in the *forward state*, increases swarm coherence, and increases swarm spreading. Based on the results summarised in Figures 5 and 6, we can confirm the expected positive influence of the sensor range. With better sensors, MAVs stay longer in the *forward state* and so travel a longer distance until they have to turn back. Due to the reduced time spent in the *coherence state*, which increases probability to break an MAV apart from the rest of the group, the swarm coherence is increased. With the longer range, MAVs can keep bigger spacing between them, which increases the swarm spreading.

Smaller angle of view of the sensors for relative localization decreases swarm coherence, decreases swarm spreading, and increases the time spent in the *coherence state*. The influence of the view angle of the relative localization sensors is an important phenomena that needs to be studied, since simple and light weight sensors carried onboard of micro aerial vehicles (such as simple monocular cameras) often do not offer 360 degrees view. Even more importantly, the possibility to deploy swarming algorithms with such a limited sensors enables to study swarming behaviour in nature, where animals have almost always limited sensing capabilities in this way. Two parameters settings were used to evaluate the hypothesis. The first settings ($\alpha = 7$ and $R = 8m$) provided the best performance with a full-view sensor in the previous experiments. As shown in Figure 7 (the initial position of robots is shown in Figure 8 - Left), such initialised swarm maintains a sufficient performance also with significantly reduced angle of view. The second setting ($\alpha = 5$ and sensor range $R = 4m$) performs much worse with a narrower field of view, although presented a sufficient coherence with the 360 degrees sensor. To conclude, we can confirm that smaller angle of view decreases coherence in general, but this effect may be neglected for well set up algorithm parameters. With this ability, the proposed algorithm also surpasses the BOID approaches, which are highly dependent on the limited field view as shown in our prior work with the same MAV platform controlled by BOID [17], [18]. Besides, the experiments in Figure 7 confirmed that swarm spreading is decreased with limited field of view and the time spent in the *coherence state* was increased, while the occurrence of the *avoidance state* remains roughly the same.

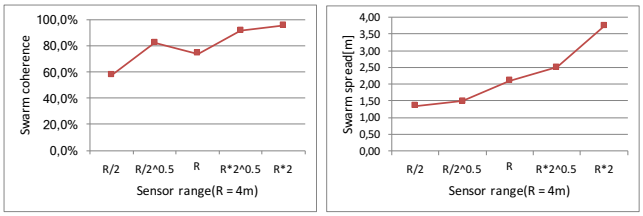


Fig. 5. Swarm coherence spreading with different sensor range. $R = 4m$ is the initial sensor range used in all other experiments in this paper.

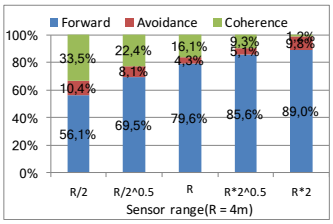


Fig. 6. Swarm distribution during experiments from Figure 5.

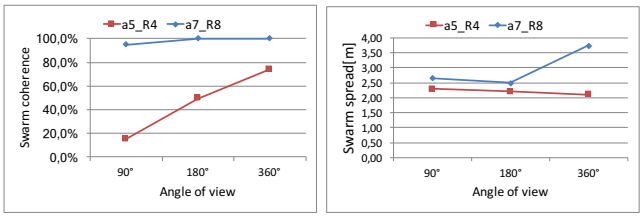


Fig. 7. Swarm coherence and spreading for different configurations of α and sensor range (a5_R4 stands for $\alpha = 5$ and sensor range $R = 4m$).

C. Obstacle avoidance ability

Finally, let us analyse the avoidance behaviour using an environment with a ring of obstacles (Figure 8 - Right). The spacing between the obstacles is smaller than the range

of the obstacle detection sensor Ra, and bigger than the MAV diameter. Using this setting, the MAV can exit the obstacle ring only if the evasive manoeuvre fails, which can be automatically detected. The robustness of the algorithm has been tested by measuring the minimal distance between the MAVs and between the MAV and the nearest obstacle. An example of these experiments is shown in `http://youtu.be/W9QcrnLVI8Y` with data in Figure 9.

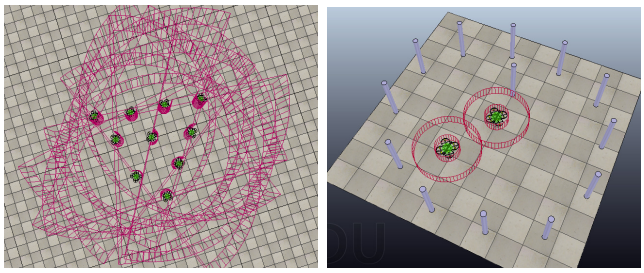


Fig. 8. Left: Initial position of a swarm with sensors with a limited angle of view. Video record of one of the experiment from Figure 7 is available at `http://youtu.be/uHPyTaaaqqE`. Right: An environment with obstacles to test the avoidance ability, for video of the simulation see `http://youtu.be/W9QcrnLVI8Y`.

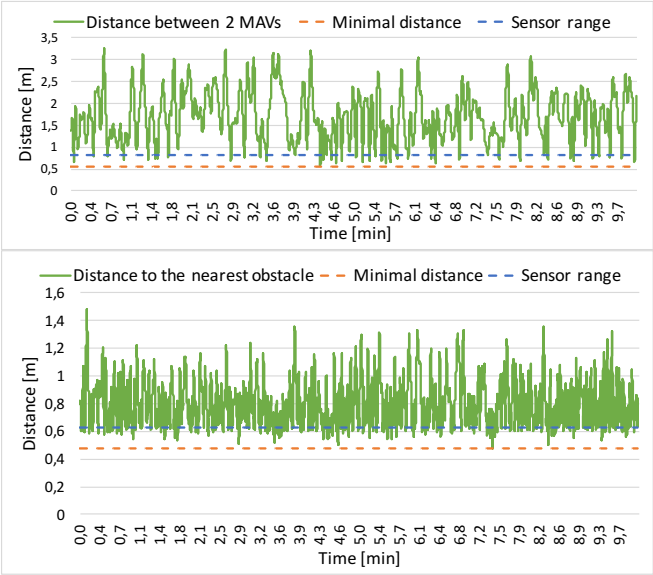


Fig. 9. Distance between the MAVs and the closest distance between an MAV and an obstacle during the simulation in Figure 8 - Right.

IV. Experimental results

The proposed algorithm has been tested in a complex environment depicted in Figure 10 to show its effectiveness and usability. First the swarm is navigated through a window 4 meters wide (the diameter of the stabilized swarm is approximately 10 meters). Then the target automatically moves to redirect the swarm to fly along a long wall. After this movement, the target is again moved to navigate the swarm across an open space with columns and a moving obstacle. The moving obstacle is slower than the MAV during its evasion manoeuvre to be able to avoid it. The distance from the center of the swarm to the actual position of the target is shown in Figure 12 - Left. As may be also seen in Figure 11, where some snapshots and a path passed by the swarm center during one of the simulation runs are displayed (for full record of the swarm movement see `https://youtu.be/gqFxtVEcEdc`), the proposed algorithm was able to repeatedly navigate the swarm through the environment without any collision with obstacle or within the swarm members.

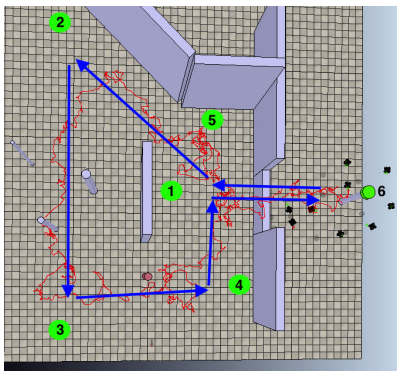


Fig. 10. Trajectory of the center of the swarm passed through the environment with obstacles. The blue arrows represent the intended trajectory, the red curve represents the real trajectory of the swarm during one simulation. The numbers indicate the successive positions of the target. For snapshots see Figure 11.

In the previous experiments, from safety reasons required for the initial HW tests (safety pilots can better secure the experiment if the MAVs stay in the same horizontal plane), all MAVs were flying at the same altitude. Nevertheless, the method works properly also in a 3D space as shown in the example of the simulation runs in Figure 13. The 3D position of the center of the swarm is shown in Figure 12 - Right. The avoidance manoeuvre performed by the MAVs to avoid collisions is more aggressive in the vertical direction because the MAVs must keep a larger security margin along the Z axis due to the air flow interference between MAVs. This causes that the altitude of the swarm oscillates more than the positions along the two horizontal axis.

V. Conclusion

A minimalist fully decentralised coherent swarming algorithm for control of MAV swarms with minimum sensory requirements and without any communication was proposed in this paper. A robust target following mechanism was designed, implemented and verified, enabling the swarm to move in an environment with obstacles into a given location and requiring only minimal computational resources. The overall system was verified in numerous simulations in realistic robotic simulator V-REP.

Acknowledgments

The work has been supported under the grant No. SGS17/187/OHK3/3T/13, and by the Czech Science Foundation (GACR) under research project No. 17-16900Y.

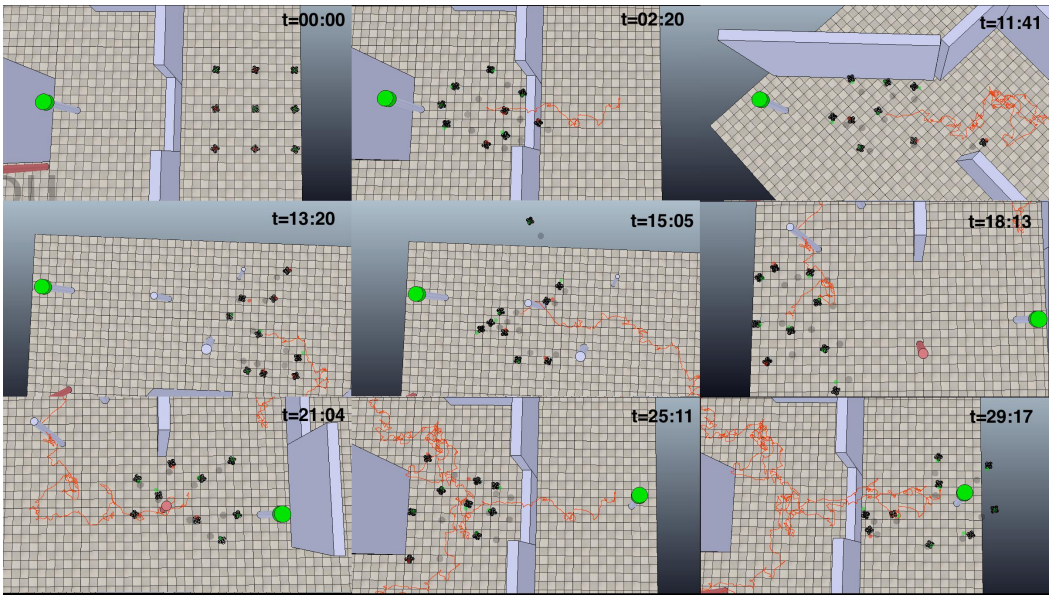


Fig. 11. Simulation of the target following algorithm in a environment with obstacles. The trajectory of the center of the swarm is depicted in red. Complete video available at `https://youtu.be/gqFxtVEcEdc`.

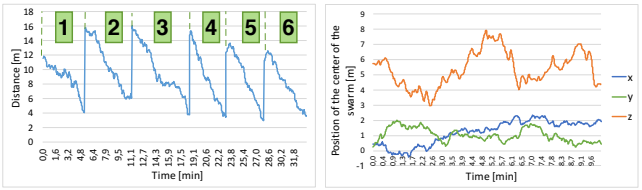


Fig. 12. Left: Distance of the center of the swarm to the particular target positions (labelled according to their index from Figure 10) during one simulation. Right: Position of the swarm in space in the experiment in Figure 13.

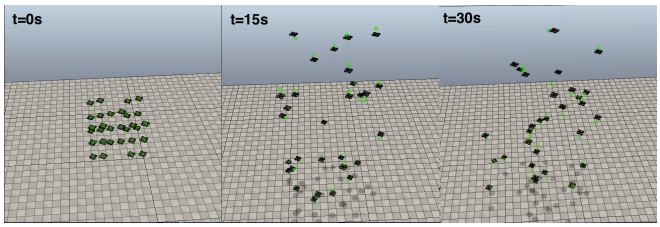


Fig. 13. 3D swarming. For video see `http://youtu.be/o0v5oe6ekVY`.

REFERENCES

[1] J. Faigl, T. Krajnik, J. Chudoba, L. Preucil, and M. Saska, "Low-Cost Embedded System for Relative Localization in Robotic Swarms," in *IEEE ICRA*, 2013.

[2] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[3] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil, "Coordination and Navigation of Heterogeneous MAV-UGV Formations Localized by a hawk-eye-like Approach Under a Model Predictive Control Scheme," *International Journal of Robotics Research*, vol. 33, no. 10, pp. 1393–1412, September 2014.

[4] V. Spurny, T. Baca, and M. Saska, "Complex manoeuvres of heterogeneous mav-ugv formations using a model predictive control," in *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2016.

[5] "From swarm intelligence to swarm robotics," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, 2005, vol. 3342.

[6] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987.

[7] R. Russell, "Heat trails as short-lived navigational markers for mobile robots," in *IEEE ICRA*, 1997.

[8] Y.-S. Ryuh, G.-H. Yang, J. Liu, and H. Hu, "A school of robotic fish for mariculture monitoring in the sea coast," *Journal of Bionic Engineering*, vol. 12, no. 1, pp. 37 – 46, 2015.

[9] Y. Jia and L. Wang, "Experimental implementation of distributed flocking algorithm for multiple robotic fish," *Control Engineering Practice*, vol. 30, no. 1, pp. 1 – 11, 2014.

[10] M. Saska, J. Langr, and L. Preucil, "Plume Tracking by a Self-stabilized Group of Micro Aerial Vehicles," in *MESAS*, 2014.

[11] B. Yang, Y. Ding, and K. Hao, "Area coverage searching for swarm robots using dynamic voronoi-based method," in *CCC*, 2015.

[12] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles," *Journal of Intelligent & Robotic Systems.*, vol. 84, no. 1, pp. 469–492, 2016.

[13] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny, and L. Preucil, "Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups," *Journal of Intelligent and Robotic Systems*, vol. 73, no. 1-4, pp. 603–622, 2014.

[14] M. Saska, T. Baca, and D. Hert, "Formations of unmanned micro aerial vehicles led by migrating virtual leader," in *14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016.

[15] A. Renzaglia and A. Martinelli, "Potential Field based Approach for Coordinate Exploration with a Multi-Robot Team," in *IEEE SSRR*, 2010.

[16] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. Kosmatopoulos, "Cognitive-based Adaptive Control for Cooperative Multi-Robot Coverage," in *IEEE IROS*, 2010.

[17] M. Saska, J. Vakula, and L. Preucil, "Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization," in *IEEE ICRA*, 2014.

[18] M. Saska, "MAV-swarms: unmanned aerial vehicles stabilized along a given path using onboard relative localization," in *ICUAS*, 2015.

[19] J. Nembrini, A. Winfield, and C. Melhuish, "Minimalist coherent swarming of wireless networked autonomous mobile robots," in *IC-SAB*, 2002.

[20] T. Baca, G. Loianno, and M. Saska, "Embedded model predictive control of unmanned micro aerial vehicles," in *MMAR*, 2016.

Risk and Comfort Management for Multi-Vehicle Navigation using a Flexible and Robust Cascade Control Architecture

Charles Philippe[1,2], Lounis Adouane[1], Benoît Thuilot[1], Antonios Tsourdos[2] and Hyo-Sang Shin[2]

***Abstract*— This paper presents a new cascade control architecture formulation for addressing the problem of autonomous vehicle trajectory tracking under risk and comfort constraints. The integration of these constraints has been split between an inner and an outer loop. The former is made of a robust controller dedicated to stabilizing the car dynamics while the latter uses a nonlinear Model Predictive Control (MPC) scheme to control the car trajectory. The proposed structure aims to take into account several important aspects, such as robustness considerations and disturbance rejection (inner loop) as well as control signal and state constraints, tracking error monitoring and tracking error prediction (outer loop). The proposed design has been validated in simulation while comparing mainly with common kinematic trajectory controllers.**

I. INTRODUCTION

Autonomous vehicles are getting more and more important in the academic field as well as in the industry. Constructors such as BMW, Volvo, Tesla and other companies such as Uber are trying to reach the next step of autonomy for consumer cars. Recent incidents or accidents [15] have highlighted the need for safe and robust architectures.

The objectives of this paper are to describe the development of a control architecture for autonomous vehicle under the constraints specific to urban passenger transportation. The two major categories of constraints are the safety and the comfort of the passengers. Indicators of tracking performance and health monitoring will be developed to allow for the future development of a supervision layer in the architecture.

In the end, the proposed architecture aims to be a generic and easily transposable solution for single and multi-vehicle navigation. These aims will be reached with a combination of an MPC controller for the tracking and a robust H_∞ controller for yaw stabilization.

The numerical applications are done for the VIPALAB vehicles, which are autonomous electric vehicles for urban transportation (cf. Fig. 1).

The remainder of this paper will be organized as follows. Section II will give an overview on the works on autonomous vehicle control and will describe the pertinence and novelty of the proposed architecture. Section III will explain the design of the two blocks of the cascade control architecture. Section IV will show comparative simulations in a range situations for different controllers and architectures.

[1] Institut Pascal, Université Clermont Auvergne, Clermont-Ferrand, France name.surname@uca.fr

[2] Cranfield University, Cranfield, United Kingdom name.surname@cranfield.ac.uk



Fig. 1. VIPALAB autonomous transport vehicles vehicles in a coordinated maneuver

II. TOWARDS A FLEXIBLE AND ROBUST CONTROL ARCHITECTURE

A. Related works

The existing works on autonomous vehicle lateral control can be separated in two main categories. There are trajectory/target tracking algorithms on one side and yaw stabilization algorithms on the other side. Some approaches will be described as *integrated* and aim to fulfill those two tasks at the same time. For the tracking task, kinematic controllers are used in [14] and [3]. Depending on the implementation they can cover a range of speeds and situations but the lack of consideration of dynamic effects can be problematic for highly dynamic situations. Yaw stabilization (or active steering) schemes include [6] and [5] which use respectively the linear robust control framework and the MPC framework. Both approaches show very good performances, at the expense of a robustness proof in [5] with the MPC design. As for integrated approaches, a first example based on a nonlinear kinematic controller is presented in [8]. Empirical terms have been added to a trajectory controller for dynamic effects compensation. In [7] is presented another integrated approach based on linear adaptive control. In [2] is presented an approach based on MPC. The choice of the technique mainly depends on the design objectives. Some other works cover the two tasks but with separate controllers for each one, which has the advantage to separate the objectives for each task and give an adapted answer. These approaches usually use two controllers in a cascade architecture. Such works for ground vehicles include [12] and [10]. Linear control is often used for the inner control loop. A similar cascade architecture for aerospace applications is presented in [4]. It is a common approach in Unmanned Aerial Vehicles (UAV)

design. Usually, these architectures fail to take into account the comfort, safety and implementability at the same time since they are more performance based. This is what the proposed design aims to do.

B. Architecture design

Compared to integrated design, the advantage of cascade architectures is the flexibility for multi-vehicle navigation and the natural separation between kinematic and dynamic phenomenons. Moreover, the trajectory tracking error dynamics are not on the same time scale than the car yaw dynamics. This separation is analoguous to the Guidance/Control framework [9] even though it is more a kinematic/dynamic separation.

The MPC has been used with great success to address trajectory tracking and yaw stabilization as an integrated approach. However it seems more relevant in this application to separate the yaw stabilization in a cascade architecture.

The inner loop for yaw stabilization has to deal with the uncertainty on the vehicle physical parameters. For instance the weight and inertia of the vehicle will undergo wide variations because of the variable passenger repartition. The friction coefficient μ will also be unknown and unmeasurable.

For these reasons the linear robust control framework has been chosen to design the inner loop controller. It enables the robustness assessment of the controller under the model uncertainty, which is a big asset for ensuring safety.

The designed architecture is shown in Fig. 2. The MPC controller generates a desired yaw rate r_d and a desired speed v_d to track the reference trajectory X_{ref}. The inner controller tracks the signal (r_d, v_d) by generating a desired steering angle δ_d and acceleration a_d to the vehicle. The state X_{dyn} (resp. X_{kin}) is the dynamic (resp. kinematic) state of the vehicle. It will be defined in section III-A (resp. III-B).

III. PROPOSED CASCADE CONTROL ARCHITECTURE

This architecture will have to fulfill the tasks of trajectory following as well as target tracking in order to be fit for multi-vehicle navigation. Trajectory following is the action of following a predefined line (defined in (x, y) coordinates) with a reference speed v_{ref} at each point. For target following the same information is available but only at the current instant. More states can be known about the target, whether by sensing or communication. As mentioned in section II, the proposed architecture will have to handle the two tasks while ensuring passenger comfort and safety. The inner loop design will be described first and then the outer loop design, since it depends on the former.

A. Inner loop design

A mixed sensitivity H_∞ controller has been chosen because it is an optimal design dechnique, and it has explicit disturbance rejection and frequency domain performance specifications. The disturbance rejection characteristics are interesting because it will dictate the vehicle's behaviour under wind gusts. Too strong of a reaction could be seen as dangerous and uncomfortable.

TABLE I
PARAMETER VALUES FOR UNCERTAIN CAR PLANT STUDY

parameter	notation	value
wheelbase	l_b	3m
inertial radius	i_r	1.5m
cornering stiffnesses	c_f, c_r	700N/deg
actuator time constant	τ_{act}	0.6s
mass	m	$600\text{kg} \pm 30\%$
CG position to front axle	l_f	$1.4\text{m} \pm 20\%$
speed	v	$3\text{m/s} \pm 50\%$
friction coefficient	μ	$0.65 \pm 50\%$

For this application the model in the state space form is shown in (1). It is based on the kinematic bicycle representation. It consists of a 2 degree of freedom (DoF) model for the sideslip β and yaw rate r dynamics and a first order model of time constant τ_{act} for the evolution of δ, the steering angle. The overall state is $X_{dyn} = (\beta, r, \delta)^T$ (resp. sideslip, yaw rate and steering angle defined in Fig. 3). The input is the desired steering angle δ_d. This model has been presented in depth in [1]. It is suitable for low speed situations such as urban traffic.

$$\dot{X}_{dyn} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & -\tau_{act}^{-1} \end{pmatrix} X_{dyn} + \begin{pmatrix} 0 \\ 0 \\ \tau_{act}^{-1} \end{pmatrix} \delta_d \quad (1)$$

The coefficients a_{ij} and b_i depend on the constants $c_f, c_r, m, v, J, l_b, l_f$ defined in Table I.

The following range of configurations has been considered:

- from zero passenger to four passengers of 100kg each
- speeds from 1.5m/s to 4.5m/s
- friction coeff in $[0.3, 1]$ (from a slippery wet road to a dry road)
- Centre of Gravity (CoG) from 1.1m to 1.6m to front axle (because of the passenger repartition)

The corresponding uncertain variables have been summarized in Table I. As a way to reduce the number of uncertain variables, J has been considered proportional to m with the intermediate of the inertial radius i_r (c.f. [1]).

The result of the H_∞ design is shown in (2) with K_{CF} being the feedforward filter and K_{DR} being the feedback filter. The filters have been approximated by second order transfer functions to make the implementation faster.

$$\begin{cases} K_{CF}(s) &= \dfrac{s^2 + 21.2s + 158.2}{s^2 + 20s + 156.3} \\ K_{DR}(s) &= \dfrac{250(s+124)(s+1.67)}{s(s+17.5)} \end{cases} \quad (2)$$

The robustness analysis shows that the design is robust to the modeled uncertainty. The performance analysis shows that the rise time for the yaw rate is always between $0.3s$ and $0.8s$ with a nominal value at $0.5s$ and the system is always well damped (as seen in Fig. 4).

Since the system is always well damped (there is no overshoot), it will be approximated by a first order system

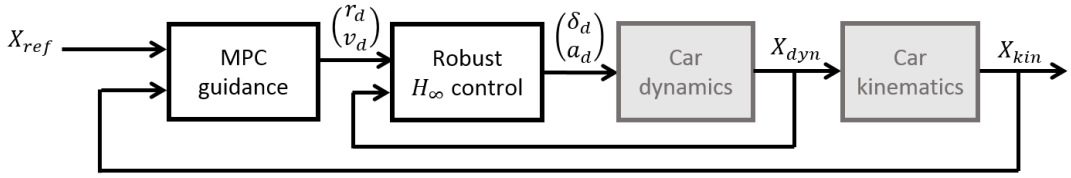


Fig. 2. Designed cascade architecture

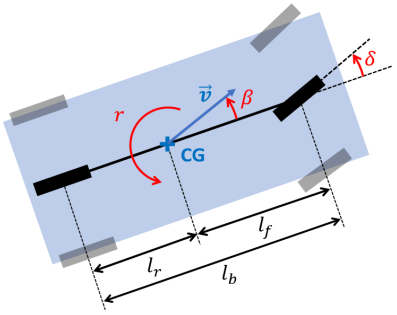


Fig. 3. Dynamic bicycle model conventions

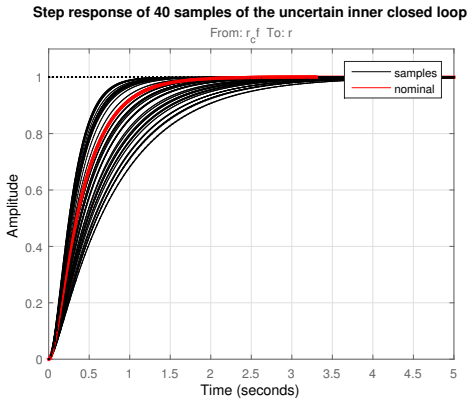


Fig. 4. Closed-loop step response envelope

in the following developments, making for a simpler model for the MPC.

B. Outer loop design

The outer loop controller (cf. Fig 2) needs to address the problem of stabilizing the vehicle around a moving target or a reference trajectory.

Even though it is a classical approach, MPC has good qualities to address our problem: it finds an optimal solution, can inherently deal with non linear processes such as car steering and can include constraints on the states and control signal (linked to comfort and safety). The prediction is also valuable to deal with slow dynamics and to anticipate future situations.

The main problem of MPC is that it always needs a reference trajectory over a finite horizon. In multi-vehicle navigation it is not always available and thus needs to be predicted to enable the use of MPC. Fig. 5 shows a situation where a reference trajectory for the virtual target T that V_F (the *follower*) has to follow is only partially known over the MPC horizon. In this situation the aim is to keep constant offsets Δy_{ref} and Δs_{ref} with the trajectory of V_L (the *leader*). As the prediction horizon of the MPC depends on the model dynamics, it is not correlated to how far the target is to the leading vehicle. Thus a big MPC horizon and a small leader/target distance can lead to such a situation.

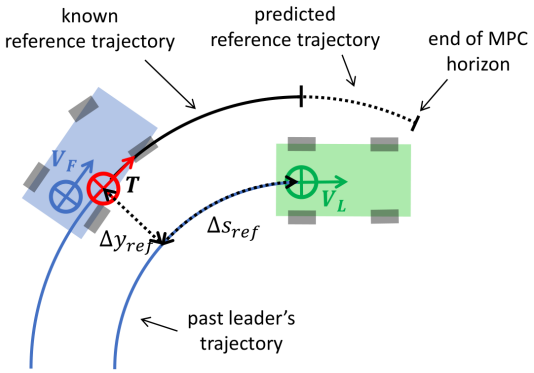


Fig. 5. Partial availability of a reference trajectory in leader/follower navigation

The reference trajectory prediction is based on the hypotheses that the yaw rate r and speed v of the leader are constant. The states for the prediction are the same as the MPC presented later in (3). It generates a reference trajectory $\hat{\mathbf{X}}_{\text{ref}}(k+N|k)$ over the MPC prediction horizon N. The MPC scheme can then be used seamlessly whether there is a reference trajectory over the whole prediction horizon or not and makes the architecture more flexible.

The chosen model for the MPC is shown in (3). At a timestep k, the state is $X_{kin}(k) = (x_k, y_k, \psi_k, r_k, v_k)^T$ where x_k, y_k and ψ_k are defined in Fig. 6, r_k is the yaw rate of the car and v_k its speed) and the input is $U = (r_d, v_d)^T$ (the desired yaw rate and speed). The only parameters that describe the vehicle's dynamics are τ_r and τ_v, the time constants for the yaw rate and the speed responses which have been identified on the close inner loop (cf. Table II). The last parameter is the sampling time T_s of the loop. These responses are assumed to be described by first order models. It is a realistic hypothesis as long as the real responses are well damped. This is the case here as seen in Fig. 4 thanks to the inner loop designed in section III-A.

$$\begin{cases} x_{k+1} &= x_k + T_s v_k \cos\psi_k \\ y_{k+1} &= y_k + T_s v_k \sin\psi_k \\ \psi_{k+1} &= \psi_k + T_s r_k \\ r_{k+1} &= r_k + \tau_r^{-1}(r_d - r_k) \\ v_{k+1} &= v_k + \tau_v^{-1}(v_d - v_k) \end{cases} \tag{3}$$

The MPC scheme finds a control input $\mathbf{U}_{opt}$ that minimizes a cost function $J(\mathbf{U})$, where $\mathbf{U} = (U_k, ..., U_{k+N})$ is series of control signals to test over the prediction horizon N. This function is usually composed of a term that penalizes the errors to the reference trajectory and other terms to smooth the control signal.

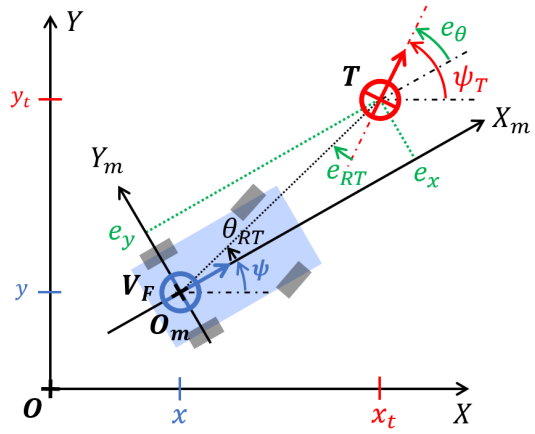


Fig. 6. Navigation errors definition

The penalized terms are:

- $\tilde{\mathbf{X}}$, the matrix of differences w.r.t. the reference trajectory over the prediction horizon N
- $\hat{\mathbf{E}}(k+N|k)$ the navigation errors over the prediction horizon N. These errors are defined in Fig. 6 and further detailed in [14]. They are a convenient way to work in a local frame that is in the vehicle's orientation.
- $\tilde{\mathbf{U}}$ the difference between the test input signal and the previous optimal input signal
- $\mathbf{U}_\Delta$ the differences between two successive input values in the tested input signal (a.k.a. control effort)

The chosen cost function is defined in (4). It is a weighted sum of the penalty terms:

$$J(\mathbf{U}) = \tilde{\mathbf{X}}^T Q \tilde{\mathbf{X}} + \hat{\mathbf{E}}(k+N|k)^T S \hat{\mathbf{E}}(k+N|k) + \tilde{\mathbf{U}}^T R \tilde{\mathbf{U}} + \mathbf{U}_\Delta^T R_\Delta \mathbf{U}_\Delta \quad (4)$$

The penalties on $\tilde{\mathbf{U}}$ and $\mathbf{U}_\Delta$ tend to smooth the tracking and are often encountered in nonlinear MPC schemes. The penalty on the navigation errors allows to separately penalize the lateral and longitudinal errors (e_x and e_y) and have been preferred to the penalty on the state difference $\tilde{\mathbf{X}}$. The values of the weight matrices Q, S, R and R_Δ are compiled in Table II. The raw state difference $\tilde{\mathbf{X}}$ has not been penalized except for the speed difference, which was found to smooth the longitudinal tracking.

For comfort and safety, the following constraints have been introduced:

- $|r_d| \leq r_{max}$
- $|\dot{r}_d| \leq dr_{max}$
- $v_{min} \leq v_d \leq v_{max}$
- $|a_{y,d}| = |v_d r_d| \leq a_{y,max}$
- $|a_{x,d}| = \Delta v_d / T_s \leq a_{x,max}$
- $|\kappa_d| = |r_d / v_d| \leq \kappa_{max}$

TABLE II
MPC CONTROLLER PARAMETERS

N	14
τ_r	0.5s
τ_v	1.4s
Q	diag(0, 0, 0, 0, 0.1)
S	diag(1, 2, 0, 0)
R	diag(0, 0)
R_Δ	diag(15, 15)
r_{max}	30 deg/s
dr_{max}	50 deg/s^2
v_{max}	4.5 m/s
$a_{y,max}$	5 m/s^2
$a_{x,max}$	3 m/s^2
$e_{x,max}$	0.5 m
$e_{y,max}$	0.2 m

The constraints on the yaw rate r_d, the yaw rate derivative $\dot{r}_d$ and the lateral/longitudinal accelerations $a_{y,d}$ and $a_{x,d}$ are here for comfort. The constraints on the speed v_d and the curvature κ_d are physical limitations of the vehicle. Two additional constraints on the tracking errors have been added: $|e_x| \leq e_{x,max}$ and $|e_y| \leq e_{y,max}$. These constraints are used to ensure the vehicle will stay within given bounds around the target. For example, the constraint on e_y will depend on the road width. The numerical values used for the constraints are compiled in Table II. The optimization function used for the simulations is the *fmincon* Matlab optimization function under constraints. The input constraints will always be respected (it restricts the search space) and the function will try to find a solution that respects the additional constraints on the tracking errors, unless impossible. In the latter case, the optimization function's output will mention that the constraints were not respected. This information can be used to prevent the violation of constraints before it actually happens. Finally, the MPC scheme runs approximately in 20 seconds for a 10 seconds Simulink simulation with a modern computer (a 2013 Intel i5 equipped laptop) and a loop rate of 10Hz.

C. Conclusion

The proposed architecture is comprised of two loops in cascade that aim to solve the trajectory tracking/target following problems under comfort and safety constraints. The task separation allows to split the constraints and to focus on different aspects of the problem at each stage as well as allowing a simpler design overall.

IV. SIMULATIONS AND RESULTS

A. Model Used

The model used for the simulations consists of a 3DoF bicycle chassis model with a linear tyre model (for both the longitudinal and lateral forces). The simulations have been performed in Simulink.

The model for the simulations is based on the one presented in section III-A. The rotational wheel dynamics have been added in order to have a realistic longitudinal behaviour. Thus the state vector is defined as $X = (\beta, r, v, \omega_f, \omega_r)^T$ with the three first states already defined in (1)and ω_f (resp.

ω_r) is the front (resp. rear) wheel rotational speed. The input vector is $U = (\delta_d, a_d)^T$, the desired steering angle and longitudinal acceleration. These inputs are transformed into actual wheel angle and acceleration by the $\{\text{actuator} + \text{controller}\}$ systems modeled by first order transfer functions of time constant 0.6s (resp 1s) and damping 0.8. The front and rear axles inertias are $J_f = J_r = 0.45\text{kg.m}^{-2}$ and the longitudinal tyre stiffnesses are $C_{l,f} = C_{l,r} = 1\text{e}^4\text{N}$. The other model parameters will be within the range of values used for controller design (cf. Table I). Unless specified otherwise, the nominal values have been taken.

B. Simulation scenarios

Comparisons will be made with two kinematic controllers. The first controller is the one presented in [14] and [13]. For simplicity it will be referenced as the "Vilca" controller. It is a nonlinear control law designed for both dynamic target following and waypoint navigation. It is based on a Lyapunov function design and is a recent example of a flexible kinematic controller. The other one is the "Pure Pursuit" controller [11], a widely used kinematic controller for trajectory tracking because of its simplicity and efficiency. It is a non linear controller that computes a curvature to reach a point on the trajectory at a look-ahead distance. This distance is usually proportional to the vehcile speed with a coefficient k_{PP} within a lower and upper bound (cf. Table III). Both algorithms compute the steering angle corresponding to the desired curvature under kinematic hypotheses, thus not taking into account actuator delays and slip.

For the simulations, the virtual target follows a sinusoidal path at a constant speed (cf. Fig 7). For the MPC controller it is assumed that no information of the target's future path is available to put it in difficult conditions. As a consequence, the prediction module described in section III-B will be used. It is on the other hand assumed that the trajectory is entirely available when using the pursuit controller, thus giving it more favorable conditions.

Two test case are presented:

- *Behaviour comparison*: A test at low speed to compare the behavior of the three controllers. For this test the nominal values for the car model will mostly be used.
- *Safety and comfort assessment*: A test at a higher speed and non-nominal car model values to check the robustness of the approaches. This test will also serve to check if the comfort constraints with our approach are respected in more agressive maneuvers.

The common (resp. variable) parameters for the simulation scenarios are compiled in Table III (resp. Table IV-B).

C. Behaviour comparison

All the nominal parameters for the model have been taken except the speed which is $2m/s$. The MPC shows a very good tracking compared to the two other methods (cf Fig. 8). However the control signals (Fig. 9) and the comfort indicators (Fig. 10) are approximately of the same magnitude for the three approaches. In easy maneuvers like this one the proposed architecture behaves well but has an edge on neither the pursuit controller or on Vilca's controller in terms of comfort.

TABLE III

COMMON SIMULATIONS PARAMETERS

parameter	**value**
initial vehicle position	$(1, 1)$ (m)
initial vehicle heading	$30°$
target path curvature variation rate	0.1 Hz
max target path curvature	$1/15\text{m}^{-1}$
Vilca's law coefficients K_V	$(1, 2.2, 8, 0.1, 0.01, 0.6)$
Pursuit law look-ahead coeff. k_{PP}	0.5
Pursuit law look-ahead dist. bounds	$[1, 5]$ (m)
outer loop sampling time $T_{s,g}$	1/10s
inner loop sampling time $T_{s,c}$	1/50s

TABLE IV

VARIABLE SIMULATION PARAMETERS

parameter	**value**	
	First simulation	*Second simulation*
initial target position	$(1.5, 1.5)$ (m)	$(2, 2)$ (m)
initial target heading	$30°$	$40°$
Target speed	2m/s	4m/s
vehicle mass	600kg	750kg
friction coefficient μ	0.65	0.4

D. Safety and comfort assessment

In this simulation, the safety has been assessed by checking the tracking performance under a change of mass, friction coefficient and speed. The respect of the comfort constraints with the proposed cascade architecture has been tested and compared with the behaviour of the other controllers. In this series, the target initial position was further from the vehicle initial position (cf. Table IV-B) to study the behaviour of the controllers when a more agressive maneuver is required to follow the target.

The tracking performance of the designed cascade architecture is now far better than both the Pursuit controller and Vilca's controller (cf. Fig. 11). The latter one shows an unstable oscillatory behaviour at these higher speeds because it neither anticipates the trajectory nor the actuator delay. The performance of the cascade architecture also shows the

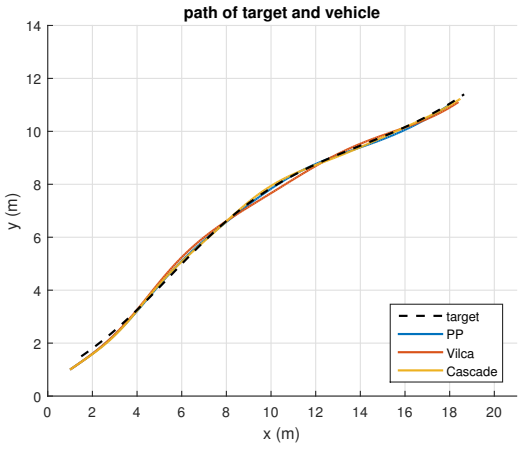


Fig. 7. Example of path for the simulations (target and follower)

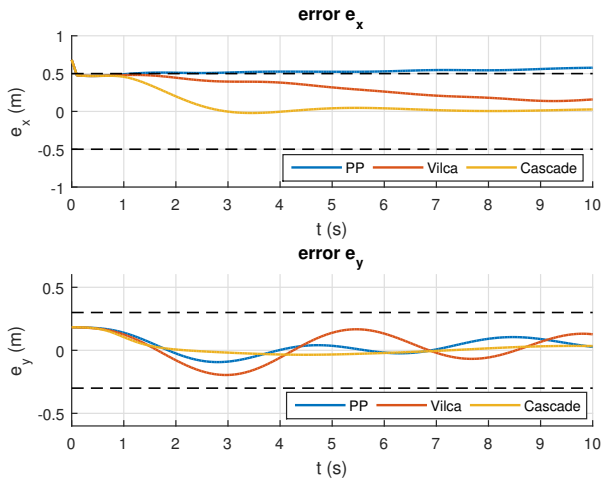


Fig. 8. Tracking performance (1st series)

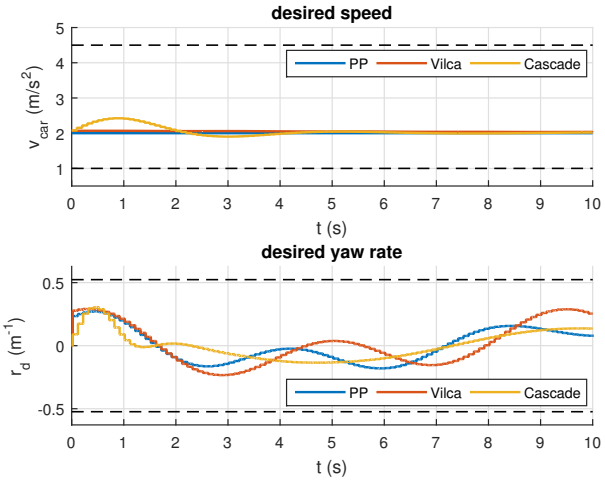


Fig. 9. Control signals (1st series)

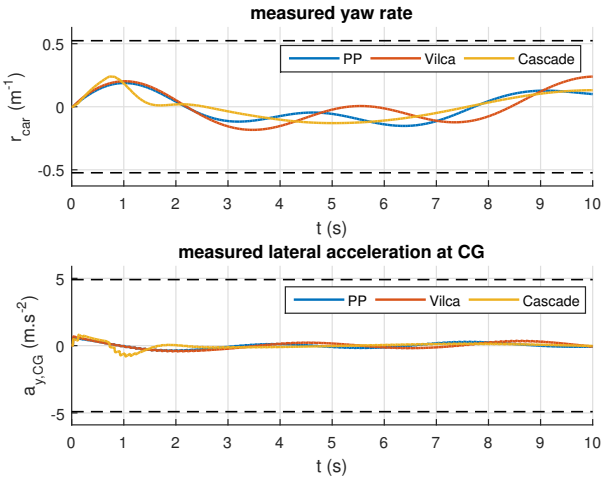


Fig. 10. Comfort indicators (1st series)

effectiveness of the inner controller to stabilize the yaw dynamics with a non nominal model. The control signals (Fig. 12) show an effective capping of both the desired speed and yaw rate with the cascade architecture, leading to a faster tracking errors convergence while respecting the introduced constraints. The comfort indicators (cf. Fig. 13) show undamped oscillations for both the Pursuit and Vilca's controller, and a slight overshoot for the cascade design. It could be removed with finer tuning of the MPC controller or the use of the real target's trajectory information.

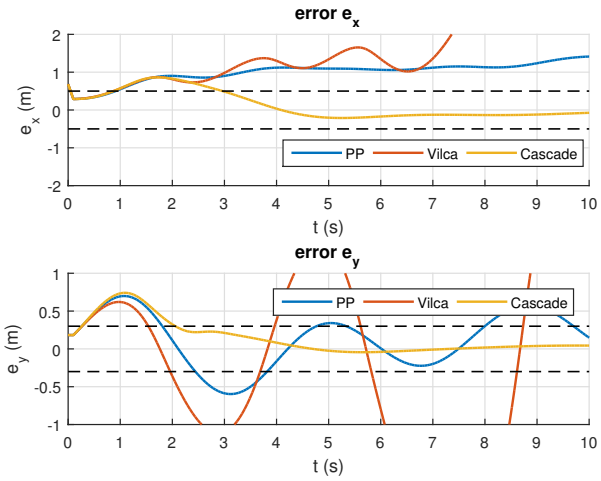


Fig. 11. Tracking performance (2nd series)

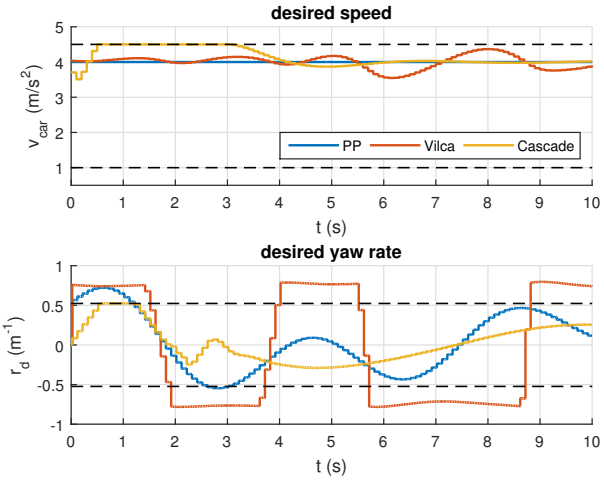


Fig. 12. Control signals (1st series)

V. CONCLUSIONS

In this paper has been proposed a cascade architecture for autonomous vehicle navigation. This architecture seamlessly fills the tasks of trajectory/path tracking as well as dynamic target following and thus can cope with multi-vehicle scenarios. The architecture is divided into a robust low-level yaw stabilization controller that focuses on the vehicle's dynamics and a high-level tracking MPC controller that

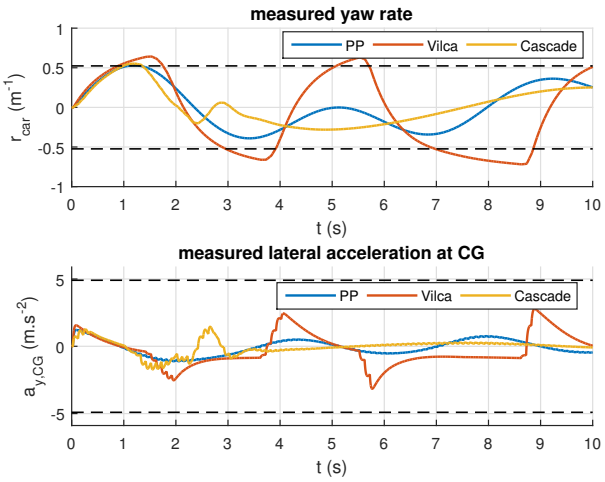


Fig. 13. Comfort indicators (2nd series)

focuses mainly on the kinematics. This architecture shows an improvement in tracking performances, safety and flexibility compared to usual kinematic controllers for trajectory tracking. It is not intended to have an edge on performances compared to integrated approaches for trajectory tracking but to improve robustness and implementability. Further work will be carried out to check the performances of a linear MPC controller as well as a robustification of the MPC scheme. Real time implementability will be evaluated and Linear Parameter Varying (LPV) techniques for the low-level controller (parametrized by speed) will be investigated to improve its performance and operational envelope.

ACKNOWLEDGMENT

This work has been sponsored by the French government research program Investissements d'avenir through the RobotEx Equipment of Excellence (ANR-10-EQPX-44) and the IMoBS^3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2014-2020 (FEDER - AURA region) and by the AURA region.

REFERENCES

[1] Jürgen Ackermann. *Robust Control for Car Steering*. Tech. rep. German Aerospace Center, Institute of Robotics and System Dynamics, 1999.

[2] Pedro Aguiar, Andrea Alessandretti, and Colin N Jones. "Trajectory-tracking and path-following controllers for constrained underactuated vehicles using Model Predictive Control". In: *European Control Conference* (2013), pp. 1371–1376.

[3] A. Broggi et al. *Autonomous vehicles control in the VisLab Intercontinental Autonomous Challenge*. 2012.

[4] Dongkyoung Chwa and Jin Young Choi. "Adaptive Nonlinear Guidance Law Considering Control Loop Dynamics". In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (2003), pp. 1134–1143.

[5] Paolo Falcone and Francesco Borrelli. "Low complexity MPC schemes for integrated vehicle dynamics control problems". In: *International Symposium on Advanced Vehicle Control* 1 (2008), pp. 875–880.

[6] Simon Hecker. "Robust Hinf-based vehicle steering control design". In: *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*. IEEE, Oct. 2006, pp. 1294–1299.

[7] Pushkar Hingwe et al. "Linear parameter varying controller for automated lane guidance: Experimental study on tractor-trailers". In: *IEEE Transactions on Control Systems Technology* 10.6 (Nov. 2002), pp. 793–806.

[8] Gabriel M. Hoffmann et al. "Autonomous Automobile Trajectory Tracking for Off-Road Driving : Controller Design , Experimental Validation and Racing". In: *Proceedings of the American Control Conference* (2007), pp. 2296–2301.

[9] Farid Kendoul. "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems". In: *Journal of Field Robotics* 29.2 (Mar. 2012), pp. 315–378. arXiv: 10.1.1.91.5767.

[10] Alexey S. Matveev, Hamid Teimoori, and Andrey V. Savkin. "A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance". In: *Automatica* 47.3 (2011), pp. 515–524.

[11] Jesús Morales et al. "Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2D laser scanner". In: *Eurasip Journal on Advances in Signal Processing* 2009.1 (2009), p. 935237.

[12] Joshué Pérez, Vicente Milanés, and Enrique Onieva. "Cascade architecture for lateral control in autonomous vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 12.1 (Mar. 2011), pp. 73–82.

[13] José Miguel Vilca, Lounis Adouane, and Youcef Mezouar. "A novel safe and flexible control strategy based on target reaching for the navigation of urban vehicles". In: *Robotics and Autonomous Systems* 70 (Aug. 2015), pp. 215–226.

[14] José Miguel Vilca, Lounis Adouane, and Youcef Mezouar. "An Overall Control Strategy Based on Target Reaching for the Navigation of an Urban Electric Vehicle". In: *International Conference on Intelligent Robots and Systems (IROS)* (2013), pp. 728–734.

[15] Danny Yadron and Dan Tynan. "Tesla driver dies in first fatal crash while using autopilot mode". In: *The Guardian* (2016).

Reactive Dubins Traveling Salesman Problem for Replanning of Information Gathering by UAVs

Robert Pěnička[1], Martin Saska[1], Christophe Reymann[2] and Simon Lacroix[2]

***Abstract*— We introduce a novel online replanning method for robotic information gathering by Unmanned Aerial Vehicles (UAVs) called Reactive Dubins Traveling Salesman Problem (RDTSP). The considered task is the following: a set of target locations are to be visited by the robot. From an initial information gathering plan, obtained as an offline solution of either the Dubins Traveling Salesman Problem (DTSP) or the Coverage Path Planning (CPP), the proposed RDTSP ensures robust information gathering in each given target location by replanning over possible missed target locations. Furthermore, a simple decision making is a part of the proposed RDTSP to determine which target locations are marked as missed and also to control the appropriate time instant at which the repair plan is inserted into the initial path. The proposed method for replanning is based on the Variable Neighborhood Search metaheuristic which ensures visiting of all possibly missed target locations by minimizing the length of the repair plan and by utilizing the preplanned offline solution of the particular information gathering task. The novel method is evaluated in a realistic outdoor robotic information gathering experiment with UAV for both the Dubins Traveling Salesman Problem and the Coverage Path Planning scenarios.**

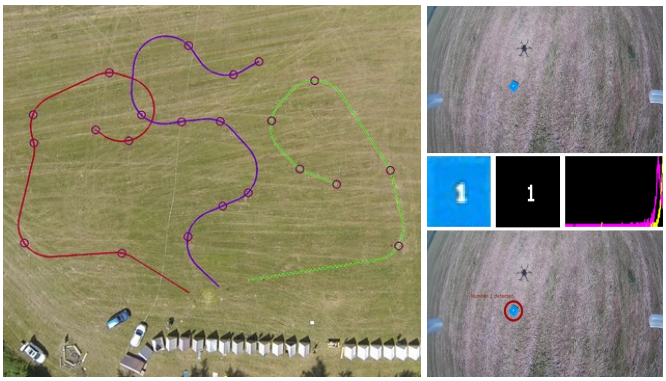


Fig. 1: Example of information gathering as a solution of Dubins Traveling Salesman Problem for multiple UAVs (MDTSP) on left, and the data collection as a camera detection of colored objects with number recognition (on right)

I. INTRODUCTION

This paper presents a new robust approach for solving the robotic information gathering by Unmanned Aerial Vehicles. In this task, a set of locations is to be visited by the robot to collect sensory data. The proposed method called Reactive Dubins Traveling Salesman Problem (RDTSP) is an online approach that uses the UAV's onboard computational resources for replanning and decision making to ensure that all given target locations are visited, regardless of possible disturbances during the information gathering task, that can lead to missing measurements from some of the target locations.

The first and typical application of the proposed method arises in the Coverage Path Planning (CPP) [1] scenarios where a predefined region is given, and the task is to find appropriate waypoints (sensor measurement locations) such that the whole area is scanned by the onboard sensor. For the aerial robotics, a simple zigzag path (further also called 'sweeping' path) is usually sufficient as there is no need for avoiding obstacles during the coverage mission.

The second considered scenario is the information gathering specified as the Dubins Traveling Salesman Problem (DTSP) [2], where its multi-robot case is shown in Figure 1. The DTSP is a variant of the well known Traveling Salesman Problem (TSP) [3] for Dubins vehicle, which we use as a simple model of considered UAV for constructing smooth constant speed paths. Contrary the CPP, the DTSP uses a predefined set of target locations, and the task is to find a minimal length path over all target locations. Notice that after finding the appropriate waypoints in the area, the CPP is a special case of the TSP or in our case a special case of the DTSP.

Using UAVs for obtaining sensory data in such predefined locations in the environment is always influenced by uncertainties in robot motion, due to wind disturbances and imprecision of the dynamic model, among other factors. Another source of uncertainty is introduced into the system by imperfect sensors used for the particular application. All these aspects together may cause that some places of interest are not perceived, or the obtained information is not sufficient. Often it is possible to detect online (during the mission) that a required location was not visited properly by identification of a deviation from the pre-planned trajectory, or that the obtained sensory measuring is not valid. The current approach is to collect all the locations with identified missing sensory information after the mission and to plan an additional flight over these places to complement the data. The proposed RDTSP, however, identifies the missing target locations online and applies the repair plans during the mission.

[1]Authors are with Faculty of Electrical Engineering, Czech Technical University, Technicka 2, Prague, Czech Republic {robert.penicka|martin.saska}@fel.cvut.cz

[2]Authors are with LAAS-CNRS, INSA, Université de Toulouse, 7 Avenue du Colonel Roche, Toulouse, France {christophe.reymann|simon.lacroix}@laas.fr

The typical solution of the information gathering mission either for CPP or DTSP is done offline prior to the mission, and during the execution, a simple trajectory following is used. However, defining the path for sweeping scenarios is rather simple by using the zigzag path, the solution of NP-hard DTSP is computationally demanding. The optimal solution of the DTSP (for a given sampling of heading angles of Dubins vehicle) can be obtained using a transformation of the DTSP into an Asymmetric TSP (ATSP) [4] and then solved by e.g. Concorde TSP solver [5]. Therefore the further introduced RDTSP method uses the offline precalculated path for the information gathering and replans over the missed target locations such that the new repair part of the plan (generally a constrained solution of DTSP over missed locations) is inserted into the original plan.

The proposed Reactive Dubins Traveling Salesman Problem is an online algorithm that uses the predefined path from either CPP or DTSP and ensures the visit of all target locations by replanning over the missed target locations. Since the optimal solution of the DTSP is computational demanding, the proposed RDTSP method utilizes the precomputed plan and inserts the repair plan over the missed target locations into the existing plan between adjacent target locations. The existing approaches for the information gathering and surveillance by UAVs [6], [7], [8], [9] focus on the planning of the whole mission offline (as a solution of the DTSP) and do not consider possible replanning over missed target locations. The proposed VNS-base DTSP planner keeps planning during the mission and minimizes the overall path length not only by considering a different sequence of visiting the missed targets, but also by finding the location of applying the repair plan. Furthermore, a decision making approach is used to determine when the replanning is applied to revisit the missed locations and also to determine which target locations are considered as missed.

The remainder of this paper is organized as follows. The problem statement of the proposed RDTSP is specified in the next section. The overall system architecture is introduced in section III. Section IV describes the proposed method for RDTSP. First experimental results are presented in V and section VI concludes the paper.

II. Problem Statement

The proposed Reactive Dubins Traveling Salesman Problem is based on the idea of utilizing the offline computed initial plan for information gathering in the online iterative improvement of the repairing plan that covers the missed target locations. The RDTSP method uses an initial path P_{init} which is a feasible Dubins path for information gathering over the required target locations. We expect the target locations $S = \{s_1, \cdots, s_n\}$ to be ordered according to the occurrence inside P_{init}.

For the purpose of RDTSP for the targeted Unmanned Aerial Vehicle, we use the kinematic model of Dubins vehicle [10]. The state of the vehicle $q = (p, \theta)^T = (x, y, \theta)^T$ is described by the position $p = (x, y) \in \mathbb{R}^2$ and the heading angle $\theta \in \mathbb{S}^1$. We expect a constant velocity v of the information gathering vehicle that is controlled by input u to either go straight or turn.

$$\dot{q} = \begin{bmatrix} \dot{p} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, u \in [-1, 1] \ . \quad (1)$$

The Dubins vehicle (1) is specific for its minimal turning radius ρ which can be, for the considered UAV, derived from the equation of circular motion $\rho = v_d^2/a_{max}$, where v_d is desired constant speed and a_{max} is the maximal acceleration allowed by the UAV. The distance $\mathcal{L}_d(q_{\sigma_i}, q_{\sigma_j})$ between two states q_{σ_i} and q_{σ_j} of the Dubins vehicle is then the shortest Dubins maneuver out of the six possible maneuvers [10].

For the RDTSP we expect an online deployment where the replanning of the whole initial path after missing some locations is not possible due to the fact that the DTSP is NP-hard, thus very computationally demanding for a large number of target locations. Instead, we propose the VNS-based method that uses the initial DTSP plan, and try to find a Dubins path over the missed locations such that the path can be inserted into the original plan between two adjacent locations.

After starting the execution of the initial plan, the current position inside the plan is described by an index number $c \in (1, n)$ that belongs to the location s_c which is the last location that has been attempted to visit. The set S_m contains k missed target locations. A solution of the RDTSP can be described as a permutation $\Sigma = (\sigma_1, \cdots, \sigma_{k+2})$ to visit the missed locations in S_m, where $\sigma_1 > c$ is location on the initial plan somewhere in the future, $\sigma_{k+2} = \sigma_1 + 1$ is the location where the repair plan connects back to the initial plan, and $s_{\sigma_i} \in S_m$ for $i \in (2, k+1)$ being the missed target locations.

Furthermore the considered Dubins vehicles requires determining respective heading angles at the target locations from Σ. The heading angles can be described as a vector $\Theta = (\theta_{\sigma_1}, \cdots, \theta_{\sigma_{k+2}})$, where θ_{σ_1} and $\theta_{\sigma_{k+2}}$ are set to the values from the initial plan P_{init} and the heading angles at the missed target locations need to be found together with the permutation of the missed locations Σ.

In other words we want to find the starting location inside the future part of the plan, from which the repair plan would be started, together with the sequence of visits to the missed locations such that the produced path has minimal length. The addressed RDTSP can be described as an optimization problem to minimize the length L_{path} of the Dubins tour over the missed target locations:

$$\begin{aligned} \underset{\Sigma,\Theta}{\textit{minimize}}\ \mathcal{L}_{path} &= \sum_{i=2}^{k+2} \mathcal{L}_d(q_{\sigma_{i-1}}, q_{\sigma_i}) \\ \textit{subject to}\ \sigma_1 &= s \in (c+1, n) \ , \\ \sigma_{k+2} &= \sigma_1 + 1 \ . \end{aligned} \quad (2)$$

III. System Architecture

The software architecture of the proposed system is built on top of the system developed at Czech Technical University and University of Pennsylvania for our participation at the MBZIRC competition in Abu Dhabi (see [11] for a description of the initial version of the CTU system primarily designed for formation flying [12], [13], [14] and swarm applications [15], [16], [17]; experiments with the current version of the MBZIRC system can be found at http://mrs.felk.cvut.cz/projects/mbzirc). The core of the system, based on Robot Operating System (ROS), is a Model Predictive Control (MPC) algorithm capable of following any given trajectory feasible for UAVs (the MPC controller was built from our solution designed for flying in GPS-denied environment, which is capable of UAV stabilization and trajectory tracking using only embedded micro-controller [18], see http://mrs.felk.cvut.cz/projects/cesnet for deployment of the system in task of scanning of large historical buildings). The system allows giving a trajectory specified by a sequence of waypoints sampled at a predefined frequency, which is post-processed and smoothed using known UAV motion constraints. For precise trajectory following, a state estimation mechanism is integrated fusing data from a variety of sensors. GPS data, IMU output, an output from image processing from two cameras, a range finder for altitude measurement, and in some applications also Differential GPS can be used in the feedback of the controller.

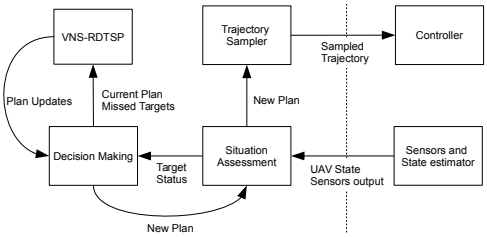


Fig. 2: Software architecture. Arrows represent messages, blocks ROS components. The dashed line symbolizes communication with lower level components.

Figure 2 illustrates the components of the system relevant to the scope of this paper. The mission consists of visiting a predefined list of targets (positions where the sensory data have to be captured), which are collected into a sweeping trajectory or for which a predefined DTSP solution can be computed in advance and fed as an input to the system. The *situation assessment* node processes the sensory outputs as well as the current estimated UAV state to track the status of the current target in the plan. When flying over a target, this node will assess if the measurements are good enough and upon failure of the sensor reading or error in position exceeding an allowed threshold, declares the target as *missed*.

This status is passed to the *decision making* node, which gathers the list of missed targets. Positions of the missed targets are used by the *VNS-DTSP* node (we use VNS - Variable Neighborhood Search based DTSP) solving the defined RDTSP problem to repair the plan by optimizing a *DTSP* tour over these targets. Each time the *VNS-DTSP* node finds a better solution, in the form of a local plan update including at least one previously missed node, a decision is made whether to use it or wait for a potentially better one. When the updated plan is deemed good enough for its execution, it is fed back to the *situation assessment* node which waits for the UAV to be in a state that allows seamless switching to the new plan, which is then sampled, post-processed and sent to the controller.

IV. Proposed Approach for the RDTSP

Because the Dubins TSP problem is NP-hard, the exact optimization scheme requires, in general, a long computation time and is not suitable to find good solutions in online settings. In fact, no algorithm is capable of finding optimal solutions to the DTSP (without sampling the heading angles) for very large number of points at this moment, which is the case of our predefined overall mission plan. Moreover, in most of the information gathering applications, a simple sweeping or DTSP trajectory over the required places of interest is preferred, and there is no need to change the overall plan completely. Therefore we seek only to locally repair the given plan by re-visiting the missed sensory locations during the mission.

As new missed targets may come in at any moment, and due to the anytime nature of the VNS algorithm, we need to be able to decide when to stop the optimization and switch to the current best plan. We control the *VNS-DTSP* scheme from the *decision making* node by providing:

- the current plan, serving as a blueprint for plan updates
- a list of missed targets S_m, to be inserted in the current plan
- current position c inside the current plan

The decision to stop optimizing the current set of missed targets and to execute the newly adapted repair plan is then taken based on three criteria: the proximity to the node at which the current plan update starts (the plan is updated just before reaching the repair plan start), the time elapsed since the current best solution was found, and based on number of missed locations together with chance of missing currently approaching location.

A. Reactive Dubins Traveling Salesman Problem

The proposed solution of the Reactive Dubins Traveling Salesman Problem (RDTSP) is based on the Variable Neighborhood Search (VNS) metaheuristic for combinatorial optimization [19]. The used VNS method iteratively switches between *shake* and *local search* procedures, where the *shake* randomly changes the actual best solution of the problem and *local search* then tries to find on such randomly created solution a new and better solution than the currently best one. VNS employs a predefined neighborhood structures $N_{1,\dots,l_{max}}$ further described as an operation on the problem solution which is the planned repair path. A Randomized variant of the VNS (RVNS) is used, which tries iteratively a number of same random operations during the *local search* procedure.

For finding the appropriate heading angles at the missed target locations, we propose a sampling based approach where the heading angles $\theta_i \in \langle 0, 2\pi)$ are equidistantly sampled into m values. The heading angles are then determined by a graph search over all heading samples for location sequence Σ and the ones with the minimal path length are used.

The proposed VNS-based solution of RDTSP uses following neighborhood structures. The randomized *shake* procedure uses:

- **Path Move** (l=1) randomly selects part of the repair plan between s_{σ_2} and $s_{\sigma_{k+1}}$, and moves it to a different position inside the plan.
- **Path Exchange** (l=2) works similar to the path move, but selects two random non overlapping parts of the plan between s_{σ_2}, $s_{\sigma_{k+1}}$ and exchange their position.
- **Start Exchange** (l=3) is a special neighborhood structure that randomly changes the location s_{σ_1} where the repairing path starts which also changes the end of repairing plan to $s_{\sigma_{k+2}} = s_{\sigma_1} + 1$. To limit the computational demand and also to focus on prior start of the repairing path, the start exchange is limited by the maximal distance d_{max} which is a number of locations ahead of the current one s_c to which the s_{σ_1} is randomly changed.

During one iteration of the *local search* procedure, the Randomized VNS tries one of the following neighborhood structures for a number of time that is equal to k^2.

- **Point Move** (l=1,3) randomly moves one location inside the solution into different position inside the plan.
- **Point Exchange** (l=2) selects randomly two target locations and switch their positions.

The VNS-based algorithm for the proposed RDTSP is summarized in Alg. 1. The planning algorithm is started every time S_m contains at least one missed location and the local optimization keeps trying to improve the repair path until it is applied by the *decision making* node which clears the set of missed target locations S_m.

The algorithm is written for the online execution during the mission, where both the set of missed locations S_m and the current index of location c are changing. During the deployment, the current index c increases which requires to change the considered staring locations s_{σ_1} and ending locations $s_{\sigma_{k+2}}$ of the repair path to be $s_{\sigma_1} > c$ and $s_{\sigma_{k+2}} = s_{\sigma_1} + 1$ which is done by *adjustStart* on line 3. Whenever the c increases, the algorithm changes automatically the starting position to the one inside d_{max} that produces the shortest repair path. Furthermore, the introduction of new missed locations into S_m requires addition of the location to the existing repair plan (*addToPath* on line 5) which is done greedily. By this manner, the proposed VSN-base algorithm for RDTSP benefits from reusing the existing repair plan during the deployment and tries to optimize it to minimize the additional path length required to revisit the missed target locations.

Algorithm 1: VNS based method for the RDTSP

Input : P_{init} – initial plan
Input : d_{max} – start lookahead distance
Input : l_{max} – maximal number of neighborhoods
Input : m – Number of heading angles samples
Changing Input: S_m – Set of missed locations
Changing Input: c – actually passed location index in P_{init}
Output : P – Actual RDTSP repair plan

```
1  P ← createInitialPath(S_m) ;                          // greedily
2  while Repair plan is not applied do
3      if c changes then
4          adjustStart(P,c) ;                  // ensures s_σ1 > c
5      if new missed target s_i ∈ S_m then
6          addToPath(P,s_i) ;                             // greedily
7      l ← 1
8      while l ≤ l_max do
9          P' ← shake(P, l)
10         P'' ← localSearch(P', l)
11         if L_path(P'') < L_path(P) then
12             P ← P''
13             l ← 1
14         else
15             l ← l + 1
```

V. Experimental Results

The proposed RDTSP method has been tested in a realistic outdoor experiment with the UAV depicted in Fig. 3. The Dubins vehicle model is used for the hexarotor UAV to produce smooth paths over the target locations using constant speed trajectories that are preferred due to the precision of sensory measurements in the information gathering scenario.



Fig. 3: The hexarotor UAV used during the experimental verification of the proposed RDTSP in scenario with initial plan obtained as the DTSP (for more details see http://mrs.felk.cvut.cz/ecmr17rdtsp)

Two different scenarios have been used to verify the performance of the online replanning of the RDTSP. The first scenario is based on precomputed coverage path plan ('sweeping' plan) where the given area is covered by the simple zigzag plan (see section V-A). The second scenario uses an offline open-loop DTSP plan over specified target locations. Section V-B describes the second scenario.

For both scenarios[1], the same configuration of the proposed RDTSP has been used. The maximal distance in which the VNS-based method considers the disconnection from the original plan and applying the repair path has been set to $d_{max} = 20$. The nominal forward speed of UAV during the experiment has been set to constant speed $v_c = 2.5\frac{m}{s}$. The RDTSP further uses the number of sampled heading angles $m = 16$ and turning radius of the Dubins vehicle $\rho = 4m$. The turning radius has been selected with respect to the maximal acceleration $a_{max} = 3\frac{m}{s^2}$ of the used hexarotor UAV.

The *decision making* node in the conducted RDTSP experiments has been used for determining whether the specified target locations were visited and also for determining when the calculated repair plan is applied. The comparison of the target locations position and UAV odometry was used to assess the missing of a target location with decision tolerance distance of 1m. Furthermore, to test the real replanning ability of the proposed method, the decision making used an artificially introduced probability $p_{miss} = 0.2$ of missing the target location (which simulates an error in the sensory data measurements). Furthermore, the decision of applying the repair path (also done by the *decision making* node) has been set to a simple variant where the new plan is applied every time the UAV reaches a starting position of the repair path with two or more missing target locations.

A. RDTSP in coverage scenario

The conducted experiment of RDTSP in the Coverage Path Planning scenario uses a simple zigzag plan over the specified area with equidistantly sampled target locations (see Figure 4).

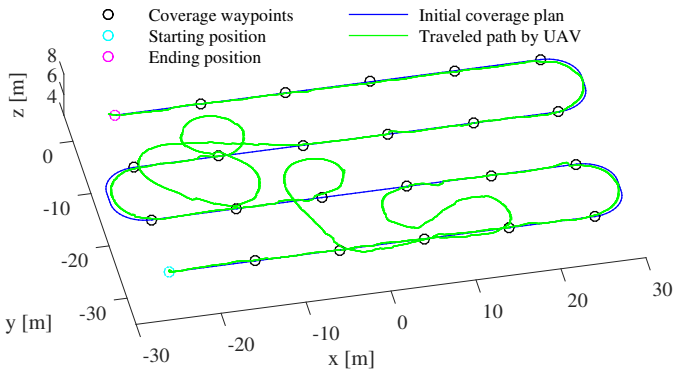


Fig. 4: Initial coverage zigzag path plan between starting and ending positions with equidistantly sampled target locations. The positions of the UAV ('Traveled path by UAV') shows the odometry measurements of the UAV that includes the two applied repair plans to revisit the target locations missed during the mission.

[1]We refer to `http://mrs.felk.cvut.cz/ecmr17rdtsp` for more information and video from the outdoor experiment that further visualizes the real time performance of RDTSP method.

During the experiment, a total number of six target locations were missed, either due to the real displacement of the UAV position from the original plan or due to the artificially introduced random missing of the target. The initial three targets, as shown in Fig. 5, were revisited by the first repair path and the other three by the second replanning, always by connecting the repair plan between two adjacent target locations on the original coverage plan.

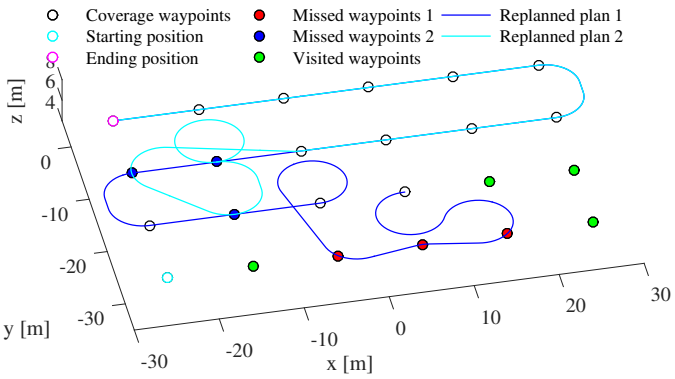


Fig. 5: Two repair plans applied by the RDTSP method during the execution of the coverage scenario.

B. RDTSP in DTSP scenario

In the RDTSP scenario with the initial path as the Dubins Traveling Salesman Problem, a total number of 20 target locations were specified. The initial plan, an open-loop version of the DTSP, was found by conversion of the DTSP into the ATSP and then optimally solved using the Concorde solver. The results of the experiment are depicted in Figure 6 and Figure 7, where the two missed target locations at the beginning of the mission are revisited by one repair path that is significantly shorter compared to the case of visiting the missed locations after the mission.

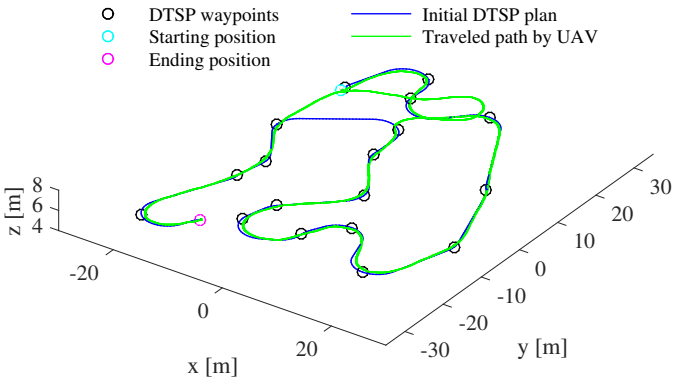


Fig. 6: The initial Dubins Traveling Salesman Problem plan over specified target locations together with the odometry measurements of the UAV including the additional repair path over missed target locations.

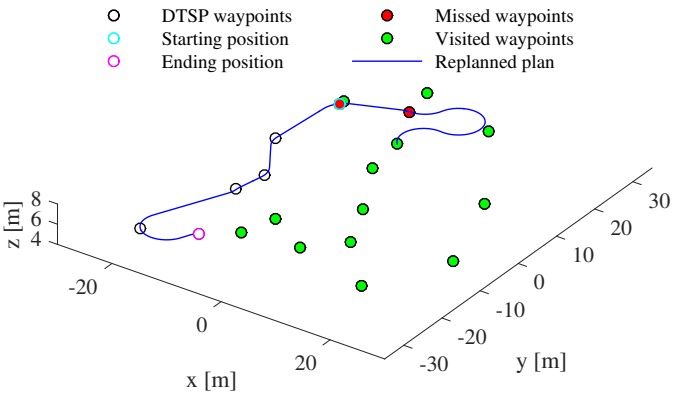


Fig. 7: The repair path applied after missing two target locations in the beginning of the DTSP scenario

The experimental verification shows that the proposed RDTSP is a robust method for the information gathering tasks, both for the CPP and DTSP scenarios, where all the missed target locations were revisited by the repair paths. The introduced online method minimizes the overall length by connecting the repair plan into the initially computed offline plan. Especially Figure 6, with the DTSP scenario, shows the benefits of the online replanning with lookahead distance, where the final path length is minimized by selecting the appropriate target location of connection the repair plan.

VI. CONCLUSIONS

In this paper, we introduced a novel method for information gathering by UAVs called Reactive Dubins Traveling Salesman Problem. The method uses a precomputed information gathering path from either the Coverage Path Planning for scanning of a given area or from the Dubins Traveling Salesman Problem where the length of the path over all given target locations is minimized. The proposed VNS-based online algorithm for the RDTSP uses the predefined path during the mission execution and ensures the robustness of information gathering task by replanning the initial plan for visiting possible missed target locations due to disturbances in sensory measurements or localization. The proposed method was verified in realistic outdoor information gathering experiments with hexarotor UAV and shows the robustness of the proposed online replanning approach in both CPP and DTSP scenarios. For the future work, we intend to investigate different strategies and improvements for the decision making, where the application of the repair plan is based on the real sensor measurements.

ACKNOWLEDGMENT

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 17-16900Y. The support of the Grant Agency of the Czech Technical University in Prague under grant No. SGS17/187/OHK3/3T/13 to Robert Pěnička is also gratefully acknowledged. The support of the Czech Ministry of Education, Youth and Sports and of the French Ministries of Foreign Affairs for the Czech-France MOBILITY project 7AMB16FR017 is greatly appreciated.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258 – 1276, 2013.

[2] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for dubins' vehicle," in *ACC*, 2005, pp. 786–791 vol. 2.

[3] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427 – 441, 2011.

[4] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[5] "Concorde tsp solver," citied on 2017-05-14. [Online]. Available: http://www.math.uwaterloo.ca/tsp/concorde.html

[6] K. J. Obermeyer, "Path planning for a UAV performing reconnaissance of static ground targets in terrain," in *AIAA Conf. on Guidance, Navigation and Control*, Chicago, IL, USA, Aug. 2009.

[7] J. Faigl and P. Váňa, *Self-Organizing Map for the Curvature-Constrained Traveling Salesman Problem*. Cham: Springer International Publishing, 2016, pp. 497–505.

[8] X. Yu and J. Y. Hung, "A genetic algorithm for the dubins traveling salesman problem," in *2012 IEEE International Symposium on Industrial Electronics*, May 2012, pp. 1256–1261.

[9] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "Autonomous uav surveillance in complex urban environments," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, Sept 2009, pp. 82–85.

[10] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[11] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization," *Autonomous Robots.*, vol. 41, no. 4, pp. 919–944, 2017.

[12] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil, "Coordination and Navigation of Heterogeneous MAV&UGV Formations Localized by a "hawk-eye"-like Approach Under a Model Predictive Control Scheme," *International Journal of Robotics Research*, vol. 33, no. 10, pp. 1393–1412, 2014.

[13] M. Saska, V. Spurny, and V. Vonasek, "Predictive control and stabilization of nonholonomic formations with integrated spline-path planning," *Robotics and Autonomous Systems*, vol. 75, no. Part B, pp. 379–397, 2016.

[14] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny, and L. Preucil, "Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups," *Journal of Intelligent and Robotic Systems*, vol. 73, no. 1-4, pp. 603–622, 2014.

[15] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles," *Journal of Intelligent & Robotic Systems.*, vol. 84, no. 1, pp. 469–492, 2016.

[16] M. Saska, J. Vakula, and L. Preucil, "Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[17] M. Saska, "MAV-swarms: unmanned aerial vehicles stabilized along a given path using onboard relative localization," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

[18] T. Baca, G. Loianno, and M. Saska, "Embedded model predictive control of unmanned micro aerial vehicles," in *MMAR*, 2016.

[19] P. Hansen and N. Mladenovi, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

Autonomous Task Execution within NAO Robot Scouting Mission Framework

Anja Babić, Nikola Jagodin and Zdenko Kovačić

Abstract— **A polygon for movement and task execution by the humanoid robot NAO was defined in the form of a 2D map. A series of tasks was designed for the robot to accomplish during its scouting mission. A localization algorithm using markers and the robot's camera was developed, as well as algorithms for navigation, path planning, and robot motion. A GUI for mission definition, supervision, and control, along with manual robot tele-operation, was developed. A finite automaton was defined which enables switching between autonomous, semi-autonomous, and manual operation modes. The developed modules were integrated with modules for audio and visual perception and the complete system was tested on a chosen scouting mission.**

Keywords: NAO, humanoid robot, localisation, navigation, GUI, autonomous task execution

I. Introduction

Robotic technology is advancing in large steps, enabling a variety of robot applications in unstructured environments such as scouting, inspecting, rescuing, and intervening in dangerous situations. International robotic competitions such as DARPA and ARGOS challenges have set new standards for professional service robots, requiring from the robots full autonomy of operation, advanced manipulation skills, awareness of the environment, learning capability, adaptability, and the ability of collaboration with other robots and humans [1, 2]. This leads to increased demand for creating a robot with a set of functionalities including artificial intelligence, environment awareness, locomotion, energy efficiency, control mode switching, and mission comprehension. Costs of using advanced human-like and human-size professional service robots for implementing and validating new control concepts are not affordable for most academic research laboratories. Fortunately, small scale humanoid robots such as NAO [3], Bioloid [4], and Darwin-OP [5] have recently become technologically and economically acceptable enough to be used instead.

The work presented in this paper was inspired by the ARGOS challenge (Autonomous Robot for Gas and Oil Sites) announced in 2014 by the French oil company Total [2]. In line with the request for a 24/7 robot-based inspection of an off-shore oil platform or similar oil/gas facility, the aim of this research was to explore the possibility of the NAO humanoid robot autonomously performing a scouting mission in a previously known space and to develop a supervisory system with a human-machine interface. To perform its task, the robot must be able to locate itself in the given space, plan its path and navigate to the selected positions, and perform a series of tasks for which the space is equipped. While collecting information about the environment, the robot is expected to manoeuvre without human intervention and be safe for humans, the environment, and itself. An autonomous robot can also have adaptive abilities, and by learning it can adapt to the surrounding environment even if the environment changes. In addition, it can find new ways for faster or better fulfilment of tasks [6]. The scouting mission of a NAO robot comprises several actions such as visiting known locations with measuring instruments (e.g. temperature, pressure), reading them, reporting to the surveillance system, signalling possible alarms, and tackling issues related to valves. During the inspection mission the robot should respond to the sound of the alarm by going into a predefined safe area and should also take care of its own battery level by visiting a charging station and recharging itself. If the robot hears the sound of a steam leak or gas leak, it first goes to the place where it knows the valves are located and checks if any of them are open. If all the valves are closed, it goes to the nearby pipeline to look for cracks and, if circumstances demand, the mission results are signalled to the surveillance system.

The paper is structured in the following way. The second section contains important technical specifications of the robot used as well as some details of the background of the software implementation. The third section contains a description of the used spatial localisation algorithm, and in the fourth section the planning of the walking path and the motion control of the robot are described. The fifth section describes the entire developed system, from an overview of the finite automaton used to a brief explanation of the roles of the implemented classes and modules. After describing the course of the mission the robot manages autonomously, several conclusions and plans for further work are presented.

II. Hardware and Software Used in Work

NAO H25 robots (see Fig. 1) are small size humanoid robots characterized by an appealing appearance and the ability of verbal and visual communication with people as well as cooperation with other NAO robots. Their ability to communicate and collaborate also allows them to be an integral part of cooperative human-robot societies.

As with humans, the most desirable ability for successful inclusion of NAO robots in cooperative human-robotic systems is watching and quickly interpreting what is seen. Therefore, in order to successfully navigate, manipulate objects, and solve complex tasks, the robot needs to be capable of advanced image processing, locating and identifying objects in 2D and 3D space, intelligent grasping, listening, interpreting and executing commands, mastering social skills, coping with irregular situations, and acquiring specific skills related to the performance of a given task.

A. Babić and Z. Kovačić are with the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (e-mail: {anja.babic; zdenko.kovacic}@fer.hr).

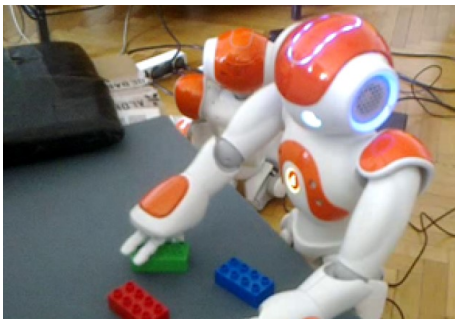


Figure 1. NAO H25 robot

The NAO H25 has 25 degrees of freedom. It possesses 2 loudspeakers and LED indicators to interact with the user and environment. It also has two cameras, four microphones, two ultrasound sensors (sonar), two infrared transceivers / receivers, and nine tactile and eight pressure sensors distributed over its body. Through a Wi-Fi network, the robot is able to communicate with a control centre and other NAO robots connected to the network. The NAO robot software (NAOqi, Choregraphe) comes pre-packaged with the robot and contains a number of ready-to-use functions and behaviours, but also allows and encourages adding new ones created using common high-level programming languages (Python, C++). In this work some ready-to-use functions available under the GNU General Public License (GPL), developed within the MRPT (Mobile Robot Programming Toolkit) project [7,8] were also used.

For implementation of a human-robot interface (HRI) with widgets some ready-to-use functions available under the GNU General Public License (GPL) within the Qt framework [9] were used. For this purpose a specific "signals and slots" way of communication among HRI components was adopted [10]. This has allowed for prompt reactions of the NAO robot to sudden events in the environment, based on the implemented finite state automaton.

III. LOCALIZATION PROBLEM

Localisation of a mobile robot is a crucial condition for achieving any degree of autonomy. In a typical case of indoor robots moving on a flat surface, localisation is defined as the estimate of robot poses, i.e. its x and y coordinates and the orientation given by the angle θ.

In this paper, global localisation is achieved using unique markers deployed in the environment (Fig. 2), which make it possible to determine the robot's current position without knowing its initial position or its position in the previous step, and ensure that robot autonomy is not compromised even in the so-called "robot kidnapping scenario" in which a mobile robot is suddenly moved to an arbitrary location while working, creating great difficulties and introducing insecurity in most localisation algorithms [11].

The main assumption of the robot localisation process is that there is a well-known map of the space where the mission and all the planned movements take place. In this paper, a probabilistic grid occupancy map is used [12]. It is a metric type of map, which means that it represents the geometric properties of the space it describes, by dividing it into a network of fields of the same size and assigning each of them a numerical value. This value represents the probability that that part of the space is occupied - that is, that it represents a barrier to robot movement [13]. In order to make the implementation more efficient and avoid numeric errors due to computations with very close probability values ranging from 0 to 1, the data in the matrix representing the grid occupancy map is entered using the natural logarithm of the so-called *odds* function, or function of chance (Fig. 3).

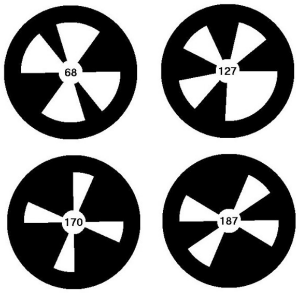


Figure 2. Example markers used for global localization

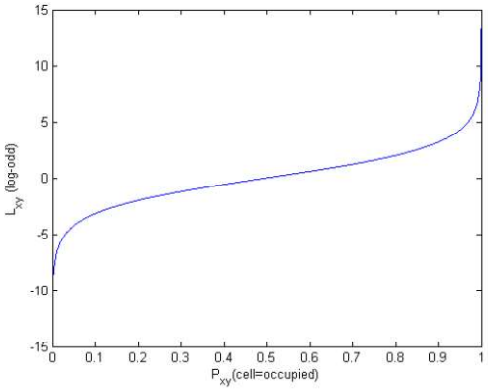


Figure 3. The logarithm of the *odds* function [8]

At any given moment, from the known logarithm of the *odds* function l_k written in the map matrix, it is possible to retrieve the probability of the field occupancy represented by the random variable S_{ij}, which is in the row i and column j, with a simple operation:

$$P\left(S_{ij} = occupied\right) = \frac{1}{1+e^{-l_k}} \tag{1}$$

The maps used in this paper have been constructed so that one pixel in the picture represents one centimetre in reality, and its brightness is a measure of probability of space occupancy, with a wholly black field meaning a safe obstacle, and a wholly white field representing a free space.

The polygon built for robot movement and equipped with objects to carry out the mission is shown in Fig. 4, while the grid occupancy map corresponding to the displayed space is given in Fig. 5. Following the conventions of tagging in the field of image processing, the origin of the global coordinate system is located in the upper left corner of the image, and its y axis points down.



Figure 4. The indoor environment for robot inspection

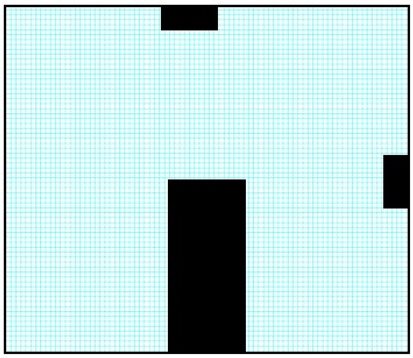


Figure 5. Grid occupancy map for space shown in Fig. 4

A list of markers present in the mission space and their coordinates in the global coordinate system are well-known to the robot. When attempting localisation, the robot turns its head until it sees a total of two markers it recognises from its list of known markers, then uses the developed local module *LRDetectionMarker* to retrieve their coordinates in its own local coordinate system.

Incorporating the robot's sonar into the localisation process proved to be unhelpful, because the robot's ultrasonic sensors were too unreliable, even with filtering outliers using the median of a certain amount of collected sensor readings, and disrupted the accuracy of the implemented algorithms. The probable cause for this is multiple ray reflection from relatively close walls with many corners, and it is possible that results would be better in less crowded spaces.

A. Calculation of global coordinates

After the robot has successfully detected two well-known markers and determined their coordinates in its local coordinate system, it is possible to determine the coordinates and the robot's orientation in the global coordinate system. The observed case is two-dimensional, which simplifies parts of the calculation.

Coordinate systems are shown in Fig. 6, with (x_{1g}, y_{1g}) and (x_{2g}, y_{2g}) as coordinates of the markers in the global coordinate system, (x_{1l}, y_{1l}) and (x_{2l}, y_{2l}) as the markers' coordinates in the local coordinate system of the robot, and (x_r, y_r) as coordinates of the robot (the origin of the local coordinate system of the robot) in the global coordinate system. The angle θ is the angle between the two x coordinate system axes and represents the orientation of the robot.

Between the robot coordinate system and the global coordinate system, the only possible transformations are Euclidean (rotation and translation):

$$\begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_g \\ y_g \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_g \\ y_g \\ 1 \end{bmatrix} \tag{2}$$

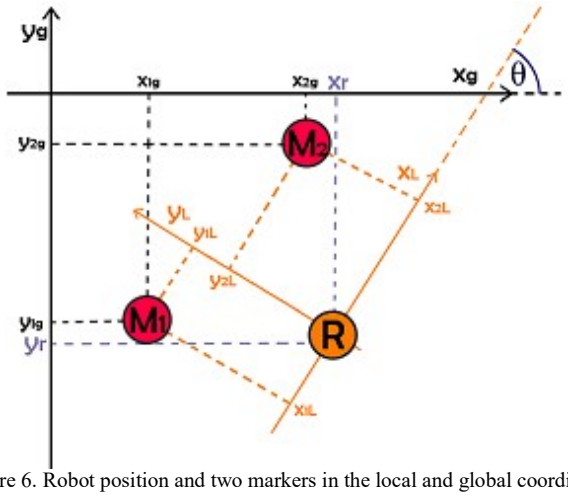


Figure 6. Robot position and two markers in the local and global coordinate system

A very valuable feature of rigid transformations – a class which Euclidean transformations belong to - is that only two pairs of corresponding points are needed for uniquely determining the transformation from one coordinate system to another. Specifically, in this case, the pairs are the known coordinates of each of the two found markers in both the local and the global coordinate system.

From this, it is possible to write a system of four equations with four unknowns:

$$\begin{bmatrix} x_{1l} \\ y_{1l} \\ x_{2l} \\ y_{2l} \end{bmatrix} = \begin{bmatrix} x_{1g} & y_{1g} & 1 & 0 \\ y_{1g} & -x_{1g} & 0 & 1 \\ x_{2g} & y_{2g} & 1 & 0 \\ y_{2g} & -x_{2g} & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \\ t_x \\ t_y \end{bmatrix} \tag{3}$$

By solving this system, the $\cos\theta$, $\sin\theta$, t_x, and t_y values are obtained, thus establishing the connection between the two coordinate systems - the requested matrix **H**.

To determine the position of the robot in the global coordinate system, it is necessary to transform the local coordinates (0, 0) using the inverse of matrix **H**:

$$\begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4}$$

For fast computation of the inverse matrix, single value decomposition (SVD) is used, which returns the result even when the matrix is singular or near singular. Robot orientation θ is determined from the now known values of $\sin\theta$ and $\cos\theta$ using the atan2 function that takes into account the quadrant in which the angle θ is located. Finally, because of the aforementioned definition of the global coordinate system in Fig. 6, the obtained global y coordinates change the sign, i.e. mirroring over the x axis is applied.

IV. NAVIGATION PROBLEM

The exploration mission of the robot can be set by the operator as a destination point on a map or as a set of sites that the robot has to navigate (patrol). Confirming the selected position will trigger path planning to that point and allow the NAO robot to walk. During patrolling, the robot takes the next point in the sequence, plans a path, and tries to move to the appropriate position in the polygon. Upon arrival, the visited point is deleted from the sequence and the patrol procedure is repeated until all the given points of the patrol mission have been visited.

During patrol mission planning or single destination setting it is only possible to select free fields on the map. If the field is free, or if the following condition is valid:

$$P\left(S_{ij} = occupied\right) < 0.5 \tag{5}$$

then its closest environment is checked. Considering the size of the robot itself, all fields within a 25 cm radius of the considered field need to be free in order for the field to be included in the mission, i.e. it is demanded that the field be sufficiently distant from any obstacle so that the robot can approach the position safely and without interfering with the obstacle in any way.

Path planning is done using the MRPT tool grid map processing methods and takes two steps:

- The obstacles represented by the occupied map fields are enlarged so that the size of a single field corresponds to the safety radius of the robot for which the path is planned, which ensures that even one free field on the map means that there will be no collision of the robot with obstacles in the environment – in this work, as mentioned earlier, this value is set to 25 cm.
- The shortest path to the destination is searched for using the value-based iteration algorithm based on the Markov decision-making process, with the maximum permissible distance between the two endpoints set at a length of 20 cm, which emerged as the optimal choice during experiments [8].

A. Robot movement

For motion control and walking, the built-in ALMotion module is used, covering four main control areas [14]:

- Control of motor torques (stiffness)
- Control of joint positions (path generation, interpolation between given positions)
- Control of walking (stride length and frequency)
- Control of the effectors in the Cartesian space (built in inverse kinematic solver, Whole Body Control function).

Each walk started with the ALMotion module commands starts and ends with a phase in which both legs provide support in a 0.6 second time interval. The foot path within the step (during the leg swing phase) is determined by SE3 interpolation, which, similarly to spline interpolation taking into account the initial and final velocities on the trajectory segments, ensures smooth transitions [14].

The robustness of the robotic walk with regards to relatively small external disturbances is ensured by closing the feedback loop with the sensors in the leg joints of the robot. This control loop is designed to reduce the oscillation of the torso resulting from the walking movement of the robot. The NAO can thus pass over different surfaces while walking, e.g. can cross the edge of a carpet, parquet or tiles, which opens up many possibilities for use in environments that are not specifically adapted to the robot's functioning.

Before the start of each walk, the robot is placed in a neutral, stable initial position for walking, so called *standInit*. Then the *moveTo* command gives the desired x and y coordinates and orientation θ in the local coordinate system of the robot. The embedded ALMotion modules then perform the planning of the walk to the given stride frequency.

As all path planning is performed in the global coordinate system while the destination targeting and selection happens in the local coordinate robot system, the target coordinates calculated during the path planning are transformed into local coordinates using the transformation matrix **H** defined by (4).

Robot localization is performed after each successfully completed walking interval in order to ensure that the transformation matrix is updated regularly and all calculations are performed with the latest available data from the environment. The main reason for this is the exceptional unreliability of feedback obtained from the robot's odometry, which leads to a significant error in estimating the robot's real-world position even after very small distances and can have serious consequences, such as collisions leading to damage to the robot or the environment.

Since the scouting mission includes tasks related to reading instrument values and performing a visual inspection of the pipes and valves, at the times when the robot moves to perform these tasks it must be ensured that the camera used for each task (top camera in the case of inspecting pipes and instruments, bottom camera in the case of inspecting valves) is correctly aligned.

Additionally, as the global positions and orientations of objects relevant to the mission are known, it is necessary, upon approaching areas close to which it is possible to perform the desired task, to confirm the accuracy of the robot position by re-localisation and re-centering using the most recently acquired data, the robot is commanded to turn around its own orientation-determining axis by the angle difference between its current and desired orientation.

Experiential evaluation of distances acceptable for successful performance of tasks has indicated reliable performance for up to 40 cm away from the observed object, with a viewing angle not exceeding 60 degrees. The important positions on the map and the orientation that the robot needs to assume in each are shown in Fig. 7.

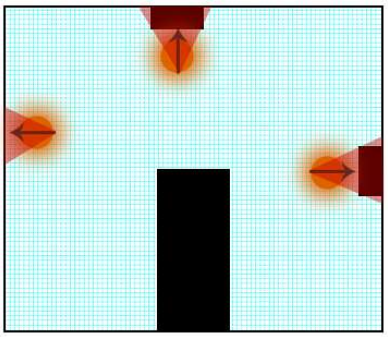


Figure 7. Positions and orientations for performing tasks (from left to right: pipe, valves and instrument)

V. CONTROL SYSTEM STRUCTURE

Controlling the entire system is achieved by using a mission controller represented by a finite automaton (Fig. 8). The states MANUAL, AUTO, and SEMI represent the basic operating modes of the system. At start-up, the system immediately switches to MANUAL mode. The states marked with GO TO SAFETY and GO TO RECHARGING represent the system states after reacting to critical events (alarm and low battery) in which the system will remain (final states). The state OPERATOR NOTICE is a strictly transient state. Transitions between the three main states may occur at the request of a user - when the status selection key is pressed - or as a reaction to external events (e.g. recognition of the sound of steam exiting a pipe, or the robot's message that it has failed to localise itself).

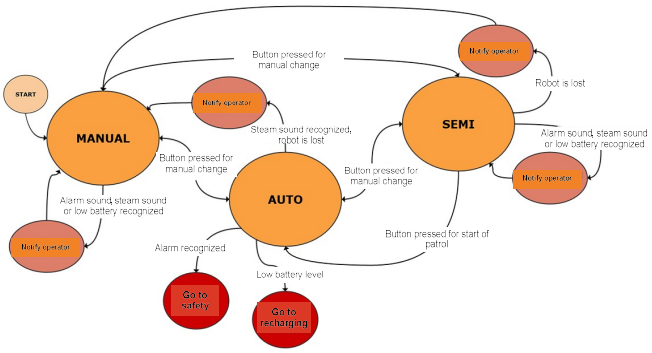


Figure 8. System states and transitions within finite automaton

A. Manual mode

In MANUAL mode, the user interface displays a live video feed from the top camera of the robot (Fig. 9). By pressing the arrow keys, the robot moves in the predetermined direction for as long as the key is pressed, and when the key is released the movement stops. The buttons marked with STRAFE allow the robot to be pushed sideways, while the left and right arrow keys mean turning the robot around its own orientation axis.

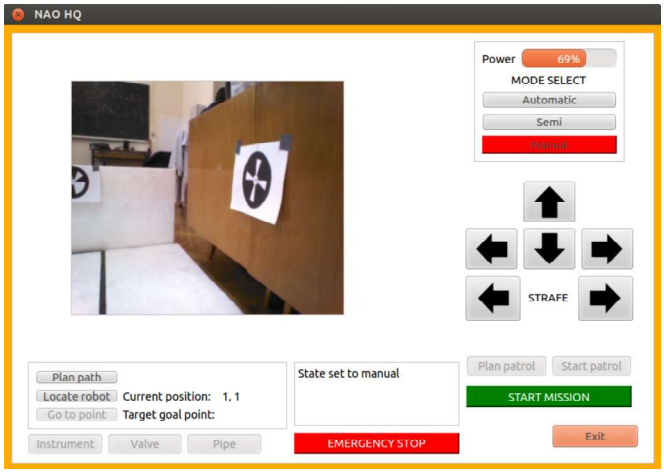


Figure 9. User interface in manual mode

In all modes, the current battery level is displayed and refreshed every 30 seconds, and pressing the emergency stop button marked EMERGENCY STOP interrupts any currently active motion of the robot.

B. Semi-automatic mode

In semi-automatic mode (SEMI), robot localisation can be manually started, and paths to a desired point on the map can be planned. After a path has been successfully planned, it is shown on the map and it becomes possible to press the appropriate button to send the robot to the given location.

Patrol planning and starting a predefined mission are performed in this mode. The patrol is defined by pressing the left mouse button on the desired location on the map to select a target point, which is then added to the number of points the robot must check during the patrol. Points from the patrol can be erased one by one, or the patrol planning can be completely cancelled and point selection restarted. It is also possible to manually request tasks such as reading temperature value from a presumably visible instrument, checking valve states, and looking for cracks in pipes.

C. Automatic mode

In the automatic mode (AUTO), a patrol or predefined mission is performed (Fig. 10). In this mode, the robot responds to the alarm, steam sound, and low battery level by planning its route and moving to the appropriate places, while in all other modes it only notifies the operator about these important events through the user interface.

Significant events during system operation and events that are being reported by the robot are recorded in the log file, together with a timestamp. The functionality of the entire system is realised using a series of developed C++ classes and modules [15]. The modules *LRMarkerDetection*, *LRInstrumentsValvesDetection,* and *LRBatteryStatus* that serve for audio-visual perception of the environment are described in detail in [16]. Image processing algorithms have been developed to allow the robot to observe valves in order to see if they are open or closed, and to find and count cracks on an observed pipe. Apart from this, the robot can detect the analogue measuring instrument in its environment and read

the value shown by its hand. The robot is capable of detecting a loud noise and, by using different features calculated from recorded short audio snippets, it classifies the detected sound and recognizes whether it is an alarm sound or the hiss of steam or leaking gas.

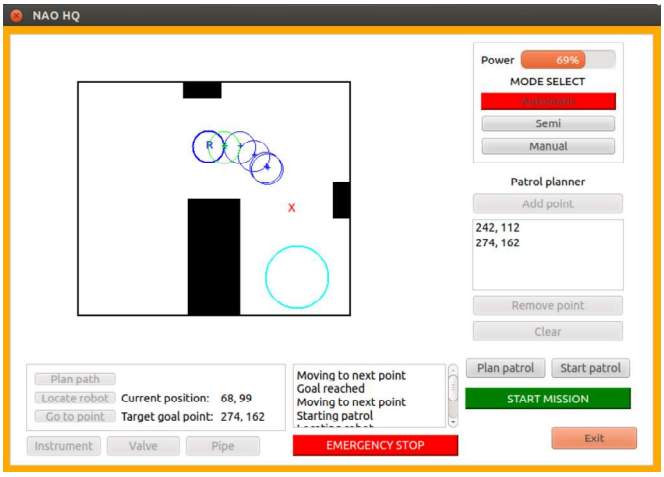


Figure 10. User interface in automatic mode

D. *Mission description*

A mission that relies on the aforementioned modules has been conceived. NAO fulfils the role of the inspection robot for the given space by visiting it, checking the room temperature, and responding to different stimuli from the environment. The mission flow is shown in Fig. 11.

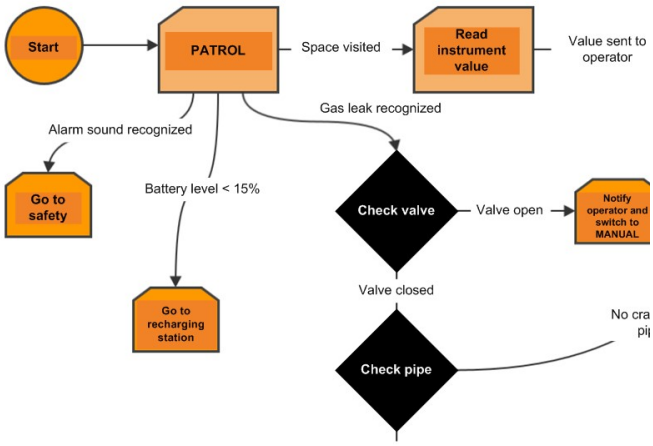


Figure 11. Mission flow

VI. Experimental Results

For the purposes of testing the developed system and all its functionalities, a special polygon was prepared (Fig. 4). The mission cannot start until the robot is given at least one point as part of a patrol. By pressing the start key, the system switches to automatic mode and the robot starts the given patrol. After visiting all the given points, the robot moves to the known instrument location - a thermometer where it must read the temperature value (Fig. 12). By successfully performing this part of the task and reporting the readout value to the user interface, and thus to the operator monitoring the system operation, the mission is considered completed and the robot enters the semi-automatic mode and waits for further instructions. If the instrument is not detected or the value has not been read successfully, the supervisory interface is notified.

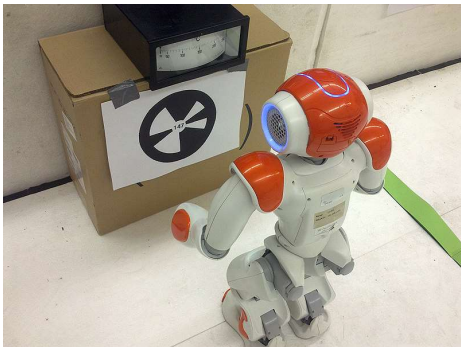


Figure 12. The robot reads the value displayed on the instrument

If the robot receives an alarm while it is on the go, it at once interrupts everything that it is doing, tells the interface that the alarm is received, plans the fastest way to the previously defined safe space and goes to safety. Similarly, if the battery level falls below the default 15% threshold, the robot interrupts the patrol and mission and goes to the recharging slot that has been pre-defined (Fig. 13).

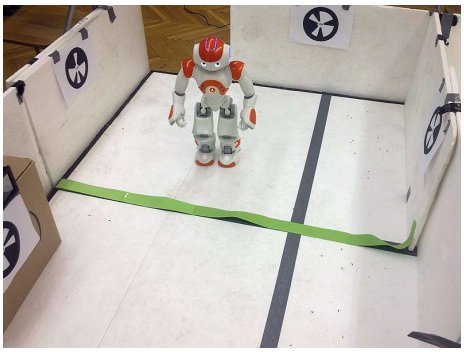


Figure 13. The robot is in the safe place

If the sound of steam or gas leak is observed, the robot interrupts the patrol and goes to the place where it knows the valves are located in order to check them (Fig. 14).

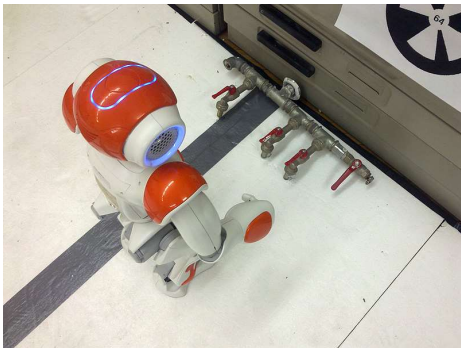


Figure 14. The robot checks if the valve is open

If none of the valves are open, the robot moves to the place where it can observe the pipes in order to find cracks on them (Fig. 15). If there are no cracks on the pipes, the robot enters the semi-automatic mode and waits for further instructions. If it detects an open valve or cracks on the pipes,

it notifies the user interface and enters the manual mode so that in the potential future implementation of a complete remote robot operator interface, the operator could control the robot to repair the failure or close valves if necessary.



Figure 15. The robot seeks cracks on the pipes

VII. Conclusion

The aim of this work was to investigate the possibility of developing a system consisting of a series of separate elements that allows a humanoid robot NAO to independently perform scouting missions in the space provided. The problem of robot localisation was resolved by using cameras and markers. Planning movement paths while patrolling the environment was implemented, enabling the robot to autonomously operate in a known indoor space. Developed modules for interaction with the environment and collecting audio and visual information were integrated into the user and control interface. A series of automated reactions to environmental events and behaviours triggered in critical conditions have been achieved. A graphical user interface has been developed, with a finite automaton mission controller handling automatic, semi-automatic, and manual modes of operation, so that the operator can monitor the work of the robot or, if necessary, take control over it.

For closer study of the developed system, a testing polygon was prepared. The most common problems showing up in the work are the result of poor lighting because the robot heavily relies on its cameras and subsequently the quality of the images retrieved from them. Robot autonomy is greatly limited due to the relatively short battery life of the NAO robot. For this reason, battery level-based control has been introduced as a safety measure. Going to the recharging point in the current system implementation is only for experimental purposes because the NAO robot cannot recharge itself. Using the newly developed NEST [17] autonomous recharging equipment, the functioning of the entire system could be significantly improved. The problem that arises is the emergence of major deviations from the desired path during a robot walk, especially in terms of orientation. The robot's feet are slippery, and the built-in odometry is very unreliable, so that safe walking is possible only over small distances before it becomes necessary to introduce corrections by repeating the localisation process, which significantly slows the movement. By introducing repeating localisation and correction of robot orientation before performing certain tasks that require the robot to closely look at the subject, the performance of tasks during the mission has greatly improved.

For further development, it would be beneficial to include control of the robot's arms in the interface, which would allow for a broader range of tasks to be executed autonomously or remotely. Robot navigation could be enhanced with the development of a predictive robot motion model that would allow corrections both during and after the motion, and with the definition of a new step sequence, the robot's walk would be improved and stabilised. Localisation using pre-known maps in can be replaced by simultaneous localization and mapping algorithms (SLAMs), which, though essentially much more demanding in implementation and computing, significantly increase the degree of robot autonomy and allow for adaptation to larger changes in the environment, especially with the addition of more advanced modes of robot movements to the system (e.g. climbing stairs, tilting and crawling under an obstacle).

Some of the results obtained while testing the control algorithms and realised software solutions are shown in the video clip of this work which can be watched at [18].

References

[1] DARPA Robotics Challenge 2015 Finals, URL https://web.archive.org/web/20160428005028/http://www.darparoboticschallenge.org

[2] ARGOS (Autonomous Robot for Gas and Oil Sites) Challenge, URL http://www.argos-challenge.com/en

[3] Softbank Robotics (Aldebaran Robotics): Who is NAO, URL https://www.ald.softbankrobotics.com/en/cool-robots/nao

[4] Robotis Anthropomorphic Robot Bioloid, URL http://support.robotis.com/en/product/bioloid/gp_kit.htm

[5] Robotis Dynamic Anthropomorphic Robot with Intelligence (Darwin-OP), URL http://support.robotis.com/en/product/darwin-op.htm

[6] G.A. Bekey, Autonomous robots: from biological inspiration to implementation and control. MIT press, 2005.

[7] J. L. B. Claraco, Development of Scientific Applications with the Mobile Robot Programming Toolkit. The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of Málaga, Málaga, Spain, 2008.

[8] J. L. Blanco, Mobile Robot Programming Toolkit (MRPT), 2011. URL http://www.mrpt.org/.

[9] Trolltech Qt software documentation, "Signals and Slots", 2014. URL http://qt-project.org/doc/qt-4.8/signalsandslots.html.

[10] J. Blanchette and M. Summerfield, C++ GUI programming with Qt 4, Prentice Hall Professional, 2006.

[11] K. JooHyun, Place recognition and kidnapped robots - Visual Recognition and Search. University of Texas, Austin, USA, 2008. URL http://www.cs.utexas.edu/~grauman/courses/spring2008/slides/Joohyun_place_recognition.pdf.

[12] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer 22 (6), 46-57, 1989.

[13] S. Thrun, Robotic mapping: A survey. Exploring artificial intelligence in the new millennium, 2002.

[14] NAO Software 1.14.5 documentation. Aldebaran Robotics, 2014. URL https://developer.aldebaran-robotics.com/doc/1-14/.

[15] A. Babić, Autonomous Task Execution within NAO Robot Scouting Mission Framework, M.Sc. thesis, FER, Zagreb, 2014 (in Croatian).

[16] N. Jagodin, Audio-visual Perception of Environment during NAO robot Scouting Mission, M.Sc. thesis, FER, Zagreb, 2014 (in Croatian).

[17] S. Toto, Mini Humanoid NAO Recharges Himself Autonomously, 2011. URL http://techcrunch.com/2011/09/20/video-mini-humanoid-nao-recharges-himself-autonomously/.

[18] Video attachment: Autonomous Task Execution within NAO Robot Scouting Mission Framework, https://youtu.be/MDYWION009o, May 2017.

Scalable Multirotor UAV Trajectory Planning using Mixed Integer Linear Programming

Jorik De Waen[1], Hoang Tung Dinh[2], Mario Henrique Cruz Torres[2] and Tom Holvoet[2]

***Abstract*— Trajectory planning using Mixed Integer Linear Programming (MILP) is a powerful approach because vehicle dynamics and other constraints can be taken into account. However, it is currently severely limited by poor scalability. This paper presents a new approach which improves the scalability regarding the amount of obstacles and the distance between the start and goal positions. While previous approaches hit computational limits when the problem contains tens of obstacles, our approach can handle tens of thousands of polygonal obstacles successfully on a typical consumer computer. This performance is achieved by dividing the problem into many smaller MILP subproblems using two sets of heuristics. Each subproblem models a small part of the trajectory. The subproblems are solved in sequence, gradually building the desired trajectory. The first set of heuristics generate each subproblem in a way that minimizes its difficulty, while preserving stability. The second set of heuristics select a limited amount obstacles to be modeled in each subproblem, while preserving consistency. To demonstrate that this approach can scale enough to be useful in real, complex environments, it has been tested on maps of two cities with trajectories spanning over several kilometers.**

I. Introduction

Trajectory planning for multirotor UAVs is a complex problem because flying is inherently a dynamic process. Proper modeling of velocity and acceleration are required to generate a feasible trajectory that is both fast and safe, that is, the UAV should be able to effectively navigate corners while maintaining momentum. The fastest trajectory is not always the shortest one, since the UAV's velocity may be different. The UAV dynamics are often not the only constraints placed on the trajectory. Different laws in different countries also affect the properties of the trajectory. The operators of the UAV may also wish to either prevent certain scenarios or ensure that specific criteria are always met.

In this paper we present a scalable approach which is capable of generating fast and safe trajectories, while also being easily extensible by design. We model the trajectory planning problem as a Mixed Integer Linear Program (MILP). The trajectory is represented in discrete time steps where each step describes the UAV's dynamic state at that moment. An objective function encodes one or more properties, like time or trajectory length, to be optimized. A general solver is then used to find the optimal solution for the problem. Because the problem is defined declaratively, additional constraints can easily be added.

[1]Jorik De Waen is a student at KU Leuven, 3001 Leuven, Belgium jorik.dewaen@student.kuleuven.be

[2]Hoang Tung Dinh, Mario Henrique Cruz Torres and Tom Holvoet are with imec-DistriNet, KU Leuven, 3001 Leuven, Belgium {hoangtung.dinh, mariohenrique.cruztorres, tom.holvoet}@cs.kuleuven.be

We demonstrate our approach in 2D environments. We assume that all obstacles are polygons, static and known in advance. Our algorithm is designed for offline planning, ensuring that a feasible trajectory exists before the UAV starts executing its task. Other papers have used MILP for trajectory planning [1], their approaches could not be used to generate long trajectories through complex environments. Our main contribution is an approach which improves the scalability by dividing the problem into many MILP subproblems. Each subproblem models only a part of the trajectory. The subproblems are solved sequentially. A first set of heuristics uses a Theta* path to generate the subproblems. The second set of heuristics select which obstacles should be modeled in each subproblem, limiting the amount of obstacles that need to be modeled while ensuring that no collisions can occur.

Schouwenaars et al. [1] were the first to demonstrate that MILP could be applied to trajectory planning problems. They used discrete time steps to model time with a vehicle moving through 2D space. Obstacles are modeled as grid-aligned rectangles. To limit the computational complexity, they presented a receding horizon technique so the problem can be solved in multiple steps. However, this technique is essentially blind and could easily get stuck behind obstacles. Bellingham [2] recognized that issue and proposed a method to prevent the trajectory from getting stuck behind obstacles, even when using a receding horizon. However Bellingham's approach still scales poorly in environments with many obstacles.

Flores [3] and Deits et al. [4] use Mixed Integer Programming with functions of a higher order to model the trajectory as a continuous curve. The work by Deits et al. is especially relevant to this paper, since they also use convex safe regions to solve the scalability issues when faced with many obstacles. Recent work [5], [6], [7] has focused on online operation and control of the individual rotors of the UAV. Other than the work by Deits et al. [4], we did not find any literature which attempts to improve MIP trajectory planning performance in scenarios with many obstacles.

The paper is structured as follows: In Section II, we discuss the MILP trajectory planning model we used in the algorithm. Section III introduces our new algorithm which segments the problem to improve performance. In Section IV, we discuss the performance of the algorithm using several testing scenarios.

II. Modeling Path Planning as a MILP problem

This section covers how a trajectory planning problem can be represented as a MILP problem. The problem representation is based on the work by Bellingham[2].

A. Time and UAV state

The trajectory planning problem can be represented with N discrete time steps and a set of state variables at each time step [2]. The number of time steps determines the maximum amount of time the UAV has to reach its goal.

$$\boldsymbol{p}_0 = \boldsymbol{p}_{start} \tag{1}$$

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n + \Delta t * \boldsymbol{v}_n \quad 0 \leq n < N-1 \tag{2}$$

Eq. (1) and (2) represent the state of the UAV at each time step. For each time step n, the position in the next time step p_{n+1} is determined by the current position p_n, the current velocity vector v_n and the time step size Δt. Velocity, acceleration and other derivatives are represented the same way. The number of derivatives needed depends on the specific use case.

B. Objective function

The objective is to minimize the time before the UAV reaches the goal position.

$$\text{minimize} \quad -\sum_{n=0}^{N-1} done_n \tag{3}$$

Eq. (3) shows the objective function [2]. Reaching the goal causes a state transition from not being done to being done. This is represented as the value of binary variable $done_n$. When $done_n = 1$, the UAV has reached its goal on or before time step n.

$$done_0 = 0, \quad done_{N-1} = 1 \tag{4}$$

$$done_{n+1} = done_n \vee cdone_{n+1}, \quad 0 \leq n < N-1 \tag{5}$$

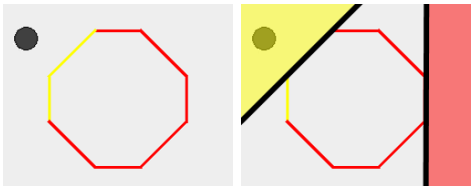
Eq. (4) states that the UAV must reach its goal eventually. Lamport's state transition axiom method [8] was used to model state transitions. In Eq. (5), the state will be done at time step $n+1$ if the state is done at time step n or if there is a state transition from not done to done at time step $n+1$, represented by $cdone_{n+1}$.

$$cdone_n = |x_n - x_{goal}| < \epsilon_p \wedge |y_n - y_{goal}| < \epsilon_p, 0 \leq n < N \tag{6}$$

The goal position requirement is fulfilled if the UAV is closer than ϵ_p to the goal position in both dimensions [2].

C. UAV state limits

Modeling the maximum velocity of a UAV requires calculating the the velocity vector's 2-norm which is non-linear. However, the maximum velocity constraint can be approximated to an arbitrary degree using multiple linear constraints [2]. The acceleration and other vector properties of the UAV can be limited in the same way.



(a) (b)

Fig. 1: A visual representation of how obstacle avoidance works. Fig. 1a shows the UAV's current position as the black circle. The color of the edges of the obstacle represent whether or not the UAV is in the safe zone for that edge. An edge is yellow if the UAV is in the safe zone, and red otherwise. Fig. 1b shows the safe zones defined by a yellow and red edge in yellow and red respectively.

D. Obstacle avoidance

The most challenging part of the problem is modeling obstacles. Any obstacle between the UAV and its goal will inherently make the search space non-convex. Because of this, integer variables are needed to model obstacles. Assuming that obstacles are convex polygons, for each obstacle, the UAV needs to be on the "safe" side of at least one edge to not collide with the obstacle (see Fig. 1). Indicator constraints were used to model obstacle avoidance. This requires one boolean variable $slack$ per edge. If $slack$ is false, the UAV is on the safe side of the edge. For each obstacle at least one of the $slack$ variables need to be false. For every convex obstacle o with the coordinates of vertex i being $x_{o,i}$ and $y_{o,i}$ [2]:

$$dx_{o,i} = x_{o,i} - x_{o,i-1}, \quad dy_{o,i} = y_{o,i} - y_{o,i-1}$$

$$a_{o,i} = \frac{dy_{o,i}}{dx_{o,i}}, \quad b_{o,i} = y_{o,i} - a_{o,i}x_{o,i} \quad 0 \leq i < N_{vertices}$$

$$slack_{o,i,n} \Rightarrow \begin{cases} b_{o,i} \leq p_{n,y} - a_{o,i}p_{n,x} & dx_{o,i} < 0 \\ b_{o,i} \geq p_{n,y} - a_{o,i}p_{n,x} & dx_{o,i} > 0 \end{cases} \tag{7}$$

$$\neg \bigwedge_i slack_{o,i,n} \quad 0 \leq n < N \tag{8}$$

Modeling obstacles this way is problematic because an integer variable is needed for every edge of every obstacle, for every time step. MILP scales exponentially with the amount of integer variables [9], so performance is mostly determined by the amount of obstacles and time steps.

III. Segmentation of the MILP problem

In this section we propose a preprocessing pipeline that makes the problem more scalable. Alg. 1 shows the outline of the algorithm.

First, we find an initial path with the Theta* algorithm (line 2). Unlike a trajectory, a path is not time-dependent and does not take dynamic properties into account. Then, we find all the turns in that path (line 3). After that, we generate path segments based on those turns (line 4). Each path segment contains the information needed to construct

Algorithm 1 General outline

1: $T \leftarrow \{\}$ ▷ The list of solved subtrajectories
2: $path \leftarrow$ THETA*($scenario$)
3: $events \leftarrow$ FINDTURNEVENTS($path$)
4: $segments \leftarrow$ GENSEGMENTS($path$, $events$)
5: **for each** $segment \in segments$ **do**
6: UPDATESTARTSTATE($segment$)
7: GENSAFEREGION($scenario$, $segment$)
8: GENSUBMILP($scenario$, $segment$)
9: $T \leftarrow T \cup \{$ SOLVESUBMILP $\}$
10: **end for**
11: $result \leftarrow$MERGETRAJECTORIES(T)

a MILP subproblem. Finally, for each segment, a MILP subproblem is constructed and solved (line 8-9). Before the MILP subproblem is solved, a heuristic selects several obstacles to be modeled in the problem. A genetic algorithm generates a convex safe region which is allowed to overlap those selected obstacles only (line 7). Because the UAV must stay within the safe region, it cannot collide with obstacles. For all but the first segment, the starting state for the UAV in the MILP problem is updated to match the final state of the UAV in the previous segment (line 6). Once all the segments have been solved, their trajectories are merged into the final result (line 11). Because the starting state in a segment is the same as the final state in the previous segments, the resulting trajectories can be appended to each other without further processing.

Our goal is to divide the problem into subproblems so that only a minimal number of obstacles need to be modeled in each subproblem, while still resulting in a relatively fast trajectory. Subproblems based on shorter segments with fewer obstacles are easier to solve, but the UAV will need to travel at a lower velocity. This is because there is no information available about the next segment. If the next segment contains a tight turn, the UAV may not be able to slow down enough if it is going too fast. Longer segments allow the UAV to travel faster, but they need more time to solve.

For the best results, we want to find segments which are as large as possible but contain as few obstacles as possible. By generating a segment for each turn in the Theta* path, we can make the segments just large enough so the UAV can always slow down in time to execute the turn. This way the UAV will always turn efficiently, without making the segments too large to solve in an acceptable amount of time.

An important downside of segmenting the problem is that our algorithm is unlikely to find the optimal trajectory.

A. Finding the initial path

The first step in Alg. 1 is finding the Theta* path (line 2), which will be used to divide the problem into segments. The MILP problem generated from each segment needs an intermediate goal to guide the UAV closer the final goal position.

We use Theta* [10] to find an initial path which connects the start and goal positions. Theta* is a variant of A* which eliminates the jagged paths associated with A* as demonstrated in Fig. 2. This path does not take any of the vehicle dynamics into account. Multirotor UAVs can hover, so the UAV can always follow this path by moving in straight lines and stopping at each node of the path. This ensures that successful navigation to the goal position is possible.

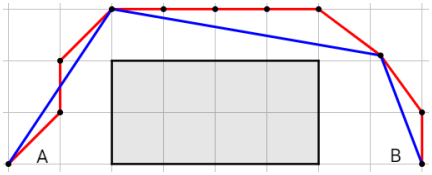


Fig. 2: A typical A* path in red compared to a Theta* path in blue. The gray rectangle is an obstacle.

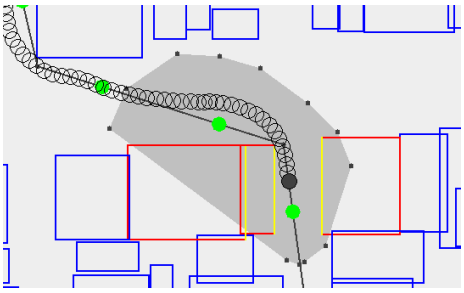


Fig. 3: The red/yellow shapes are the obstacles modeled in the MILP problem, using the same color scheme as in Fig. 1a. The blue shapes are the remaining obstacles. The green circles depict the transitions between segments. The dark gray shape is the convex safe area generated by the genetic algorithm. The solid black circle represents the current position of the UAV, with the hollow circles showing the position in previous time steps.

B. Detecting turn events

Because the shortest path between two points in Euclidian geometry is a straight line, any turn at a node in the Theta* path must have at least one obstacle on the inside of that turn. A turn without an obstacle on the inside can always replaced by a shorter, straight line segment. This means that by definition these ”inner” obstacles make the search space non-convex. A shape is convex if every point on the line between any two points inside the shape is also inside the shape. Because moving in a straight line between two valid positions on either side of the turn is not possible, the search space must be non-convex if turns in the Theta* path exist. Non-convexity of the search space is the main cause for the poor performance of Mixed Integer Programming [4]. We have shown that the turns in the path correspond with parts of the problem which must be non-convex. Because of this, we have chosen to generate the segments such that there is at most a single turn in each segment. This limits the non-convexity for each subproblem, significantly improving

execution time.
In some cases, a Theta* path contains multiple nodes for a single turn. Alg. 2 groups those nodes together into turn events.

Algorithm 2 Finding Turn Events

```
1:  function FindTurnEvents(path)
2:      Δmax ← max. acc. distance * turn tolerance
3:      events ← {}  ▷ The list of turn events found so far
4:      i ← 1  ▷ Skip the start node, it can't be a turn
5:      while i < |path| − 1 do  ▷ Skip the goal node
6:          event ← {path(i)}  ▷ Start new turn event
7:          turnDir ← TurnDir(path(i))
8:          i ← i + 1
9:          while i < |path| − 1 do
10:             if ||path(i − 1) − path(i)|| > Δmax then
11:                 break  ▷ Node is too far from previous
12:             end if
13:             if TurnDir(path(i)) ≠ turnDir then
14:                 break  ▷ Node turns in wrong direction
15:             end if
16:             event ← event ∪ {path(i)}  ▷ Add to event
17:             i ← i + 1
18:         end while
19:         events ← events ∪ {event}
20:     end while
21:     return events
22: end function
```

In the Theta* path, all but the first and last nodes are turns in the path. The second node is always the first node in a new turn event (line 5). Subsequent nodes in the path which are not too far away from the previous node (line 9) and turn in the same direction (line 12) are added to the current turn event (line 15). The maximum distance between nodes in the same turn depends on the maximum acceleration distance of the UAV and a turn tolerance parameter (line 2). The maximum acceleration distance is the distance the UAV needs to accelerate from zero to its maximum velocity, or slow down from the maximum velocity to zero. Once a node is found which does not belong in the event, the event is stored (line 18), a new event is created for that node (line 5) and the process repeats until no more nodes are left.

C. Generating path segments

Each path segment contains the information needed to construct a MILP subproblem. Alg. 3 constructs the segments using turn events. Each segment needs to be large enough so the UAV can safely approach and exit each turn. A multirotor UAV can always safely navigate a turn if it can come to a complete stop before the turn. That is satisfied if the segment starts at least the maximum acceleration distance away from the turn. If the segment starts even earlier, the UAV has more space to maneuver and can navigate the turn more efficiently. However, as the segment gets larger, so does the difficulty of the segment. The approach margin multiplier determines the expansion distance around turn events, based on the maximum acceleration distance.

Algorithm 3 Generating the segments

```
1:  function GenSegments(path, events)
2:      segments ← {}
3:      catchUp ← true
4:      lastEnd ← path(0)
5:      for i ← 0, |events| − 1 do
6:          event ← events(i)
7:          if catchUp then
8:              expand event.start backwards
9:              add segments from lastEnd to event.start
10:             lastEnd ← event.start
11:         end if
12:         nextEvent ← events(i + 1)
13:         if nextEvent.start is close to event.end then
14:             mid ← middle between event & nextEvent
15:             add segment from lastEnd to mid
16:             lastEnd ← mid
17:             catchUp ← false
18:         else
19:             expand event.end forwards
20:             add segment from lastEnd to event.end
21:             lastEnd ← event.end
22:             catchUp ← true
23:         end if
24:     end for
25:     add segments from lastEnd to path(|path| − 1)
26:     return segments
27: end function
```

To generate the segments, Alg. 3 considers each turn event in turn. It keeps track of the end point of the last segment it generated with $lastEnd$. When constructing a segment, it considers the distance between the end of the current turn event and the start of the next turn event. On line 13, two events are too close to each other if they are separated by less than three times [1] the expansion distance. In that case, the segment is constructed to end in the middle between the current and next event (line 14-16). If the events are far enough apart, the end of the current event is expanded forwards by the full expansion distance(line 19-21). Because the next turn event is a long distance away, the $catchUp$ flag is set to true (line 22), ensuring that one or more segments are added to catch up to the start of the next event (line 7-10). To limit the size of segments, they can be no longer than the distance the UAV can travel at maximum velocity in T_{max} time.

D. Generating the safe region for each segment

The last step of preprocessing determines which obstacles will be modeled in the MILP subproblem for each segment. Not all obstacles need to be modeled in the MILP problem

[1] Requiring three (instead of two) times the expansion distance as separation between turn events ensures that the segment between those turns is also at least as long as the expansion distance. This prevents some issues that can occur with very short segments.

Algorithm 4 Genetic Algorithm

```
1: function GenSafeRegion(scenario, segment)
2:     pop ← SeedPopulation
3:     for i ← 0, N_gens do
4:         pop ← pop∪ Mutate(pop)
5:         Evaluate(pop)
6:         pop ← Select(pop)
7:     end for
8:     return BestIndividual(pop)
9: end function
10: function Mutate(pop)
11:     for each individual ∈ pop do
12:         add vertex with prob. P(add vertex)
13:         OR remove vertex with prob. P(remove vertex)
14:         for each gene ∈ individual.chromosome do
15:             randomly nudge vertex
16:             if new polygon is legal then
17:                 update polygon
18:             else
19:                 try again at most N_attempts times
20:             end if
21:         end for
22:     end for
23:     return BestIndividual(pop)
24: end function
```

to prevent collisions. Obstacles on the inside of turns in the Theta* path "cause" those turns and make the search space non-convex. However, collisions with obstacles which are further away or on the outside of the turn can be avoided without the reducing the convexity of the search space.

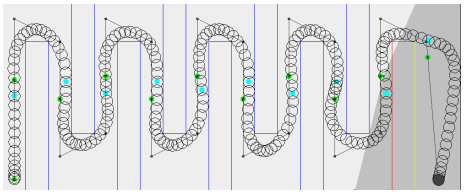
We select the obstacles to be modeled in the MILP by constructing the convex hull of the start and goal positions of the segment, as well as all Theta* path nodes between them. Any obstacle which overlaps this convex hull will be modeled.

The convex hull can be considered a safe region. If the UAV stays inside this region, it cannot collide with obstacles since any obstacle that overlaps with the safe region is modeled in the MILP problem. However, this safe region restricts the movements of the UAV more than necessary.

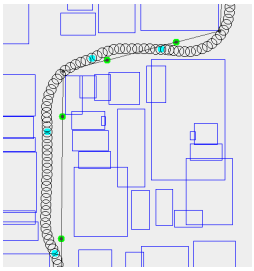
To make the safe region less restrictive, we use a genetic algorithm which attempts to grow it. We use a genetic algorithm because it can provide acceptable solutions with a minimum of development effort. In our implementation (Alg. 4), each individual in the population represents a single legal polygon. A legal polygon is convex, does not self-intersect, can only overlap with the selected obstacles and contains every node in the Theta* path for that specific segment. The latter requirement prevents the polygon from drifting off. Each individual has a single chromosome, and each chromosome has a varying number of genes. Each gene represents a vertex of the polygon.

The only operator is a mutator (line 4). Contrary to how mutators usually work, the mutation does not change the original individual. This means that the every individual can be mutated in every generation, since there is no risk of losing information. This mutator can add or remove vertices of the polygon by adding or removing genes (lines 12-13), but only if the amount of genes stays between a specified minimum and maximum. The mutator attempts to "nudge" every vertex/gene to a random position inside a circle around the current position (line 15). If the resulting polygon is not legal, it retries a limited number of times (line 16-19).

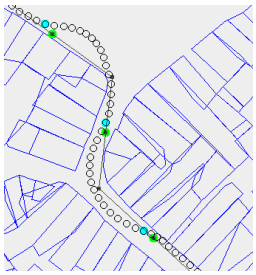
Tournament selection is used as the selector, with the fitness function being the surface area of the polygon (line 5-6). Fig. 2 Shows the obstacles modeled in the MILP problem in yellow and red, as well as the polygon generated by the genetic algorithm in dark gray.



(a) Up/Down Scenario



(b) A small part of the San Francisco Scenario



(c) A small part of the Leuven Scenario

Fig. 4

IV. Results

We test our algorithm in several different scenarios. Each scenario was tested with two different problem sizes. All tests were executed on an Intel Core i5-4690K running at 4.4GHz with 16GB of 1600MHz DDR3 memory. The reported times are averages of 5 runs. The machine runs on Windows 10 using version 12.6 of IBM CPLEX. Fig. 4 shows these scenarios visually. Table I shows a table with detailed information about the scenarios and execution times. Table II shows the parameters used in the execution of the algorithm.

A. Up/Down Scenario

The first test scenario has very few obstacles, but lays them out in a way such that the UAV needs to slalom around them. The small scenario has only 5 obstacles, while the larger one has 9. Fig. 4a shows the large variant. This is a challenging scenario for MILP because every obstacle causes a turn, making the problem significantly less convex. Without segmentation on the small version of the scenario, the solver

scenario	#obstacles	world size	path length	#segments	Theta* time	GA time	MILP time	score
Up/Down Small	5	25m x 20m	88m	7	0.09s	1.10s	20.8s	26.6s
Up/Down Large	9	40m x 20m	146m	11	0.14s	1.62s	40.1s	43.6s
SF Small	684	1km x 1km	1392m	28	2.04s	9.56s	59.2s	105.7s
SF Large	6580	3km x 3km	4325m	84	18.14s	18.21s	231s	316.0s
Leuven Small	3079	1km x 1km	1312m	29	2.29s	29.83s	152s	95.9s
Leuven Large	18876	3km x 3km	3041m	61	18.74s	83.69s	687s	217.6s

TABLE I: The experimental results for the different scenarios

grid size	$2m$	turn tolerance	2
approach multiplier	2	population size	10
# generations	25	max. nudge distance	$5m$
min. # vertices	4	max. # vertices	12
P(add vertex)	0.1	P(remove vertex)	0.1
max nudge attempts	15	T_{max}	$5s$
time step size	$0.2s$		

TABLE II: The parameters used for testing

does not find the optimal trajectory within 30 minutes. If execution is limited to 10 minutes, the best trajectory it finds takes 26.0s to execute by the UAV. That is less than a second faster than the segmented result while it took more than 20 times more execution time to find that trajectory. For the larger scenario with 9 obstacles, the solver could not find a trajectory within 30 minutes. This scenario clearly shows the advantages of segmentation, even if there only are a few obstacles.

B. San Francisco Scenario

The San Francisco scenario covers a 1km by 1km section of the city for the small scenario, and 3km by 3km section for the large scenario. Fig. 4b shows the small variant. All the obstacles in this scenario are grid-aligned rectangles laid out in typical city blocks. Because of this, density of obstacles is predictable. This scenario showcases that the algorithm can scale to realistic scenarios with much more obstacles than is typically possible with a MIP approach.

C. Leuven Scenario

The Leuven scenario also covers both a 1km by 1km and 3km by 3km section, this time of the Belgian city of Leuven. This is an old city with a very irregular layout. The dataset, provided by the local government[2], also contains full polygons instead of the grid-aligned rectangles of the San Francisco dataset. While most buildings in the city are low enough so a UAV could fly over, it presents a very difficult test case for the trajectory planning algorithm. The density of obstacles varies greatly and is on average much higher than in the San Francisco dataset. The algorithm does slow down compared to the San Francisco dataset, but still runs in an acceptable amount of time. As visible in Fig. 4c, there are many obstacles clustered close to each other, with many edges being completely redundant. For a real application, a small amount of preprocessing of the map data should be able to significantly reduce both the amount of obstacles as the amount of edges.

[2] https://overheid.vlaanderen.be/producten-diensten/basiskaart-vlaanderen-grb

V. Conclusion

Trajectory planning using MIP was previously not computationally possible in large and complex environments. The approach presented in this paper shows that these limitations can effectively be circumvented by dividing the path into smaller segments using several steps of preprocessing. The final trajectory is generated by a solver so the constraints on the trajectory can easily be changed to account for different use cases. The experimental results show that the algorithm works well in realistic, city-scale scenarios, even when obstacles are distributed irregularly and dense.
We demonstrate that our new approach can be used to improve the scalability of MILP trajectory planning. However, more work is required to use the algorithm with an actual UAV. Extending the algorithm to 3D is the next step. We expect the extension to 3D to bring more performance challenges due to the higher dimensionality of the solution space. A complimentary, short-term online planner is necessary for a physical UAV to execute the generated trajectory.

References

[1] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Control Conference (ECC), 2001 European*, pp. 2603–2608, IEEE, 2001.

[2] J. S. Bellingham, *Coordination and control of uav fleets using mixed-integer linear programming*. PhD thesis, Massachusetts Institute of Technology, 2002.

[3] M. E. Flores, *Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational B-spline basis functions*. PhD thesis, California Institute of Technology, 2007.

[4] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 42–49, IEEE, 2015.

[5] Y. Hao, A. Davari, and A. Manesh, "Differential flatness-based trajectory planning for multiple unmanned aerial vehicles using mixed-integer linear programming," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 104–109, IEEE, 2005.

[6] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor uav," in *Control Conference (ECC), 2007 European*, pp. 4001–4008, IEEE, 2007.

[7] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2520–2525, IEEE, 2011.

[8] L. Lamport, "A simple approach to specifying concurrent systems," *Communications of the ACM*, vol. 32, no. 1, pp. 32–45, 1989.

[9] R. M. Karp, "Reducibility among combinatorial problems," in *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pp. 85–103, 1972.

[10] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.

Probabilistic Modeling of Gas Diffusion with Partial Differential Equations for Multi-Robot Exploration and Gas Source Localization

Thomas Wiedemann[1], Christoph Manss[1], Dmitriy Shutin[1], Achim J. Lilienthal[2], Valentina Karolj[1], Alberto Viseras[1]

***Abstract*— Employing automated robots for sampling gas distributions and for localizing gas sources is beneficial since it avoids hazards for a human operator. This paper addresses the problem of exploring a gas diffusion process using a multi-agent system consisting of several mobile sensing robots. The diffusion process is modeled using a partial differential equation (PDE). It is assumed that the diffusion process is driven by only a few spatial sources at unknown locations with unknown intensity. The goal of the multi-robot exploration is thus to identify source parameters, in particular, their number, locations and magnitudes. Therefore, this paper develops a probabilistic approach towards PDE identification under sparsity constraint using factor graphs and a message passing algorithm. Moreover, the message passing schemes permits efficient distributed implementation of the algorithm. This brings significant advantages with respect to scalability, computational complexity and robustness of the proposed exploration algorithm. Based on the derived probabilistic model, an exploration strategy to guide the mobile agents in real time to more informative sampling locations is proposed. Hardware-in-the-loop experiments with real mobile robots show that the proposed exploration approach accelerates the identification of the source parameters and outperforms systematic sampling.**

***Index Terms* — multi-agent exploration, gas source localization, mobile robot olfaction partial differential equation, factor graph, sparse Bayesian learning, message passing**

I. Introduction

Robotic platforms are particularly suitable for exploration of dynamic gas distributions. In our understanding and in the context of this paper, such an exploration task covers issues like mapping of gas distributions and identification of gas sources. Specifically, this paper examines the task of finding gas sources, e.g. leaks. While gas distribution maps appear as a byproduct of our approach, gas distribution mapping is not in the main focus of this paper. In searching for the sources we exploit measurements of the gas concentration in the environment. As an application one may think of a technical accident or disaster response scenario, where toxic material is leaking and the location has to be identified. Such a scenario implies threats for a human operator and indicates the benefits of an automated sampling of the gas concentration. Therefore, we consider a robotic platform equipped with a gas sensor for exploring gas distributions.

[1]Institute of Communications and Navigation of the German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany, thomas.wiedemann@dlr.de, christoph.manss@dlr.de, dmitriy.shutin@dlr.de, valentina.karolj@dlr.de, alberto.viserasruiz@dlr.de

[2]Mobile Robotics and Olfaction Lab, Orebro University, 70182 Orebro, Sweden, achim.lilienthal@oru.se

Employing robotic platforms for chemical sensing and gas source localization is an active and emerging research field (see [1] for a survey). Where a single robot would be able to only serially sample the gas concentration at different times, a multi-robot system is capable of taking measurements at different locations at the same time. This is of advantage for analyzing the dispersion process, since the gas dispersion is dynamic, and the time-variant nature of gas dispersion is an important property, that should be taken into account [2]. Besides, a multi-robot system has additional advantages: i) multiple robots can achieve the exploration task faster; ii) a multi-robot system is more robust, since it possesses natural redundancies; and finally iii) the individual robots can make use of synergies, e.g. share the computational costs of algorithms. Even though it seems reasonable to employ a multi-agent system, an open issue is the design of an automated exploration strategy. This exploration strategy should guide a single or a swarm of robots to the sampling locations in an efficient way and should avoid unnecessary measurements. The development and analysis of such an intelligent distributed exploration algorithm to identify the sources of the gas dispersion is the main focus of this paper.

The exploration or sampling problem is closely related to optimal sensor placement techniques. In literature, some approaches consider this as an observer design problem. In those cases an observer performance is optimized by adapting the sensor location, e.g. in [3] to estimate a distributed process described by a partial differential equation. Another approach is a probabilistic treatment of the sampling process. In this context there are different criteria to measure the informativeness of the samples [4]. In literature, this is also referred to as an optimal experimental design problem [5]. More details on these topics can be found in [6], where the author gives an introduction to optimal sensor location and experimental design problems. A probabilistic or statistical view is also a contemporary and relevant research topic in the context of gas distribution mapping [7], [8]. Further, the benefit of using statistical properties of the gas distribution in an adaptive sampling strategy has already been shown in first real-world experiments [9].

The contribution of this paper is twofold: First, we propose a new adaptive model-based exploration strategy for multiple mobile robots. Second, we present an approach to evaluate this exploration strategy within a hardware-in-the-loop experiment. For the exploration we are following the ideas of an uncertainty or entropy driven exploration [10]. This means that the sampling scheme of the robots prefers regions

with high uncertainty. In other words, if the system has currently little knowledge about a region in the environment, it will guide robots to this location in order to carry out the next measurements. The approach is similar to our previous work [11]. However, in this paper robots possess a global view of the whole environment in contrast to the greedy algorithm in [11] where the robots only consider their direct neighborhood for a new measurement, Still, it is necessary to be able to quantify the uncertainty of our model parameter in different regions in the considered environment. To this end, we propose to make use of a model of the gas diffusion in the environment. In particular, since the gas diffusion is time and space variant, we use a partial differential equation (PDE) as a mathematical model. For the evaluation of the proposed multi-robot exploration, we developed a hardware-in-the-loop experimental setup. In general, it is difficult to evaluate gas distribution exploration in realistic experiments due to the difficulty in measuring ground truth gas concentrations. In addition, interesting gas distribution for realistic application may be of toxic nature and dangerous to handle. Therefore, we simulate the gas dispersion in the robots environment. In contrast to the gas dispersion, the robots themselves are not simulated. We employ a real multi-robot system to test our system and algorithm with "real world" constraints. In this way we are able to analyze how our distributed algorithm cope with realistic communication constraints like data rate, package delay or jitter. Further, limitation arising from the robotic platform, e.g. speed limits, collision avoidance and low-power on board computers, can be studied.

The outline of the paper is as follows: First, in Section II we explain the exploration strategy in detail and how it is implemented in a distributed fashion. In Section III we describe the experimental setup of the hardware-in-the-loop experiments and evaluates the performance of the proposed exploration strategy.

II. Exploration Strategy

This chapter describes the design of the exploration strategy. We first introduce the gas diffusion model, which is used in a second step to quantify the uncertainty in different regions.

A. Environment Model

As a mathematical model for the gas dispersion we use the 2D diffusion equation which can be described with the linear parabolic partial differential equation (PDE) [12]:

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t} - \kappa \Delta f(\boldsymbol{x},t) = u(\boldsymbol{x},t). \tag{1}$$

This equation models the dynamic behavior of gas concentrations f at a location $\boldsymbol{x}$ and time t given the diffusion coefficient κ. The right hand side $u(\boldsymbol{x},t)$ represents the inflow of material, i.e. the spatial source strength distribution we want to identify. In a first step the PDE is numerically approximated via the Finite Difference Method (FDM) [13]. Therefore, we discretize space and time by dividing our region into grid cells and considering temporal evolution of gas concentration in each cell at discrete time steps n. Since we used the FDM, our numerical approximation results in a linear equation r_c for each grid cell c:

$$r_c(\boldsymbol{f}[n], \boldsymbol{f}[n-1], u_c[n]) = 0. \tag{2}$$

The FDM treats the concentration and source strength across a cell as constant. The 1D vectors $\boldsymbol{f}[n]$ and $\boldsymbol{u}[n]$ aggregate the 2D concentration field and 2D source strength distribution for all grid cells at time stamp n. Both are unknown and have to be estimated based on measurements. This is the purpose of the exploration. A measurement done by a robot will be modeled as:

$$y_k[n] = \boldsymbol{M}[n]\boldsymbol{f}[n] + \xi, \tag{3}$$

with $\boldsymbol{M}[n]$ being a measurement matrix selecting the cell from $\boldsymbol{f}[n]$ that is currently visited by the robot. Additionally, ξ represents a measurement noise mostly defined by the sensor characteristics.

B. Uncertainty Quantification

For the uncertainty driven exploration procedure, we need a way to distinguish two cells with respect to their uncertainty. In order to quantify the uncertainty, we transform our model to a probabilistic representation. In the probabilistic framework all variables are treated as random variables and could be described by probability density functions (PDF). This permits us computing second order moments of the PDFs, e.g. the variance of a Gaussian distribution. The second order moments can be interpreted as measure of entropy or informativeness of a parameter. It is the second order moment of the source strength we propose to use as an uncertainty quantification of a cell.

For the probabilistic formulation, the linear equations of (2) are relaxed. Thereby we assume that the equations hold only with a certain precision τ_s, with some random deviations from zero. We assume that those derivations or residuals in all grid cells are spatially and temporally white and statistically independent from each other. The precision τ_s reflects some uncertainty of our model assumptions. In this way the conditional probabilistic density function for all concentrations and sources can be formulated as follows:

$$p(\boldsymbol{f}[n]|\boldsymbol{f}[n-1], \boldsymbol{u}[n]) \propto \prod_{c=1}^{C} \mathrm{e}^{-\frac{\tau_s}{2}(r_c(\boldsymbol{f}_c[n], \boldsymbol{f}_c[n-1], u_c[n]))^2}. \tag{4}$$

The measurement model (3) could be transformed to

$$p(\boldsymbol{y}[n]|\boldsymbol{f}[n]) \propto \prod_{k=1}^{K} \mathrm{e}^{-\frac{\tau_m}{2}(\boldsymbol{M}[n]\boldsymbol{f}[n] - y_k[n])^2}, \tag{5}$$

where the sensor noise is modeled as white with a variance $1/\tau_m$.

In our approach we do not need any assumption regarding the exact number of sources, their position or strength. We only utilize the prior knowledge that sources are sparsely distributed in the considered environment. In other words, we do not know the exact number of sources, but there are

only few of them and their number is of the same order of magnitude as the number of robots. In the probabilistic setting we are able to introduce prior knowledge according to the Bayes theorem. Here, we make use of a sparse prior and Sparse Bayesian Learning techniques [14] (SBL). SBL applies a hierarchical prior, where the actual prior itself is parametrized by an additionally introduced hyper-parameter $\boldsymbol{\gamma}[n]$ as

$$p(u_c[n]\big|\gamma_c[n]) = N(u_c[n]|0, \gamma_c^{-1}[n]). \tag{6}$$

By the combination of this prior and a hyper-prior $p(\gamma_c[n]) \propto 1/\gamma_c[n]$ the Automatic Relevance Determination (ARD) version of SBL [15] leads to emergence of sparsity in the Bayesian context of $\boldsymbol{u}[n]$. For a more descriptive interpretation, the prior can be compared to a regularization method. Whenever no information is available in a grid cell, the zero centered Gaussian prior will drive the source strength to zero.

Putting all together according to the Bayes Theorem the posterior PDF of our problem becomes:

$$\begin{aligned} &p(\boldsymbol{f}[0]...\boldsymbol{f}[N], \boldsymbol{u}[0]...\boldsymbol{u}[N], \boldsymbol{\gamma}[0]...\boldsymbol{\gamma}[N]|\boldsymbol{y}[0]...\boldsymbol{y}[N]) = \\ &\prod_{n=0}^{N} p(\boldsymbol{y}[n]|\boldsymbol{f}[n]) p(\boldsymbol{f}[n]|\boldsymbol{f}[n-1], \boldsymbol{u}[n]) \\ &\prod_{c=1}^{C} p(u_c[n]\big|\gamma_c[n]) \prod_{c=1}^{C} p(\gamma_c[n]). \end{aligned} \tag{7}$$

In order to evaluate the source strength in a single grid cell, it is possible to calculate the marginal PDF based on the posterior. Generally, the marginal results from integrating over all other variables and parameters. In contrast, in this paper we use a distributed algorithm that enables the multi-robot system to cooperatively calculate all marginal distributions. The marginal PDF for all $u_c[n]$ can be approximated by a Gaussian distribution. So we can use the variance of the PDF $p(u_c[n])$, i.e. the second order central moment, as a gauge for the uncertainty of cell c. More precise: the higher the variance the higher the uncertainty regarding the estimated source strength in a cell.

C. Distributed Implementation

Let us consider how to calculate the marginal PDF of all cells based on the probabilistic formulation of the posterior, since these marginals are the foundation of the uncertainty quantification. Further, we want to calculate them in a distributed fashion in order to take advantage of the multi-agent system. Therefore, in a first step we introduce a graphical representation of our posterior PDF. This representation is used in a second step, where we apply a message passing algorithm to calculate the marginals in a distributed fashion.

1) Factor Graph Representation: An interesting property of our posterior PDF (7) is the fact that the function is nicely factorized. There are three factors for each cell and one factor for each measurement. A factorized function can be graphically represented with a Factor Graph (FG) [16]. A FG is an undirected bipartite Bayesian network being composed of value nodes, which represent random variables, and factor nodes, which model functional dependencies between them. (See [17] for a more detailed introduction to FG.) In our case the FG models the relationships between the different random variables $f_c[n]$, $u_c[n]$ and $\gamma_c[n]$. Figure 1 shows the part of the overall graph for one cell and one time instance.

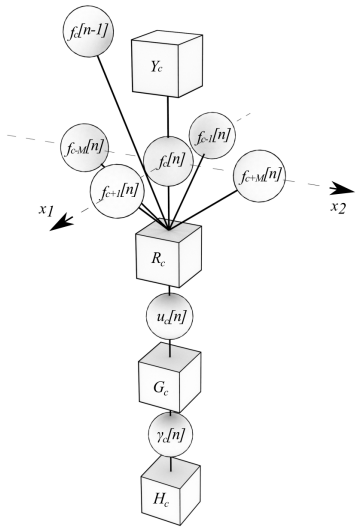


Fig. 1: Factor Graph: This graph represents the part of the posterior PDF associated with a single grid cell. It models the relations between variable nodes (spheres) by factor nodes (cubes).

The node Y_c represents a measurement and it only exists if a measurement was taken in the corresponding cell. The node R_c puts the neighboring concentration values of the grid cell into a relation with the source strength. This relationship arises from our PDE and the numerical approximation. The nodes G_c and H_c model the factors of the posterior PDF that correspond to the parametric prior $p(u_c[n]\big|\gamma_c[n])$ and, respectively, the hyper-prior $p(\gamma_c[n])$.

2) Message Passing Algorithm: The FG is in general the foundation of message passing algorithms. In the probabilistic context, these algorithms are a powerful tool to calculate marginal distributions. Mostly, it is used in coding theory for error detection and correction. In this paper we make use of them to calculate the marginal distribution of our posterior PDF (7). More precisely, we make use of the sum product algorithm [18] (also called loopy belief propagation) and variational message passing [19]. Messages are sent between nodes of the factor graph along the edges. There are two possible types of messages: messages from factor nodes to variable nodes and vice versa. The sum product algorithm provides update rules according to which the messages are calculated. In general the outgoing messages of a node are functions of incoming messages (For the specific calculation rules see [17], [19]). By iteratively exchanging messages between nodes, the outgoing messages of variable nodes converge to the marginal distribution of the corresponding variable or parameter. For messages corresponding to our hierarchical prior in the graph, analytical tractability is not given. However, for this part of the graph we use variational

message passing (VMP) [19] to circumvent the issue by analytical approximation techniques. The messages themselves represent beliefs, i.e. probabilistic distributions. Since we designed our posterior PDF properly and thanks to the VMP, all messages stay in the same class of distribution. E.g. the outgoing message of a node $u_c[n]$ is always Gaussian distributed and therefore could be parametrized by only two values: mean and variance. We stress that the overall factor graph contains a lot of edges and a lot of message-updates have to be calculated. But all updates can be calculated in closed-form and correspond to relatively simple calculations.

Moreover, the main advantage of the message passing algorithms is the fact that it could be easily implemented in a distributed fashion. Even so many messages have to be calculated, they can be updated in random order or parallel. Thus, the framework is very suitable for a distributed implementation.

Actually, we divide the overall factor graph in different parts. A simplified version of the overall factor graph and the partitioning of this graph are shown in Fig. 2. The different parts correspond to different 2D region of our environment. Each region is assigned to one robot of our multi-agent system and each robot is able to calculate all messages of its own part of the graph. The robot only has to exchange the messages along the border of its partition (red arrows) with its neighbors. By iteratively calculating and exchanging messages the outgoing messages of $u_c[n]$ converges to the marginal PDF. Based on the variances each robot proposes a certain number of relevant points in the region it is responsible for to all other robots.

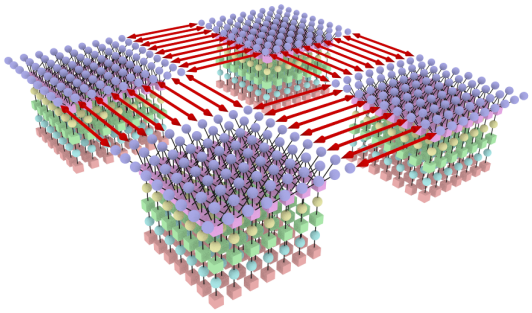


Fig. 2: Distributed Factor Graph: This figure illustrates the overall factor graph. This is a simplified version without measurement nodes and time dependencies. The graph is spatially split into four regions, where the big arrows represent messages sent between different regions.

D. Exploration Procedure

Our exploration strategy utilizes the uncertainty quantification of the estimated source strength for each cell $c \in C$ with C being the set of all cells in our environment. The cells are rated according to the inverse variance, i.e. precision $\tau_c[n]$, of the source strength marginal $p(u_c[n]) \propto N(\hat{u}_c[n], \tau_c[n])$. A set P of cells with the lowest precisions - i.e. with the highest uncertainties - serves as a proposal for new way points for the robots. Therefore, each robot a proposes K cells from the region $C_a \subseteq C$ of the environment it is responsible for:

$$\tau_{c_1} \leq \tau_{c_2} \leq ... \leq \tau_{c_K} \leq \tau_{c_{(K+1)}} \leq ...; \tau_{c_i} \in C_a$$

$$P_a = \bigcup_{i=1}^{K} c_i \tag{8}$$

The K most uncertain points of each region are combined to the set $P = \{c \in \bigcup P_a\}$ for the whole environment. From this set P each robot selects one way point. We like to emphasize that therefore the movement of a robot is not limited to the region it is responsible for with respect to the calculations. Actually, the robot can choose from the whole set P. Their choice is based on two criteria: i) Each robot adds to the precision of a proposed way point a penalty based on its distance to the way point and selects the one with the lowest value. In this way the robots not only favor points with high uncertainty but also those which are close-by. ii) If another robot is already on its way to a point, this point is neglected. Therefore it is necessary that a robot communicates its decision to all other robots.

After the decision is made, a robot moves to the selected way point. To avoid collisions with other robots, we implemented a reactive collision avoidance mechanism. On its way, the robot monitors the distance to all other robots based on received position information. If the distance drops below a defined safety threshold, the robot stops and selects another way point from the proposals that would increase the critical distance. Finally, when a robot reaches its goal, it takes a measurement which is incorporated in the probabilistic solution of the PDE.

The overall procedure is depicted in Fig. 3. As can be seen, the robotic navigation and solving the PDE are actually two separate loops. The loops are connected by the data exchange of proposed way points and measurements. We would like to stress that in this way the navigation part of each robot could be realized in an asynchronous fashion, where no robot has to wait for results of others.

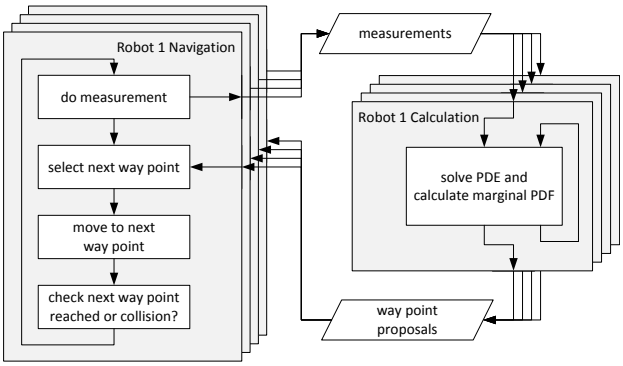


Fig. 3: Exploration Procedure: The exploration is implemented with two main loops. The right loop solves the PDE and produces new way point proposals, the left one controls the individual robots.

III. Evaluation

In order to evaluate our approach we designed a hardware-in-the-loop experiment. This means that we use a real robotic system but we simulate the gas dispersion taking place in the environment of the robots.

A. System Setup

We developed a small robot for our experiments. The main part is a Raspberry Pi 2, a low power single-board computer with Linux OS (900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM). From this computer it is possible to send velocity commands to a micro-controller that implements a velocity controller for two motors driving the tracks of the robot. The experiments were done in a laboratory with the commercial optical tracking system. This system is able to track active infra-red LEDs on the robot and provides the current position and orientation of each robot with high accuracy. On the robot's computer a position controller is implemented that compares the actual pose from the tracking system and the nominal pose demanded by the exploration strategy. We employ five robots. Their computers are connected to an 802.11 wireless LAN communication system. By this, they get their own position data and are able to exchange messages among each other. For the software implementation and the inter-process communication we make use of the Robot Operating System (ROS) [1].

The gas diffusion is simulated for a two dimensional case. The data are generated according to equation (1). Whenever a measurement is demanded by the exploration procedure for one robot, this equation is evaluated at the current position of the robot. Additionally, we disturb the measurement by the additive white noise ξ. For the evaluation of the PDE a Finite Volume Method solver [20] is used. At the boundary we have chosen a Dirichlet boundary condition $f(x,t) = 0$, except for the right border, where we use a Neumann boundary condition $\frac{\partial f(x,t)}{\partial x} = 0$. For the virtual gas simulation we considered the concentration and source strength unit-less. The discrete grid size, the time difference between two discrete time stamps and the diffusion coefficient κ are set to 1 in the simulation. However, later on the concentration field is fitted to our laboratory with a scale of $6m$ times $2.4m$.

B. Results

We evaluate the proposed exploration strategy in comparison against exploration with a predefined sweeping trajectory. The sweeping trajectories are generated by dividing the environment into five equal regions and fitting a meander into each region. In this way the measurements will fully cover the whole environment after a certain time, i.e. each grid cell is measured at least once. This strategy might be reasonable, if no prior knowledge or model assumptions are available. We compare the performance by means of speed and quality of the estimates. Regarding speed we consider the number of measurements needed to converge. In order to see how well the spatially distributed sources $\boldsymbol{u}[n]$ are identified,

[1] http://www.ros.org/

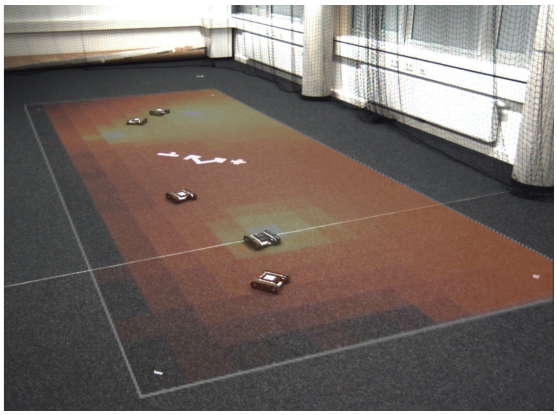


Fig. 4: Lab Environment: The picture shows our lab during an experiment. The simulated concentration field is projected to the ground in a post-processing step. A video of an experiment can be found here: https://youtu.be/UJYwdzrDTL4

we use the so-called earth mover's distance (EMD) measure, which is an analog of a Wasserstein metric for discrete distributions [21]. The EMD measures the effort to "displace" one distribution onto another one and is particularly useful for comparing sparse functions. Specifically, we compare the estimated vector $\boldsymbol{u}[n]$ with the ground truth vector $\hat{\boldsymbol{u}}$ with all elements set zero except for the three cells containing a source ($\hat{u}_c = 1.0$ at $x = -0.2m, y = -1.8m$; $\hat{u}_c = 1.0$ at $x = -0.6m, y = 1.0m$; $\hat{u}_c = 0.8$ at $x = 0.4m, y = 1.6m$).

The results are shown in Fig. 5 with the aid of an example experiment. Fig. 5a and 5b visualize the trajectories of the meander and the proposed exploration strategy. The proposed exploration strategy is adaptive. In other words it reacts to measurements. Therefore, the trajectory is not deterministic and not predictable. The trajectory in Fig. 5b is an example and will look different in another simulation run, because of different initial conditions and randomness caused by measurement noise. The trajectories are plotted as an overlay above the simulated concentration field at the time, when the sources are correctly identified. From the figure it gets obvious that this distribution is driven by three sources located at the concentration peaks. Fig. 5c depicts the performance of both strategies. The curves plot the difference between the estimated source distribution $\boldsymbol{u}[n]$ and the true source distribution $\hat{\boldsymbol{u}}$ in relation to the number of collected measurements.

With the meander trajectory the multi-agent system was able to identify the source distribution after approximately 340 measurements in our example. This indicates the step in the convergence curve towards zero in Fig. 5c. In general, the performance of the meander highly depends on the position of the sources. If they are already covered at the beginning of the trajectory, of course fewer measurements are needed. However, to be conservative the worst case has to be considered and this means a full coverage of the region. In our example, 360 measurements would be necessary for that. In contrast, the curve for the proposed exploration strategy converges with only 230 measurements in Fig. 5c. This indicates that robots were able to identify

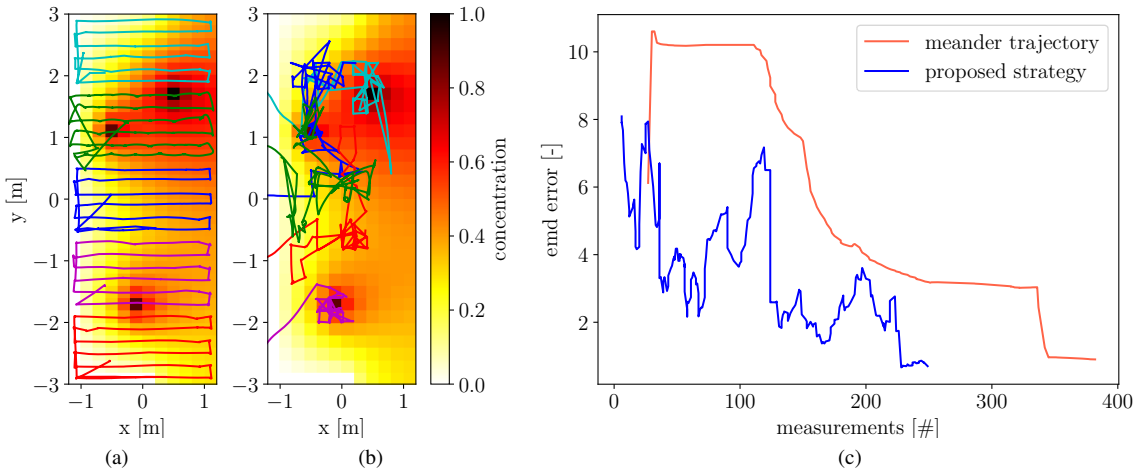


Fig. 5: Results: The figure compares the performance of the meander trajectory (a) and the proposed exploration strategy (b). The trajectories superimpose the simulated concentration field. In (c) the error regarding the estimated source distributions are plotted for both cases. The proposed exploration is able to identify the sources with less measurement than the meander.

the sources with fewer measurements. As can be seen from the trajectory in Fig. 5b the measurements are concentrated around the source locations. These measurements contain more information regarding the sources. This is the reason for a better performance.

Additionally, the hardware-in-the-loop experiments enable us to analyze other performance indicators of our algorithm and system properties. For example we can measure the gross data rates containing all overheads caused by ROS, OS, TCP etc. Concretely, in our case a data rate of less than $70kBytes/s$ is required for a communication link between two robots. This provides us with the information to create specifications for a communication system for future real-world experiments. Similarly, we can investigate the processor load of the on-board computers caused by the algorithm. In particular, the on-board computers were able to generate way point proposals with a update-rate of $1.0Hz$.

IV. Discussion and Conclusion

The results of the experiments have shown that a model-based exploration is of advantage for sampling a gas diffusion process. By an intelligent exploration strategy, the number of required measurements to identify gas sources can be reduced. This property is favorable to applications, where a measurement is expensive or consumes a lot of time.

The potentials of the presented approaches arise from the uncertainty driven strategy for taking new measurements in combination with the assumption that the sources are sparsely distributed in the environment and their number is small. This assumption is encoded with a prior PDF that assumes a source to have zero amplitudes with unknown variance. Under this assumption each found source with non-zero posterior amplitude effectively "contradicts" this prior assumption; as such the uncertainties of the estimated sources in the corresponding regions grow. As a consequence, the robots concentrate their measurements on informative regions around the sources according to the proposed exploration strategy. Additionally, this implies a need to employ a multi-robot system with more agents as gas sources, since single robots may got stuck in a neighborhood of a source and do not discover new sources. Multiple rovers are able to effectively reduce the uncertainty around the sources by multiple simultaneous measurements, and individual robots can "escape" from the source location.

For future development of the method a key problem to be addressed is a mismatch between the used dispersion model and real gas dispersion under the influence of possible turbulence and advection mechanisms that are not explicitly represented with model (1). To account for these effects several tactics can be explored. In particular, advection-based diffusion can be incorporated directly into the PDE through a convection term. In the presented framework, this is easy to achieve, since only equation 1 and 2 have to be modified appropriately, with the rest of the approach left intact. Turbulence, on the other hand, can be treated as a random effect and accounted for in the probabilistic formulation with the model uncertainty τ_s in (4).

In summary, let us mention that realistic gas dispersion problems are quite complex dynamical processes. From a practical perspective having a very complex model that adequately represent reality might lead to computationally very complex inverse problems. Instead, simpler models, like the one used in this work, can be seen as a numerically feasible approximation, which can be estimated based on concentration measurements in an adequate (in the context of robotic exploration) amount of time.

Acknowledgment

This work has partly been supported within H2020-ICT by the European Commission under grant agreement number 645101 (SmokeBot)

REFERENCES

[1] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, Nov 2012.

[2] A. Marjovi and L. Marques, "Multi-robot odor distribution mapping in realistic time-variant conditions," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3720–3727, May 2014.

[3] M. A. Demetriou and D. Ucinski, "State Estimation of Spatially Distributed Processes Using Mobile Sensing Agents," *American Control Conference (ACC)*, pp. 1770–1776, 2011.

[4] D. J. C. MacKay, "Information-Based Objective Functions for Active Data Selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[5] F. Pukelsheim, *Optimal Design of Experiments*. John Wiley & Sons, Inc., 1993.

[6] D. Uciński, *Optimal Measurement Methods for Distributed Parameter System Identification*. CRC Press, 2004.

[7] A. J. Lilienthal, M. Reggente, M. Trincavelli, J. L. Blanco, and J. Gonzalez, "A statistical approach to gas distribution modelling with mobile robots-the kernel dm+ v algorithm," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 570–576, 2009.

[8] J. G. Monroy, J.-L. Blanco, and J. Gonzalez-Jimenez, "Time-variant gas distribution mapping with obstacle information," *Auton. Robots*, vol. 40, no. 1, pp. 1–16, Jan. 2016.

[9] P. P. Neumann, S. Asadi, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, "Micro-Drone for Wind Vector Estimation and Gas Distribution Mapping," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 1, pp. 50–61, 2012.

[10] P. Whaite and F. Ferrie, "Autonomous exploration: driven by uncertainty," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 193–205, mar 1997.

[11] T. Wiedemann, C. Manss, and D. Shutin, "Multi-Agent Exploration of Spatial Dynamical Processes under Sparsity Constraints," *to appear Journal Autonomous Agents and Multi-Agent Systems*, 2017.

[12] J. Crank, *The Mathematics Of Diffusion*, 2nd ed. Oxford: Clarendon Press, 1975.

[13] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2004.

[14] M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, 2001.

[15] D. Wipf and S. Nagarajan, "A New View of Automatic Relevance Determination," *Advances in Neural Information Processing Systems 20*, pp. 1625–1632, 2008.

[16] F. R. Kschischang, B. J. Frey, and H. a. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[17] H.-a. Loeliger, "An Introduction to Factor Graphs," *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 28 – 41, 2004.

[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Francisco: Kaufmann, 1988.

[19] J. Winn and C. M. Bishop, "Variational Message Passing," *Journal of Machine Learning Research*, vol. 6, pp. 661–694, 2005.

[20] A. Logg, K.-A. Mardal, and G. Wells, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Springer Publishing Company, Incorporated, 2012.

[21] Y. Rubner, C. Tomasi, and L. J. Guibas, "A Metric for Distributions with Applications to Image Databases," *Computer Vision, 1998. Sixth International Conference on*, pp. 59–66, 1998.

Evaluation of the Force-Current Relationship in a 3-Finger Underactuated Gripper

Simone Giannico, Nicola Castaman and Stefano Ghidoni
Intelligent Autonomous Systems Laboratory
Department of Information Engineering, University of Padova, Italy

***Abstract*—This paper provides a detailed analytic evaluation of the force-current relationship for a real underactuated gripper, whose geometry presents some differences with respect to the cases usually considered in the literature. Differently from other approaches, the model proposed can work in two ways: calculating the current needed to exert a given force, and calculating the force applied by the gripper when a known current is impressed to the motor. Calculating the current as a function of the forces is not a trivial task; however, this is possible in the proposed solution thanks to the use of a single input parameter describing the set of forces applied by the gripper. The proposed approach was tested in real experiments, demonstrating that the proposed model is capable of providing a very good estimation in several working conditions.**

***Index Terms*—Underactuated gripper, grasping, force measurement.**

I. Introduction

Manipulation and grasping have been key topics in robotics for a long time, with a large number of applications in service robotics, human-machine interaction and industrial applications. They are often coupled with perception, that can be used to acquire prior knowledge on the environment and driving path planning [1]. In several cases, perception is also used as a feedback for manipulation and grasping: in this case, besides 2D and 3D vision sensors, tactile feedback is also very common, and it is usually provided by force/pressure sensors [2], [3]. Furthermore, accurate sensing is of paramount importance when manipulation and grasping applications are addressed to fragile and deformable objects, which is the new frontier in grasping [4].

The use of dedicated sensors guarantees high accuracy measurements and is the primary method for providing feedback to the manipulation and grasping algorithms. However, such Direct Sensing (DS) method often requires to install extra elements that can alter the shape of the gripper, limit its movements and require extra work for the installation and setup: if this is rather common in a laboratory, these overheads are unacceptable when an industrial scenario is targeted. An alternative to this solution is viable when cooperative robots are being employed. Cooperative robots are meant to work together with humans, and are equipped with a sensory system that can detect collisions and stop the robot before it causes damage; such sensing capability is often achieved exploiting the actuators as sensors. For example, measuring the current flowing through an electric engine, coupled with a priori knowledge about robot geometry and engine characteristics, provides information about the force applied by a robot arm. This solution, based on Indirect Sensing (IS), offers lower sensitivity and accuracy with respect to real sensors, but sensor integration comes at no cost and no effort.

The scenario just outlined demonstrates that IS is worth being investigated. This can be applied to both manipulator arms and grippers, as long as their drivers enable inspecting the motor status. A major difference between IS and DS is that the former requires accurate a priori knowledge of the robot physical characteristics to estimate the forces, while the latter is based on direct force measurement. It is important to observe that mechanical modeling plays a key role in IS, and strongly depends on the physical characteristics of the robot.

This paper focuses on the mechanical modeling of a 3-finger underactuated gripper with three phalanxes. A novel mechanical modeling including features that are generally neglected in the literature is introduced, in order to better apply the theoretical analysis to the mechanical structure of a real case.

This paper presents three main contributions. The first one is the possible misalignment among the pivots of the median actuating hinge, which is present in the real case considered in this work. A discussion on how the mechanical modeling presented in previous papers can be adapted to consider this non ideality is reported. The second aspect considered is computational efficiency, which is crucial when dealing with real world applications. Indeed, a computationally inexpensive algorithm for the position analysis of the finger is proposed. Its performance is compared against a typical iterative solution.

The third contribution is about the relationship between the force applied by the gripper and the currents of its electric motors: usually, the force is expressed as a function of the currents, that is, given the currents it is possible to calculate the forces. Differently, this paper faces the inverse problem: a solution for evaluating the motor current needed to obtain the desired force on the three phalanxes is presented. This problem is not addressed in the literature discussed in Section II.

The theoretical model presented in this paper was experimentally tested using a 3-finger gripper and sensors based on a FSR (Force Sensitive Resistor) together with a mechanical structure to properly shape the finger in the desired configuration.

The paper is organized as follows. Section II discusses related works, especially focusing on the forces applied by underactuated grippers. Section III presents the mechanical

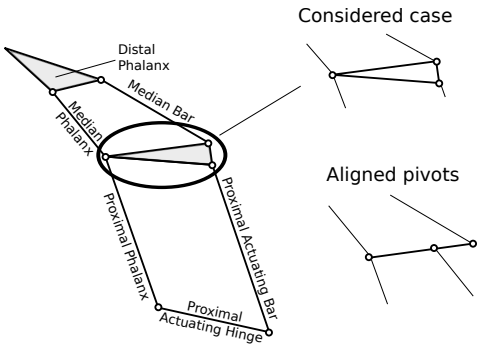


Fig. 1. Differences in the median actuating hinge.

modeling, including the influence that some non idealities usually neglected in the literature have on the gripper geometry. Experimental validation of the proposed analysis is detailed in Section IV, while final remarks are presented in Section V.

II. Related Work

Object manipulation is a fundamental feature for modern robots. At the beginning, robotic grippers were developed with the aim of matching the human hand in terms of dexterity and adaptation capabilities. Early models include the Stanford/JPL hand [5], the Utah/MIT hand [6], and the NASA Robonaut hand [7]. In all these models, significant efforts were made to find designs that were simple enough to be easily built and controlled, in order to obtain practical systems.

Aiming to reduce the cost due to the control architecture needed for complex mechanical systems with often more than ten actuators, newer grippers were designed reducing the number of Degrees of Freedom (DOFs), thus decreasing the number of actuators. The Graspar hand [8] and the Cassino finger [9] are significant examples.

Other prototypes adopt a different approach involving a lower number of actuators without decreasing the number of DOFs. This approach, called underactuation, adopts passive elements such as springs or mechanical levers in order to adapt the finger to the shape of the object to be grasped. Notable examples include the BarrettHand [10], the SARAH hand [11], the RTR-II-Hand [12], the SPRING-Hand [13], and the TH-3R Hand [14].

In the literature, few works address the analysis of underactuated grippers. Moreover, they either consider non-real cases, or they propose experiments using several different fingers, but lacking a detailed mechanical analysis. In [15] a study on an experimental gripper composed of two phalanxes is presented. Both the gripper and the analysis are very different from the present approach. Particularly, the fingers of the gripper have only two phalanxes. Furthermore, there is little focus on the analysis of the theoretical forces.

More recent work [16], [17] studies the force capability of a particular class of underactuated fingers, initially focusing on a generic 2-DOF finger and its ability to seize objects with a secure grasp. In a further work [18], the authors generalize the analysis to n-DOF underactuated fingers. In these works

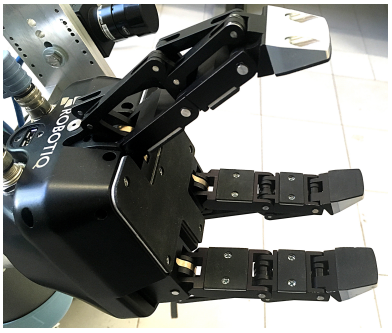


Fig. 2. Robotiq 3-Finger gripper.

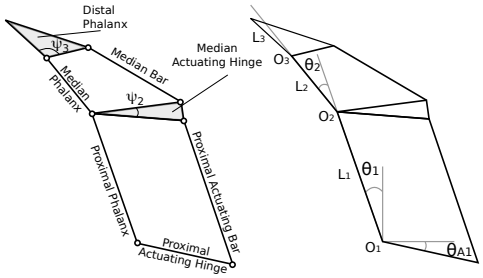


Fig. 3. Schematics for the underactuated finger.

force capability is defined as the ability to generate an external wrench onto a fixed object with a given set of phalanxes. The relationship between force and torque is analyzed, but the inverse relationship is not considered. Other works [19], [20] consider a theoretical approach to underactuated grippers based on a tendon pulley transimission mechanism.

III. Analysis of the Force-Current Relationship

This study is focused on the 3-Finger Adaptive Gripper by Robotiq[1], an underactuated gripper whose fingers are composed of three phalanxes: proximal, median and distal, as shown in Figure 3. In particular, since the three fingers are identically shaped, a single finger of the hand is hereby considered. The gripper can operate either in grasp mode (i.e. fingers wrapped around the grasped object) or pinch mode (i.e. only the distal phalanxes make contact with the object). Two different mechanisms actuate the finger: the first is meant to keep the distal phalanx in proper position while closing the finger in "pinch mode"; the second mechanism closes the finger in "grasp mode". Since our work is limited to grasping, the second mechanism is the core subject of the analysis.

We consider the case of a stable grasp, so each element of the mechanism is in equilibrium. The virtual power theorem can be applied to obtain an equation for input torques and output forces:

$$\begin{bmatrix} T_A & T_2 & T_3 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{A1} \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \qquad (1)$$

[1] http://robotiq.com/products/industrial-robot-hand/

where T_A is the actuating torque, while T_2 and T_3 are the spring torques. The latter can be calculated as follows:

$$T_2 = -K_{2,\text{spring}} \cdot (\theta_2 + \theta_{2,\text{preload}}) \,, \tag{2}$$
$$T_3 = -K_{3,\text{spring}} \cdot (\theta_3 + \theta_{3,\text{preload}}) \,. \tag{3}$$

A matrix J can be defined, such that:

$$[J] \begin{bmatrix} \dot{\theta}_{A1} & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^T = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \,. \tag{4}$$

Considering only forces normal to the phalanx, the relationship between torques and forces becomes bijective. Each element j_{ih} of the Jacobian matrix represents the moment arm between the i-th force and the h-th revolute-joint.

$$J = \begin{bmatrix} k_1 & 0 & 0 \\ k_2 + l_1 \cos(\theta_2) & k_2 & 0 \\ k_3 + l_2 \cos\theta_3 + l_1 \cos(\theta_2 + \theta_3) & k_3 + l_2 \cos(\theta_3) & k_3 \end{bmatrix} \tag{5}$$

A matrix $[W]$ is to be found, such that:

$$\begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^T = [W] \begin{bmatrix} \dot{\theta}_{A1} & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^T \,. \tag{6}$$

It is evident that the identity matrix differs from [W] only for the first row. Therefore, it is necessary to determine $w_{11} = \frac{\partial \dot{\theta}_1}{\partial \dot{\theta}_{A1}}$, $w_{12} = \frac{\partial \dot{\theta}_1}{\partial \dot{\theta}_2}$ and $w_{13} = \frac{\partial \dot{\theta}_1}{\partial \dot{\theta}_3}$.

To obtain w_{11} one can think of blocking the variables θ_2 and θ_3. The resulting mechanism has only 1 DoF, so it is sufficient to simply calculate the ratio between $\dot{\theta}_1$ and $\dot{\theta}_{A1}$. Since this 1 DoF mechanism can only revolute around O_1 (see Fig. 3), it rotates as a solid body around O_1. Therefore it is $w_{11} = 1$.

With a similar approach, variables θ_{A1} and θ_3 can be blocked, so the second four-linkage system behaves as a solid body. Therefore the variation of the absolute position of the second phalanx ($\delta\theta_2^{\text{abs}}$) equals the variation of the absolute position of its actuating hinge ($\delta\theta_{A2}$), as in (7):

$$\begin{aligned} \theta_2 &= \theta_2^{\text{abs}} - \theta_1 \,, \\ \delta\theta_2 &= \delta\theta_2^{\text{abs}} - \delta\theta_1 \,, \\ \delta\theta_{A2} &= \delta\theta_2^{\text{abs}} = \delta\theta_2 + \delta\theta 1 \,. \end{aligned} \tag{7}$$

As we can see in Fig. 4, the virtual displacement of O_2 can be put in a relation with both the rotation of the proximal phalanx and that of the median actuating hinge, as in (8), (9):

$$\overline{O_2 O_2'} = -l_1 \delta\theta_1 \,, \tag{8}$$
$$\overline{O_2 O_2'} = h_2 \delta\theta_{A2} = h_2(\delta\theta_2 + \delta\theta_1) \,. \tag{9}$$

From (8), (9) it results:

$$\begin{aligned} & h_2(\delta\theta_2 + \delta\theta_1) = -l_1 \delta\theta_1 \,, \\ & \frac{\delta\theta_1}{\delta t} = -\frac{h_2}{h_2 + l_1} \cdot \frac{\delta_2}{\delta t} \Rightarrow \frac{\partial \dot{\theta}_1}{\partial \dot{\theta}_2} = -\frac{h_2}{h_2 + l_1} = w_{12} \,. \end{aligned} \tag{10}$$

Finally, w_{13} can be obtained recursively as in (11):

$$w_{13} = (-\frac{h_2}{h_2 + l_1}) \cdot (-\frac{h_3}{h_3 + l_2}) = \frac{h_2 h_3}{(h_2 + l_1)(h_3 + l_2)} \,. \tag{11}$$

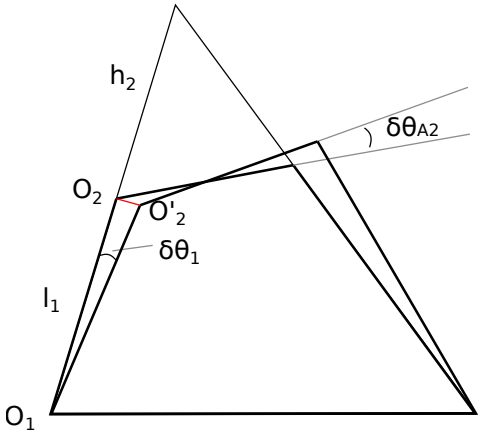


Fig. 4. Calculating w_{12}.

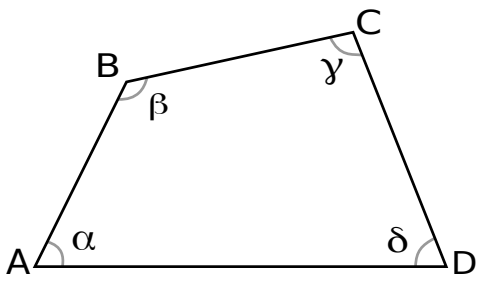


Fig. 5. Quadrilateral nomenclature.

A. Position Analysis of the Finger

In order to calculate the elements of the [W] matrix, the parameters h_2 and h_3 have to be obtained from the position analysis of the finger. This is where the geometry of the finger, which differs from the geometry considered in previous literature, influences the results of the analysis. An algorithm for the four-linkage system is presented, as it will be used twice for the position analysis of the finger. It implements a direct approach taking advantage of trigonometry. Other possible solutions can use numeric methods for non-linear systems such as Newton method, leading to a slower computation. A performance comparison will be presented later. Referring to a generic quadrilateral as in Fig. 5, of which all the four sides are known constant, the algorithm shall calculate angles α, γ and δ corresponding to the known variable angle β. In (12) and (13) angles δ and α are calculated using trigonometry. Particularly, from (13) it is possible to obtain a solution for $\cos\alpha$ without the use of iterative methods.

$$\begin{aligned} \cos\delta = & -\frac{1}{2}(\overline{AB}^2 + \overline{BC}^2 - \overline{AD}^2 - \overline{CD}^2 \\ & - 2 \cdot \overline{AB} \cdot \overline{BC} \cdot \cos\beta) \\ \delta = & \arccos(\cos\delta) \quad , \qquad \delta \in [0, \pi] \\ A = & \overline{AB} + \overline{BC} \cdot \cos(\pi - \beta) \\ B = & \overline{BC} \cdot \sin(\pi - \beta) \\ C = & \overline{CD} \cdot \cos\delta - \overline{AD} \end{aligned} \tag{12}$$

$$A \cdot \cos\alpha + B \cdot \sqrt{1-\cos^2\alpha} + C = 0 \tag{13}$$

The algorithm can now calculate $\alpha = \arccos(\cos\alpha)$ (given that $\alpha \in [0, \pi]$) and $\gamma = 2\pi - \alpha - \beta - \delta$ so the position analysis for the 4-linkage system is complete.

This paper proposes a method (12),(13) for computationally inexpensive solution of the four-linkage systems that compose the finger. In the experiment section, the performance of the presented method is compared with that of iterative algorithms for non linear systems, in particular with a C++ implementation of the Newton method.

Once the four-linkage angles are determined, the *h* parameter can be calculated as in (14):

$$h = \overline{AD} \cdot \frac{\sin\delta}{\sin(\pi - \alpha - \delta)} - \overline{AB} \tag{14}$$

The presented algorithm, which calculates the angles and the h parameter for a given quadrilateral, is to be applied at first to the four-linkage system of the median phalanx. In this case $\beta_{\text{med}} = \pi + \theta_3 - \psi_3$ is the input. Parameter h_3 is calculated as well as the angles, of which α_{med} is needed for β_{prox} that serves as an input for the quadrilateral of the proximal phalanx. See Fig. 3 for ψ_2 and ψ_3. The next input (to calculate h_3) is:

$$\beta_{\text{prox}} = \pi + \theta_2 - \alpha_{\text{med}} - \psi_2 \tag{15}$$

Having determined all the parameters needed for [J] and [W] matrices, the forces can be calculated:

$$\begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}^T = [J]^{-T}[W]^{-T} \begin{bmatrix} T_A & T_2 & T_3 \end{bmatrix}^T \tag{16}$$

Differently from the ideal general case in [18], the *median actuating hinge* of this finger has non-aligned pivots. Part of our work addressed this problem, demonstrating that this difference is entirely contained in (15).

B. Obtaining the Current as a Function of the Desired Force

As it was observed, with the hypothesis of pure normal forces the relation between $\begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}^T$ and $\begin{bmatrix} T_A & T_2 & T_3 \end{bmatrix}^T$ becomes bijective, therefore any set of three torques has a corresponding set of forces and *vice versa*. However, even though both sets are of rank 3, the status of the system is partially determined by the configuration of the hand, which is expressed in the 2 coordinates θ_2 and θ_3. As a result, for a given configuration, the relation is actually of rank 1. While this does not complicate the task of obtaining the forces as a function of the torque (or motor current), it can in fact make things more difficult when trying to obtain the torque (or current) as a function of the forces exerted (or to be exerted) by the phalanxes. In the first case (torque to forces), each element of the input set (the torques) is in a bijective relation with either the input current or one of the two configuration coordinates. This means that the high level input is as simple as a torque (or the corresponding motor current) for a given configuration. Vice versa, when calculating the actuating torque (or current) as a function of the desired forces, it is not obvious how to express the input forces. Indeed, for a given position the input is actually of rank 1, so the user can not arbitrarily choose a set of three forces. In the presented approach, a single parameter is defined in order to properly represent the set of forces at a high level.

A desirable feature in a robotic hand is the ability to control the maximum force applied. Therefore, a solution is here proposed which takes the maximum desired force as a rank 1 input for the function, obtaining the corresponding motor current as the output. A possible approach is to try different values for the input current (in the function that gives the forces as an output), until the desired force value is matched within a given error. However, to reduce the number of iterations, the following method can be applied. The proposed algorithm tries to consider each phalanx as the one exerting the maximum force; it then verifies this assumption by calculating the corresponding actuating torque with (17), and the set of three forces that the latter torque generates. If the force exerted by the considered phalanx has actually the maximum value in the set, then the corresponding motor current is taken as the solution.

$$\begin{aligned}
T_{A,\text{prox}} &= \frac{f_1 k_1 k_2 w_{11}}{j_{21} x_2} + \frac{w_{11} T_2}{w_{12}} + \\
&\quad + \frac{(j_{31}k_2 - j_{21}j_{32}) \cdot (T_3 w_{11} - w_{13})}{j_{21} w_{12} k_3} - \frac{k_2}{j_{21} w_{12}}\,, \\
T_{A,\text{med}} &= (f_2 - \frac{T_2}{k_2} + \frac{j_{32} T_3}{k_2 k_3}) \cdot \frac{k_2 k_3 w_{11}}{j_{32} w_{13} - w_{12} k_3}\,, \\
T_{A,\text{dist}} &= (-f_3 + \frac{T_3}{k_3}) \cdot \frac{k_3 w_{11}}{w_{13}}\,.
\end{aligned} \tag{17}$$

Where each f_i has to be substituted with the desired value. The same results can be used to implement an algorithm for the minimum applied force.

1) Calculate $T_{A,\text{prox}}, T_{A,\text{med}}, T_{A,\text{dist}}$ with (17);
2) Calculate the set of forces for each torque with (16), until a set is obtained where the input force is higher than the consequential two others;
3) Pick the torque (among $[T_{A,\text{prox}}, T_{A,\text{med}}, T_{A,\text{dist}}]$) corresponding to such a set of forces.

This method can be avoided when the three functions $f_1(T_A), f_2(T_A), f_3(T_A)$ can be reasonably assumed as monotonic in the workspace. In this case it is sufficient to apply (17) to each phalanx and pick the minimum actuating torque in the resulting set of three.

The ability of evaluating *a priori* the current corresponding to the desired force is very important both for open-chain control and for speed optimization of feedback systems with pressure sensors.

C. Further Considerations on the Real Case Application

The tools and functions that have been described, extend general previous work. Some peculiarity is added here and some solutions are proposed for real-case problems such as different geometry (in relation to [18]), and the need for a useful and meaningful inverse function. However, a few additional

considerations have to be done. First of all, the spring torques T_2 and T_3 can be preloaded, as it is for the gripper from Robotiq. In this case, it will be $T_i = -K_{i,\text{spring}} \cdot (\theta_i + \theta_{i,\text{preload}})$, except for resting position (i.e. $\theta_i = 0$) in which case it will be $T_i = 0$. Secondly, regarding the current control, we can simply relate it with the input torque (T_A) considering that for a DC brushed motor the torque curve is generally linear. So it is sufficient to multiply the current by a constant that includes the torque/current ratio and the reduction ratio of the gears in the specific model. In this case the motor is coupled with a 14:1 planetary gear; plus another 40:1 ratio between the endless screw at the motor shaft and the worm gear attached to the proximal actuating hinge. Finally, in subsection *B*, the angles of the quadrilateral have been assumed $\in [0, \pi]$. If adapting this work to different cases, the convexity of the quadrilaterals should be verified for the configurations of interest.

IV. Experiments

In this section two different sets of experiments are presented. The first one concerns the position analysis of the finger. It compares the performance of the algorithm proposed in Subsection III-A against an implementation of the Newton method for non-linear systems. On the other hand the second experiment evaluates the effectiveness of the proposed analysis for current-force relationship estimation. The test consists in a comparison between the results of the algorithm and the actual force applied by the gripper.

A. Algorithm for the Position Analysis of the Finger

The performance of the algorithm presented in Subsection III-A was compared (in a C++ implementation) against an implementation of the Newton method. The programs implementing the two approaches were tested on a PC running Debian 7.8.0 on a Pentium® Dual-Core CPU T4400 @ 2.20 GHz. For a set of 52 different input angles, the average execution time was of 3000 ns and 5823.53 ns for the proposed algorithm and the Newton method respectively. Please note that the accuracy of the non-iterative method is directly related to data-type precision of the used variables. The iterative solution has been set to obtain a comparable accuracy, as it can be seen by the reported errors.

B. Experimental Setup

As mentioned in the Section I, the proposed analysis was tested on the 3-Finger Adaptive Gripper produced by Robotiq[2]. The 3-Finger gripper from Robotiq has three phalanxes per finger. Each finger has only one motor, which actuates the Proximal Actuating Hinge (see Figure 3). Therefore the mechanism has only 1 DoA (Degrees of Actuation) against 3 DoF (Degrees of Freedom).

The current range in the motor for the grasping motion goes from 225 mA to 590 mA, in a linear relation with integer values from 0 to 255. The gripper was connected to the computer via Ethernet and controlled with ROS[3] (Robot Operating System) and ROS-Industrial Robotiq package[4].

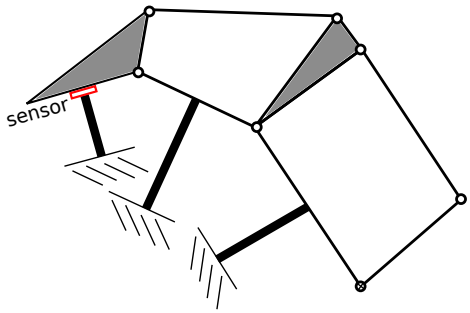


Fig. 6. Measurement apparatus schematics

The real force applied by the gripper was measured by means of a force-sensing resistor (Interlink FSR 402[5]) which has a declared range of 20 N. However, using proper conditioning, it was possible to obtain good results up to 40 N. Such value is only sufficient for measurements on the median and distal phalanxes, while forces on the proximal phalanx exceed the system range. The sensor choice was made in favor of compactness, which is very useful in order to obtain stable grasp configurations to observe. The analog signal from the conditioning circuit was fed to an Arduino Nano[6] microcontroller board, which was programmed to process the values (being the curve non-linear) and send them to a PC.

The measures were taken with the gripper in horizontal position, with the two inactive fingers holding the experimental apparatus. Please note that the structure that simulated the grasped object, was designed in order to obtain proper contact points on each finger, as in Figure 6. Motor current feedback was received as an integer value between 0 and 255, then converted via the previously mentioned relation. Angles between the phalanxes were calculated by measuring the distance occurring between markers placed at the middle of each finger.

C. Experimental Results

The proposed experiments were used to test the newly introduced model, and should be considered as a first step towards its complete and thorough experimental validation. For a complete validation of the proposed model, experiments strongly reducing the non-idealities should be planned.

Figure 7 summarizes test results for the forces on the distal phalanx. For each execution, angles between the phalanxes, θ_2 and θ_3, are represented on the axes x and y respectively, so each point of the X-Y plane corresponds to a configuration of the finger. The relative error $\left(\frac{\Delta \text{F}}{\text{F}_{\text{meas}}}\right)$ is proportional to the size of the bubbles. Being the axes scaled on degrees, and being the relative error adimensional, the diagram does not explicitly represent the error, but it is meant to compare the errors

[2]http://robotiq.com/products/industrial-robot-hand/

[3]http://www.ros.org/

[4]https://github.com/ros-industrial/robotiq

[5]http://interlinkelectronics.com/force.php

[6]https://www.arduino.cc/en/Main/ArduinoBoardNano

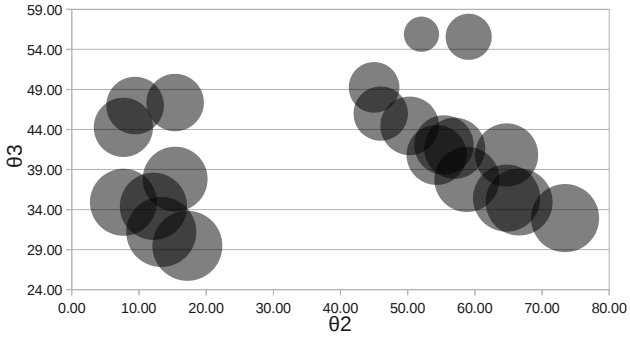


Fig. 7. Relative errors for the distal phalanx

obtained for different configurations. For most executions, the relative error is more than 50%, which is too high to be completely ascribed to the uncertainty of the force sensor. Similar results were observed for the median phalanx. Such a gap is caused, in our opinion, by the presence of forces parallel to the phalanxes. These friction forces significantly affect the system by generating a moment about the point where the actuating torque is applied, therefore altering the balance of the virtual works. The reaction force generated by friction also has a stabilizing effect on the system.

The results provided in this paper are a first contribution to the understanding of the influence of non-idealities in such a theoretical model, an investigation that was missing in the state of the art.

V. Conclusions

This paper proposes a mechanical modeling for underactuated grippers whose fingers are composed of three phalanxes. Tests were performed on a 3-finger gripper, but can be extended to grippers with any number of fingers. An algorithm for a computationally inexpensive position analysis of the finger was also proposed, together with a method for calculating the motor currents to be employed for applying a desired force. This is very useful for driving robotic applications, in particular when grasping targets soft and deformable objects. The experimental results highlighted the strong influence of non idealities in the model proposed, that would probably affect also other similar models. In order to overcome the strong influence of non-idealities during the experiments, future work will make use of a device to minimize friction on the phalanxes.

Acknowledgements

The authors would like to thank Elisa Tosello for the fruitful discussions and the precious advice. This work was partially funded by the project EURECA, H2020 CleanSky 2 under grant agreement n. 738039.

References

[1] D. Holz, M. Nieuwenhuisen, D. Droeschel, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke, "Active recognition and manipulation for mobile robot bin picking," in *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*. Springer, 2014, pp. 133–153.

[2] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Şucan, "Towards reliable grasping and manipulation in household environments," in *Experimental Robotics*. Springer, 2014, pp. 241–252.

[3] P. N. Koustoumpardis, K. X. Nastos, and N. A. Aspragathos, "Underactuated 3-finger robotic gripper for grasping fabrics," in *Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–8.

[4] M. Mohammadi, T. L. Baldi, S. Scheggi, and D. Prattichizzo, "Fingertip force estimation via inertial and magnetic sensors in deformable object manipulation," in *Haptics Symposium (HAPTICS), 2016 IEEE*. IEEE, 2016, pp. 284–289.

[5] J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *The International journal of Robotics research*, vol. 1, no. 1, pp. 4–17, 1982.

[6] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers, "Design of the utah/mit dextrous hand," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3. IEEE, 1986, pp. 1520–1532.

[7] M. A. Diftler and R. O. Ambrose, "Robonaut: A robotic astronaut assistant," in *Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), 2001 6th International Symposium on*, vol. 2001. Citeseer, 2001.

[8] J. D. Crisman, C. Kanojia, and I. Zeid, "Graspar: a flexible, easily controllable robotic hand," *IEEE Robotics Automation Magazine*, vol. 3, no. 2, pp. 32–38, Jun 1996.

[9] G. Figliolini and M. Ceccarelli, "A novel articulated mechanism mimicking the motion of index fingers," *Robotica*, vol. 20, no. 01, pp. 13–22, 2002.

[10] N. T. Ulrich, "Methods and apparatus for mechanically intelligent grasping," 9 1990, uS Patent 4,957,320. [Online]. Available: https://www.google.com/patents/US4957320

[11] T. Laliberté and C. M. Gosselin, "Underactuation in space robotic hands," in *Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), 2001 6th International Symposium on*, 2001, pp. 18–21.

[12] B. Massa, S. Roccella, M. C. Carrozza, and P. Dario, "Design and development of an underactuated prosthetic hand," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 3374–3379.

[13] M. C. Carrozza, C. Suppo, F. Sebastiani, B. Massa, F. Vecchi, R. Lazzarini, M. R. Cutkosky, and P. Dario, "The spring hand: development of a self-adaptive prosthesis for restoring natural grasping," *Autonomous Robots*, vol. 16, no. 2, pp. 125–141, 2004.

[14] W. Zhang, D. Che, H. Liu, X. Ma, Q. Chen, D. Du, and Z. Sun, "Super underactuated multifingered mechanical hand with modular selfadaptive gearrack mechanism," *Industrial Robot: An International Journal*, vol. 36, no. 3, pp. 255–262, 2009.

[15] H. Shimojima, K. Yamamoto, and K. Kawakita, "A study of grippers with multiple degrees of mobility: Vibration, control engineering, engineering for industry," *JSME international journal*, vol. 30, no. 261, pp. 515–522, 1987.

[16] L. Birglen and C. M. Gosselin, "On the force capability of underactuated fingers," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, Sept 2003, pp. 1139–1145 vol.1.

[17] C. M. Gosselin, "Analysis of underactuated mechanical grippers," *Journal of Mechanical Design*, vol. 123, pp. 367–374, 2001.

[18] L. Birglen and C. M. Gosselin, "Kinetostatic analysis of underactuated fingers," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 211–221, April 2004.

[19] R. Rizk, S. Krut, and E. Dombre, "Grasp-stability analysis of a two-phalanx isotropic underactuated finger," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3289–3294.

[20] G. A. Kragten and J. L. Herder, "Equilibrium, stability, and robustness in underactuated grasping," in *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2007, pp. 645–652.

From Monocular SLAM to Autonomous Drone Exploration

Lukas von Stumberg[1], Vladyslav Usenko[1], Jakob Engel[2], Jörg Stückler[3], and Daniel Cremers[1]

***Abstract*— Micro aerial vehicles (MAVs) are strongly limited in their payload and power capacity. In order to implement autonomous navigation, algorithms are therefore desirable that use sensory equipment that is as small, low-weight, and low-power consuming as possible. In this paper, we propose a method for autonomous MAV navigation and exploration using a low-cost consumer-grade quadrocopter equipped with a monocular camera. Our vision-based navigation system builds on LSD-SLAM which estimates the MAV trajectory and a semi-dense reconstruction of the environment in real-time. Since LSD-SLAM only determines depth at high gradient pixels, texture-less areas are not directly observed so that previous exploration methods that assume dense map information cannot directly be applied. We propose an obstacle mapping and exploration approach that takes the properties of our semi-dense monocular SLAM system into account. In experiments, we demonstrate our vision-based autonomous navigation and exploration system with a Parrot Bebop MAV.**

I. INTRODUCTION

Most autonomous micro aerial vehicles (MAVs) to date rely on depth sensing through e.g. laser scanners, RGB-D or stereo cameras. Payload and power capacity are, however, limiting factors for MAVs, such that sensing principles are desirable that require as little size, weight, and power-consumption as possible.

In recent work, we propose large-scale direct simultaneous localization and mapping (LSD-SLAM [1]) with handheld monocular cameras in real-time. This method tracks the motion of the camera towards reference keyframes and at the same time estimates semi-dense depth at high gradient pixels in the keyframe. By this, it avoids strong regularity assumptions such as planarity in textureless areas. In this paper, we demonstrate how this method can be used for obstacle-avoiding autonomous navigation and exploration for a consumer-grade MAV. We integrate our approach on the recently introduced Parrot Bebop MAV, which comes with a 30 fps high-resolution fisheye video camera and integrated attitude sensing and control.

Our proposed two-step exploration strategy is specifically and directly suited for semi-dense reconstructions as obtained with LSD-SLAM. A simple but effective local exploration strategy, coined *star discovery*, safely discovers free and occupied space in a local surrounding of a specific position in the environment. In contrast to existing autonomous exploration approaches, our method takes the semi-dense depth measurement principle of LSD-SLAM based on motion parallax into account. A global exploration strategy then determines interesting volume for further local explorations in order to sequentially discover novel parts of the environment. We demonstrate the properties of our exploration strategy in several experiments with the Parrot Bebop.

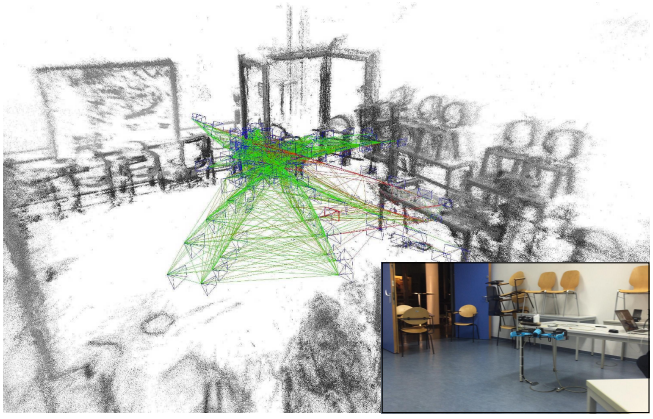


Fig. 1: We propose vision-based autonomous navigation and exploration for small low-cost micro aerial vehicles equipped with monocular cameras. Our approach is based on large-scale direct SLAM which determines a semi-dense reconstruction of the environment. We integrate our approach on a commercially available Parrot Bebop MAV.

II. RELATED WORK

Autonomous exploration by mobile robots has been investigated over many years, mainly relying on laser scanner sensors. Yamauchi [2] proposed in his seminal work the so-called frontier-based exploration strategy that favors exploring the frontiers of the unexplored space in the map. Some methods define a utility function [3], [4], e.g., on paths or view poses that, for instance, trade-off discovered area with travel costs. The approaches in [5], [6], [7] combine the probabilistic measure of information gain with travel cost in a measure of utility. Rekleitis et al. [8] optimize a utility function that favors the reduction of uncertainty in the map, and at the same time tries to achieve a fast exploration of the map. All of the above methods rely on a dense map representation of the environment which is acquired using 2D and 3D laser range sensors. In our case, these exploration methods are not directly applicable. The exploration process needs to additionally consider how dense

This work has been supported by ERC Consolidator Grant 3D Reloaded (649323).

[1]The authors are with the Computer Vision Group, Computer Science Institute 9, Technische Universität München, 85748 Garching, Germany {stumberg, usenko, cremers}@in.tum.de

[2]The author is with Oculus Research, Redmond, USA jajuengel@gmail.com

[3]The author is with the Computer Vision Group, Visual Computing Institute, RWTH Aachen University, 52074 Aachen, Germany stueckler@vision.rwth-aachen.de

map information can be obtained from the visual semi-dense depth measurements of our SLAM system.

Very recently, autonomous exploration by flying robots has attracted attention [9], [10], [11], [12]. Nuske et al. [11] explore rivers using an MAV equipped with a continuously rotating 3D laser scanner. They propose a multi-criteria exploration strategy to select goal points and traversal paths. Heng et al. [12] propose a two-step approach to visual exploration with MAVs using depth cameras. Efficient exploration is achieved through maximizing information gain in a 3D occupancy map. At the same time, high coverage of the viewed surfaces is determined along the path to the exploration pose. In order to avoid building up a dense 3D map of the environment and applying standard exploration methods, Shen et al. [9] propose a particle-based frontier method that represents known and unknown space through samples. This approach also relies on depth sensing through a 2D laser scanner and a depth camera. Yoder and Scherer [10] explore the frontiers of surfaces measured with a 3D laser scanner. In [13] a 3D occupancy map of the environment is acquired using an on-board depth sensor. Next best views for exploration are selected by growing a random tree path planner in the free-space of the current map and choosing a branch to explore that maximizes the amount of unmapped space uncovered on the path. Also these approaches use dense range or depth sensors which allow for adapting existing exploration methods from mobile robotics research. Desaraju et al. [14] use a monocular camera and a dense motion stereo approach to find suitable landing sites of a UAV.

We propose an exploration method which is suitable for lightweight, low-cost monocular cameras. Our visual navigation method is based on large-scale direct SLAM which recovers semi-dense reconstructions. We take special care of the semi-dense information and its measurement process for obstacle mapping and exploration.

III. AUTONOMOUS QUADROCOPTER NAVIGATION USING MONOCULAR LSD-SLAM

We build on the TUM ARDrone package by Engel et al. [15] which has been originally developed for the Parrot ARDrone 2.0. We transferred the software to the Parrot Bebop platform which comes with similar sensory equipment and onboard control. The Parrot Bebop is equipped with an IMU built from 3-axis magnetometer, gyroscope, and accelerometer. It measures height using an ultrasonic sensor, an air pressure sensor and a vertical camera, similar to the Parrot ARDrone 2.0. The MAV is equipped with a fisheye camera with wide 186° field-of-view. The camera provides images at 30 frames per second. A horizontally stabilized region-of-interest is automatically extracted in software on the main processing unit of the MAV, and can be transmitted via wireless communication with the attitude measurements.

1) State Estimation and Control: The visual navigation system proposed in [15] integrates visual motion estimates from a monocular SLAM system with the attitude measurements from the MAV. It filters both kinds of messages using a loosely-coupled Extended Kalman filtering (EKF) approach. Since the attitude measurements and control commands are transmitted via wireless communication, they are affected by a time delay that needs to be compensated using the EKF framework. Waypoint control of the MAV is achieved using PID control based on the EKF state estimate. In monocular SLAM, the metric scale of motion and reconstruction estimates are not observable. We probabilistically fuse ultrasonic and air pressure measurements and adapt the scale of the SLAM motion estimate to the observed metric scale [15].

2) Vision-Based Navigation Using Monocular LSD-SLAM: LSD-SLAM [1] is a keyframe based SLAM approach. It maintains and optimizes the view poses of a subset of images, i.e. keyframes, extracted along the camera trajectory. In order to estimate the camera trajectory, it tracks camera motion towards a reference keyframe through direct image alignment. This requires depth in either of the images, which we estimate from stereo correspondences between the two images within the reference keyframe. The poses of the keyframes are made globally consistent by mutual direct image alignment and pose graph optimization.

A key feature of LSD-SLAM is the ability to close trajectory loops within the keyframe graph. In such an event, the view poses of the keyframes are readjusted to compensate for the drift that is accumulated through tracking along the loop. This especially changes the pose of the current reference keyframe that is used for tracking, also inducing a change in the tracked motion estimate. Yet, the tracked motion estimate is used to update the EKF that estimates the MAV state which is fed into the control loop. At a loop closure, this visual motion estimate would update the filter with large erroneous velocities which would induce significant errors in the state estimate. In turn this could cause severe failures in flight control. We therefore compensate for the changes induced by loop-closures with an additional pose offset on the visual estimate before feeding it into the EKF.

In order to initialize the system, the MAV performs a look-around maneuver in the beginning by flying a 360° turn on the spot while hovering up and down by several centimeters. In this way, the MAV already obtains an initial keyframe map with a closed trajectory loop (Fig. 6).

IV. AUTONOMOUS OBSTACLE-FREE EXPLORATION WITH SEMI-DENSE DEPTH MAPS

Autonomous exploration has been a research topic for many years targeting exploration of both 2D and 3D environments. In most 3D scenarios an exploration strategy works with a volumetric representation of the environment, such as a voxel grid or an octree, and uses laser-scanners or RGB-D cameras as sensors to build such a representation.

In this paper we devise an exploration strategy that builds on a fundamentally different type of sensor data – semi-dense depth maps estimated with a single moving monocular camera. The difference to previously mentioned sensors lies in the fact that only for the image areas with strong gradients the depth can be estimated. This means that especially initially during exploration, large portions of the map will

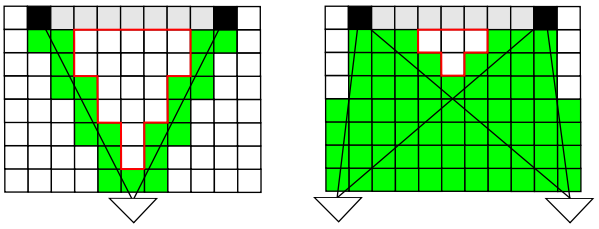


Fig. 2: With LSD-SLAM, only semi-dense depth measurements at edges and high-texture areas can be obtained. It provides no depth in textureless areas such as walls. This creates indentations of unknown space in the occupancy map (free space green, indentations red). Lateral motion (right picture) towards the measurable points, however, allows for reducing the indentations.

remain unknown. The exploration strategy has to account for the motion parallax measurement principle of LSD-SLAM.

A. Occupancy Mapping with Semi-Dense Depth Maps

In this work we use OctoMap [16] that provides an efficient implementation of hierarchical 3D occupancy mapping in octrees. We directly use the semi-dense depth maps reconstructed with LSD-SLAM to create the 3D occupancy map. All keyframes are traversed and the measured depths are integrated via ray-casting using the camera model.

Since LSD-SLAM performs loop closures, the poses at which the depth maps of keyframes have been integrated into the map may change and the map will become outdated. We therefore periodically regenerate the map using the updated keyframe poses. While this operation may last for several seconds, the MAV hovers on the spot and waits until proceeding with the exploration.

Each voxel in the occupancy map stores the probability of being occupied in log-odds form. In order to determine if a voxel is free or occupied, a threshold is applied on the occupancy probability (0.86 in our experiments). During the integration of a depth measurement, all voxels along the ray in front of the measurement are updated with a probability value for missing voxels and measuring free-space. The voxel at the depth measurement in turn is updated with a hit probability value. Note that LSD-SLAM outputs not only the computed depths but also the variance of this estimate. Although measurements with a high variance can be very noisy, they still contain information about the vicinity of the sensor. Therefore we insert only free space on a reduced distance for these pixels which assures that no wrong voxels are added. Fig. 11a shows an example occupancy map.

B. Optimal Motion for Exploration and Mapping

By using semi-dense reconstructions, we do not make strong assumptions such as planarity on the properties of the environment in textureless areas. On the other hand, the use of semi-dense reconstruction in visual navigation leads to indentations of unknown volume which occur between the rays of free-space measured towards depth readings (Fig. 2). As we will analyze next, these indentations can be removed

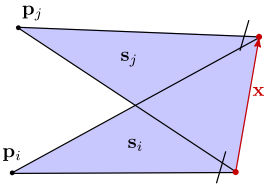


Fig. 3: Discovered area $\mathbf{s}_i$, $\mathbf{s}_j$ for measured points $\mathbf{p}_i$, $\mathbf{p}_j$ in relation to the direction of motion $\mathbf{x}$.

through lateral motion towards the measurable structures – an important property that we will exploit in our exploration strategy.

Figure 3 illustrates the problem of finding the direction of motion $\mathbf{x} \in \mathbb{R}^3$ with $\|\mathbf{x}\|_2 = 1$ such that it maximizes the observed free space in a 2D setting (without loss of generality). Assuming the camera center at the origin of the coordinate frame, the volume that is observed free in front of the measured point is cut by the triangle formed by the motion $\mathbf{x}$ of the camera and the measured point $\mathbf{p}_i$. The magnitude of the vector

$$\mathbf{s}_i = \frac{1}{2}(\mathbf{p}_i \times \mathbf{x}) = \frac{1}{2}\hat{\mathbf{p}}_i \mathbf{x} \tag{1}$$

equals the observed free area, where $\hat{\mathbf{p}}_i$ is a skew-symmetric matrix formed from $\mathbf{p}_i$ such that its product with $\mathbf{x}$ corresponds to the cross-product between the two vectors.

To find the optimal direction we maximize (two times) the sum of squared areas of the triangles formed by all observed points,

$$S(\mathbf{x}) = 2\sum_{i=1}^{n} \mathbf{s}_i^\top \mathbf{s}_i = \frac{1}{2}\sum_{i=1}^{n} \mathbf{x}^\top \hat{\mathbf{p}}_i^\top \hat{\mathbf{p}}_i \mathbf{x}, \tag{2}$$

$$\|\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{x} = 1. \tag{3}$$

Since we want to determine the optimal motion direction independent from its magnitude, we optimize the direction subject to a normalization constraint.

This constrained optimization problem can be solved using Lagrange multipliers,

$$L(\mathbf{x}, \lambda) = S(\mathbf{x}) - \lambda(\mathbf{x}^\top \mathbf{x} - 1), \tag{4}$$

so that the optimal solution for $\mathbf{x}$ should satisfy the equations

$$\nabla_{\mathbf{x}} L^\top = \left(\sum_{i=1}^{n} \hat{\mathbf{p}}_i^\top \hat{\mathbf{p}}_i\right)\mathbf{x} - 2\lambda\mathbf{x} = \mathbf{M}\mathbf{x} - 2\lambda\mathbf{x} = \mathbf{0}, \tag{5}$$

$$\frac{\partial L}{\partial \lambda} = \mathbf{x}^\top \mathbf{x} - 1 = 0. \tag{6}$$

This implies that $\mathbf{x}$ should be a (unit) eigenvector of the matrix $\mathbf{M}$. Moreover, the vector that corresponds to the largest eigenvalue produces the largest observation of the free space.

We perform Monte-Carlo simulations to further analyze the optimal motion direction. Without any prior knowledge about the environment structure, we assume a uniform distribution of depths in the pixels. We sample 600 points $\mathbf{p}_i$ according to the following distribution: $u \sim U(0, 640), v \sim$

TABLE I: Summary of eigenvectors and eigenvalues of $\mathbf{M}$ over 100 random simulations where we draw image coordinates and depths from uniform distributions $u \sim U(0,640)$, $v \sim U(0,480)$, $d \sim U(0.5,5)$, respectively.

eigenvalue	eigenvector
7440.5 ± 275.6	$(0.012, 0.996, -0.001) \pm (0.088, 0.005, 0.014)$
7093.7 ± 260.0	$(0.996, 0.012, -0.002) \pm (0.005, 0.088, 0.018)$
1230.1 ± 66.1	$(0.002, 0.001, 0.997) \pm (0.018, 0.015, 0.001)$

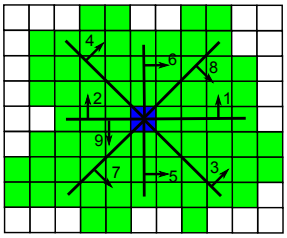
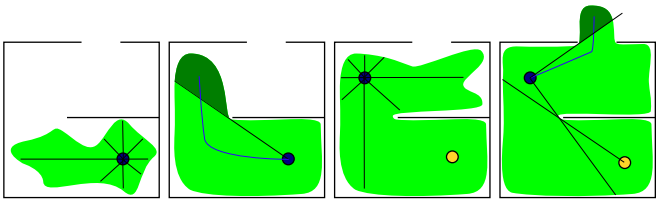


Fig. 4: Local star discovery exploration strategy. The MAV flies to the end of the lines in the order defined by the numbers, always facing the direction of the arrow corresponding to the line.

$U(0,480), d \sim U(0.5,5)$, where (u,v) are the image coordinates, d is the distance, and U denotes uniform distribution. The points are reprojected into 3D space using the camera model and used for computing $\mathbf{M}$. Statistics for the eigenvalues and eigenvectors for 100 random simulations is accumulated in Table I. It demonstrates that the optimal direction for increasing the observed free space is a motion parallel to the image plane, i.e. sidewards or up-down motion.

C. Obstacle-Free Local Exploration through Star Discoveries

The characteristics of our semi-dense SLAM approach prevent the direct application of existing exploration approaches such as next-best view-planning or frontier-based exploration. Frontiers of known space (occupied or measured free) occur at indentations of unknown space as well as between measured edges and textures on flat walls. Simply flying to those boundaries would not allow discerning unknown from free or occupied space at the textureless boundaries as monocular SLAM requires motion parallax towards measurable structures and depth is only measured along the line-of-sight of semi-dense depth readings. Next-best view planning aims at choosing a new view that maximizes the discovered unknown space at the new view pose. Since measuring depth requires motion in our monocular SLAM system, it could be extended to measure the discovered space along the path to the new view point. This procedure would be computationally very expensive, since for each potential view pose many ray-casting operations would need to be performed. We propose a simpler but effective local exploration strategy that we call *star discovery*, which discovers the indentations in the unknown volume around a specific position in the map.



(a) First (b) Second (c) Third (d) Fourth

Fig. 5: Four steps of the proposed exploration strategy. (a) The exploration starts at the blue circle. After an initial look-around, the green volume is marked as free. The black lines illustrate the paths of the first star discovery. (b) All voxels that are free and in line-of-sight from the origin of the star discovery are marked light green. The remaining voxels (dark green) are marked interesting. A path towards an interesting voxel is determined (blue line). (c) A second star discovery (black lines) is executed at the new origin (blue). (d) The dark green volume marks again the interesting volume. The algorithm finds a way out of the room.

In star discovery, the MAV flies a star-shape pattern (Fig. 4). In order to generate motion parallax for LSD-SLAM and to discover indentations in the unknown volume, the MAV flies with a 90° heading towards the motion direction. Clearly, the MAV can only fly as far as the occupancy map already contains explored free-space.

The star-shape pattern is generated as follows: We cast m rays from a specific position in the map at a predefined angular interval in order to determine the farest possible flight position along the ray. The traversability of a voxel is determined by inflation of the occupied voxels by the size of the MAV. In order to increase the success of the discovery, we perform this computation at l different heights and choose the result of maximum size.

Only if the star discovery is fully executed, we redetermine the occupancy map from the updated LSD-SLAM keyframe map. This also enables to postpone loop-closure updates towards the end of the exploration process, and provides a full 360° view from the center position of the star discovery.

Our exploration strategy is also favorable for the tracking performance of LSD-SLAM. For instance, flying an outward facing ellipse of maximum size instead it could easily loose track because the MAV will only see few or no gradients when it flies close to an obstacle while facing it.

D. Global Exploration

Obviously, a single star discovery from one spot is not sufficient to explore arbitrarily shaped environments, as only positions on the direct line-of-sight from the origin can be reached (Fig. 5). This induces a natural definition of interesting origins for subsequent star discoveries. We denote a voxel *interesting* if it is known to be free but not in line-of-sight of any previous origin of star discovery.

We determine the interesting voxels for starting a new star discovery as follows: For every previously visited origin of a star discovery, we mark all free voxels in direct line-of-sight as visited. Then all free voxels in the inflated map are

traversed and the ones that have not been marked are set to interesting. With m being the number of star discovery origins, the whole algorithm runs in $O(n^3 \cdot (m + hor^2 \cdot ver))$, where *hor* and *ver* are the number of voxels inflated in the horizontal and vertical directions. We define n as the number of voxels along the longest direction of the bounding box of the occupancy map.

Afterwards, we search a path in the occupancy map to one of the interesting voxels. We look at several random voxels within the largest connected component of interesting voxels and choose the one from which we can execute the largest star discovery afterwards.

As discussed above, frontier-based exploration would not be suitable with our monocular SLAM system, as the frontiers could be located on non-observable occupied structures such as textureless walls. In order to discover these structures, we propose star-shaped local exploration moves. Our global exploration strategy determines new origins for these moves where freespace has been measured behind semi-dense structures that are not on the direct line-of-sight from the previous star discovery origin.

V. RESULTS

We evaluate our approach on a Parrot Bebop MAV in two differently sized and furnished rooms (a lab and a seminar room). We recommend viewing the accompanying video of the experiments at https://youtu.be/fWBsDwBJD-g.

A. Experiment Setup

We transmit the live image stream of the horizontally stabilized images of the Bebop to a control station PC via wireless communication. The images are then processed on the control station PC to implement vision-based navigation and exploration based on LSD-SLAM. All experiments were executed completely autonomous.

We report results of two experiments. The first experiment has been conducted in a lab room and demonstrates our star discovery exploration strategy. In this simpler setting, we neglected depth measurements with high variance estimates and plan the star discovery only in a single height. The second experiment evaluates local (star discovery) and global exploration strategies in a larger seminar room. We enhanced the system in several ways towards the first experiment to cope with the larger environment. We use depth measurements with high variance estimates as free space measurements for occupancy mapping as described in Sec. IV-A instead of neglecting them. In order to increase the possible coverage of the star discovery it is computed on several different heights and the one with the largest travel distance is used. When computing interesting voxels we as well use multiple heights for the center points. The robustness of the star discovery was improved by slightly reducing the maximum distance to the origin and by sending intermediate goals to the PID-controller. Finally, we start LSD-SLAM only right after takeoff, we improved the accuracy of the scale estimation and we readjusted the parameters of the PID-controller, the autopilot and the look-around maneuver.



Fig. 6: Semi-dense reconstruction after the look-around in the first experiment.

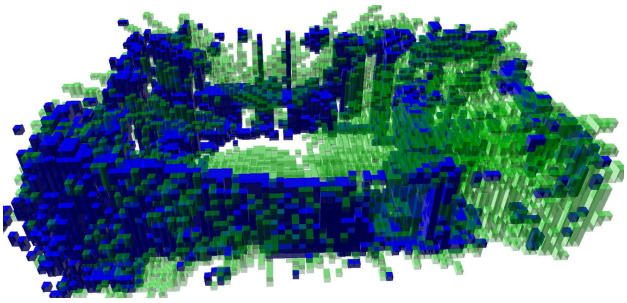


Fig. 7: The difference between 3D occupancy map before and after star discovery. Occupied voxels are shown blue, free voxels that were unknown before the star discovery are green.

B. Qualitative Evaluation

1) Star Discovery: In the first experiment, we demonstrate autonomous local exploration using our star discovery strategy in our lab room. There was no manual interaction except triggering the discovery and the landing at the end. At first, the MAV performs a look-around maneuver. In Fig. 6 one can see the semi-dense reconstruction of the room obtained with LSD-SLAM. Based on a 3D occupancy map, a star discovery is planned (Fig. 9). In this case, we used three voxels in horizontal direction and one voxel in vertical direction to inflate the map.

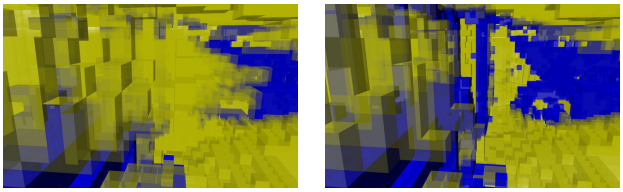
Fig. 10a shows the planned waypoints of the star discovery overlaid with the actual trajectory estimate obtained with LSD-SLAM. Fig. 7 shows how the executed trajectory has increased the number of free voxels in the occupancy map.

2) Full Exploration Strategy:

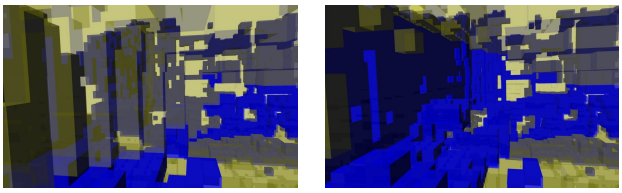
In the second experiment, we demonstrate a star discovery with subsequent repositioning at an interesting voxel in a larger seminar room. First, the MAV took off, initialized the scale, and performed a look-around maneuver. Afterwards, the MAV executed a star discovery. Fig. 10b shows the planned discovery motion and the flown trajectory estimated with LSD-SLAM. We explain the differences by LSD-SLAM

TABLE II: Quantitative results on run-time and occupancy map statistics for the two experiments.

experiment	1		2		
	look-around	star discovery	look-around	star discovery	new origin
occupancy map	5.65s	13.06s	4.25s	38.43s	41.09s
inflating map	0.69s	0.76s	1.40s	2.15s	2.86s
mark voxels in sight	-	-	-	4.93s	4.52/6.63s
way to new origin	-	-	-	0.024s	0.065s
#voxels in bounding box	195048	211968	449565	728416	1312492
#free voxels	36071	46021	75113	106944	159294
#occupied voxels	8259	11477	6102	9673	10816
#free ÷ #known	0.81	0.80	0.92	0.92	0.94
#free ÷ #bounding box	0.18	0.22	0.17	0.15	0.12
#keyframes (approx.)	66	162	54	236	257
total #points	15411528	37828296	3152358	13776972	15002889



(a) before star discovery (b) after star discovery



(c) before star discovery (d) after star discovery

Fig. 8: Indentations in unknown volume before and after the star discovery in the first experiment. Occupied voxels are blue and unknown voxels are yellow. The number of unknown (yellow) voxels is reduced through the discovery.

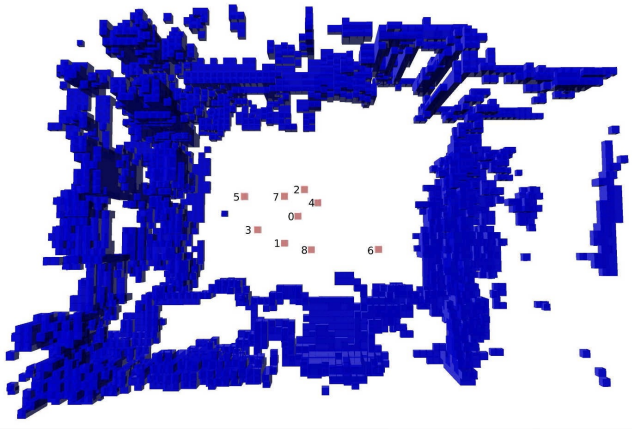


Fig. 9: Exploration plan of the star discovery in the first experiment. Occupied voxels are blue, approached voxels are red and numbered according to their approach order (0: origin).

pose graph updates.

After the star discovery, we obtain the maps and interesting voxels in Fig. 11 and Fig. 12. The largest connected component found by our algorithm is the one outside the room. The MAV planned a path towards it and autonomously executed it. In Fig. 13 we depicted the planned path and the actually flown trajectory estimated with LSD-SLAM.

After reaching the interesting point the battery of the MAV was empty and it landed automatically. The step that our algorithm would have performed next is the star discovery depicted in Fig. 14.

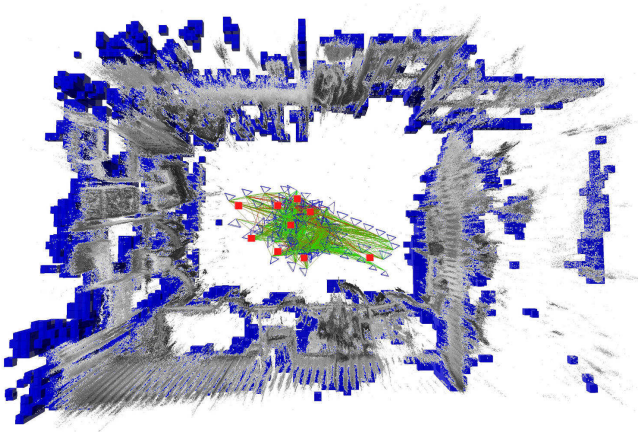
C. Quantitative Evaluation

Table II gives results on the run-time of various parts of our approach and properties of the LSD-SLAM and occupancy mapping processes for the two experiments. The creation of the occupancy map is visibly the most time-consuming part of our method, especially at later time steps when the semi-dense depth reconstruction becomes large. In the second experiment modified parameters were used for the creation of the occupancy map. While they proved to perform better they also further increased the time consumption. The remaining parts are comparatively time efficient and can be performed in a couple of seconds. Our evaluation also shows that star discoveries significantly increase the number of free voxels in the map.

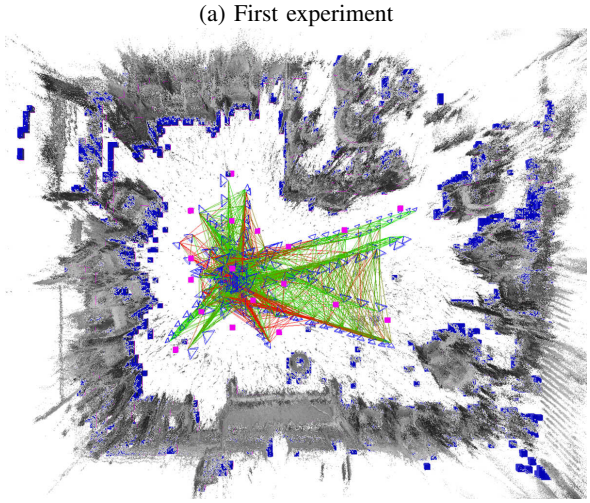
VI. CONCLUSIONS

In this paper, we proposed a novel approach to vision-based navigation and exploration with MAVs. Our method only requires a monocular camera, which enables low-cost, lightweight, and low-power consuming hardware solutions. We track the motion of the camera and obtain a semi-dense reconstruction in real-time using LSD-SLAM. Based on these estimates, we build 3D occupancy maps which we use for planning obstacle-free exploration maneuvers.

Our exploration strategy is a two-step process. On a local scale, star discoveries find free-space in the local surrounding of a specific position in the map. A global exploration strategy determines interesting voxels in the reachable free-space that is not in direct line-of-sight from previous star

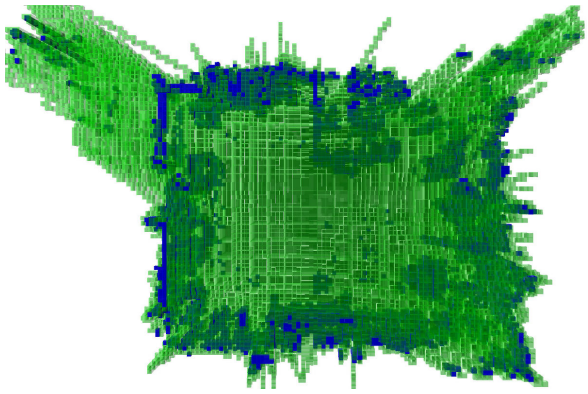


(a) First experiment

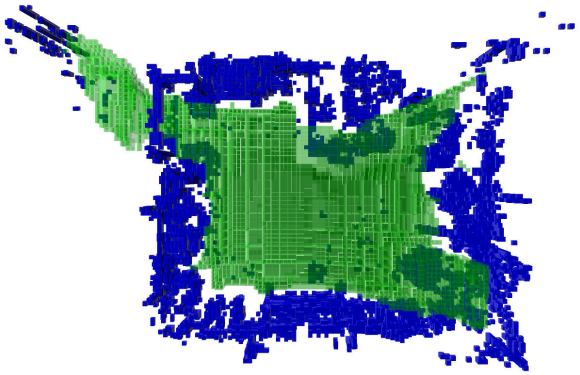


(b) Second experiment

Fig. 10: 3D occupancy map manually overlaid with semi-dense reconstruction, planned waypoints of the star discovery and executed trajectory (planned waypoints in red or pink, occupied voxels in blue, actual trajectory as estimated by LSD-SLAM as blue cameras, loop closure constraints as green lines).



(a) Free voxels (green)



(b) Traversable voxels (green) in the inflated map

Fig. 11: Occupied (blue), free and traversable voxels in the second experiment.

discovery origins. In experiments, we demonstrate the performance of LSD-SLAM for vision-based navigation of a MAV. We give qualitative insights and quantitative results on the effectiveness of our exploration strategy.

The success of our vision-based navigation and exploration method clearly depends on the robustness of the visual tracking. If the MAV moves very fast into regions where it observes mostly textureless regions, tracking can become difficult. A tight integration with IMU information could benefit tracking, however, such a method is not possible with the current wireless transmission protocoll for visual and IMU data on the Bebop.

Also a more general path planning algorithm based on the next best view approach is desirable. This however requires a more efficient way to refresh the occupancy map when pose graph updates happen.

In future work we will extend our method to Stereo LSD-SLAM [17] and tight integration with IMUs. We may also use the method for autonomous exploration on a larger MAV with onboard processing.

References

[1] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, 1997.

[3] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *Internaional Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[4] N. Basilico and F. Amigoni, "Exploration strategies based on multi-criteria decision making for searching environments in rescue operations," *Autonomous Robots*, vol. 31, no. 4, pp. 401–417, 2011.

[5] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

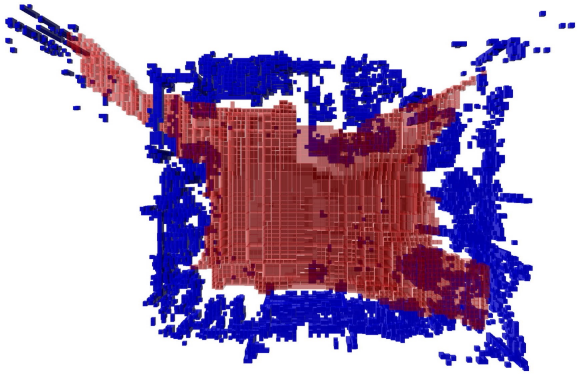
[6] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard, "Autonomous exploration for 3D map learning," in *Autonome Mobile Systeme (AMS)*, 2007.

[7] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," in *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2005.

[8] I. M. Rekleitis, "Single robot exploration: Simultaneous localization and uncertainty reduction on maps (slurm)," in *CRV*, 2012.

[9] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 9–15.

[10] L. Yoder and S. Scherer, "Autonomous exploration for infrastructure modeling with a micro aerial vehicle," in *Field and Service Robotics*, 2015.

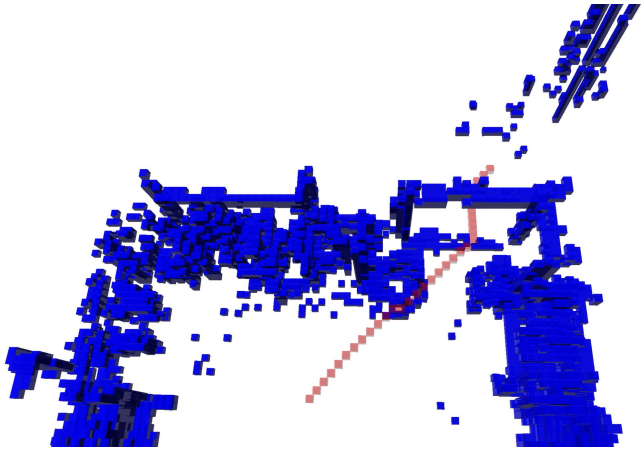


(a) Traversable voxels (red) in line-of-sight from the star discovery origin.

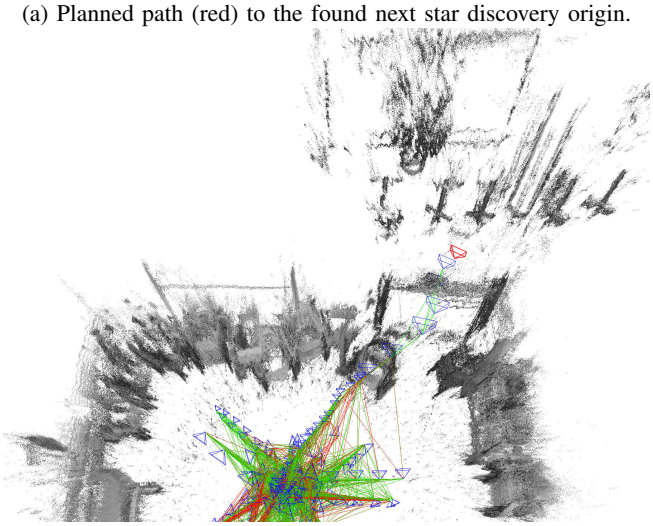


(b) Interesting voxels (pink)

Fig. 12: Voxels in line of sight and interesting voxels in the second experiment.



(a) Planned path (red) to the found next star discovery origin.



(b) Flown trajectory (blue) to the interesting point estimated with LSD-SLAM.

Fig. 13: Planned and executed path in the second experiment.



Fig. 14: Plan (red voxels) for the second star discovery in the second experiment.

[11] S. Nuske, S. Choudhury, S. Jain, A. Chambers, L. Yoder, S. Scherer, L. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for a UAV navigating rivers," *Journal of Field Robotics*, 2015.

[12] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1071–1078.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468.

[14] V. R. Desaraju, N. Michael, M. Humenberger, R. Brockers, S. Weiss, J. Nash, and L. Matthies, "Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing," *Autonomous Robots*, vol. 39, no. 3, pp. 445–463, 2015.

[15] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems (RAS)*, vol. 62, no. 11, pp. 1646–1656, 2014.

[16] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[17] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *IROS*, 2015.

COACHES: An assistance Multi-Robot System in public areas

L. Jeanpierre*, A.-I. Mouaddib*, L. Iocchi†, M.T. Lazaro†, A. Pennisi‡, H. Sahli‡, E. Erdem§, E. Demirel§ and V. Patoglu§
*GREYC Lab, University of Caen, France
†Dept. of Computer, Control and Management Engineering, Sapienza University of Rome, Italy
‡Vrije Universiteit Brussel, Belgium
§Sabanci University, Turkey

***Abstract*— In this paper, we present a robust system of self-directed autonomous robots evolving in a complex and public spaces and interacting with people. This system integrates high-level skills of environment modeling using knowledge-based modeling and reasoning and scene understanding with robust image and video analysis, distributed autonomous decision-making using Markov decision process and Petri-Net planning, short-term interacting with humans and robust and safe navigation in overcrowding spaces. This system has been deployed in a variety of public environments such as a shopping mall, a center of congress and in a lab to assist people and visitors. The results are very satisfying showing the effectiveness of the system and going beyond just a simple proof of concepts.**

I. Introduction

Public spaces in large cities are increasingly becoming complex and unwelcoming environments. Public spaces progressively become more hostile and unpleasant to use because of the overcrowding and complex information in signboards. It is in the interest of cities to make their public spaces easier to use, friendlier to visitors and safer to increasing elderly population and to citizens with disabilities. Meanwhile, we observe, in the last decade a tremendous progress in the development of robots in dynamic, complex and uncertain environments. The new challenge for the near future is to deploy a network of robots in public spaces to accomplish services that can help humans. Inspired by the aforementioned challenges, COACHES project addresses fundamental issues related to the design of a robust system of self-directed autonomous robots with high-level skills of environment modeling and scene understanding, distributed autonomous decision-making, short-term interacting with humans and robust and safe navigation in overcrowding spaces.

COACHES project provides a modular architecture integrated in robots. We deployed COACHES at Caen city in the "Rives de l'Orne" shopping mall. It is a cooperative system based on fixed cameras and mobile robots. The fixed cameras can do object detection, tracking and events detection (objects or behavior). The robots combine these information with the ones perceived via their own sensors, to provide information through its multi-modal interface, guide people to their destination, show tramway stations and transport goods for elderly people, etc.... The COACHES robots use different modalities (speech and displayed information) to interact with the mall visitors, shopkeepers and mall managers. The project has enlisted an important end-user (Caen la mer) providing the scenarios where the COACHES robots and systems will be deployed.

II. Overall system description

A. General Architecture and functionalities

The general architecture of the embedded software, in each robot, has the following components (Figure 1): (i) *Modelling and reasoning*: Modelling a variety of static/dynamic knowledge about the environment formally in three knowledge bases (KB): (SEM) static properties of entities in the environment and their relations, (COM) general static common sense knowledge about shopping malls (taxonomic knowledge and defaults), and (TEMP) short-term/long-term temporary knowledge about entities in the environment obtained from observations and interactions with humans. Utilizing SEM and TEMP, COM also involves common sense knowledge to derive simple goals of assistance, advertisement and security. These goals are sent to the planner. (ii) *Perception*: A system for detecting and tracking people using fixed cameras has been implemented. Such a system is able to detect particular events such as entering in a particular area of the shopping mall and sending an advertising message to the robot. (iii) *Interaction interface*: A GUI has been developed allowing a person to interact with the robot and expressing his requests in terms of assistance. The GUI has been realized in order to be fully customizable for different scenarios and different user profiles, allowing for personalized short-term interactions as described in [1]. (iv) *Markov Decision Process and Petri-Net Plans planning*: a task language based on Progressive Reasoning Units (PRUs) has been developed to express all assistance tasks of the robot, an appropriate MDP-based planning algorithm to computed the policy to accomplish the task and PNP-based planning to transform a policy into an execution Petri-Net plan to deal with execution error. Details on the implementation of this framework are provided in [2]. For the development of the software robotics components, we used the Robot Operating System (ROS) (www.ros.org), which is the standard middleware for robotics applications. In particular, we used the last stable version ROS Indigo and the last LTS (Long Term Support) version of the Linux/Ubuntu Operating System.

We then extend this architecture to multi-robot settings by developing the following general principles : let robots $A = \{A_1, A_2, \ldots, A_n\}$ receive from different KB modules

a set of goals $G = \{g_1, g_2, \ldots, g_k\}$ where goals concern advertisement, patrolling, assisting and escorting. We consider that goals are sorted according to their type assuming that the type convoys some goal's priority. In our case, we consider that security goals have a higher priority than assistance goals and the advertisement goals have the lowest priority.

Let assume that KB modules **communicate** their local generated goals to each other leading to the same set of goals handled by different decision-making modules of different robots (Figure 1). The general principale consists in receiving information from external sensor (external cameras in our case) by communicating these information through the network (wireless). The KB modules receive the same information and thus generate the same list of goals. Each KB sends this list of goals to its decision making module to generate a policy of accomplishing one or many goals.

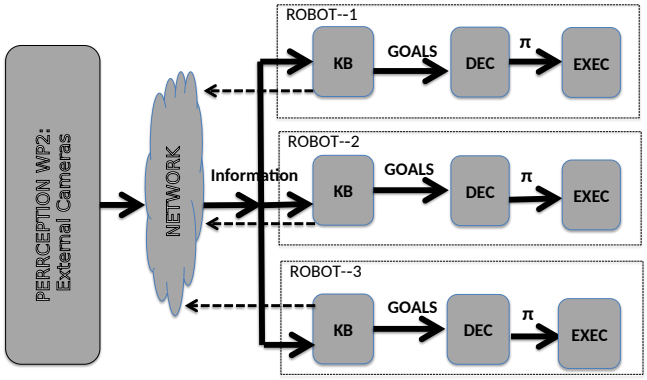


Fig. 1. General architecture of multi-robot decision-making module

B. Communication

KB modules communicate to exchange the information concerning the status of execution and also the level of interruptibility allowing, at the receipt of a list of goals, to consider only robot that could accomplish the goals according to their current status. The list of messages exchanged between different modules (KB, DEC and EXEC) and between robots are : $msg_new_facts_\#id$ coming from the local perception to the local KB; $msg_new_external_facts_\#id$ coming from the local KB to the other robots; $msg_end_local_goals_\#id$ coming from the local EXEC module to the local KB; $msg_end_external_goals_\#id$ coming from the local KB to the other robots; $msg_selected_goals_\#id$ coming from the local DEC module to KB local; $msg_selected_external_goals_\#id$ coming from the local KB to the other robots; $msg_goals_values_\#idrobot$ coming from the other robots.

The general principal is depicted in Figure 2 where each robot has a local Knowledge base, a local decision maker and a local executor interacting each other and exchanging information with the other robots through a communication infrastructure. In our current case, we consider a direct communication between robots assuming that they evolve in an environment where their ranges of communication cover all the space as in the mall. Limited communication ranges issues are let to the future work. However, we use a procedure allowing robots to move towards a central point to establish the communication and update their KB.

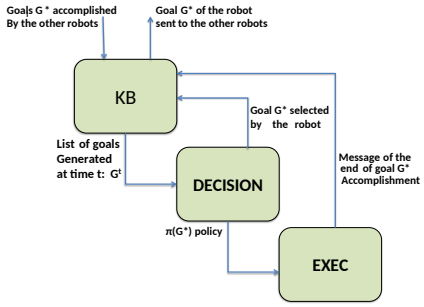


Fig. 2. Communication between KB, decision and execution modules

III. Image and Video Analysis for Perception

In this section, we describe the adopted techniques for detecting and tracking people from fixed and mobile cameras. Fixed cameras are mounted in the shopping mall to analyze the behavior of the people in order for detecting particular events (e.g. interactions, standing in front of a shop for a while, etc.). Mobile cameras, mounted on the robot, are used for detecting and tracking people in the environment during the navigation of the robot and are useful for approaching people. A detailed description of both the systems is provided in the following sections.

A. Fixed Camera System

The fixed cameras are used for detecting specific events, like entering a specific area of the shopping mall, and sending the coordinates of the detected event to the robot. The system is based on the following steps: (i) Background Subtraction, (ii) Person Detection, (iii) Re-projection on the map, (iv) Tracking, and (vi) Event Detection (see the general scheme in Fig. 3).

In order to reduce the person search space in the image, a background subtraction method has been adopted. In particular the Fastest Adaptive Foreground EXtraction (FAFEX) method of Pennisi *et al.* [3] has been used. The output of FAFEX is a foreground mask containing the blobs, which are the possible candidates to be people. Each blob is classified as *person* or *not a person* by using a Support Vector Machine (SVM) classifier, based on Histograms of Oriented Gradients (HOG) [4], trained on the INRIA Person dataset [1]. Then, the detections are reprojected onto a map of the shopping mall by using a Homography technique. To track the people inside the environment, a tracker, called PTracking [5], has been used. Then, in order to detect events such as entering a particular area of the shopping mall, an event detection module has been developed. By selecting specific areas of the shopping mall, such a module is able to recognize if a person entered one of those areas, which could be used as inputs to the navigation system of the robot.

[1] `http://pascal.inrialpes.fr/data/human/`

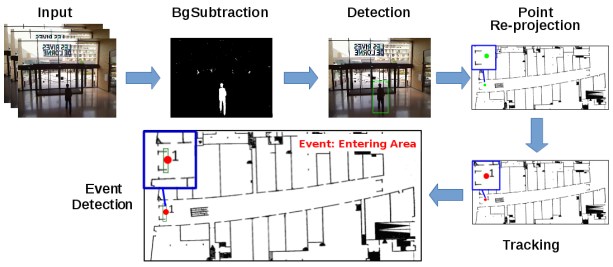


Fig. 3. Fixed Camera System scheme.

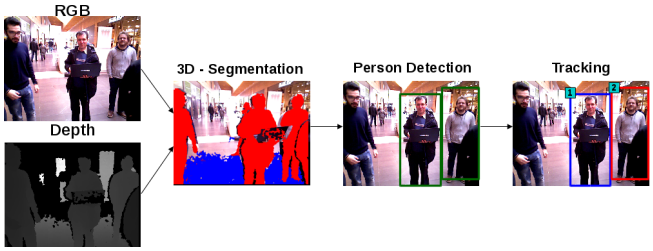


Fig. 4. The 3D Person Detection and Tracking module is based on 3 main steps: 1) 3D Segmentation, 2) Person Detection, and 3) Tracking.

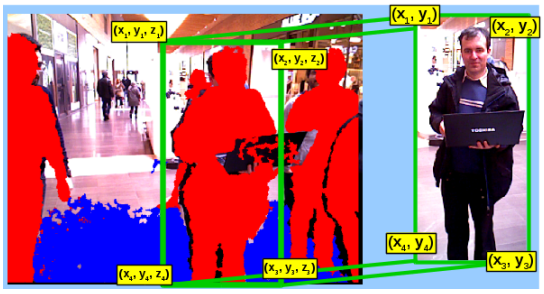


Fig. 5. The bounding box of the 3D cluster is converted in 2D image coordinates for extracting the correspondent RGB patch.

B. Robot Perception

The cameras mounted on the robot are used for detecting and approaching the people inside the shopping mall. To this end, we developed a framework for 3D person detection and tracking framework as shown in Fig. 4. The system is based on three steps: (i) scene segmentation, (ii) person detection, and (iii) person tracking. The robot is equipped with an RGB-D camera (i.e. Asus Xtion), which is mounted on the top of the robot. Thanks to the depth and camera location information, the system is able to straighten the 3D scene and to compute the point cloud. Then, the points under $5cm$ of height are excluded from the cloud assuming that they belong to the floor. The same assumption is made for all the points above $2m$ of height (we assume that a person is tall less than 2 meters). The remaining points are grouped by using a clustering method based on the Euclidean distance [6]. Each cluster is considered as a candidate to be a person. In order to verify this, RGB patches are extracted by converting the 3D bounding box of each cluster in a 2D image bounding box (see Fig. 5).

To recognize if a patch is a person, an approach based on the Aggregate Feature Channels (ACF) [7] has been adopted. The ACF method computes the features as the aggregation of multiple features. In our current implementation we used

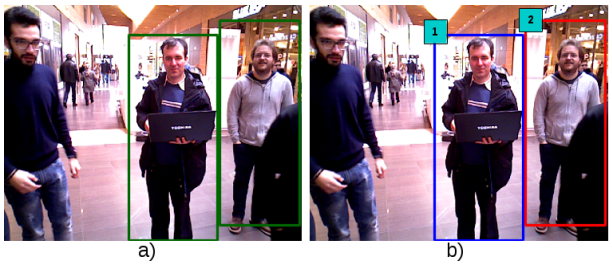


Fig. 6. a) Detection: the green bounding boxes represent the detections and b) the numbered box are the tracks.

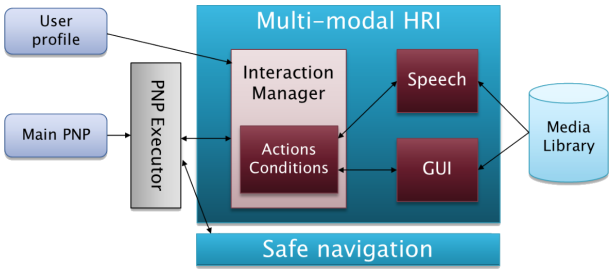


Fig. 7. Multi-modal HRI architecture.

the L, U, and V color components, the HOG features and, the gradient magnitude. All these features are integrated into the Fastest Pedestrian Detector in the West (FPDW) [8]. Then, to classify a single patch, a boosting tree classifier has been trained by using the INRIA person dataset. Such a strategy increases the detector speed maintaining robust detection performance (see Fig. 6a). Finally, a multiple target tracker is used for tracking the detected people. The tracker uses the appearance model of each detection together with a euclidean distance approach for carrying out the data association step. The target appearance is represented by using the combination of the RGB color information and the Speed Up Robust Features (SURF) [9]. Such features are reduced by using a sparse dictionary. Then, an on-line Adaboost classifier, one for each detection, is trained using such a dictionary. A Kalman filter is used for filtering out the noise of each detection (see Fig. 6b). A track is assigned to the current detection if, the confidence of one classifier is the largest possible and is greater than a minimum threshold c (experimentally evaluated), and if the euclidean distance between the related Kalman prediction and the current detection position is less than a predefined threshold e (experimentally evaluated).

Thanks to such a system, the robot is able to track the people inside the environment and to understand if a person wants to approach it by measuring the relative proximity distance.

IV. Multi-Modal Human-Robot Interaction

The interactions realized in the COACHES scenario are characterized by being short in time with many users who are not expert and who have not been trained about the capabilities of the robot. In this context, the use of an HRI system offering multiple modalities of interaction can greatly increase its usability and improve user experience, since the users can choose the modality more comfortable to them.

Figure 7 illustrates the overall architecture of our HRI system. Inputs to the system are a Petri Net Plan (PNP) describing the robot behavior, a user profile and a multimedia library.

The PNP is generated by the reasoning and planning module of the system (as it will be described in next section V) and contains both *robotic* actions (e.g., move, goto) and *interaction* actions. The execution of this plan is managed by the PNP Executor module which derives the execution of each action to the safe navigation component in the case of robotic actions or to the multi-modal HRI component in the case of interaction actions.

The execution of interaction routines is managed by the Interaction Manager (IM). The IM acts as a server that executes the interaction actions when requested by the PNP Executor and returns the input of the user in the form of PNP conditions which are evaluated by the PNP Executor to enable the corresponding transitions to make evolve the course of the PNP.

Currently, the IM manages two components: a C# speech server using the multi-language Microsoft Speech Recognition and Synthesis engine and, a Python GUI on a touch-screen. This allows for the use of the following modalities of interaction: output of information is provided visually in the form of texts or images displayed on the Python GUI or by voice using a Text-to-Speech (TTS) component while the input from the user can be given by using the touch-screen or via spoken commands interpreted by the Automatic Speech Recognition (ASR) system.

V. Distributed reasoning, decision-making and execution

A. Semantic reasoning

Since the relevant knowledge about the environment is heterogeneous (e.g., static/dynamic, spatial/nonspatial), we classify the knowledge bases into three parts: Semantic Map (SEM), Commonsense Knowledge Base (COM), and Temporary Knowledge Base (TEMP). SEM and COM represent the static knowledge about the environment while TEMP represents the temporary knowledge and can be updated due to observations or human-robot interactions.

1) Knowledge Base for Semantic Map: We define a semantic map (SEM) for an environment which consists of a set of entities, a set of spatial relations and a set of non-spatial relations. The entities include access points (e.g., doors) of places like stores, restaurants, elevators, escalators, restrooms, etc. The spatial relations include qualitative spatial relations, like "next-to" and "up-down" directions. The nonspatial relations include the names of stores, what kind of objects they sell and wheel-chair accessibility condition. Overall, these two sorts of relations describe static knowledge about the shopping mall.

Entities of a shopping mall can be represented by a set of atoms of the form `entity(entityID)`. **Qualitative spatial relations** of the entities in the environment can be represented by a set of atoms, like `acc(store1, store2)` (to describe accessibility of `store1` from `store2`) or `dir(store1, store2, next-to)` (to describe relative direction of `store1` from `store2`). **Nonspatial relations** of entities can be represented by atoms, like `name(store1ID, abc)`, `sells(abc, shirts)`, and `hasRamp(store1ID)`. We represent the names of the stores, which product they sell and if they have a ramp for stroller or wheelchair access as nonspatial relations. These nonspatial relations are necessary for computing a personalized path. For example, if a customer with stroller asks "Where can I buy shoes?", the robot should consider stores which sell shoes and has ramp for stroller access. After finding the possible goal locations, the path finder module computes the path. This computation also requires nonspatial relations, because the path should not include routes without stroller access.

We can represent the entities and qualitative spatial relations as a directed graph. The vertices of the graph denote the entities, whereas the edges denote the accessibility relations of the entities. We can also label the edges with the directionality relations.

2) Knowledge Base for Commonsense Knowledge: We define commonsense knowledge base (COM) in a shopping mall with two parts. The first part is about taxonomic relations between entities (e.g., "French restaurant is a restaurant"). We represent these relations as an ontology. The second part is about default knowledge related to shopping malls (e.g., "Children usually desire toys"). Since these relations necessitate nonmonotonicity, we represent them as ASP rules.

Taxonomic knowledge of a shopping mall consists of hierarchies of classes and their relationships. We model the nonspatial relations of entities, and taxonomic commonsense knowledge about these entities as a formal Shopping Mall Ontology. We represent this ontology in OWL (Web Ontology Language) [10], [11], and use DL reasoners, such as PELLET [12], to extract relevant knowledge from the ontology using the query language Sparql [13]. **Default knowledge** about entities in a shopping mall can be represented in Answer Set Programming (ASP) [14]. For instance, we can represent the default commonsense knowledge "Normally, a package belongs to the adult next to it." by the following rule:

```
belongs(X,P) :- object(X,package), nextTo(X,P),
  instanceOf(P,adult), not -belongs(X,P).
```

Similarly, we can represent "Normally, a package is suspicious if it does not belong to anyone." by a rule in ASP. Then, if the robot sees a package which does not belong to anyone, it can infer (using the ASP solver CLINGO [15]) that the package is suspicious with the commonsense knowledge.

3) Knowledge Base for Temporary Knowledge: We define temporary knowledge base (TEMP) in a shopping mall as ASP facts, like `disabled(c1)`, `promotionAt(store1)`, `noAccess(elevator)`, and `interestedToBuy(c1, cosmetic)`. These knowledge can be obtained from observations via perception or from human-robot interactions. The mall manager may tell the robot that the elevator is broken. The robot can recognize

that the customer asking a question is at a wheelchair. The shopkeepers may tell the robot that they have some promotions over the weekend. These temporary knowledge can be represented as follows:

The robot will use this temporary knowledge, in addition to the knowledge bases SEM and COM, to infer a list of possible goals. Whenever a customer asks for a place, we add that place with `goalLoc(X)` predicate. Whenever a customer asks for a product, we add that product with `interestedToBuy(C,X)` predicate.

B. Distributed decision-making

Once local KB synchronized, each robot computes the value of its optimal policy to accomplish a goal and thus communicates a vector of values to the other robots. Each robot computes the vector values of the goals and receives from the others their vector values. From these exchanged information, each robot maintains a matrix of values of the couple (robot, goal). This global information gathered from local information allows each robot to select the best goal using a distributed market-based auctioning. Indeed, each robot i maintains a matrix M_i^h per goal priority h. Robots concern goals of lower priority when goals of higher priorities are allocated. The matrix is constructed as follows:

1) each robot i computes the optimal value V_i^{*,g_l} to accomplish goal g_l. Value vector $(V_i^{*,g_1}, V_i^{*,g_2}, \ldots, V_i^{*,g_k})$ represents the values of optimal policies accomplishing goals in the list. This vector represents the line i of the matrix.
2) Each robot j sends its value vector to each others, allowing them to complete their matrix
3) Each robot i has thus a matrix :

$$M_i^h = \begin{pmatrix} V_1^{*,g_1} & V_1^{*,g_2} & \ldots V_1^{*,g_k} \\ V_2^{*,g_1} & V_2^{*,g_2} & \ldots V_2^{*,g_k} \\ \ldots & & \\ V_n^{*,g_1} & V_n^{*,g_2} & \ldots V_n^{*,g_k} \end{pmatrix}$$

The allocation of goals to the robots is performed by a distributed decision-theoretic market auction, where each robot i computes for each goal g a value $\mathbf{V_i(g)}$)of following a policy accomplishing it. The agent α proposing the best value is elected to accomplish this task.

$$(\alpha, g^*) = argmax_{A_i, g_k} V_{A_i}(g_k) \quad (1)$$

It's possible there exist many robots α able to accomplish the goal g^* with the same value and thus the equation has a lot of solutions. When many robots optimize the accomplishment of the goal g^*, we allocate the goal to the robot with the minimum regret. The *Regret* of not accomplishing a goal g^* is a loss in value when accomplishing the best goal other than g^*. More formally,

$$regret_j(g) = V_j^{\pi^*}(g) - \max_{g' \neq g} V_j^{\pi^*}(g')$$

Let S_g be the set of robots α optimizing the value of accomplishing the goal g (solutions of Equation 1), the best robot to which we allocate the goal g is given by the following equation:

$$\alpha^* = \min_{\alpha \in S_g} regret_\alpha(g)$$

If this equation has many solutions, we can proceed in the same way with the other goals and so on.

C. Execution monitoring

A crucial feature for deploying robots in public spaces is their ability of reliably executing their plans in presence of uncertainty about the world and the user interactions and of perception noise.

During the COACHES project, we have studied a method for improving robustness of the plan generated by the decision making modules. This processing step (called *Robustification*) during the plan generation phase aims at improving the robustness of the plan to situations that are not modelled in the planning domain.

To this end, we use the Petri Net Plan (PNP) formalism [16] and the plan execution framework is formed by the following elements: 1) Plan translation into PNP [17], 2) Execution Rules (ER) [2], 3) ROS action execution.

Briefly, the policy or the conditional plan generated by the planner is first transformed into a PNP. This is a straightforward procedure since Petri Nets can easily represent trees and DAG. Finally, the actual execution of the robot actions and interactions is performed by the PNP engine. Each action name in PNP is mapped into an action that can be either a robotic action or an interaction action (interpreted by MODIM, as described in Section IV).

VI. Evaluation methodology and results on real robots

A. Evaluation methodology

Evaluation of a complex system like the one developed in the COACHES project requires a specific methodology and setup. While the evaluation of the components presented in the previous sections of this paper is described in the relative papers, here we want to focus on the overall evaluation of the entire system by the actual users (i.e., customers of the shopping mall).

For such a user evaluation, we have designed the experimental protocol that is described in this section. The actual execution of the experiment is planned for July 6-9, 2017 in a public event in the shopping mall Rives de l'Orne in Caen, where we aim at involving around 100 users[2]. The experimental protocol we propose here has been successfully applied in the different scenario of a teaching assistant robot [18], where several COACHES components were used to implement the system.

More specifically, the experimental protocol will be based on the Godspeed Questionnaire Series (GQS) [19], a common evaluation method for HRI. GQS will be used to asses the success of the robot, evaluating the emotional states

[2]Results of these experiments will thus be ready by mid-July 2017.

of people during the interaction. The questionnaire will be submitted to users (customers and vendors of the shopping mall) characterized by different features (independent variables). The measure of the GQS features (dependent variables) and the consequent statistical analysis will allow for an important user evaluation of the COACHES robots.A sketch of the experimental methodology is the following: (1) Definition of the independent variables (e.g., type of user: customer/vendor, gender, age, task to be executed, type and level of interaction with the robot, etc.); (2) Selection of the users, following a *between-user* approach, each user will participate to a single test; (3) Filling the paper questionnaires by the users before and after the experience with the robot; (4) Statistical analysis of all the collected data.

The output of the statistical analysis will be useful to assess the effectiveness of the COACHES approach in achieving the considered tasks as well as the user feelings about the system.

B. Public demonstration

We ran a demonstration of the two robots collaborating to guide visitors to researchers offices and to various services in our lab with the same tasks as in the mall. The mission was made of 3 layers: first, the robots come to predefined wait-points near entrances, where they offer assistance to visitors. Second, visitors use the touchscreen (fig.8) to ask for services like a specific office. Finally, the robot escorts the visitor (fig. 9.a) to the destination and then returns to a free wait-point. Each step is planned jointly by the two robots to avoid conflicts (fig. 9.b).



Fig. 8. Interaction with visitors



(a) (b)

Fig. 9. a) Escorting to office; b) Robots coordination

VII. Conclusion and future works

We presented a practical and novel models for cooperative robots for assistance in public areas by sharing tasks and interacting with people. We developed a framework allowing a fleet of robots to reason and synchronize their local static and dynamic KBs by exchanging appropriate information and to develop an augmented MDP using a value matrix for a marked-based auctioning to better coordinate the robot activities. We also presented a module of perception dedicated to the people detection and tracking, face detection and 2D escorting with different algorithms and a multi-modal human robot interaction. We developed a software allowing a fleet of robots to assist in a cooperative way a group of peoples by sharing autonomously tasks and maintain interaction with people they assist. A video is available on the web site presenting the overall behavior of the system with two robots at `greyc.coaches.fr`.

References

[1] L. Iocchi, M. T. Lázaro, L. Jeanpierre, and A.-I. Mouaddib, "Personalized short-term multi-modal interaction for social robots assisting users in shopping malls," in *Int. Conf. on Social Robotics*, Paris, France, Oct 26-30 2015.

[2] L. Iocchi, L. Jeanpierre, M. T. Lázaro, and A.-I. Mouaddib, "A practical framework for robust decision-theoretic planning and execution for service robots," in *Proc. of Int. Conf. on Automated Planning and Scheduling (ICAPS)*, London, UK, June 12-17 2016, pp. 486–494.

[3] A. Pennisi, F. Previtali, D. D. Bloisi, and L. Iocchi, "Real-time adaptive background modeling in fast changing conditions," in *12th IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, 2015, pp. 1–6.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 - Volume 01*, 2005, pp. 886–893.

[5] F. Previtali and L. Iocchi, "Ptracking: Distributed multi-agent multi-object tracking through multi-clustered particle filtering," in *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 2015, pp. 110–115.

[6] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, Oct. 2009.

[7] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1532–1545, 2014.

[8] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *Proceedings of the British Machine Vision Conference*, 2010, pp. 68.1–68.11.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[10] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From shiq and rdf to owl: the making of a web ontology language," *J. Web Sem.*, vol. 1, no. 1, pp. 7–26, 2003.

[11] G. Antoniou and F. van Harmelen, "Web ontology language: Owl," in *Handbook on Ontologies*, 2004, pp. 67–92.

[12] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007, software Engineering and the Semantic Web.

[13] E. PrudHommeaux, A. Seaborne, *et al.*, "Sparql query language for rdf," *W3C recommendation*, vol. 15, 2008.

[14] G. Brewka, T. Eiter, and M. Truszczynski, "Answer set programming at a glance," *Commun. ACM*, vol. 54, no. 12, pp. 92–103, 2011.

[15] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. T. Schneider, "Potassco: The potsdam answer set solving collection," *AI Commun.*, vol. 24, no. 2, pp. 107–124, 2011.

[16] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara, "Petri net plans - A framework for collaboration and coordination in multi-robot systems," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 3, pp. 344–383, 2011.

[17] V. Sanelli, M. Cashmore, D. Magazzeni, and L. Iocchi, "Short-term human robot interaction through conditional planning and execution," in *Proc. of International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

[18] P. Ferrarelli, M. T. Lazaro, and L. Iocchi, "Design of robot teaching assistants through multi-modal human-robot interactions." in *Proc. of International Conference on Robotics in Education (RiE)*, 2017.

[19] A. Weiss and C. Bartneck, "Meta analysis of the usage of the godspeed questionnaire series," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2015, pp. 381–388.

Semantic Monte-Carlo Localization in Changing Environments using RGB-D Cameras

Marian Himstedt, Erik Maehle

***Abstract*—The localization with respect to a prior map is a fundamental requirement for mobile robots. The commonly used adaptive monte carlo localization (AMCL) can be found on most of the mobile robots ranging from small cleaning robots to large AGVs. While achieving accurate pose estimates in static environments, this algorithm tends to fail in the presence of significant changes. Recently published extensions and alternatives to AMCL observe the environment over longer times while building complex spatio-temporal models. Our approach, in contrast, utilizes object recognition and prior semantic maps to enable robust localization. It exploits the fact that putative changes in the environment can be *predicted* based on prior semantic knowledge. Our system is experimentally evaluated in a warehouse environment being subject to frequent changes. This emphasizes the importance of our algorithm for challenging industrial applications.**

I. Introduction

Mobile robots require a robust estimate about their pose with respect to a prior map to provide services and to ensure safe navigation. Occupancy grid maps are commonly used since they provide valuable input not only for localization, but also for path planning and obstacle avoidance. A number of requirements for all of these modules have to be met in order to obtain a robust system while avoiding to maintain multiple maps for each of them. Special attention is required in changing environments. For localization, it is necessary to find sufficient correspondences among the prior map and the current observation. For global path planning, it is beneficial to avoid all known obstacles, while local planning layers can account for minor changes or dynamic objects. It appears rather important to ensure that the global topology in terms of traversability is not violated.

Existing systems for map-based localization can be summarized as follows, they:

1) assume a static environment or
2) continuously map and preserve the latest state for future tasks or
3) have to observe their working space over longer times in order to identify and predict systematic changes

It is rather unlikely that implementations based on (1) are able to provide robust localization results in changing environments. Depending on the kind and extent of change, the pose estimation can slightly drift or completely diverge from the true pose. Systems based on category (2) are likely to risk the above mentioned requirement of global traversability of path planning. If, for instance, a large obstacle is placed inside a hallway, which is observed by a mobile robot, this subspace in the map will be labeled *occupied* based on a single observation. This, in turn, might entail that parts of the map become inaccessible from the path planning's perspective until the update will be reverted based on novel observations. This event can be noticed in numerous robotic applications, e.g. pallets and parked vehicles in warehouse hallways or humans in populated public spaces. Without further evidence about the nature of change, the systems based on (2) might ensure localization performance but potentially generate unnavigatable maps. The observation and incorporation of changes in map subspaces over time is investigated by approaches based on the category (3). These were demonstrated to achieve promising results in localization while still maintaining valid maps for navigation. The underlying models, however, typically require a quantitative number of observations for achieving relevant information about map subspaces that can be utilized for predictions and uncertainty estimation. Due to the missing domain knowledge, the process can become of substantive complexity when analyzing individual grid cells of large maps while raising the crucial question whether robotic navigation can actually benefit from all this information. This becomes particularly apparent when modeling the free space of large corridors being frequently passed by humans or robots. Visited cells of this space constantly undergo state transitions. How is this distinguished from transitions caused by semi-static objects as, for example, pallets at reloading points or parked vehicles?

Contrary to the systems based on (1) - (3), we introduce Semantic Monte Carlo Localization (SMCL) which augments existing methods by object recognition to enable robust localization while maintaining valid maps for navigation in changing environments. Our approach copes without the need of long-term observations or continuous mapping. The key idea is to recognize objects being commonly present in the target environment. Our approach utilizes semantic maps being generated once based on the common object classes. Our measurements are originated from a RGB-D camera providing depth and color data from which object classes can be inferred. The probabilistic association of places in the map and observations is supported by this information. Measurements are incorporated according to the expected contribution of the associated object class which can be determined a-priori.

Our approach is experimentally evaluated on an automated

All authors are with the Institute of Computer Engineering, University of Lübeck, Germany. E-mail: himstedt@iti.uni-luebeck.de

guided vehicle (AGV) in a warehouse environment which is subject to frequent changes. This application requires reliable localization over periods of time. Full autonomy is required once the AGV has passed a teach-in drive with the support of a human supervisor. We demonstrate that SMCL is able to robustly estimate the AGVs global pose throughout all experiments. We expect that our approach is valuable for service robots in environment types being subject to changes whose roots can be determined a-priori: e.g. humans in populated environments, palleted goods in warehouses or cars on streets.

Our key contributions:

- Robust localization in changing environments using semantic perception
- No continuous observation or mapping of environment
- Use of RGB-D cameras instead of laser range finders

II. Related Work

This sections provides an overview of work which is related to localization in changing environments. This mainly addresses the research field of map-based localization, however, for the sack of completeness, we further include relevant work in place recognition and SLAM focusing on changing environments.

A. Map-based Localization

The most common method for map-based localization is given by the Monte Carlo localization (*mcl*), often implemented and referred to as adaptive Monte Carlo Localization (*amcl*) [19]. The majority of work in this field are extensions of *amcl.* The early work of Thrun et al. suggests to explicitly detect and model short readings being a frequent source of association problems due to dynamic objects such as humans [19]. Thanks to this extension, the authors reported on more reliable pose estimations for an automated tour guide robot in a museum [2]. Meyer-Delius et al. presented the idea of temporal maps being able to capture dynamic changes over multiple time frames through agglomerated incorporation of temporary local maps and a global map [12]. With the experience gained from this idea, Tipaldi et al. further suggested the use of Hidden Markov Models (HMM) to model state transitions of individual grid cells of a global map [20]. Their system learns state transitions «*free* $\rightarrow$ *occupied*» and vice versa. A Rao-Blackwellized particle filter (RBPF) is used to infer the robot pose in this map representation. Saarinen et al. presented the concept of independent markov chains which, similarly to [20], are learnt for each grid cell [16]. The stochastic process counts events of state transitions and based on that allows to predict probabilities of dynamic changes. In [17], the authors further demonstrated the robustness in dynamic environments based on a MCL variant utilizing a Normal Distribution Transform (NDT). More recently, Krajnik et al. proposed Frequency Map Enhancement (*fremen*) which builds probabilistic functions of time to model environments [10]. A mixture of *amcl* and the SLAM algorithm *gmapping* [5] is used to continuously build spatio-temporal maps and enable global localization w.r.t. a prior map. Based on sets of observations made at varying times and timescales, this representation learns to predict putative changes and frequently incorporates recent information. In order to ensure robust state estimates over longer periods of time, the authors suggest to regularly re-initialize the global localization (*amcl)* at a known place (charging station).

Atanasov et al. incorporate semantic observations in localization using the Matrix Permanent [1]. Similarly to our approach, the authors match object observations to a prior map, however with the main objective of recognizing places rather than enabling robustness in changing environments by means of a-priori knowledge about potential object relocations.

B. Place Recognition

Since place recognition can be used for the global localization of a robot w.r.t. a prior map. The majority of the state-of-the-art in this field differs from our approach since they rather enable topological localization and loop closure detection than locally accurate metric localization. Nevertheless they share the same challenge: The association of places in the environment being subject to change. Milford et al. suggested to identify place correspondences across substantive changes by utilizing sequences of low-resolution image representations [13]. Churchill et al., in contrast, make use of high-level image features and incorporate varying environmental conditions as new experience [3]. Also the authors of [14], [18], [15] presented methods for establishing robust correspondences across changes in seasons, day time and weather conditions.

C. SLAM

The application of SLAM for long-term robot pose estimation has been investigated by several researchers. Labbe et al. [11] and Hartmann et al. [6] proposed solution for SLAM with appearance-based loop closure detection which were reported to be capable of multi-session mapping. Einhorn et al. utilize the normal distribution transform (NDT) to infer loop closure candidates and estimate transformations of graph nodes [4]. Either of the authors relocalize a mobile robot in a SLAM graph [11], [4], [6] while constantly incorporating new observations in the map.

D. Summary

Algorithms for map-based localization incorporate changes based on continuous observations of the environment. The systems presented in [10] and [14] are able to predict changes once the environment has been sufficiently observed. The robustness in [10] is achieved based on frequent re-initializations and repeated mapping. None of the related approaches incorporates semantic perception to enable robust localization in the presence of significant changes right after building an initial map. Existing SLAM-based systems keep running all the time while constantly overriding the map.

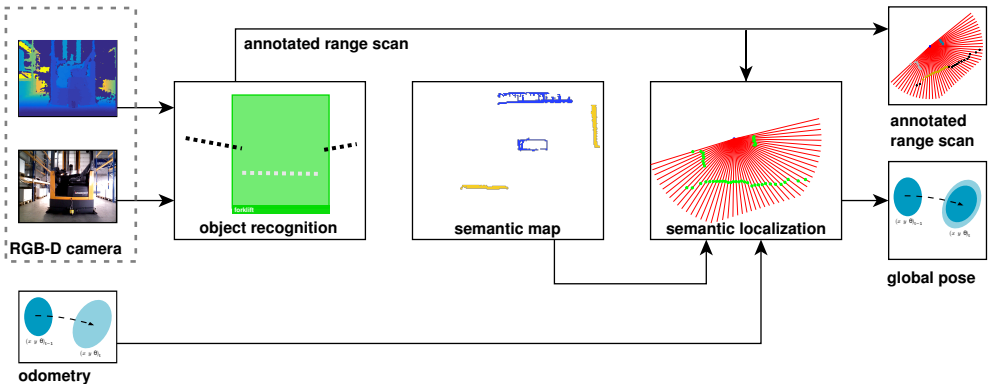


Figure 1: The graph provides an overview of our semantic perception framework. It demonstrates how the key contribution of this paper, the semantic localization, is integrated. The input RGB-D sensor data is passed to the first component in order to recognize known objects (object recognition). This module generates a 2D projection of the 3D point cloud with measuring points being at either a specified height for unknown objects or at object-specific height references, e.g. the first horizontal pillar of a rack. This range scan is supplemented by object class labels for each beam. SMCL evaluates the annotated range scan and estimates the vehicle's pose w.r.t. the prior semantic map.

Without appropriate semantic perception this is undesirable for a number of applications since the maps rapidly become unnavigable.

III. Semantic Perception

The semantic perception framework is a key element of our system. It consists of the components object recognition, semantic mapping and semantic localization. Their functionalities are described in the following. Fig. 1 illustrates the overview of the entire system highlighting also the parts of object recognition and semantic mapping. This demonstrates their positions within the overall system and its connections to other components.

A. Object Recognition

The object recognition is the fundamental requirement of our semantic navigation system. An exhaustive presentation of the object recognition system the reader is referred to our previously published work in [8]. The principle is outlined in the following:

1) Firstly, the input depth data is processed by an efficient contour-based segmentation which identifies segment boundaries based on range and height discontinuities.
2) Geometric properties of segments, specifically the object height and width, are estimated.
3) The a-priori given object class database is searched for correspondences based on the geometric features.
4) For all putative matching object classes:
   a) A region of interest (ROI) in the RGB image is defined around the object observation with coordinates being obtained by step (1).
   b) Features are extracted from the ROI which are obtained from the fc7-layer of a Convolutional Neural Network.
   c) The feature vector is evaluated using a multi-class Support Vector Machine.
   d) The class with minimum distance is selected.
   e) If the distance is below a threshold τ_{cl} then those points associated with the observed object are assigned the corresponding class labels.

For the application of warehouse environments as it presented in this paper, we aim at recognizing the following object classes: (high-level) *rack*, *gate*, *pallet*, *forklift*, *human*.

Each of the object classes is of particular interest for robotic navigation. The storage goods placed inside a rack are constantly moved which potentially decreases the localization accuracy within warehouse hallways. Gates are typically opened and closed multiple times a day. Due to their widths they significantly contribute to the pose estimate when approaching them. Bypassing humans and even more parked or moving forklifts can cause increased localization errors. The class forklift comprises all relevant vehicles that can potentially be observed. The detection of pallets is beneficial since they are frequently moved in warehouses which results in major changes in the environment, particularly at reloading areas whose appearances constantly switch from widely open to occupied. We observed that the mentioned object classes are the most common and relevant ones for this type of environment. All other objects are labeled as unknown.

B. Semantic Maps

A semantic map m is built based on a regular grid with each grid cell m_i describing an occupancy probability and an associated class label. It is generated based on odometry readings and RGB-D data using a graph-based SLAM system. During mapping, we constantly search for objects being of interest for the target environment, particularly for navigation purposes. Measurement point sets associated with recognized objects are maintained by the SLAM algorithm.

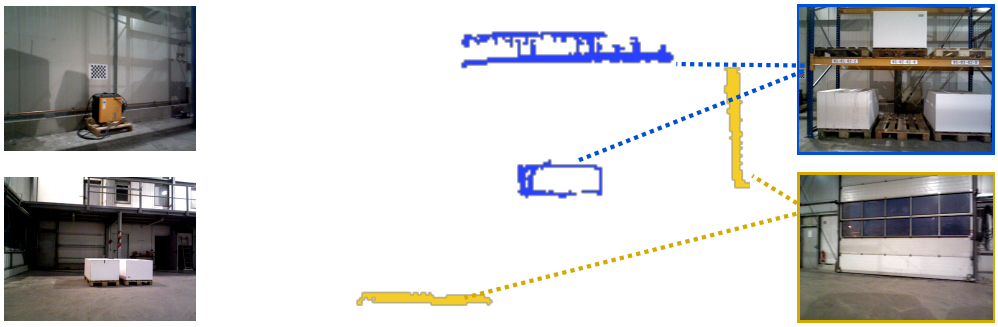


Figure 2: The semantic layer is plotted on top of an occupancy grid map. Grid cells are supplemented by object class labels. *Gates* and *racks* are identified based on our object recognition system. *Reloading areas* (red) are inferred from multiple pallet observations outside of racks. Single pallets as well as those placed at reloading areas are incorporated in the map but ignored for localization purposes. *Charging stations* (purple) tagged with a checkerboard sign are automatically mapped. They are used as initialization of SMCL in this work.

The final points in the map coordinate frame are projected into semantic map grid. Grid cells being hit by the object observations are assigned the corresponding class label. The object class is estimated from a putative uncertain set of observations. The semantic maps incorporate the classes *rack* and *gate* detected by our object recognition system. These are supplemented by *charging stations* and *reloading areas*. Charging stations are recognized based on predefined signs whereas reloading areas are inferred from the observation of multiple pallets in open space areas. Even though this is technically possible, we omit storing objects which are likely to change their positions. Here this addresses the object classes pallet, forklift and human. Fig. **2** shows an example of a semantic map. A more exhaustive description of the semantic map generation can be found in our prior work [9].

IV. Localization

Given a semantic map m as described in Section III-B, a particle filter is used in order to perform global localization. The robot's state at a discrete time step t is expressed by its pose $x_t = (x, y, \vartheta)$ containing the 2D location (x, y) and orientation ϑ. The state x_t is predicted based on odometric readings which are incorporated by the motion model. The observation model evaluates sensor measurements z_t obtained from the RGB-D sensor with respect to the prior map.

A. Motion model

The motion $u_t = (v_t, \omega_t)^T$ carried out by the robot is modelled by its translational velocity v_t and rotational velocity ω_t. The odometry is subject to noise which can occur due to a variety of sources, for instance: wheel slip, varying inflation pressures of tyres or load balance of the vehicle. In order to incorporate this uncertainty in our motion model, we add zero mean Gaussian noise σ with standard deviation σ as detailed in [19]:

$$\begin{pmatrix} \hat{v}' \\ \hat{\omega}' \end{pmatrix} = \begin{pmatrix} v + \epsilon_{\alpha_1|v|+\alpha_2|\omega|} \\ \omega + \epsilon_{\alpha_3|v|+\alpha_4|\omega|} \end{pmatrix} \tag{1}$$

The state transition from the previous state x_{t-1} to the predicted state x'_t after Δt units of time is defined by the following motion model:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + \hat{v}\Delta t \cos(\theta + \hat{\omega}\Delta t) \\ y + \hat{v}\Delta t \sin(\theta + \hat{\omega}\Delta t) \\ \theta + \hat{\omega}\Delta t \end{pmatrix} \tag{2}$$

B. Observation model

The observation model utilizes the latest range sensor measurement to evaluate each particle x_t^k. Given the map m and the observation z_t, we estimate the observation likelihood $p\left(z_t \,|\, x_t^k, m\right)$ for the k-th particle.

The key contribution of our work affects the observation model. Traditionally it consists of a range model projecting end points of sensor beams in the map coordinate frame originated at the particle's state x_t^k. The occupancy value of the hit grid cell m_i is evaluated and the distance r_{m_i} is estimated. Our approach additionally considers object class labels of semantic grid maps to evaluate correspondences of range measurement points and grid cells. We will further refer to the sensor observation of an individual beam k at time t as $z_t^k = \{r, o\}_t^k$ with r being the measured range and o the estimated object class for the k-th beam.

1) Range model: This model evaluates the particle set according to the differences of the measured range r_t^k and the range projected from the particle's pose to the map grid cell m_i being hit by r_t^k. The closest cell m_i is found using nearest neighbour search as exhaustively described in [19].

Given the ranges r_t^k and r_{m_i}, the likelihood p_r of measuring r_t^k given m and particle state x_t can be estimated as:

$$p_r\left(r_t^k \,|\, x_t, m\right) = \exp\left(-\frac{|r_t^k - r_{m_i}|}{2\sigma_r^2}\right) \tag{3}$$

with σ_r^2 describing the expected measurement uncertainty. This range model is the core of the likelihood field sensor model which is also used in AMCL providing the basis to compute a particle's weight.

2) Object model: The semantic perception is integrated inside the object part of the observation model. It incorporates class-specific observation uncertainties which can occur due to objects of similar visual appearance or geometric properties. The object correspondence matrix describes this a-priori knowledge which expresses the probability of measuring a certain object class given the object class of the cell o_{m_i}:

$$p_o\left(o_t^k \equiv o_{m_i}\right) = p\left(o_t^k \mid o_{m_i}\right) \tag{4}$$

The object correspondence matrix does not have to be symmetric, hence $O_{ij} \neq O_{ji}$ is a valid constraint. This enables to specifically account for single-sided observation uncertainties. The matrix used for our system is given by Table I.

Predicted Class						
Rack	Pallet	Gate	Forklift	Human	Unknown	True Class
0.99	ϵ_{min}	0.5	ϵ_{min}	ϵ_{min}	0.5	Rack
–	–	–	–	–	–	Pallet
0.5	ϵ_{min}	0.99	ϵ_{min}	ϵ_{min}	0.5	Gate
–	–	–	–	–	–	Forklift
–	–	–	–	–	–	Human
0.5	ϵ_{min}	0.5	ϵ_{min}	ϵ_{min}	0.99	Unknown

Table I: The object correspondence matrix describes the a-priori probabilities of measuring certain object classes. As already mentioned, the dynamic classes *pallet, forklift* and *human* are not stored in the map and hence their observation does not contribute to the pose estimate. The value ϵ_{min} is a small nonzero value to avoid numerical issues.

3) Model fusion: The presented sub-models for range and object perception are probabilistically fused within a common sensor model. The individual components of this mixture are weighted in order to account for sensor and environment characteristics. For instance, the weights of the range model can be adjusted according to the measurement accuracy of the depth sensor. The weight for the object model can be set with respect to the classification accuracy or overall uncertainty being expected.

The mathematical derivation for the mixture model p^k given the measurement z_t^k can be expressed as follows:

$$p^k(z_t^k \mid x_t, m) = z_r \cdot p_r + z_o \cdot p_o \tag{5}$$

with z_r and z_o denoting the prior weighting factors for the individual components. All measurements z_t at time t are incorporated according to:

$$p(z_t \mid x_t, m) = \prod_k p^k \tag{6}$$

4) Summary: The presented semantic sensor model provides an extension to the generic likelihood field model. More precisely, the association of putative correspondences of projected measurement end points and map grid cells is established using the latter. Instead of purely evaluating particles based on the estimated spatial displacements, the semantic model considers additional information that can be observed using RGB-D cameras. Similarly to the generic likelihood field, the pose estimate will be more accurate and robust the more correspondences are found. The key difference, however, is that the semantic sensor model significantly mitigates or even disables the contribution of false correspondences which is not the case for the generic sensor model. The latter literally matches any observed obstacle to the closest one in the map which may result in significant pose deviations. For estimating the pose w.r.t. to the prior map it is important to use stable correspondences. Points inside a subspace of the map being subject to frequent changes, which are not necessarily predictable, rather confuse the localization algorithm than contributing to robustness or accuracy.

V. Experiments

We evaluate the presented approach for semantic localization by experiments carried out in a warehouse which consists of a multitude of common objects expected for this type of environment. The data is collected using a reach truck which has been fully automated for a research project. For the purpose of semantic mapping and localization, we manually steered the vehicle inside the warehouse. The RGB-D data utilized by our system is recorded using two Asus Xtion cameras with one being aligned forwards and one sidewards with respect to the vehicle's direction of travel. We use the following parameters for our evaluation: $\alpha_1, ..., \alpha_4 = [1.0, 1.0, 1.0, 1.0]$, $\sigma_r = 0.2m$, $z_r = 0.5$, $z_o = 0.5$, $p_{occ} = 0.7$, $p_{free} = 0.3$, $\tau_{svm} = 0.4$

A. Datasets

A number of datasets were recorded given the described setup. This set consists of different experiments focusing on particular problems the state-of-the-art algorithm AMCL faces in changing environments. We will shortly describe each of the datasets in the following.

1) Static: This dataset was captured subsequently to the mapping process with no objects being moved in order to estimate the baseline localization accuracy of the presented approach.

2) Gate: The vehicle is steered towards an open sectional gate which was closed within the mapping process. Three pallets are placed below the open gate area. However, the boundaries of the pallets are $0.65m$ apart from the actual gate and hence are outside of the recorded map area. This demonstrates a common highly dynamic area inside a warehouse with the gate being frequently opened and closed.

3) Rack: The reack truck is driven through a hallway surrounded by high-level racks. The pallets of the lower sections are slightly moved about $0.1m$ towards the hallway. This experiment investigates the impact of changing content inside racks on the localization accuracy. This effect can be observed due to accumulated errors occurred during automated reloading processes or more obviously due to

pallets being stocked by humans in environments with both automated systems and human operators.

4) Reloading: Particularly larger free space areas of warehouses are subject to changes due to palleted goods being frequently moved. We recorded a trajectory which passes a typical reloading area of a warehouse in order to analyze the algorithm's robustness in the presence of significant changes.

5) Mixed: This dataset investigates the algorithm's performance over longer time covering a trajectory length of about 269 m and period of time of about 28 min. The environment is constantly reconfigured during this experiment.

B. Ground truth

The ground truth trajectory for each dataset is estimated using a SICK S-300 laser range finder. Thanks to the large field of view ($270°$), long range ($30m$) and sub-centimeter accuracy of the sensor, we are able to provide high-resolution ground truth data. Since also a laser-based localization system (such as AMCL) would be affected by the environment changes being investigated, a SLAM algorithm is used to generate individual maps and estimate the traversed paths. The underlying graph-based SLAM framework which makes use of a joint map and pose estimation is described in **[7]**. In order to align the trajectories to a common global frame, a checkerboard detection system with known absolute pose is placed on top of a charging station. The latter is also utilized as additional, highly accurate, loop closure detection. Note that this system is not utilized in any way by the localization algorithms being evaluated here. The accuracy of the ground truth is below $0.03m$.

C. Results

Figure 3 illustrates the results obtained for SMCL and AMCL respectively.

1) Static : Both, AMCL and SMCL provide robust and accurate results. The position accuracy is at about $0.21m$ in the mean.

2) Rack : The results of AMCL are adequate for minor changes inside the racks. However, once a significant amount of storage content is moved, the localization error of AMCL increases. Since AMCL does not distinguish particular objects, it estimates the vehicle's pose w.r.t. the rack's changing content which can be palleted goods or the pillar. SMCL, in contrast, correctly identifies the high-level racks and relies on the first horizontal pillar for the localization which ensures robust and accurate results within warehouse hallways. The error of SMCL remains constantly to its baseline whereas the error of AMCL increases by about $0.1m$ which is related to the amount and degree of the change inside the racks.

3) Reloading: It can be observed that AMCL fails once the vehicle enters the spacious reloading area in front of the gate. The pose estimate significantly deviates due to a notable amount of change being present in this area. The error of AMCL increases to about $2.38m$. The particle filter of AMCL diverges in this experiment and does not recover without re-initialization. SMCL correctly identifies pallets in this area and robustly tracks the vehicle's pose.

4) Gate : As expected, AMCL literally pushes the vehicle towards the occupied map area being actually covered by the gate. SMCL correctly recognizes that the perceived objects are pallets instead of parts of the gate and hence its pose estimate does not deviate. AMCL deviates by about $0.83m$ at the gate area and continues with an increased error. This confirms the necessity of storing gates in the map.

5) Mixed : AMCL frequently deviates during this experiment up to a maximum error of about $1.47m$. SMCL performs with a better mean error of about $0.25m$. In some situations during this experiment SMCL deviates up to a maximum error of about $0.76m$. We expect that this has happened during continuous loops in the reconfigured reloading area. Due to the limited perception field, SMCL was unable to observe sufficient stable points and and hence relied on odometric measurements.

D. Discussion

Either of the algorithms achieve a baseline accuracy of about 0.21m. This differs from those results that can be obtained using laser range finders (LRFs) due to the following reasons. Firstly, the measurement accuracy of an LRF is significantly higher over the entire operating range compared to RGB-D sensors, particularly in the case of Kinect-like cameras as being used in our experiments. Secondly, the operating range and field-of-view of the LRF is substantially larger, e.g. (30m; 270deg) for a SICK S-300 vs. (5m; 58deg) for a ASUS Xtion camera. This difference becomes particularly apparent in areas of larger free space where RGB-D cameras can hardly perceive any obstacle whereas LRFs can typically observe a large area over a long time. The limited sensing properties of RGB-D sensors also cause an increased uncertainty and hence limit the accuracy of the maps even when using customized SLAM methods such as **[7]**. Nevertheless, the use of these cameras enables a number of benefits, such as reduced costs and a high information density for object recognition. The latter provides a substantial requirement for the semantic perception layer of SMCL.

AMCL performs well as long as sufficient static obstacles are present and observable (e.g. large walls), otherwise it fails or becomes inaccurate. The error of AMCL in the gate and reloading area (up to $2.38m$) is not acceptable for automated systems. Drive requests might not be fulfilled since targets are failed. The consequences of association errors at a gate can be enormous since an automated vehicle can be attracted outside the warehouse into an unmapped area which might be inaccessible for itself and not expected by humans residing in this area. The minor errors of AMCL close to the racks (about 0.32m) might be acceptable if an additional system can be utilized for actual reloading processes, e.g. by measuring pallet poses as targets and apply reactive approaching strategies. However, the increased uncertainty in combination with the lack of semantic perception can cause

more significant errors or filter outages, if more dynamic objects such as other AGVs or unexpected pallets are present. Pose estimates of SMCL also deviate in the presence of significant changes, but orders of magnitude less than AMCL does. The uncertainty of SMCL increases but still covers the true pose which enables SMCL to correct its pose and minimize its uncertainty once sufficient correspondences are found.

The higher robustness of SMCL can be achieved taking into account a slight increase of computational times. The estimation of the object part within the sensor model requires about $5\,ms$ for each iteration which can be neglected. The object recognition system requires about $273\,ms$ in the mean, and $498\,ms$ in the maximum resulting in a runtime of about $4Hz$ using solely CPUs.

VI. Conclusion

Semantic maps provide a number of substantial benefits for mobile robots. This paper presents an approach utilizing the latter as prior for the map-based localization of an AGV. The most commonly used algorithm AMCL provides accurate results and enables global localization. However, the underlying sensor model hampers its use in dynamic environments being subject to numerous changes over time. The goal here is to incorporate the semantic information of the map inside the core of the localization. The association of map and sensor data is augmented by additional knowledge about known objects classes and their predicted properties in terms of dynamic changes and contribution to the global pose estimate. Our approach requires only limited additional overhead for modeling changes of the environment within the map representation since the latter are not constantly incorporated into spatio-temporal models as it is the case for the majority of related algorithms. Our paper demonstrates that these expensive models are not necessarily required if prior knowledge about commonly observed objects is available. The semantic perception layer is not exclusively designated for the localization system. A multitude of other components benefit from this as well, e.g. the human-robot-interaction, mapping of storage places or even for tailored obstacle avoidance.

The accuracy and robustness of SMCL was demonstrated to outperform the state-of-the-art in map-based localization. We expect the outcome of our paper to be directly beneficial for the emerging application of AGVs in logistic environments and motivate the utilization of semantic mapping and localization for other robotic applications. The key advantage, that is the limited number of commonly observed object classes, is feasible for many other scenarios. Detecting cars, buses and pedestrians will likely enable higher localization accuracy for on-road driving. Likewise the navigation of mobile robots in shopping malls, stations and airports can benefit from the recognition and consideration of humans, carts and suitcases.

References

[1] N. Atanasov, M. Zhu, K. Daniilidis, and G. Pappas. Localization from semantic observations via the matrix permanent. *International Journal of Robotics Research*, 2015.

[2] W. Burgard, A. B. Cremers, D. Fox, D. HÃd'hnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1):3 – 55, 1999.

[3] W. Churchill and P. Newman. Experience-based Navigation for Long-term Localisation. *The International Journal of Robotics Research (IJRR)*, 2013.

[4] E. Einhorn and H. M. Gross. Generic 2d/3d slam with ndt maps for lifelong application. In *2013 European Conference on Mobile Robots*, pages 240–247, Sept 2013.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. Proc. of the 2005 IEEE International Conference on*, Apr 2005.

[6] J. Hartmann, J. H. Klüssendorff, and E. Maehle. A unified visual graph-based approach to navigation for wheeled mobile robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1915–1922, Nov 2013.

[7] M. Himstedt, S. Keil, S. Hellbach, and H.-J. Böhme. A robust graph-based framework for building precise maps from laser range scans. In *Proceedings of the Workshop on Robust and Multimodal Inference in Factor Graphs. IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[8] M. Himstedt and E. Maehle. Camera-based obstacle classification for automated reach trucks using deep learning. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–6, June 2016.

[9] M. Himstedt and E. Maehle. Online semantic mapping of logistic environments using rgb-d cameras. *International Journal of Advanced Robotic Systems*, 14(4), 2017.

[10] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide. Long-term topological localization for service robots in dynamic environments using spectral maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[11] M. LabbÃl' and F. Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, June 2013.

[12] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard. Temporary maps for robust localization in semi-static environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[13] M. Milford and G. Wyeth. Seqslam : visual route-based navigation for sunny summer days and stormy winter nights. In *IEEE International Conferece on Robotics and Automation (ICRA 2012)*, pages 1643–1649, River Centre, Saint Paul, Minnesota, 2012. IEEE.

[14] P. Neubert, N. Sunderhauf, and P. Protzel. Superpixel-based appearance change prediction for long-term navigation across seasons. *Robotics and Autonomous Systems*, 69:15–27, August 2014.

[15] E. Pepperell, P. Corke, and M. Milford. Routed roads: Probabilistic vision-based place recognition for changing conditions, split streets and varied viewpoints. *The International Journal of Robotics Research*, 35(9):1057–1179, 2016.

[16] J. Saarinen, H. Andreasson, and A. J. Lilienthal. Independent markov chain occupancy grid maps for representation of dynamic environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012.

[17] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal. Normal distributions transform monte-carlo localization (ndt-mcl). In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[18] N. Suenderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[20] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *International Journal of Robotics Research*, 32(14), December 2013.

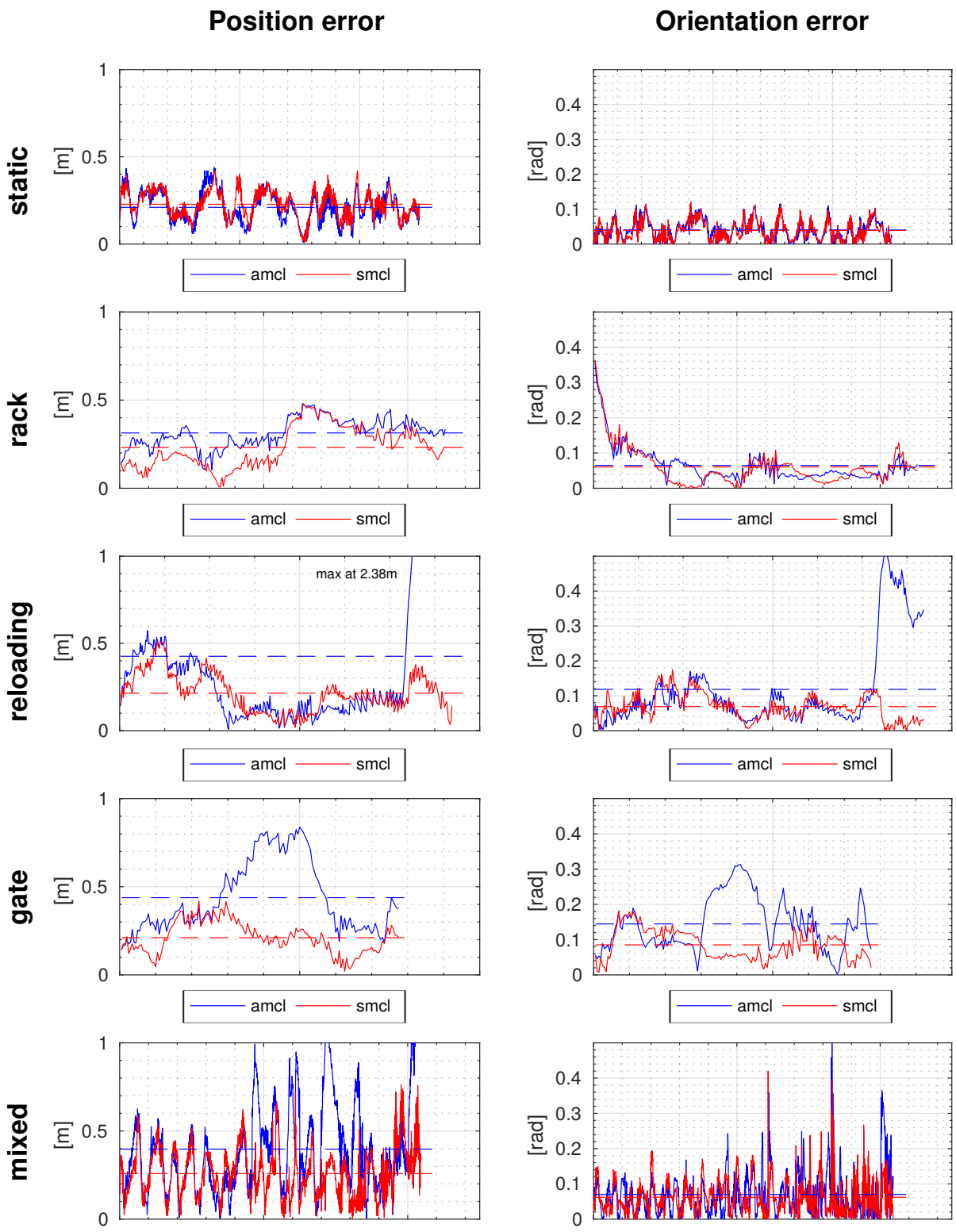


Figure 3: Experimental results obtained for the individual datasets *static*, *rack*, *reloading*, *gate* and *mixed* (one for each row). The left column shows the position error, the right column the orientation error. The dashed lines show the mean error on the individual datasets for either algorithms. The baseline accuracy of SMCL and AMCL is at about $0.21m$. Due to significant changes in the environments, such as the closing state of a gate, the relocations of pallets and parked vehicles, the error of AMCL increases enormously (see *gate*) or AMCL diverges (see *reloading*). SMCL, in constrast, is able to robustly estimate the vehicle's pose even in the presence of high dynamic changes.

An Efficient Backtracking-based Approach to Turn-constrained Path Planning for Aerial Mobile Robots

Hrishikesh Sharma[1], Tom Sebastian[1] and Balamuralidhar P.[1]

***Abstract*— Many important classes of civilian applications of Unmanned Aerial Vehicles, such as the class of remote monitoring of long linear infrastructures e.g. power grid, gas pipeline etc. entail usage of fixed-wing vehicles. Such vehicles are known to be constrained with restricted angular movement. Similarly, mobile robots such as car robots or tractor-trailer robots are also known to entail such constraint. The algorithms known so far require a lot of preprocessing for turn constraint. In this paper, we introduce a novel algorithm for turn angle-constrained path planning. The proposed algorithm uses a greedy backtracking strategy to satisfy the constraint, which minimizes the amount of backtracking involved. By further constructing an efficient depth-first brute-force algorithm for path planning and comparing against its performance, we see an improvement in convergence performance by a factor of at least 10x. Further, compared to recent LIAN suite of path-planning algorithm, our algorithm exhibits much reduced discretization offset/error with respect to shortest path length. We believe that this algorithm will form an useful stepping stone towards evolution of better path planning algorithm for specific mobile robots such as UAVs.**

I. Introduction

The usage of Unmanned Aerial Vehicles (UAVs) for remote surveillance applications especially of civil nature is gaining attention worldwide. Such mobile aerial robots are also being increasingly used in coordination with Unmanned Ground Vehicles (UGVs) as mobile robots as well [1]. Many of these applications need usage of long-endurance vehicles, especially of fixed-wing UAVs. An example bunch of such applications includes remote health monitoring of long linear infrastructures [2]. Such infrastructures, which support critical utilities, include power grid corridors, oil and gas pipeline corridors, railway corridors etc. The area of surveillance is typically vast, running into tens of kilometers. Hence planning for UAV missions for such applications has received separate attention e.g. in [3]. Such infrastructures mostly run through **complex** remote outdoor terrains, typically **non-urban**, such as forests, hilly regions, wetlands etc., for such lengths. This makes remote monitoring of such systems a very challenging task. Given the vastness, it is necessary that the mission route being planned is as short as possible, to conserve battery power.

One important constraint in controlling of fixed-wing aircrafts is maximum turn angle constraint. Such constraint is used to limit the turn rate within the UAV controller, so as to avoid various types of dangerous *aerodynamic stalls* during its flight mission [4]. However, in case of shortest path planning, limited turn rate navigation of an aircraft can lead to more number of turns in a specific mission. This in turn leads to slowing down of the vehicle, resulting in longer mission time [5]. Hence it is important to design for efficient turning along each mission's path.

In the plethora of prior works on path/route planning, A* has been the popular workhorse algorithm, in a discrete navigation space[6]. However, especially for route planning with turning constraints, A* is known to be inefficient [7]. This is because each of the neighbors, along with path can be possibly augmented, leads to turn angle which is constrained to be one of $\langle \pm 45^\circ,\ \pm 90^\circ,\ \pm 135^\circ \rangle$. In another solution [8], a graph transformation is performed first, followed by a path search on the transformed graph. But such algorithm is inefficient for simple reason: in a 2D grid for example, there are typically at maximum 1 neighbor left post transform for each node, since the turn limit is generally about 30° [9], [10], [11]. On such a sparse graph, the path which is deemed optimal is actually very suboptimal when compared to the optimal path in the continuous version of the problem, since discretization offset/error is very high. [11] solves this problem using *Bellman-Ford* algorithm, which is known to have a much higher complexity ($O(|E| \cdot |V|)$ than the ones based on Dijkstra's algorithm $O(|E|+|V|log|V|)$ such as ours. In yet another related work [12], angle-constrained motion planning is carried out along with other kinematic constraints arising out of high windy conditions. However, the due to dynamic change in the lift, the turn limit itself is a variable. Towards minimization of the *alternative* aspect of number of turns, few solutions are provided in [5], [13]. However, it is shown that such paths can be up to 22% longer than ideal solution on an average, even in 2D case. Further, none of the above works applies in a straightforward way to the case when the flying region has communication coverage holes, a problem that has been dealt separately by us [14]. A performance-wise comparison of few more of these works is provided in section VI.

One of the common problems in the workhorse path planning algorithm, A*, was the error due to discretization of the grid. Such error has recently been reduced in a modified A* algorithm, known as "**Theta***" [15], or any-angle path-planning algorithm. Many variants of this significant advancement have also come up within last decade. A significant property of this *heuristic* algorithm is that it leads to countably finite but much larger set of feasible turning angles at various points of turn along a path. We exploit this property to design a novel *heuristic* variant of

[1]All the authors are with Embedded Systems and Robotics Research Lab, Tata Consultancy Services Ltd., India. hrishikesh.sharma@tcs.com, tom.sebastian@tcs.com, balamurali.p@tcs.com

Theta*, which obeys a maximum turn angle constraint. The **novelty** of our algorithm is that we have introduced a new greedy backtracking strategy to satisfy the constraint, which locally minimizes the amount of backtracking involved. Using the *same* algorithm, we were able to design path planning in regions with coverage holes as well, another contribution. Theta* being a 2D path planning algorithm, our algorithm is first described as 2D path planning algorithm. It is important to note that such simplification is not *restrictive*: our algorithm scales to 3D space as well (c.f. section VI-C). Being a variant of Theta*, the discretization error is shown to be limited by the same limits as that of Theta*: 8% in 2D case and 13% in 3D case [16]. The additional advantage we get that of fast convergence with no additional discretization error. This is because in path planning over vast area, size of grid/graph is quite big, and hence solution time is quite important.

In the remaining paper, we first provide the model for corresponding optimization problem. We then overview the Theta* algorithm, which is followed by a presentation of our novel algorithm for angle-constrained path planning, in the next section. The results of simulation and their analysis is presented next, before we conclude the paper.

II. Problem Model

The problem is modeled as a constrained optimization problem [17]. In terms of decision variables, there is only one set involved. This set is the 3D *instantaneous* location of the UAV: $\langle x, y, z \rangle$. If the altitude is held/assumed constant, then the location is a 2-tuple, formed by latitude and longitude. In terms of constraints, there are many common constraints which a correctly planned path must conform to [18], [19]. For example, in practice, being related to "turning radius" constraint in case of motion planning, turn angle constraint only works if it is combined with a certain "minimum distance between two consecutive turns" constraint [11]. However, for lack of presentation space, in this paper, we have restricted ourselves to design for the most critical constraint, the turn angle constraint. The constraints being independent of one-another, they can be applied to a greedy solution orthogonally, to prune the search space. The (decoupled) design towards remaining constraints has been presented in [14], including "minimum distance" constraint. This paper does not deal with turn radius constraint, because of the assumption of constant speed explained later. The corresponding restricted and independent problem model that was solved for, is as follows.

minimize

$$\sum_{i=1}^{T-1} \|P_{T_{i+1}} - P_{T_i}\|$$

subject to

$$\tan^{-1}\left(\frac{y_{P_{T_i}} - y_{P_{T_{i-1}}}}{x_{P_{T_i}} - x_{P_{T_{i-1}}}}\right) -$$

$$\tan^{-1}\left(\frac{y_{P_{T_{i-1}}} - y_{P_{T_{i-2}}}}{x_{P_{T_{i-1}}} - x_{P_{T_{i-2}}}}\right) \leq \theta_{Bmax}$$

In the above model, $P_{T_i}, i \in 1, \ldots, T$ are the coordinate tuple of each *turning point* along the path in some coordinate system, e.g. world coordinate system.

A. Turn Angle Constraint

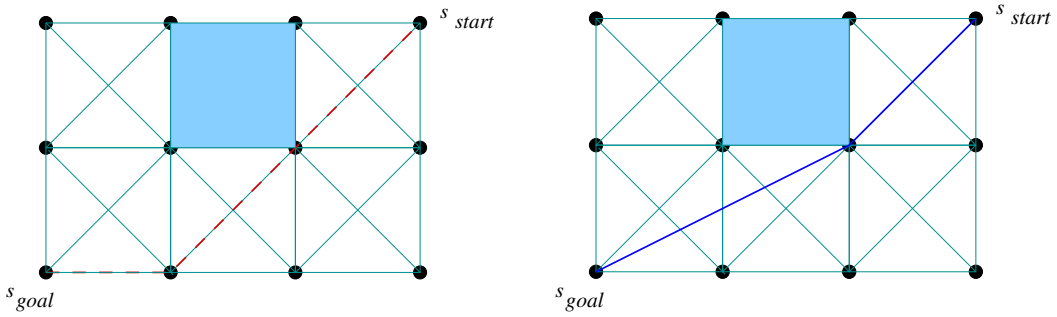
This constraint forces the generated route to make turning manoeuvre, less than or equal to a predetermined maximum turning angle. Such constraint is meant for fixed-wing vehicles, the focus of this paper. The actual kinematics behind this constraint is explained in [11].

In practice, being related to "turning radius" constraint in case of motion planning, turn angle constraint only works if it is combined with a certain "minimum distance between two consecutive turns" constraint [11]. Our algorithm straightforwardly works with this constraint, as detailed elsewhere in [14].

III. Solution Approach

As part of a larger research goal, we have designed an efficient path-planning algorithm which navigates through multiple communication coverage holes [14]. Being associated as a component in that algorithm as well, the turn-constrained path-planning algorithm presented herein is specifically designed in a way that it can handle large coverage holes. In such case, it is corroborated in [20] that the best possible speed while simultaneously optimizing for distance and communication costs is the *maximum possible speed* of the UAV. Assumption of such constant (maximum) speed decouples the motion and route planning parts of path planning. Similar philosophy is followed in the popular communication-agnostic 2-phase decoupled trajectory planning approach [6], [21], [22], where the factoring of dynamic constraints is pushed to 2nd phase, while focussing on route planning in 1st phase. Hence we have currently avoided dealing with Dubin's path design or motion primitive-based design, towards (direct) motion planning [23].

Even when it comes to route planning, finding an optimal solution is NP-hard, due to presence of obstacles [24]. Hence two philosophical lines of solution exist: combinatorial and sampling-based. Especially in motion planning, randomized sampling based approaches are preferred [21], whenever the solution space is high-dimensional. In the premises of our problem, however, it can be seen that the solution space is practically low-dimensional. This is because turn angle constraint is a holonomic constraint. Further, we impose five constraints in the overall problem [14]. Hence size of search space becomes quite tractable, along with its low dimensionality. Hence it is more prudent to follow the combinatorial, greedy approach where optimality is not asymptotic, unlike randomized sampling-based algorithms e.g. RRT*, PRM*, FMT* etc., as described below.



(a) Discrete Shortest Path using Square Grid

(b) Actual (Continuous) Shortest Path

Fig. 1: Offset between Discrete and Continuous Shortest Paths, courtesy [15]

IV. Background: Theta* Algorithm

In geographical applications in which the navigation environment is continuous, a shortest path found using popular A* on any discretized graph entails a *discretization offset/error* (c.f. Fig. 1). A recent and quite popular technique which smoothens the path as it is being greedily evolved, thus reducing this gap extensively, is **Theta*** [15]. Theta* is a variant of A*, with the key difference being that Theta* allows the parent of a vertex to be any vertex, unlike A* where the parent must be a visible neighbor. To have such extended, transitive parental relation, Theta* updates the **g**-value and parent of an *unexpanded* visible neighbor **s'** of vertex **s** by considering an alternative extension. To allow for any-angle paths, in each iteration, Theta* additionally considers the *grandfatherly* extension from parent(**s**) to **s'** in a straight line [cost = **c**(parent(**s**),**s'**)], resulting in a new length of **g**(parent(**s**)) + **c**(parent(**s**),**s'**) if **s'** has line-of-sight (**LOS**) to parent(**s**). The idea behind considering this alternative is that in case of Euclidean shortest path problems, this alternative is never longer than previous default path, due to the *triangle inequality*, if **s'** has LOS to parent(**s**). If line of sight is not present, then the extension in current iteration happens in the normal A* way, along the grid.

V. Our Algorithm

To reduce computational complexity [25], we solve the discretized version of the optimization problem by imposing a grid. The grid can be regular, or can be irregular as well. In a discrete 3D Euclidean space with obstacles, it is known that the underlying set of grid points is not convex [15]. In fact, it is proven that for spaces of dimension 3 and above, the problem is NP-hard [24]. Hence algorithms, typically inspired by shortest path algorithms in a convex domain, e.g. Dijkstra, have been researched for design for approximately optimal solutions/paths. Theta* is one such algorithm that we use this algorithm as base, and perform certain modifications to satisfy the turning angle constraint.

The algorithm is summarized in Alg. 1. The expressions surrounded by a text box are the *additional changes* over the pseudo-code of Theta*[15]. It is easy to observe that our improvements do not make the Theta* algorithm, which is a lightweight greedy algorithm, overly complex. The stages of this algorithm are explained in next few sections.

A. Angle Consistency Check at Augmentation Steps

In a greedy approach, turn is necessarily detected at each path augmentation step, to ensure that the angle constraint is continuously satisfied. To understand how turn is detected at each step, we follow nomenclature shown in Fig. 2. We also use the following theorem as we design our algorithm.

***Theorem** 1:* In greedy Theta* shortest path algorithm, the direction formed by ⟨grandparent, parent⟩ and the direction formed by ⟨parent, current⟩ are not same. That is to say, the nodes grandparent, parent, current are not collinear.

Proof: We first observe that since the evolving path has already found a path segment between parent node and current node along the line of sight between them, there is no obstacle between them that crosses the line of sight. Similarly, there is no obstacle between grandparent node and parent node that crosses their line of sight. So if these three nodes were collinear, then Theta* algorithm would have found a direct line of sight between grandparent node and current node itself, devoid of any obstacle between them. In which case, parent of current node would have been the grandparent node. Since that is not the case, we have a proof by contradiction that the nodes grandparent, parent, current are indeed not collinear, leading to a turn at parent node. ■

B. Angle Consistency with Grandparent

As is known, Theta* checks for a LOS condition between the current node's child/neighbor node, and its parent node. If line of sight exists, then using triangle inequality, the path augmented till the child node makes a "turn" at the location of the parent node. The two directions/lines which are involved and subtend a certain turn angle are a) line between current node's grandparent and the current node's parent, and b) line between current node's parent and the current node's child/neighbor being considered. Hence the first check has to make sure that before current node's parent and child are connected via a direct path, the above angle is within the limit prescribed by the constraint. If the turn

Algorithm 1 Modified, Angle-Constrained Theta* Algorithm

```
UpdateVertex(s,s')
if LineOfSight(parent(s), s') then
    ▷ /* Path 2 */
    if g(parent(s)) + c(parent(s), s') < g(s') then
        if [Angle(line(grandparent(s),parent(s)), line(parent(s),s) < angle_limit] then
            g(s') := g(parent(s)) + c(parent(s), s');
            parent(s') := parent(s);
            if s' ∈ open then
                open.Remove(s');
            end if
            open.Insert(s', g(s') + h(s'));
        else
            [goto : without_LOS]
        end if
    end if
else
[without_LOS]: ▷ /* Path 1 */
    if g(s) + c(s, s') < g(s') then
        if [Angle(line(parent(s),s), line(s,s')) < angle_limit] then
            g(s') := g(s) + c(s, s');
            parent(s') := s;
            if s in open then
                open.Remove(s');
            end if
            open.Insert(s', g(s') + h(s'));
        else
            [Find another unexplored neighbor node of s]
            if [Such node s" exists] then
                [UpdateVertex( s, s" );]
            else
                [Block Node s';]
                [Backtrack partial path to the last parent node of s, 1 unit away on grid]
            end if
        end if
    end if
end if
```

angle is 0°, then the constraint is trivially true.

valid turn at parent node possible =

$$\begin{cases} true & \text{if } \angle(\text{line(gp(s),p(s))}, \text{line(p(s),s)}) < \text{angle_limit} \\ false & \text{otherwise} \end{cases}$$

C. Angle Consistency with Parent

If the angle check fails, even if there is a LOS between parent and child of the current node, then one must fall back to the A* way for augmentation. In such case, one must check whether the angle subtended between the line from current node's parent to the current node, and the line from current node to the child being considered, is within the turn

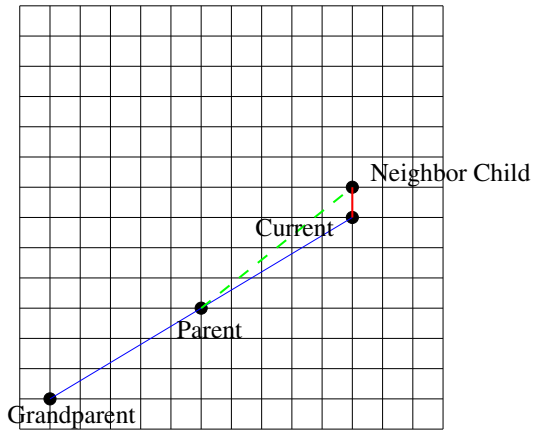


Fig. 2: Node Nomenclature for Shortest Path Algorithms

limit. Note that the line between current node's parent and current node, due to nature of Theta* algorithm, need not be aligned at all with any grid dimension or grid diagonals. Hence the possibility of success at this step is also on the higher side, than the possibility considering A*. Once again, if the turn angle is 0°, then the constraint is trivially true.

valid turn at current node possible =

$$\begin{cases} true & \text{if } \angle(\text{line(p(s),s)}, \text{line(s,neigh(s))}) < \text{angle_limit} \\ false & \text{otherwise} \end{cases}$$

D. Moving on to next Neighbor

If in both of the above cases, the neighbor node being considered is not found suitable to augment the path along, then we must look for alternatives. In that case, the child node being checked should be dropped from consideration, and another neighbor/child node of the current node is tested for possible and compatible turn constrained path augmentation (c.f. Fig 3).

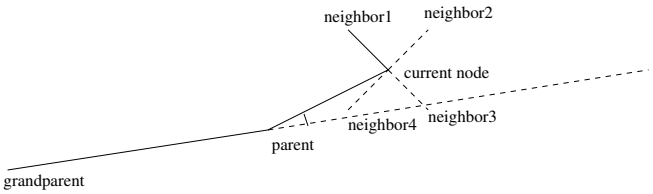


Fig. 3: Consideration for Turn Constraint in Theta*

E. Backtracking on Exhaustion of all Neighbors

There are typical scenarios in square grid where plain angle checking involving grandparent, parent and all other neighbors of current node fails, especially close to obstacles. In such case, only possible option is to backtrack along the partial path augmented so far, and try to approach the obstacle using some other neighbor of a prior node on the backtracked path. This leads to a possibility whereby

the angle of approach towards a obstacle known/discovered becomes lesser/acute.

For circumnavigation around an obstacle, [26] suggests a heuristics of putting non-uniform weights to the cell, so that any path approaching an obstacle gets *repulsed* to take a detour around the obstacle. [10] suggests another way, which boils down to having an adaptive value of cell size. Our way is different and novel since it involves *locally* minimal backtracking in order to circumnavigate via alternative route. The direction of turn is immaterial to us, and hence we do not follow any approach similar to [27]. Usage of backtracking in the path augmentation is visible in algorithm proposed in [28], but customized for a different algorithmic structure: rapidly-exploring random trees, rather than the greedoid (Theta*).

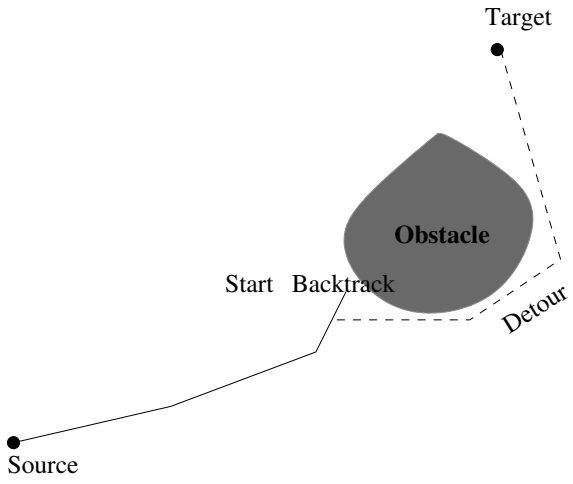


Fig. 4: Depiction of Unit-step Backtracking and detouring

In a greedy algorithm, like augmentation, philosophically, one must also diminish the path in a *locally* minimal way. Hence, to backtrack, it is not necessary that we backtrack all the way to the parent node of the current node. In Theta*, it is possible that the parent node is actually a node very far away from the current node, not necessarily at a distance of unit cell. It is imperative that the farther we backtrack and try approach via a detour, the higher the approximation error becomes. Hence via a modification to Theta*, we also keep track of which previous node participated in the triangle inequality for the current node. We iteratively go back to that specific neighbor of the current node, and try take a shorter detour when needed. Note that this variation itself is a heuristic, just like Theta*. A depiction of such detouring is shown in Fig. 4.

VI. Simulations and Results

For our algorithm, we have been able to **prove** certain theoretical properties, which include *correctness* and *completeness*, *optimality* and *complexity* of the algorithm. It uses the same framework as Theta*, and hence the proofs are similar to that for Theta*. We omit the proofs, derived in [14], here for lack of space.

***Lemma** 2:* At any point during the execution of the our modification to Theta*, following the parents from any vertex in the open or closed lists to the start vertex retrieves a path that satisfies all the problem constraints enlisted in section II, from the start vertex to this vertex in reverse.

***Theorem** 3:* When our algorithm terminates, the path extraction retrieves a path satisfying all the problem constraints, from the start vertex to the goal vertex if such a path exists.

***Lemma** 4:* If there exists a (piecewise-continuous) path between any two vertices satisfying all the constraints, then there also exists a grid path between the same two vertices, which was followed/traversed by our modified Theta* algorithm, across iterations, to arrive at the compliant path.

***Theorem** 5:* Our algorithm terminates. If a path satisfying the constraints is known to exist, the algorithm reports one such path upon termination. Else, on termination, the algorithm reports that no such path exists.

Following [16], for any square grid-based path planning algorithm, the worst-case discretization offset/error, against a continuous Euclidean shortest path, is bound by 8% in 2D case, and 13% in 3D case. Hence, while designing for an efficient algorithm, we have chosen to improve upon Theta* algorithm, whose *average-case* performance is known to be much below the above *worst-case* bounds. We experimentally show the efficiency of our algorithm over contemporary ones, for the additional approximation error arising out of circumnavigation of obstacles, later in this section. With such assurance on practical efficiency of path length, we have more importantly focussed on another performance: the runtime, or **convergence performance**. This is important since path planning algorithms over **vast** area, and hence over a mega-sized grid, one must be able to do fast planning, especially in the cases of disaster management where an immediate surveillance mission is most preferred. Even when the search space is low-dimensional as in our case, the sheer volume of space entails fast planning efforts.

To the best of our knowledge, no research work exists that combines greedy backtracking with greedy path augmentation (Theta*), for angle-constrained path planning. Hence a direct comparison with runtime performance of existing algorithms is at most crude. For a meaningful comparison, we chose to compare against a small improvement over turn-constrained planning algorithm of [7]. To incorporate backtracking and shorten the node-disjoint property of alternative paths enumerated in [7] that makes runtime very high, we designed another *naive* (**brute-force**) algorithm as follows. It is imperative that on discounting the improvements over earlier works, designed into the brute-force algorithm, the performance of our algorithm against prior works will be even higher. Before we present that comparison, a short semi-analytical comparison with repulsion-based angle-constrained path planning [26] is discussed below.

A. Cost Comparison of Repulsion versus Backtracking

As mentioned before, [26] suggests a heuristics of putting non-uniform weights to the cell, so that any path approaching an obstacle starts gets *repulsed* to take a detour around the

obstacle. [10] implements this suggestion by providing a formal way to specify this *repulsion "field"*. We have designed a different heuristics, which uses locally minimal backtracking to go along with Theta* greedy augmentation. For the former algorithm, one major disadvantage is that the set of non-uniform weights around each obstacle is dependent on the topology of the region that the obstacle represents. For example, for a concave obstacle, due to turn angle constraint, one will need to model repulsion from a farther distance than a convex obstacle of same size. Similarly, the size of obstacle is another parameter that will impact the design of obstacle-specific repulsion "field" around it. Our algorithm requires no such heavy preprocessing. Further, our algorithm does efficient, locally minimal backtracking by design. In contrast, [26] only proposes putting a repulsion "field" around each obstacle, and does not guarantee that it is of minimal size. For a quick numerical comparison, we downloaded and used the same dataset as used in [10], the *OpenStreetMaps* (OSM) dataset. We chose to only compare the *relative path length* and *success rate* as parameters, against the LIAN-5 algorithm therein (the fastest among the LIAN family). The relative path length is measured as a fraction against the direct line-of-sight distance between the source and the goal node. We chose 30 rectangular fragments in which length is few orders more than the width (size 1950m x 315m). We discretized the grid with a square cell of size 3m x 3m, as the unit cell. Like [10], we also marked cells corresponding to the areas occupied by buildings as un-traversable. We randomly chose source and goal nodes in each fragment such that the LOS distance is at least 1500m. The angle limit, θ_{Bmax}, was set to 20°. As can be seen from table I, the discretization error of 6% is within the limit of 8%, established earlier. At the very least, this comparison is able to point towards **more closeness** of our approach, to the **optimal** solution, than a very recent approach. Also, we are able to find a path in almost all cases, since our algorithm can backtrack, and hence is able to span most of the solution space.

TABLE I: Cost Comparison of LIAN-5 and our Algorithm

	θ_{Bmax} = 20°	
	success rate	**relative path length**
LIAN-5	87%	1.14
Our Algorithm	98%	1.06

B. Comparing Performance with Brute-force Path Planning

The above comparison was against a recent algorithm, which does not incorporate any backtracking strategy. A more meaningful comparison is as follows. At one extreme, we have our algorithm that prunes the feasible set of solutions in a local, greedy way, and converges fast to an approximate solution. At the other extreme, we can brute-force enumerate various solutions within the *feasible set*, and then check how many iterations it takes during the enumeration, **on an average** across multiple test cases, before we locate an efficient solution. Obviously, a lot depends on how the enumeration is performed. If we perform the enumeration in some efficient way, then comparing the performance of our algorithm to such efficient brute-force solution-finding algorithm leads to establishment of **worst relative performance** of our algorithm. Such relative performance, in turn, gives a pessimistic lower bound to our algorithm, and actual performance is expected to be better than that figure of merit. The design details of brute-force algorithm are presented in [14].

To carry out the performance comparison, we developed a simulation environment. Details of this environment are presented in [14]. The details of simulation-based comparison are as follows.

1) Algorithm for Brute Force Path Planning: This algorithm is a small improvement over planning algorithm of [7]. We model our brute-force path planning algorithm on the lines of intuitive limited backtracking described earlier. We do not perform the most naive way of brute force path planning, that is, evolving each path independently, end-to-end, and then testing each of them for being compliant to the conditions of our problem. We avoid doing that so that we can have a more realistic comparison (average case performance gain as against worst case performance gain). We again use Theta* to evolve brute force path during each iteration of brute force algorithm.

We first use basic Theta* to find a path from source node to goal node, that does not cross any obstacle. Next, we trace the solution path from source node to goal node, and at each intermediate node, check the turning angle constraint. If at a certain intermediate node, the constraint is violated, we start the search for the next optimal solution in its vicinity. That is, the sequence of nodes in current path from source till some node in vicinity of the specific intermediate node are retained in the same sequence, certain nodes immediately following the intermediate node in the current path blocked out, and alternative, remaining neighbors to the current node are re-explored to find another optimal path in certain scenarios.

Since we use Theta* as the kernel of brute force algorithm, we also need to take care of other ways via which the path evolved in current iteration relapses and coincides with the path evolved and discarded in the previous iteration. To understand that, one must have a look at Fig. 5.

As the brute force path planning for the current iteration evolves from e.g. node i_1 in the Theta* way, it is possible that ends up coinciding with the previous iterations' path. If that happens, that implies that in both paths, all segments, including the segment $i_1 - i_2$ are common and overlaid. Since the neighbor of i_1 was blocked before the current iteration started, this perfect overlay is only possible when line-of-sight based merging happens. That is, the path of current iteration progresses via $i_1 - j_2$ and $j_2 - i_2$ segments, which in turn get triangulated and shorted in lack of any obstacle along segment $i_1 - i_2$. Lack of obstacle is implied since otherwise, that segment could not have been part of the path evolved in previous iteration. To avoid such triangulation to happen, one must not only block the immediate neighbor of i_1 in previous iteration, but also the immediate segment: $i_1 - i_2$ as well. In

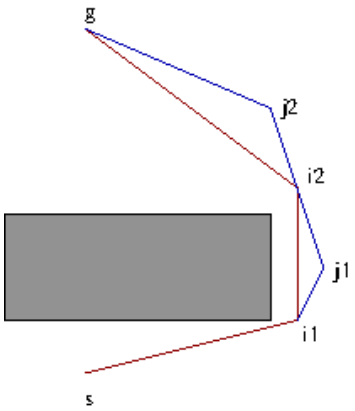


Fig. 5: Prohibited Brute Force Path Evolution: Case 1

such case, during any future iterations beyond the previous iterations, such segments cannot be exactly overlaid on any new path. With respect to Fig. 5, this means that the red-colored segment should be blocked out as well, at the end of previous iteration. The path is then forced to evolve and complete as shown in blue color in the same illustration. It is possible for such blocked red segment to be a proper subset of a longer blue segment however, because in that case, the enumeration of paths of both iterations, as successive turning points, will not exactly coincide.

2) Comparison Results: The comparison of above two algorithms was done by simulating both algorithms for a fixed, same suite of test scenarios, and observing the set of their execution times for the suite. The simulations were carried out on the same Lenovo P300 workstation. Being single-threaded, the simulations use only one (Intel core i5) processor, and hence the (performance) denominator of used architecture is same/common. The test scenarios were same as that used in section VI-A, using OpenStreetMaps dataset. We assume that the per-iteration time taken by both algorithms is similar, since the distance between goal and source node is same for both, and the backtracking strategy is *similar*. In such a case, we observed via simulations that at a coarse level of measurement, our algorithm takes just 1 iteration, while the brute force testing algorithm was found to take almost 9-10 iterations even for simple test cases. For testcases part of a different suite covered in section [14], and having at least 2 coverage holes without any obstacles, the brute-force algorithm was found to take nearly 50+ iterations. Hence the *factor of improvement* in convergence performance was seen to be in the **order of tens**.

Generalizing the above observations, it has been proven that whenever the grid size increases, the no. of iterations will increase significantly [14]. The intuition behind the proof is as follows. A naive brute-force algorithm, in average case, enumerates the solution space much more than an *efficient* randomized sampling-based approach [21], and hence its expected relative convergence is slower. The latter approach works by generating *random* paths between the source and the goal nodes, which are then tested for compliance of constraint/s and discarded/retained based on output. Further, there is no correlation between the paths generated in two successive iterations of such algorithm. Hence the computational complexity of finding one random compliant path is O($\mathbf{n^2}$), where the grid size is $n \times n$. The complexity of brute-force search algorithm is hence *at least* O($\mathbf{n^2}$). On the other hand, our backtracking-based algorithm is a best-first search algorithm, whose computational complexity is known to be O(**b+p**), where **b** is the *branching factor* and **p** is the *depth* of corresponding tree. In our case, the branching factor is a constant (4 for regular grids), and assuming wavefront propagation, depth is **n**. Assuming that for each obstacle, the degree of backtracking is local and hence limited, the overall complexity of our algorithm will be close to O(**n+4**).

If the size of grid is increased from $n \times n$ to $m \cdot n \times m \cdot n$ (latter case of a real deployment grid), then it can be seen by comparing the above two complexity figures that as **m** increases, the ratio of these two complexity figures increases. Given that the grid sizes in real deployments will be much bigger than our simulation grid sizes, the real value of our algorithm lies in real deployments.

C. Possible Scaling to 3D

As mentioned earlier, the steps involved in dealing with turn constraint also scale well for 3D case. However, most of the steps are found to be computationally heavy, e.g. finding the minimum distance to the "surface" (instead of perimeter) of a 3D hole. The complexity of this computation, and many others, increases by a factor of **n**. A specific extension to reduce the extra complexity is known as **Lazy Theta*** algorithm [16]. This algorithm can again be used as the base algorithm for our algorithm.

D. Possible Dynamic Replanning

Our algorithm can be recast to support dynamic replanning, similar to the well-known 2-phase motion planning algorithm for fixed wing aircrafts [22], to a certain extent. It can so happen that some unknown obstacle is located while the aircraft is in flight, using e.g. SONAR technology. If the current position of the aircraft is such that the two tangents to the obstacle surface do not subtend an angle more than θ_{Bmax}, then a new path is always feasible without requiring the fixed-wing UAV in flight to move back, an infeasible manoeuvre. In such case, the current position of the aircraft is deemed as the new source node, while the target node remains the same. The list of obstacles is regenerated, by including the new obstacle/s. As such, being greedy algorithm, our algorithm is computationally light. Assuming that the on-board computer has enough resources to re-execute our algorithm, a new, refined path depicting the remaining route of the UAV can be computed.

VII. Conclusion

In this paper, we have provided a novel algorithm for turn-constrained shortest path planning for mobile robots, that works even in presence of coverage holes. Such an algorithm is especially of high utility and importance when

it comes to remote surveillance of long linear infrastructures running through non-urban terrain, a popular application. The algorithm is based on improving the greedy augmentation of partial path in the Theta* way, by additional function of backtracking to a locally minimal distance, one grid unit at a time, so as to circumnavigate any arbitrary obstacle. Extensive simulation based validation and performance benchmarking of the algorithm was also carried out. The worst-case path cost deviation from best possible (continuous) path is bounded to be less than 13% [16], while the average-case performance is much superior. More importantly, for real deployments, our algorithm improves the convergence performance by order of tens. Work has already happened to design and to demonstrate that this algorithm works together with another algorithm for shortest path planning in presence of coverage holes. In reality, fixed-wing UAVs can only move along smooth paths, having a related turn-radius constraint. Hence we intend to adapt our algorithm with a fine-grained smoothening phase, towards motion planning, within the framework of 2-phase decoupled trajectory planning approaches [21]. Such adaptation shall also help in dealing with secondary factors such as dynamic disturbances such as sudden gust of wind, and system noise, which have currently been abstracted out. We are also working towards integrating a *Digital Surface Model* as a real planning environment, within the simulation environment, for real-life demonstrations. Overall, we believe that this algorithm will form an important stepping stone towards evolution of more robust algorithms for communication-aware path planning, which in turn are needed to enable many a new and important requirements and civilian applications of UAV as a remote surveillance platform.

REFERENCES

[1] J. Chen, X. Zhang, B. Xin, and H. Fang, "Coordination Between Unmanned Aerial and Ground Vehicles: A Taxonomy and Optimization Perspective," IEEE Transactions on Cybernetics, vol. 46, no. 4, pp. 959–972, 2016.

[2] C. Ezequiel and M. C. et al., "UAV Aerial Imaging Applications for Post-disaster Assessment, Environmental Management and Infrastructure Development," in International Conference on Unmanned Aircraft Systems, 2014.

[3] E. Grotli and T. Johansen, "Path- and Data Transmission Planning for Cooperating UAVs in Delay Tolerant Network," in IEEE GLOBECOM WI-UAV Workshop, Dec 2012, pp. 1568–1573.

[4] G. H. Elkaim, F. A. P. Lie, and D. Gebre-Egziabher, "Principles of Guidance, Navigation, and Control of UAVs," in Handbook of Unmanned Aerial Vehicles. Springer, 2015, pp. 347–380.

[5] H. Xu, L. Shu, and M. Huang, "Planning Paths with Fewer Turns on Grid Maps," in AAAI International Symposium on Combinatorial Search, 2013, pp. 193–201.

[6] C. Goerzen, Z. Kong, and B. Mettler, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," in IEEE International Symposium on UAVs, 2009, pp. 65–100.

[7] V. Roman, "Turn Constrained Path Planning Problems," Master's thesis, University of Nevada at Las Vegas, 2006.

[8] A. Boroujerdi and J. Uhlmann, "An Efficient Algorithm for Computing Least-cost Paths with Turn Constraints," Elsevier Information Processing Letters, vol. 67, no. 6, pp. 317–321, 1998.

[9] L. Techy, C. Woolsey, and K. Morgansen, "Planar Path Planning for Flight Vehicles in Wind with Turn Rate and Acceleration Bounds," in IEEE International Conference on Robotics and Automation, 2010, pp. 3240–3245.

[10] K. Yakovlev, E. Baskin, and I. Hramoin, "Grid-Based Angle-Constrained Path Planning," in KI 2015: Advances in Artificial Intelligence. Springer Verlag LNCS, 2015, vol. 9324, pp. 208–221.

[11] J. Krozel, C. Lee, and J. S. B. Mitchell, "Turn-Constrained Route Planning for Avoiding Hazardous Weather," ATCAI Air Traffic Control Quarterly, vol. 14, no. 2, pp. 159–182, 2006.

[12] J. A. Guerrero and Y. Bestaoui, "UAV path planning for structure inspection in windy environments," Journal of Intelligent & Robotic Systems, vol. 69, no. 1-4, pp. 297–311, 2013.

[13] Y. Li, H. Chen, M. J. Er, and X. Wang, "Coverage Path Planning for UAVs based on Enhanced Exact Cellular Decomposition Method," Elsevier Journal on Mechatronics, vol. 21, no. 5, pp. 876–885, 2011.

[14] H. Sharma and T. Sebastian, "On Communication-aware Offline Path Planning for Long-range Surveillance Missions undertaken by UAVs," arXiv Preprint https://arxiv.org/submit/1887041, 2017.

[15] K. Daniel, A. Nash, and S. Koenig, "Theta*: Any-Angle Path Planning on Grids," Journal of Artificial Intelligence Research, 2010.

[16] A. Nash, S. Koenig, and C. Tovey, "Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D," in AAAI International Conference on Artificial Intelligence, 2010, pp. 147–154.

[17] J. Nocedal and S. J. Wright, Numerical Optimization. Springer Science+Business Media Ltd., 2006.

[18] Z. Qi, Z. Shao, Y. S. Ping, L. M. Hiot, and Y. K. Leong, "An Improved Heuristic Algorithm for UAV Path Planning in 3D Environment," in International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 2, 2010, pp. 258–261.

[19] R. Szczerba, P. Galkowski, I. Glicktein, and N. Ternullo, "Robust Algorithm for Real-time Route Planning," IEEE Transactions on Aerospace and Electronic Systems, vol. 36, no. 3, pp. 869–878, 2000.

[20] A. Ghaffarkhah and Y. Mostofi, "Dynamic networked coverage of time-varying environments in the presence of fading communication channels," ACM Transactions on Sensor Networks, vol. 10, no. 3, 2014.

[21] D. Lee and D. H. Shim, "Path Planner based on Bidirectional Spline-RRT* for Fixed-wing UAVs," in International Conference on Unmanned Aircraft Systems, 2016, pp. 77–86.

[22] M. Hwangbo, J. Kuffner, and T. Kanade, "Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs," in IEEE International Conference on Robotics and Automation, 2007, pp. 1035–1041.

[23] P. Maini and P. Sujit, "Path Planning for a UAV with Kinematic Constraints in the Presence of Polygonal Obstacles," in International Conference on Unmanned Aircraft Systems, 2016, pp. 62–67.

[24] J. Canny and J. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems," in Annual Symposium on Foundations of Computer Science. IEEE, 1987, pp. 49–60.

[25] F. Li and R. Klette, Euclidean Shortest Paths. Springer Verlag, 2011.

[26] H. Kim, D. Kim, J.-U. Shin, H. Kim, and H. Myung, "Angular Rate-constrained Path Planning Algorithm for Unmanned Surface Vehicles," Elsevier Journal of Ocean Engineering, vol. 84, pp. 37–44, 2014.

[27] O. Arslan, P. Tsiotras, and X. Huo, "Solving Shortest Path Problems with Curvature Constraints using Beamlets," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 3533–3538.

[28] J. Redding, J. N. Amin, J. D. Boskovic, Y. Kang, K. Hedrick, J. Howlett, and S. Poll, "A Real-time Obstacle Detection and Reactive Path Planning System for Autonomous Small-scale Helicopters," in AIAA Navigation, Guidance and Control Conference, 2007.

Effective Target Aware Visual Navigation for UAVs

Ciro Potena, Daniele Nardi and Alberto Pretto.

***Abstract*— In this paper we propose an effective vision-based navigation method that allows a multirotor vehicle to simultaneously reach a desired goal pose in the environment while constantly facing a target object or landmark. Standard techniques such as Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) in some cases (e.g., while the multirotor is performing fast maneuvers) do not allow to constantly maintain the line of sight with a target of interest. Instead, we compute the optimal trajectory by solving a non-linear optimization problem that minimizes the target re-projection error while meeting the UAV's dynamic constraints. The desired trajectory is then tracked by means of a real-time Non-linear Model Predictive Controller (NMPC): this implicitly allows the multirotor to satisfy both the required constraints. We successfully evaluate the proposed approach in many real and simulated experiments, making an exhaustive comparison with a standard approach.**

I. Introduction

Vision-based control, or *visual servoing* (VS), of UAVs (Unmanned Aerial Vehicles) is an active research topic with many applications, including search & rescue, fire monitoring, traffic monitoring and patrolling. More specifically, in these tasks the multirotor is steered to its desired state by using visual feedbacks obtained from one or more cameras. This topic has gained even more interest in the last years, making it possible to deal with complex vision-based tasks, such as landing on moving platforms [1], flight through gaps [2], object grasping [3] and target tracking [4].

What makes VS a challenging problem for multirotors is the under-actuated dynamics of such vehicles, especially when performing agile and fast maneuvers (i.e., with high velocity and angular accelerations). During such kind of maneuvers, standard visual-based controllers focus solely on reaching the goal state *without* constantly taking into account their camera configuration with respect to the perceived environment. In other words, during the UAV flight these systems may lose for some time the line of sight with a target of interest, even if such target represents the final goal of the flight. This behavior can prevent the applicability of such controllers in activities when the re-localization of the target of interest is not a trivial task, due to the self-motion of the target (e.g., when tracking a specific person that moves in the crowd), or due to sensor aliasing (e.g., when moving toward a specific object with not unique appearance features).

This work has been supported by the European Commission under the grant number H2020-ICT-644227-FLOURISH. Potena,Nardi and Pretto are with the Department of Computer, Control, and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Italy. Email: {potena, nardi, pretto}@diag.uniroma1.it.

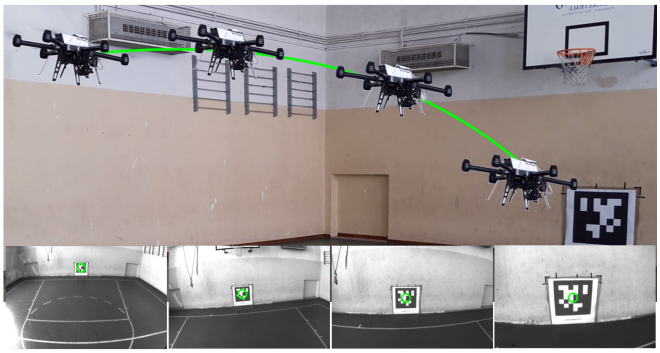


Fig. 1: An example of target aware visual navigation: the UAV is following an optimal trajectory towards the target while constantly framing the target with the camera.

In this paper, we propose an effective and robust VS controller that allows an UAV to perform fast maneuvers without losing the line of sight with the target of interest during the entire duration of the flight.
VS techniques can be split into two parallel branches: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). In PBVS, the 3D goal pose is directly obtained from a complete 3D reconstruction of the surrounding environment or from the 6D position of one or more landmarks placed in it. In contrast, IBVS formulates the problem in terms of image features locations: the goal pose is defined by means of desired features locations in the final image while the control law aims to minimize the features re-projection error during the flight. Even if IBVS does not require any full 3D estimation, it still needs the depth of the target. It has been shown that both strategies have their own weakness. In IBVS it is particularly challenging to model the relation between the vehicle dynamics and the feature projection error, especially for under-actuated systems. Furthermore, an inaccurate estimation of the object depth leads to instabilities. In PBVS, since the control law is directly designed in the state-space domain, there is a close dependence on the accuracy of the 3D environment reconstruction or on the target pose estimation. In practice this estimation may be very noisy, leading PBVS to be very sensitive to initial conditions, camera calibration parameters and image noise corruption.

Differently from the previous work, we propose a procedure that decouples the planning and the control problems. The planning task is addressed by employing a hybrid approach. Firstly, as in PBVS, we get the goal pose as the position and the relative orientation of the vehicle in the environment that allows to have the desired view of the target object.

Then, similarly to IBVS, we model the trajectory as a non-linear constrained optimization problem with a cost function that penalizes the target's location error, in order to constantly keep the target in the camera field of view.

Once a global optimal trajectory[1] has been found, we employ an NMPC framework as controller and local planner. Making use of an efficient open-source solver, our control framework is capable to solve an NMPC problem in few milliseconds, allowing us to use at each time step just the initial tuple of control inputs[2] while simultaneously re-solving the whole non-linear control problem.

We compare our method against a common PBVS approach in both simulated and real environments, getting in all experiments cutting edge results. Additionally, we make a preliminary assessment with respect to a state-of-the-art Optimal Visual Servoing (OVS) technique, suggesting that our approach can achieve comparable results.

A. Related Work

Several IBVS [4][5][6] and PBVS [7][8] approaches has been applied to control aerial vehicles in the last decades. In those standard solutions, the controller uses the visual information as the main source for the target pose computation, *without* taking into account where the target is re-projected into the image plane along the trajectory. A possible solution that usually mitigates such weakness is the kinematic limitation of the multirotor in terms of roll and pitch angles, but this penalizes the vehicle maneuverability. To this end, Ozawa *et al.* [9] present an approach that takes advantage of the rotational-dynamic of the vehicle, where a virtual spring penalizes large rotation with respect to a gravity aligned frame. Some recent approaches map the target features' dynamic into a "virtual image-plane" used to compensate the current roll and pitch angles, in order to keep them close to zero [1][3][10]. Being the re-projection error obtained from the rotation-compensated frame, it is still possible that the target, due to significant rotations, completely leaves the camera field of view.

In [11][12] the authors present two approaches based on a spherical camera geometry, allowing to design the control law as a function of the position while neglecting the angular velocity. Being solely position-based, these kind of methods suffer from the above discussed problems, since the system is still vulnerable to large rotations.

Some recent approaches are based on hybrid techniques, where image features and 3D data are fused together to develop a more stable controller than IBVS or PBVS alone. An example has been presented in [13], where the outputs from IBVS and PBVS methods are fused to form a stable Hybrid controller. Sheckells *et al.* [14] presented an approach where the desired trajectory is obtained by minimizing a cost function over the re-projection error. The proposed optimization procedure leads to computational time constraints that do not allow to constantly re-optimize the whole path while following it, penalizing the vehicle to obtain even more better tracking performance. Our work builds on a part of the problem formulation given in [14], but our solution presents significant differences: (i) the whole problem is decoupled and split into two optimization problems; (ii) The formulation of the target re-projection error assumes a slightly different form, by enabling to scale the different error components; (iii) the NMPC explicitly takes into account two dynamic effects and the low-level controller that runs on the UAV.
The coupling between perception and planning has also been addressed in [2], where an UAV has to simultaneously localize itself with respect to a gap and pass trough it. They plan a ballistic trajectory capable to satisfy both dynamic and perception constraints by maximizing the distance of the vehicle with respect to the edges of the gap.

In all the above mentioned work, except for [14], there is no guarantee that the target is constantly kept in the camera field of view because they don't directly take into account the vehicle dynamics.

B. Contributions

Our method differs from previous works under two main aspects: (i) Unlike standard VS approaches, we guarantee to constantly maintain the target as close as possible to the center of the camera field of view during the whole maneuver; (ii) Making use of a global and a local planner, we allow the multirotor to constantly stay on the optimal path.

II. UAV Dynamic Model

In this section, we describe the vehicle dynamic model that we exploit as a constraint in the optimal trajectory computation.
We express a generic position vector as x_Y^Z denoting the position of the reference frame Y expressed with respect to the reference frame Z. Furthermore, we express a rotation matrix from the reference system Y to the reference system Z as R_Y^Z. For the trajectory planning and the control of the multirotor vehicle, we make use of three main coordinate reference systems: (i) the camera frame with C; (ii) the world fixed inertial frame with I; (iii) the body fixed frame with B, that is the frame attached to the Center of Gravity (CoG) of the UAV. The UAV configuration at each time step is formulated by the position p_B^I and the linear velocities v_B^I of the vehicle CoG, both expressed in the inertial frame, and the vehicle orientation q_B^I. More specifically, the whole state of the vehicle is then expressed as $x = \{p_B^I, q_B^I, v_B^I\}$. At each time step, we also define the tuple of control inputs as $u = \{\phi_{cmd}, \theta_{cmd}, \dot{\psi}_{cmd}, T_{cmd}\}$, where the single terms stands for, respectively, the roll, pitch and yaw rate desired commands and the commanded thrust.

[1]With an abuse of notation, as in other related work, we call here and in the rest of the paper "optimal trajectory" the desired trajectory the multirotor tracks during the flight. Actually, due to the non-linear nature of the cost function, the optimization does not always guarantee the convergence to the optimal, global minimum.

[2]An NMPC provides a sequence of control inputs for a finite temporal horizon.

We employ a widely used dynamic model for multirotors, where the main forces that act on the vehicle are generated from the propellers. More specifically, each propeller generates a thrust force F_T proportional to the square of the motor rotation speed. Moreover, we take into account also two other important effects that became relevant in case of dynamic maneuvers, namely *blade flapping* and induced *drag*. Both of them introduce additional forces in the x-y rotor plane [15]. We model them into a single lumped *drag* coefficient K_D, as shown in [16], [17], leading to the aerodynamic force $F_{aero,i}$:

$$F_{aero,i} = F_{T,i} K_{drag} {R_B^I}^T v_B^I \tag{1}$$

where i stands for the propeller index, $K_{drag} = diag\{K_D, K_D, 0\}$, $F_{T,i}$ is the z component of the $i-th$ thrust force and v_B^I is the vehicle's linear velocity (in the next equations, where there is no confusion, we will omit both the superscripts and subscripts I and B). The final dynamic model of the vehicle can be expressed as follows:

$$\dot{p} = v, \tag{2.a}$$

$$\dot{v} = \frac{1}{m}\Big(R\sum_{n=0}^{n_p}(F_{T,i} - F_{aero,i}) + F_{ext}\Big) + g, \tag{2.b}$$

$$\dot{\phi} = \frac{1}{\tau_\phi}(k_\phi \phi_{cmd} - \phi) \tag{2.c}$$

$$\dot{\theta} = \frac{1}{\tau_\theta}(k_\theta \theta_{cmd} - \theta) \tag{2.d}$$

$$\dot{\psi} = \dot{\psi}_{cmd} \tag{2.e}$$

where m is the mass of the vehicle, F_{ext} are the external forces that act on the multirotor. In our system, we make use of a low-level controller that maps the high-level attitude control inputs in propellers' velocity, as the one provided with the Asctec NEO hexacopter used in the experiments. To achieve better tracking performance, we model this inner control loop as first order dynamic systems, where the model parameters τ_i and k_i are obtained by a system identification procedure [18].

III. Optimal Visual Servoing (OVS)

In this section we describe how we take into account dynamics and perception constraints in planning a trajectory and controlling the multirotor. The first step is the goal pose computation, namely the position and orientation of the vehicle that allows to get the desired view of the target. We then split the Optimal Visual Servoing (OVS) problem into two consecutive stages. First, we compute an optimal global trajectory by solving a non-linear optimization problem. In order to take into account the dynamic and perception constraints, the output trajectory is minimized over the multirotor dynamics and the target re-projection error in the image plane. To track the desired trajectory we then employ a Receding Horizon NMPC controller, where a smaller non-linear optimization problem is solved every time step and only the first control input is actually sent to the multirotor.

A. Goal Pose Computation

Before computing the optimal trajectory, the multirotor has to retrieve the goal pose it aims to reach. Such a pose depends on the task (e.g., inspection or patrolling) and it usually requires the vehicle to frame a target (e.g., landmark or object) from a specific distance and with a specific point of view. Retrieving a relative 3D transformation from the camera is a well-known problem and has been widely studied in the last decades. A widely used technique is based on the solution of a Perspective-n-Point (PnP) problem [19]: such technique requires a prior knowledge about the target object geometry and scale.

Since the choice of the goal pose computation algorithm goes behind the purpose of this work, we assume for the sake of simplicity to have a real-time "black-box" detection framework that outputs: (i) the (u, v) pixel coordinates of the target T in the camera image plane; (ii) the 3D position of the target in the camera frame p_T^C; (iii) the orientation q_C^T of the target object with respect to the camera frame C in terms of *yaw* angle. The goal pose in world I reference system can be then obtained as follows:

$$p_{goal}{}_B^I = p_B^I + q_B^I(q_C^B(d_T^C - p_T^C) + p_C^B) \tag{3.a}$$

$$q_{goal}{}_B^I = q_B^I q_C^B q_T^C \tag{3.b}$$

Where d_T^C is the desired position of the target expressed in the camera frame, while p_C^B and q_C^B are the extrinsic calibration parameters between the camera frame and the body frame.

B. Optimal Trajectory Computation

Once we have the goal pose, we need to generate a discrete trajectory composed by N tuples of the vehicle state vector $\{x_0, ..., x_N\}$ and control inputs $\{u_0, ..., u_N\}$ that minimize the functional cost J subject to the vehicle dynamics equations $f(x_k, u_k)$ described in Sec. II. The time step of such dynamic equations is given by $\frac{t_f - t_0}{N}$, where t_f and t_0 are respectively the final time and the initial time, while N is the number of steps. Additionally, the custom choice of the time variable allows us to define also the nominal speed s_{nom} (i.e. $\frac{\Delta_p}{t_f}$), namely the speed the vehicle is expected to flight. Similarly to [14], we define the cost function as:

$$J(x_{0:N}, u_{0:N-1}) = J_N(x_N) + \sum_{k=0}^{N-1} J_k(x_k, u_k) \tag{4.a}$$

where J_N is the final cost and J_k is the cost along the trajectory. At this point, we split J_k into two main terms. The first one represents the cost over the desired final state and the control effort, and it can be expressed as follow:

$$J'_K(x_k, u_k) = \frac{1}{2}(x_k - x_N)^T Q (x_k - x_N) + \frac{1}{2} u^T R u \tag{4.b}$$

where $Q \geq 0$ and $R \geq 0$ are the matrices that weight the control objectives. In addition, in the second term of J_k we introduce a cost that aims to penalize the re-projection error of the target into the camera field of view. The entire cost in the discrete time step k can be then formulated as:

$$J_K(x_k, u_k) = J'_K(x_k, u_k) + \frac{1}{2} e_i(x_k)^T H e_i(x_k) \quad (4.c)$$

$$e_i(x_k) = \mathcal{P}(x_k, P_i, \pi) - p_i \quad (4.d)$$

where $H \geq 0$ is the penalization term over the target re-projection error and $\mathcal{P}$ is a general camera projection function. Starting from the 3D position of the object in the camera frame P_i, the re-projection error is obtained by the knowledge of the intrinsic calibration parameters of the camera, denoted in Eq. 4.d as π, the extrinsic parameters between the camera reference system C sensor and the body frame B, and the desired position of the target object in the image plane p_i. Differently from [14], we make use of a weighting matrix H in place of a scalar weighting factor, allowing us to scale the different components of the re-projection error. Ideally, we want to have an H that penalizes mostly the error along the smaller dimension of the input image. We set H as follows:

$$H = \begin{pmatrix} h_x & 0 \\ 0 & h_y \end{pmatrix} \quad (4.e)$$

Let $h_{i=x,y}$ be the scale factor related to the smaller dimension $(d_i < d_j)$, we set it as follows:

$$h_i = h_j \times \sigma, \quad \sigma = \frac{d_j}{d_i} \quad (4.f)$$

This enables the UAV to cope with different camera sensor setups. Since we introduce in the cost function J the re-projection error term of the target with respect to the camera image plane, the optimal solution will implicitly allows the vehicle to constantly face the target, maintaining it as close as possible to the center of the image plane.

C. Optimal Control Solver

Once the optimal trajectory has been obtained, the multirotor must closely follow it. To this end we employ an NMPC that repeatedly solves the following optimal control problem:

$$\min_{u,x} \sum_{k=0}^{K-1} \left(\|x_k - x_f\|_Q^2 + \|u_k - u_f\|_R^2 \right) + \|x_K - x_f\|_P^2 \quad (5)$$

$$\begin{aligned} \text{subject to: } & x_{k+1} = f(x_k, u_k) + f_{ext}(d_k) \\ & d_{k+1} = d_k \\ & U_{min} \leq u_k \leq U_{max} \\ & x_0 = x_{init} \end{aligned}$$

where $Q \geq 0$ is the weight factor over the state, $R \geq 0$ is the weight factor over the control inputs and P is the weight factor over the final state. The controller is implemented in a receding horizon fashion, meaning that the aforementioned optimization problem is solved every time step over the fixed time interval $[i, i+K]$. Once the optimization problem has been solved, the optimization procedure is repeated for the time interval $[i+1, i+K+1]$ starting from the state reached in $i+1$ and by using the previous solution as initial guess. By solving this optimization procedure in real-time, the proposed framework simultaneously provides a feed-forward trajectory toward the desired state and a discrete set of control inputs which will be used by the low-level on-board controller. This means that, in practice, at the end of each optimization procedure only the first control input tuple is actually sent to the multirotor controller, then the optimization procedure is repeated.

IV. Simulation Experiments

We tested the proposed framework firstly in a simulated environment by using the RotorS simulator [20] and a simplified multirotor model with a front-facing camera. The mapping between the high-level control input and the propellers velocities is done by a low-level PD controller that aims to resemble the low-level controller that runs on the real multirotor. From the higher controller level point of view, we implemented a receding horizon NMPC [21], where the optimization problem is solved by means of the efficient ACADO solver [22]. To demonstrate the effectiveness of the proposed method, we fix the number of segments N and then flew the virtual vehicle to the desired goal pose by using the approach described in section III and a standard PBVS technique. The latter adopts a linear interpolation technique between the starting and the goal poses obtaining a vector of N intermediate poses. Such interpolated poses are then sent to the same NMPC that computes the trajectory to track them. Once the final time t_f has been fixed, by tuning N it is possible to act on the flight behavior: increasing the number of segments will involve smoother trajectories and control inputs, since the delta-pose between two adjacent desired states segments is smaller. On the other hand, increasing N also brings to higher computational cost when performing the optimal trajectory computation. We used $N = 55$ as trade-off between smoothness and computational velocity.

The goal pose is computed for each run employing an April Marker [23] attached on a virtual building. Since we aim to test our approach with different levels of aggressive maneuvers, we act on the S_{nom} parameter (i.e. changing the final time t_f). In all the experiments we set the initial state to $x = \{8, -12, 14, 0, 0, 1.918, 0, 0, 0\}$. Since the target is always kept in the same location inside the virtual environment, the computed goal state is $x = \{6 + w_x, 2 + w_y, 9.4 + w_z, 0, 0, 1.57 + w_{yaw}, 0, 0, 0\}$, where $w \in \mathbb{R}^4$ is a small white noise random component due to the target detection errors. The relative transformation between the initial and the final pose forces the multirotor to retrieve an optimal trajectory along the 4 principal motion directions of the vehicle.

A. Results

Quantitative image error trajectories for the OVS and PBVS methods for various values of t_f are reported in

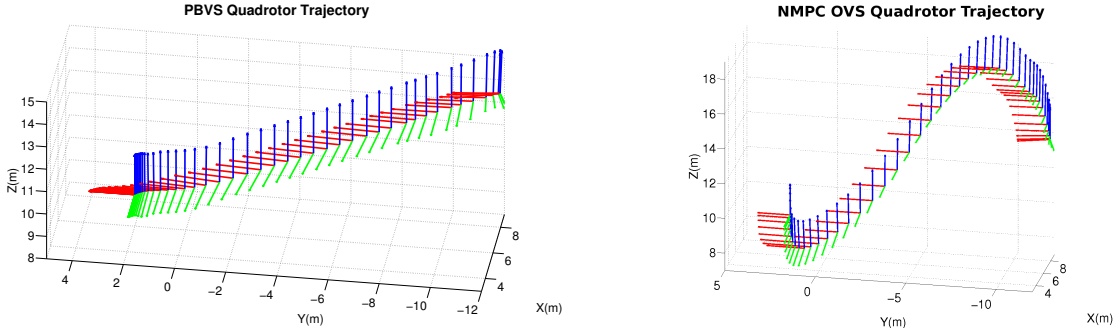


Fig. 2: Example of trajectories obtained using PBVS(a) and NMPC OVS (b) with $s_{nom} = 3.10\frac{m}{s}$: the latter constantly takes into account the target pose during the flight.

TABLE I: Comparison of simulated image error trajectory statistics for each method across different nominal speeds[3].

		Avg. Pixel Error			Max. pixel error		
t_f	S_{nom}	NMPC OVS	PBVS	Sheckells *et al.*[14][3]	NMPC OVS	PBVS	Sheckells *et al.*[14][3]
10.2	2.01	**32.5**	89.2	∼63.8	**55.05**	145.36	∼127.6
7.5	2.73	**45.03**	109.4	∼85.3	**76.91**	199.21	∼195.1
6.6	3.10	**55.2**	125.2	∼90.7	**98.76**	223.67	∼207.9
5.1	4.02	**62.5**	146.9	not available	**115.64**	323.78	not available

TABLE II: Comparison in terms of control effort between a standard PBVS approach and the proposed one.

		RMS Thrust (g)		RMS Roll Ref. (deg)		RMS Pitch Ref. (deg)		RMS Yaw Rate (rad/s)	
t_f	S_{nom}	NMPC OVS	PBVS	NMPC OVS	PBVS	NMPC OVS	PBVS	NMPC OVS	PBVS
10.2	2.01	10.8	10.56	0.15	0.11	0.08	0.075	0.34	0.33
7.5	2.73	10.95	10.79	0.31	0.25	0.33	0.29	0.43	0.46
6.6	3.10	11.21	10.98	0.5	0.43	1.47	0.38	0.49	0.48
5.1	4.02	11.54	11.13	0.9	0.75	1.82	0.69	0.61	0.60

in Table I. The reported results are obtained averaging the performance of PBVS and OVS given the same goal pose and starting from the same initial state for multiple trials. As a preliminary assessment, we also reported some results from the experiments in Sheckells *et al.* [14], showing that our method can provide results comparable with this state-of-the-art approach. It is important to highlight that, given this data, a direct comparison with [14] is not possible, since the pixel error statistics are strictly correlated with the simulation setup which has not been released by the authors.

Remarkably, the target error trajectory along both image axes is almost always lower than both the other approaches. In spite of this, from a qualitative point of view (see Fig. 3), the PBVS trajectory seems to behave better in term of pixel errors at same points. The explanation for such behavior comes from the different shape of the two trajectories. In our case, the vehicle is steered to avoid the target to leave the center of the camera field of view, preferring a constant and possibly small error. In the PBVS case, the trajectory is straightforward, involving bigger errors in the acceleration and deceleration phases, worst average and maximum errors, but sporadically smaller error compared with the OVS approach.

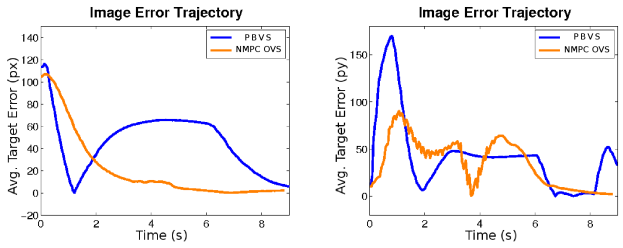


Fig. 3: Comparison of simulated PBVS and NMPC OVS pixel error trajectories for $s_{nom} = 3.10\frac{m}{s}$. Respectively error on the x-axis of the image plane in the left image, while in the right one the pixel error on the y-axis.

From the control inputs point of view, the reduced pixel error comes with an energy effort trade-off: as reported in Table II, the RMS thrust of each OVS trajectory is larger with respect to the corresponding PBVS trajectory. Similar conclusion can be drawn from the attitude points of view since maintaining the target in the center of the camera

[3]We emphasize that the statistics from [14] have been obtained with a different simulation setup, so they represent an indicative performance measure.

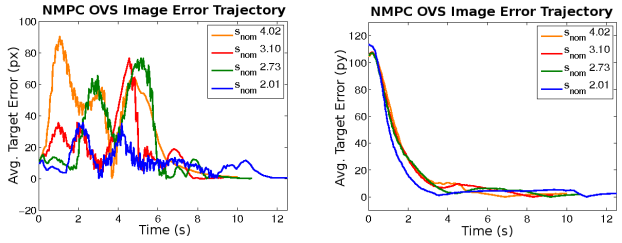


Fig. 4: Comparison of simulated NMPC OVS pixel error trajectories for different values of s_{nom}. Respectively error on the x-axis of the image plane in the left image, while in the right one the pixel error on the y-axis.



Fig. 5: The Asctec NEO hexacopter used for the real experiments.

involves larger angles and yaw rate commands. The choice of the correct behavior depends on the task requirements and, by acting on the optimization parameters Q_k, R_k and H_k, it is possible to obtain the desired trade-off between control effort and image error. Two samples of the trajectories generated by the approaches are depicted in Fig. 2. Often the bigger pixel error terms in a VS scenario occur in the initial and in the final phase, due to the attitude components required to accelerate and decelerate the vehicle. As qualitatively reported in Fig. 2, the OVS trajectory takes into account these two error sources by a small ascending phase at the same time as the forward pitch command. Similarly the trajectory dips softly at the end of the flight so that the target remains in the center of the image plane when the multirotor has to pitch backward in order to decelerate. From the PBVS point of view, the trajectory is more or less a straight line. The vehicle starts suddenly to pitch and to decrease its altitude, involving a bigger pixel error.

V. Real Experiments

We tested the proposed framework on an Asctec NEO hexacopter Fig. 5 equipped with an Intel NUC i7, where we implemented our algorithm in ROS (Robotic Operating system), running on Ubuntu 14.04. The overall weight of the vehicle is 2.8 Kg. For the state estimation we make use of a forward-looking VI-Sensor [24] and the ROVIO (Robust Visual Inertial Odometry) framework [25]. The ROVIO output is then fused together with the vehicle Inertial Measurement Unit (IMU) by using an Extended Kalman Filter (EKF) as described in Lynen *et al.* [26]. The control inputs obtained at each time step by our approach are then sent to the low-level on-board controller by using the UART connection.

We executed several experiments acting on the s_{nom} parameter. In each run the multicopter starts from the same initial state before starting to look for the fixed target. The distance between the vehicle and the target is approximatively 8 m, and the environment is an indoor closed building.
TABLE III: Comparison of image error for each method across different nominal speeds.

		Avg. Pixel Error	
t_f	S_{nom}	NMPC OVS	PBVS
6	1.79	**62.89**	77.2
5	2.11	**88.03**	112.4
4	2.73	**104.17**	146.7
3	3.21	**123.8**	179.2

A. Results

Qualitative results for trajectories with different values of s_{nom} are reported in Fig. 6, while table III reports the average error statistics between OVS and PBVS. Apart from the same error behavior that also appears in the simulation experiments, it is possible to note how the Mean Squared Error (MSE) pixel error for the OVS approach is lower than the PBVS approach. The difference in terms of pixel MSE is bigger in the initial and final phases where the PBVS, in order to accelerate and slow down, is subject to the greatest kinematic movements in terms of roll and pitch angles. Is also useful to highlights how both OVS and PBVS use a noisy target detection approach. In practice, the multirotor is not able to obtain an accurate 3D position of the target object in the environment. This accounts for a constant non-zero pixel MSE, even when the vehicle reach the goal state.

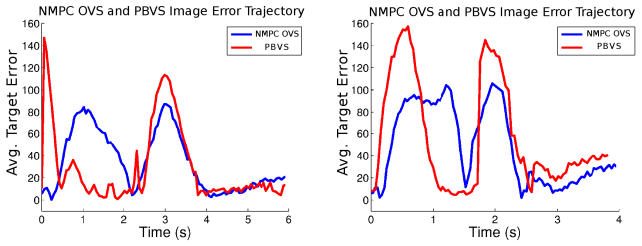


Fig. 6: Comparison of real PBVS and NMPC OVS Mean Squared Error (MSE) pixel trajectories for different values of s_{nom}. Respectively the MSE for $s_{nom} = 2.07$ in the left image, while in the right the MSE for $s_{nom} = 3.1$.

VI. Conclusions

In this work we proposed a novel OVS approach for multirotor vehicles, particularly suitable for agile maneuvers. The method splits the VS into two different optimization problems. In the first one an image-based cost function is minimized in order to find the best trajectory for the vehicle. The optimal trajectory is then tracked by means of an NMPC controller that runs in real-time. It has been shown in simulated and real-world experiments how the proposed approach achieves better performance in terms of

target re-projection error when compared to a standard PBVS approach. In both scenarios it is capable to keep the target object as close as possible to the center of the camera field of view, even when performing fast maneuvers.

As a future work, we will investigate the possibility to iteratively recompute the optimal trajectory to follow, in order to cope also with the case of a moving target and eventually take into account obstacles in the surrounding environment.

REFERENCES

[1] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing." in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[2] D. Falanga, M. Mueggler, E. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[3] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward Image Based Visual Servoing for Aerial Grasping and Perching," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[4] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. C. Cervera, "Computer vision based general object following for gps-denied multirotor unmanned vehicles." in *Proc. of the American Control Conference (ACC)*, 2014.

[5] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 187–198.

[6] N. Guenard, T. Hamel, and R. E. Mahony, "A practical visual servo control for a unmanned aerial vehicle." in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[7] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2002.

[8] L. Mejias, S. Saripalli, P. Campoy, and G. S. Sukhatme, "Visual servoing of an autonomous helicopter in urban areas using feature tracking," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 185–199, 2006.

[9] R. Ozawa and F. Chaumette, "Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[10] H. B. H. Jabbari, G. Oriolo, "Output feedback image-based visual servoing control of an underactuated unmanned aerial vehicle," in *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 2014.

[11] O. Bourquardez, R. E. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Trans. Robotics*, vol. 25, no. 3, pp. 743–749, 2009.

[12] R. Mebarki, V. Lippiello, and B. Siciliano, "Autonomous landing of rotary-wing aerial vehicles by image-based visual servoing in gps-denied environments," in *Proc. of IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR*, 2015.

[13] A. H. A. Hafez, E. Cervera, and C. V. Jawahar, "Hybrid visual servoing by boosting ibvs and pbvs," in *Proc. of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 2008.

[14] M. Sheckells, G. Garimella, and M. Kobilarov, "Optimal visual servoing for differentially flat underactuated systems," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[15] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[16] S. Omari, M. D. Hua, G. Ducard, and T. Hamel, "Nonlinear control of VTOL UAVs incorporating flapping dynamics," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[17] M. Burri, J. Nikolic, C. Hrzeler, G. Caprari, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," in *Proc. of the 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.

[18] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles," *arXiv:1611.09240*, 2016.

[19] T. Petersen, "A comparison of 2D-3D pose estimation methods," *Aalborg University, Aalborgb*, 2008.

[20] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[21] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[22] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, pp. 298–312, 2011.

[23] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[24] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[25] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[26] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

Data Collection Planning with Dubins Airplane Model and Limited Travel Budget

Petr Váňa, Jan Faigl, Jakub Sláma, Robert Pěnička

***Abstract*— In this paper, we address the data collection planning problem for fixed-wing unmanned aircraft vehicle (UAV) with a limited travel budget. We formulate the problem as a variant of the Orienteering Problem (OP) in which the Dubins airplane model is utilized to extend the problem into the three-dimensional space and curvature-constrained vehicles. The proposed Dubins Airplane Orienteering Problem (DA-OP) stands to find the most rewarding data collection trajectory visiting a subset of the given target locations while the trajectory does not exceed the limited travel budget. Contrary to the original OP formulation, the proposed DA-OP combines not only the combinatorial part of determining a subset of the targets to be visited together with determining the sequence to visited them, but it also includes challenges related to continuous optimization in finding the shortest trajectory for Dubins airplane vehicle. The problem is addressed by sampling possible approaching angles to the targets, and a solution is found by the Randomized Variable Neighborhood Search (RVNS) method. A feasibility of the proposed solution is demonstrated by an empirical evaluation on modified benchmarks for the OP instances to the scenarios with varying altitude of the targets.**

I. Introduction

The problem addressed in this paper is motivated by data collection missions in which an Unmanned Aerial Vehicle (UAV) is requested to gather information from a set of target locations while the vehicle travel budget is limited, e.g., due to a limited battery capacity or fuel tank volume. The locations are known in advance, and each location is specified as a 3D point. Moreover, each location is associated with a reward value representing the importance of the measured data at the location, which can be for example a camera snapshot in surveillance missions [1] or measurements received by a remote transmission from sensor fields [2]. Having the limited travel budget, the problem is to select the most valuable locations that can be visited with respect to the available travel budget. This type of problems can be formulated as a variant of the *Orienteering Problem* [3] which is herein extended to consider the kinematic constraints of Dubins airplane model [4] for a fixed-wing aircraft. Therefore, we call the studied problem the *Dubins Airplane Orienteering Problem* (DA-OP).

In the regular OP, a set of all possible target locations with positive rewards is given together with the prescribed initial and final locations of the vehicle. The OP stands to maximize the sum of the collected rewards while the total traveled distance is shorter or equal to the given travel budget T_{max}. Since the budget may not allow visiting all the given targets, the OP is similar to the Knapsack problem in finding the most suitable subset of the targets, such that the sum of the collected rewards is maximized. However, it is necessary to evaluate the shortest path visiting the selected target locations to ensure T_{max} constraint and maximize the collected rewards, e.g., by visiting additional locations as a result of the saved travel cost by the shortest path. Therefore, the OP is also connected to the *Traveling Salesman Problem* (TSP) in which we find the shortest path for a particular subset of the target locations. Notice, the OP becomes a decision version of the TSP when the budget is equal to the minimal required travel distance to visit all the locations. Hence, the OP is at least NP-hard [5].

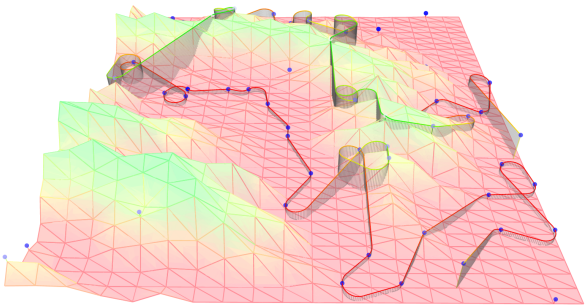


Fig. 1. An instance of the Dubins Airplane Orienteering Problem (DA-OP) together with its solution found by the proposed RVNS-based algorithm. The target locations are represented as the blue disks, and the color of the trajectory and the terrain corresponds to the specific altitude. The lowest parts are in the red while the highest parts are in green.

Unlike the regular OP, the studied DA-OP extends the target location to 3D and further consider limitations of fixed-wing UAVs. The introduced DA-OP can be considered as an extension of the existing *Dubins Orienteering Problem* (DOP) proposed in [6], which has been further generalized to the *Dubins Orienteering Problem with Neighborhoods* (DOPN) in [7], [8]. In the current paper, we formulate the OP for the Dubins airplane model in 3D and propose to solve the introduced DA-OP by a modified variant of the Variable Neighborhood Search (VNS) algorithm [9]. An example of the DA-OP solution is depicted in Fig. 1.

The rest of the paper is organized as follows. An overview of the related work on the Dubins airplane model and existing OP solvers is presented in the next section. The introduced DA-OP is formally defined in Section III and the proposed solver is described in Section IV. Results on empirical evaluation of the proposed approach are reported in Section V. Finally, Section VI concludes the paper.

The authors are with the Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic {vanapet1|faiglj|slamajak|penicrob}@fel.cvut.cz

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S.

II. Related work

The introduced *Dubins Airplane Orienteering Problem* (DA-OP) is mostly related to the two research fields which are directly utilized to address the DA-OP. The first field is related to approaches for determining the shortest possible paths for the Dubins airplane model in 3D, while the second field is a class of methods for solving the OP. The most related approaches of both these fields are further discussed in the rest of this section.

A. Shortest paths for the Dubins Airplane Model

The problem of finding the shortest curvature-constrained path in 2D was addressed by L. E. Dubins [10] in 1957. He showed that the optimal path connecting two locations with prescribed leaving and arrival headings of Dubins vehicle is composed of up to three segments which are either straight segments (S) or circle segments (C). It results in two basic path types of the so-called Dubins maneuvers: CSC and CCC, for which zero length segments are allowed. All the path types can be determined by a closed-form expression which represents a fast method for finding the optimal solution of a simple trajectory planning between two configurations of Dubins vehicle.

The shortest curvature-constrained path in 3D is studied in [11] where the authors proved that every minimizer is either a helicoidal arc or of the form CSC or CCC. This results in the analogy to the 2D curvature-constrained maneuvers since the Dubins maneuver is a special case. However, to the best of our knowledge, there exists no analytic solution of the shortest curvature-constrained path in the 3D.

A suboptimal approach for the 3D path generation is proposed in [12] which enables to satisfy arbitrary initial and final configurations of the vehicle while the constraint on the pitch angle is met. The resulting path is a CSC maneuver, and the Dubins maneuver in the plane is utilized as an initial estimation of the requested maneuver. In [13], numerical and geometric approaches for generating CSC paths in 3D are proposed. The authors claim that the numerical approach finds the optimal solution assuming the points are sufficiently far apart but no formal proof is provided.

The constraint on the pitch angle determines the maximal climb/dive angle and the authors of [4] introduce the *Dubins airplane model* to address both constraints of real UAVs: the bounded curvature and pitch angle. The proposed model treats the vertical position changes independently to the horizontal movement. The Dubins airplane model is further modified in [14] to be more consistent with a fixed-wing UAV kinematics. In [15], the authors propose to utilize 7-th order Bézier curves to compose the path as an alternative approach. However, in this paper we considered the Dubins airplane model [4] as a suitable model for extension of the curvature-constrained data collection planning in 3D.

B. Solving OP-based problems

The DA-OP is a direct extension of the recently introduced DOP [6] where the vehicle altitude is fixed to a constant value. Similarly to the DOP, the basic properties and possible approaches to address the DA-OP can be defined by the underlying combinatorial optimization problems: the *Dubins Traveling Salesman Problem* (DTSP) [16] and the regular Euclidean OP [3]. Therefore a brief overview of the existing approaches is provided in this section.

The first type of methods to the DTSP are decoupled approaches where the sequence of the visits to the targets is determined prior the optimization of the headings, e.g., by solving the Euclidean TSP as in the Alternating algorithm [16]. However, the most related approach to the DA-OP is the sampling based method [17] that allows transforming the DTSP to the Generalized Asymmetric TSP (GATSP) by using a discrete set of possible headings for each target location. Further, the GATSP can be transformed to the Asymmetric TSP (ATSP) that can be solved by existing solvers, e.g., Lin-Kernighan heuristic [18]. Finally, the DTSP can also be addressed by soft-computing techniques such as genetic [19] and memetic [20] algorithms and also by recently proposed unsupervised learning [21], [22].

The proposed DA-OP is a variant of the OP with the limited travel budget T_{max}, and thus only a subset of the target locations to be visited within T_{max} has to be determined together with the respective sequence of the visits to the selected targets. Since there probably does not exist an algorithm for solving the introduced DA-OP directly, the related existing approaches are Euclidean OP solvers for which a detailed survey can be found in [23]. The most related OP approach is the Variable Neighborhood Search (VNS) metaheuristic proposed by the authors of [9] that has been modified in [24] by Sevkli et al. to tackle the OP. Since the proposed solver of the DA-OP is leveraging on this method, it is described in the next paragraph in a more detailed way.

The basic idea of the VNS-based OP solver [24] is to search the solution space with a predefined set of neighborhood structures which are critical for the algorithm performance. The algorithm starts with an initial solution and explores the solution space by applying two procedures. The first one is the *shake* procedure, which aims to diversify the solution, and thus escapes from local minima. The second procedure is called *local search* and it utilizes the given neighborhood structures to optimize the current solution.

The original VNS algorithm exhaustively examines all possible solution modifications defined by the neighborhood structures. This can be too computationally demanding for large instances, and therefore, a randomized version of the VNS (RVNS) for the OP has been proposed in [24] to speed up a solution of the OP. The RVNS variant has been applied for solving the DOP in [6].

The introduced DA-OP is addressed by a variant of the RVNS utilized in [6], and the proposed method is presented in Section IV right after the formal introduction of the problem presented in the next section.

III. Problem Statement

The introduced *Dubins Airplane Orienteering Problem* (DA-OP) is an extension of the existing DOP [6] into

3D with respect to the constraints of the Dubins airplane model [4], which is a suitable model for a fix-wing aircraft with curvature and pitch angle constraints. The vehicle state q is represented by the configuration (p, θ, ψ), where $p = (x, y, z)$ stands for the vehicle position in 3D, $p \in \mathbb{R}^3$, $\theta \in \mathbb{S}^1$ is the vehicle heading, and $\psi \in \mathbb{S}^1$ is the vertical angle of the vehicle. The dynamics of the vehicle can be described by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \cdot \cos\psi \\ \sin\theta \cdot \cos\psi \\ \sin\psi \\ u_\theta \cdot \rho^{-1} \end{bmatrix}, \tag{1}$$

where v is a constant forward velocity of the vehicle, ρ stands for the minimum turning radius, and u_θ is the control input controlling the vehicle heading θ. The control output is considered to be limited by $u_\theta \in [-1, 1]$.

In the Dubins airplane model, the pitch angle ψ can be changed immediately to any value from the given interval $\psi \in [\psi_{min}, \psi_{max}]$, which does not fully correspond to real physical constraints but it is a suitable approximation for most of the fixed-wing UAVs.

The DA-OP follows the OP, where Dubins airplane model is utilized and the set of n target locations S is given as $S = \{s_1, s_2, \ldots, s_n\}$ for $s_i \in \mathbb{R}^3$. The DA-OP stands to find a feasible path which maximizes the sum of the collected rewards while the limited travel budget T_{max} is respected. The initial vehicle location s_1 and the final location s_n are given and their rewards are assumed to be zero $r_1 = r_n = 0$. For other locations, individual rewards are given and they are strictly positive, i.e., $r_i > 0$ for all $1 < i < n$.

The DA-OP consists of selecting a subset of k target locations from S and determination of the sequence to their visits which can be described as a sequence $\Sigma_k = (\sigma_1, \ldots, \sigma_k)$ where σ_i stands for the respective index of the target location, i.e., $1 \leq i \leq k$, $1 \leq \sigma_i \leq n$ and because of given initial and final locations, $\sigma_1 = 1$ and $\sigma_k = n$. In contrast to the Euclidean OP, the DA-OP contains also a determination of the vehicle heading for each selected location given by $\Theta_k = (\theta_{\sigma_1}, \ldots, \theta_{\sigma_k})$. The vehicle is not allowed to change its altitude while it visits a particular target location, and therefore, the pitch angle of the vehicle at the target location is prescribed to be zero. Hence, the vehicle state q_{σ_i} at the target σ_i is defined by the target location s_{σ_i} and the corresponding heading θ_{σ_i}. The DA-OP can be then defined as the following optimization problem:

$$\begin{aligned} \textit{maximize}_{\,k,\Sigma_k,\Theta_k} & \sum_{i=1}^{k} r_{\sigma_i} \\ \textit{subject to} & \sum_{i=1}^{k-1} \mathcal{L}(q_i, q_{i+1}) \leq T_{max} \ , \\ & q_i = (s_{\sigma_i}, \theta_{\sigma_i}, 0), \\ & \sigma_1 = 1, \sigma_k = n \ , \\ & \sigma_i \neq \sigma_j, \forall i, j : i \neq j \ , \end{aligned} \tag{2}$$

where $\mathcal{L}(q_i, q_{i+1})$ is the length of the maneuver of the Dubins airplane model between q_i and q_{i+1} respecting (1).

IV. Proposed solution to the Dubins Airplane Orienteering Problem (DA-OP)

The proposed algorithm for the DA-OP leverages on the recently introduced DOP [6]. The main difference of the proposed approach to the DOP [6] is in the altitude changes that are caused by the limited pitch angle necessary to connect target locations by a continuous path. This limitation has to be considered because of possible different altitudes of the waypoints corresponding to the different elevations of the ground locations, which does not occur for the DOP on a plane. Moreover, in DOP, a solution of the Euclidean OP has similar length, especially for small minimal turning radii of the vehicle. Since the Dubins airplane model has limited pitch, the Euclidean distance cannot be simply utilized for a heuristic approach.

The proposed algorithm consists of three main parts to address challenges arising from the DA-OP. The first part is related to the vehicle heading that is uniformly sampled, and all three-dimensional maneuvers are pre-computed for each pair of possible states of the vehicle. The second part is an examination of possible collisions of the computed maneuvers with the terrain, and unfeasible maneuvers are discarded. Each 3D maneuver determined by the Dubins airplane model is sampled, into a finite set of states, and for each such a state the 3D model of the vehicle is checked for a possible collision with the terrain modeled as a mesh, e.g., using RAPID library [25]. Finally, having only the feasible 3D maneuvers connecting the possible waypoints, the solution of the DA-OP is found by the randomized variant of the VNS algorithm similarly as for the DOP in [6]. The first and third parts are further described in the following subsections as the collision test is performed similarly as in any motion planning algorithm such as PRM or RRT.

A. Heading Sampling and 3D Dubins Maneuvers

In the first step of the proposed algorithm, a particular instance of the DA-OP is discretized by sampling the possible vehicle states at each waypoint covering the respective target location. We utilize uniform sampling of m_i samples of the vehicle heading. The heading samples are given by a set $H_i = \{\theta_i^1, \ldots, \theta_i^{m_i}\}$ for the i-th waypoint.

Having the sampled state of the vehicle, a 3D maneuver for the Dubins airplane model connecting each pair of the vehicle state associated with the target locations is determined. Dubins airplane maneuvers are more complex to compute than Dubins maneuvers in the plane because of the possible altitude changes. Based on [4], the maneuvers are divided into three cases according to the altitude difference between the initial and final states of the vehicle: *low altitude*, *high altitude*, and *medium altitude*. The process of determining 3D maneuvers is detailed in [14], and therefore, we provide only a short overview of the procedure here.

First, Dubins maneuver is calculated as a two-dimensional projection of the final maneuver. Then, the next step depends on the altitude change. In the *low altitude* case, Dubins maneuver is long enough for achieving required altitude gain, i.e., the maximal pitch angle is not exceeded. The

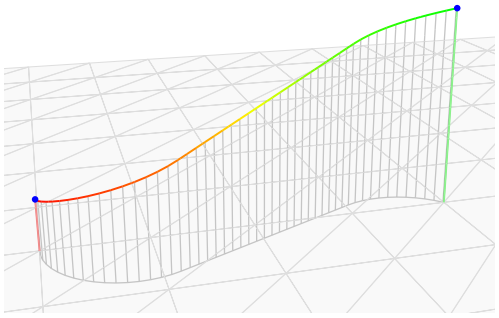


Fig. 2. A Dubins airplane maneuver in 3D with changing the vehicle altitude

calculated 2D Dubins maneuver is modified by changing the turn segments into helices segments, and the direction of the straight segment is changed accordingly. An example of such a *low altitude* maneuver is depicted in Fig. 2 where the altitude changes are highlighted by a color change.

For the *high altitude* case, the altitude gain is too high and cannot be achieved by altitude changes of the 2D trajectory. Therefore, the trajectory is modified by adding a certain number of spiral segments at the start or end of the maneuver to provide sufficient travel distance for the altitude correction according to the motion constraints of the Dubins airplane model. The maneuver is prolonged at the lower end because of minimizing potential terrain collisions.

The *medium altitude* case is a mix of the two previously described cases. The length of the generated 2D maneuver is not sufficient for the altitude change, but a necessary prolongation is smaller than one spiral turn. Therefore, in this case, the maneuver is generated by adding a third turning segment to achieve the required length for the correct altitude change respecting the Dubins airplane model.

Finally, vehicle maneuvers which intersect with the terrain are removed. Therefore, the proposed algorithm guarantees that the resulting solution is feasible and the vehicle does not collide with the ground.

B. VNS-based algorithm for the DA-OP

Having the sampled states of the vehicle at the target locations with the corresponding maneuvers for the Dubins airplane model, the instance of the DA-OP can be considered as a directed graph where each state stands for the graph node and the edges connecting the nodes are the determined maneuvers. The proposed solution of the DA-OP is based on the VNS discrete optimization technique [24] that searches for a maximal rewarding feasible tour within the graph such that the tour cost does not exceed the limited travel budget T_{max}. The VNS is a meta-heuristic algorithm which searches the solution space using neighborhood structures. Its performance depends on the neighborhood structures utilized for the sequence changes in the `shake` and `localSearch` procedures of Algorithm 1.

The current solution of the proposed algorithm is represented by a sequence P of the all targets except the initial and terminal locations which are prescribed, i.e., P is a sequence of $n-2$ targets $P = (\sigma'_1, \sigma'_2, \ldots, \sigma'_{n-2})$. P is

Algorithm 1: Randomized VNS for the DA-OP

Data: Targets S with the associated rewards r_i, the travel budget T_{max}, sampled vehicle states and the lengths of the corresponding maneuvers

Result: Solution represented by Σ_k

```
P, Σ_k := foundInitialSolution()
while termination condition is not met do
    P' := shake(P)
    P'' := localSearch(P')
    Σ''_k := selectLocations(P'', T_max)
    if L(Σ''_k) ≤ T_max and R(Σ''_k) > R(Σ_k) then
        P := P''
        Σ_k := Σ''_k
    end
end
```

a permutation of labels of the targets $S \setminus \{s_1, s_n\}$, and thus it holds that for any $\sigma'_i \in P$, $\sigma'_i \neq 1$ and $\sigma'_i \neq n$. This allows the VNS algorithm to modify the whole sequence P without maintaining the prescribed initial and terminal locations. Thus, the VNS neighborhood structures work on the all targets in P.

Due to the limited budget T_{max}, only a subset of k locations can be visited, and therefore, the first $k-2$ elements of P are considered for a solution of the DA-OP that is prolonged by the prescribed initial and terminal locations $\sigma_1 = 1$ and $\sigma_k = n$, i.e., Σ_k is constructed from P as $\Sigma_k = (\sigma_1, \sigma'_1, \sigma'_2, \ldots, \sigma'_{k-2}, \sigma_k)$. However, it is necessary to determine the number of selected targets k to be visited by the trajectory that does not exceed T_{max}. Such a value of k is iteratively determined in the `selectLocations` procedure as the highest number for which the trajectory satisfies T_{max}. The trajectory for a particular k is found using the pre-computed maneuvers as follows.

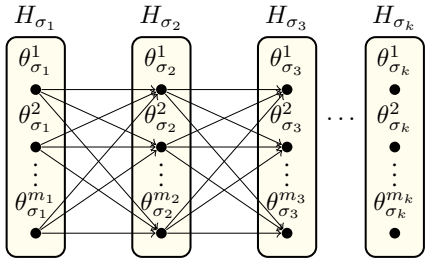


Fig. 3. An example of the graph structure for finding the shortest data collection tour for uniformly sampled headings at the target locations

Since multiple samples of the vehicle state are considered, the sequence Σ_k itself does not fully specify the data collection trajectory. Therefore, each target location has associated particular vehicle state for each sampled heading value, and we can create a graph structure representing possible paths connecting the waypoints of the particular targets in the current sequence Σ_k, see Fig. 3. Then, this graph is used to find the shortest tour by a feed-forward search which time complexity can be bounded by $\mathcal{O}(km^2)$, where m is

the number of samples per each target and k is the current number of the selected targets to be visited.

After that, the current sequence Σ_k'' is evaluated to be a new best solution found so far at the Step 6 of Algorithm 1. The length of the trajectory represented by Σ_k'' is $\mathcal{L}(\Sigma_k'')$ and the sum of the collected rewards along the sequence is $R = \sum_{i=1}^{k} r_{\sigma_i}$, which is denoted $R(\Sigma_k'')$ for brevity.

The important part of the VNS-based search algorithm is a generation of the first feasible solution in the `foundInitialSolution` procedure by an iterative insertion of the target locations to the sequence. Then, the initial solution is improved by searching for possible insertion/removal of other targets such that, a target with the highest ratio f of the reward increase to the data collection trajectory prolongation is selected and inserted to the best position in the sequence:

$$f = \frac{R(\Sigma_{k+1}) - R(\Sigma_k)}{\mathcal{L}(\Sigma_{k+1}) - \mathcal{L}(\Sigma_k)}, \tag{3}$$

where Σ_k stands for the solution sequence before the insertion of the particular target.

The VNS-based algorithm utilized four different neighborhood structures which locally changes P to maximize the sum of the collected rewards. In each iteration, the actual solution is randomly changed to explore the state space in the procedure `shake` which chooses from the first two neighborhood structures:

- **Insert** selects a random element in P and moves it to a different randomly chosen position in P,
- **Exchange** selects two random elements in P and exchanges them.

Unlike the previous procedure which aims to explore the space and escape from the local minima, the `localSearch` procedure tries to optimize the current solution and eventually reaches the global optima. In the original VNS, the exhaustive search of all possible changes is used which can be computationally demanding for instances with a high number of targets. In contrast, the utilized randomized variant of the VNS examines only n^2 randomly selected local changes, where n is the number of all targets. Two different neighborhood structures are applied to locally optimize the current solution:

- **Path insert** selects a random sub-sequence from P and moves it to a different randomly chosen position in P,
- **Path exchange** selects two random sub-sequences in P and exchanges them.

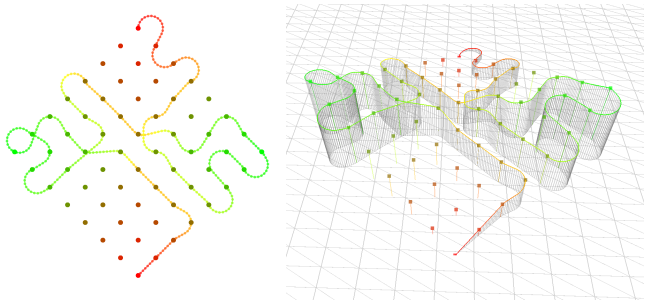
The termination condition can be chosen arbitrarily according to the application scenario. A high number of iterations provides a better solution at the cost of computational requirements. For example the algorithm can be terminated after the specified number of iterations or sooner, e.g., after a given maximum number of iteration without the solution improvement.

V. Results

The proposed VNS-based approach for the DA-OP has been evaluated using standard benchmarks for the Euclidean OP introduced by Chao et al. [26] that have been modified for the 3D problems that require the Dubins airplane model. In particular, we consider the benchmarks Set 64 with 64 target locations where a positive reward value associated with each target location is selected from the interval (0, 42] except the initial and final location which has zero rewards as in the ordinary OP. We define the altitude z_i of each target as a multiplication of the reward value and a specified constant β:

$$z_i = \beta \cdot r_i, \tag{4}$$

which means the most rewarding locations are at the highest altitudes, and thus makes the problem suitable for evaluation of the Dubins airplane model with altitude changes.



(a) 2D projection (b) 3D projection

Fig. 4. An example solution of Set 64 instance with $\beta = 0.1$ and $T_{max} = 80$. The locations at the highest altitude are the most rewarding (shown in green) and the locations at the lowest altitude with the zero reward are shown in red.

Four different values of $\beta \in \{0, 0.05, 0.1, 0.15\}$ have been utilized to show the relationship between the altitude change and the path length. The vehicle turning radius is set to $\rho = 0.7$, the minimum pitch angle to $\psi_{min} = -10°$, and the maximum pitch angle to $\psi_{max} = 20°$. The allowed ascension ratio is greater than the descension to respect properties of real vehicles. The number of heading samples is set to 16 which has been empirically found as a suitable trade-off between the computational requirements and the quality of found solutions. An example of the DA-OP instance is depicted in Fig. 4 in which the asymmetric limits of the pitch angle results in different path patterns near the start and end of the final solution.

The proposed DA-OP solver has been implemented in C++ with the Dubins airplane model according to [14]. All the presented results have been computed using a single core of the AMD Phenom 1090T CPU running at 3.2 GHz. The algorithm has been terminated after 10 000 iterations of the main loop.

Average values of the collected rewards for the particular T_{max} and β are depicted in Fig. 5. The results indicate that for small altitude changes (i.e., $\beta = 0.05$), the algorithm is capable of finding *low altitude* maneuvers for all possible maneuvers. With the increased altitude changes for $\beta = 0.1$, the ascending pitch angle is still sufficient, but the limited descenting angle causes small modifications of the original Dubins maneuver found for the 2D projection. Further, for $\beta = 0.15$, the altitude changes are so huge that additional

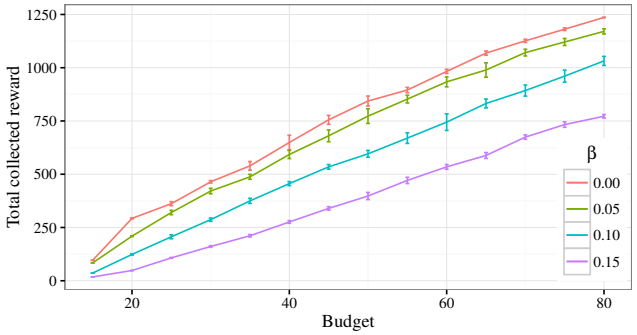


Fig. 5. Average values of the total collected rewards for the given travel budget and particular β. The results are computed from 10 trials.

spiral segments have to be inserted, which increases the length of the maneuvers, and thus the travel budget T_{max} is spent for visiting lower number of targets, and the sum of the collected rewards is lower.

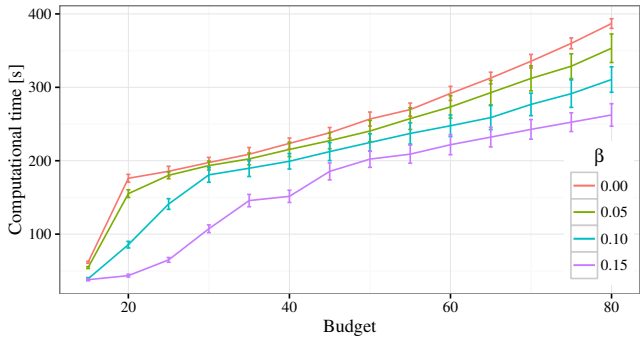


Fig. 6. Average required computational times based on the given travel budget for various β determined from 10 trials

The required computational time is shown in Fig. 6. The proposed VNS-based algorithm runs about hundreds of seconds for the examined instances with 64 targets. It can be observed that the computational time increases with the maximal travel budget, which is natural as a high travel budget allows to visit more locations, and thus more combinations, insertions, and path exchanges can be performed.

VI. Conclusion

In this paper, we introduce a problem formulation for data collection planning with Dubins airplane model suitable for 3D scenarios with fixed-wing UAVs and limited travel budget. The formulation is based on the DOP extended for the Dubins airplane model, and we call the introduced problem as the *Dubins Airplane Orienteering Problem* (DA-OP). We propose to utilize the VNS-based algorithm to solve the DA-OP in which possible vehicle headings are sampled by uniform sampling strategy. The performance of the algorithm has been evaluated in a series of DA-OP instances, and regarding the presented results, the proposed approach seems to be feasible.

Since the proposed approach is probably the first extension of the orienteering problems in 3D data collection planning with UAVs, we aim to investigate the performance of other 3D vehicle models within the presented solution framework for the DA-OP.

References

[1] K. Vicencio, B. Davis, and I. Gentilini, "Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods," in *IROS*, 2014, pp. 2985–2990.

[2] D. G. Macharet, A. A. Neto, V. F. da Camara Neto, and M. F. Campos, "Data gathering tour optimization for dubins' vehicles," in *IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1–8.

[3] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[4] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a dubins airplane," in *46th IEEE CDC*, 2007, pp. 2379–2384.

[5] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval research logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[6] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[7] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem with neighborhoods," in *ICUAS*, 2017, pp. 1555–1562.

[8] J. Faigl and R. Pěnička, "On close enough orienteering problem with dubins vehicle," in *IROS*, 2017, (to appear).

[9] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, vol. 130, no. 3, pp. 449–467, 2001.

[10] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and tetz rminal positions and tangents," *American Journal of Mathematics*, pp. 497–516, 1957.

[11] H. J. Sussmann, "Shortest 3-dimensional paths with a prescribed curvature bound," in *34th IEEE CDC*, vol. 4, 1995, pp. 3306–3312.

[12] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, A. Pironti, and M. Virgilio, "Algorithms for 3d uav path generation and tracking," in *45th IEEE CDC*, 2006, pp. 5275–5280.

[13] S. Hota and D. Ghose, "Optimal path planning for an aerial vehicle in 3d space," in *49th IEEE CDC*, 2010, pp. 4902–4907.

[14] M. Owen, R. W. Beard, and T. W. McLain, "Implementing dubins airplane paths on fixed-wing uavs," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, pp. 1677–1701.

[15] A. A. Neto, D. G. Macharet, and M. F. Campos, "3d path planning with continuous bounded curvature and pitch angle profiles using 7th order curves," in *IROS*, 2015, pp. 4923–4928.

[16] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for dubins' vehicle," in *American Control Conference*. IEEE, 2005, pp. 786–791.

[17] K. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based roadmap methods for a visual reconnaissance uav," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 7568.

[18] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[19] X. Yu and J. Hung, "A genetic algorithm for the dubins traveling salesman problem," in *IEEE International Symposium on Industrial Electronics*, 2012, pp. 1256–1261.

[20] X. Zhang, J. Chen, B. Xin, and Z. Peng, "A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets," *Chinese Journal of Aeronautics*, vol. 27, no. 3, pp. 622–633, 2014.

[21] J. Faigl and P. Váňa, "Self-organizing map for the curvature-constrained traveling salesman problem," in *ICANN*, 2016, pp. 497–505.

[22] J. Faigl and P. Váňa, "Unsupervised learning for surveillance planning with team of aerial vehicles," in *IJCNN*, 2017, pp. 4340–4347.

[23] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[24] Z. Sevkli and F. Sevilgen, "Variable neighborhood search for the orienteering problem," *Computer and Information Sciences–ISCIS*, pp. 134–143, 2006.

[25] "Rapid - robust and accurate polygon interference detection," [cit. 15-05-2017]. [Online]. Available: http://gamma.cs.unc.edu/OBB/

[26] I.-M. Chao, B. L. Golden, and E. A. Wasil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, 1996.

Motion planning for long reach manipulation in aerial robotic systems with two arms

A. Caballero[1], M. Bejar[2], A. Rodriguez-Castaño[1], A. Ollero[1]

***Abstract*—In this paper an aerial robotic system with two arms for long reach manipulation (ARS-LRM) while flying is presented. The system consists of a multirotor with a long bar extension that incorporates a lightweight dual arm in the tip. This configuration allows aerial manipulation tasks increasing considerably the safety distance between rotors and manipulated objects. The objective of this work is the development of planning strategies to move the ARS-LRM system for both navigation and manipulation tasks. With this purpose, a simulation environment to evaluate the algorithms under consideration is required. Consequently, the ARS-LRM dynamics has been properly modeled with specific methodologies for multi-body systems. Then, a distributed control scheme that makes use of nonlinear control strategies based on model inversion has been derived to complete the testbed. The motion planning problem is addressed considering jointly the aerial platform and the dual arm in order to achieve wider and safer operating conditions. The operation of the planner is given by an RRT*-based algorithm that optimizes energy and time performance in cluttered environments for both navigation and manipulation tasks. This motion planning strategy has been tested in a realistic industrial scenario given by a riveting task. The satisfactory results of the simulations are presented as a first validation of the proposed approach.**

I. INTRODUCTION

Among the numerous applications in which unmanned aerial vehicles (UAVs) can be used, aerial manipulation is arousing much interest. Potential applications in this field include instrument deployment, maintenance operation and contact inspection in industrial sites in which the access is very dangerous or costly. The motivation is to decrease risks and operational costs in theses scenarios with the support of aerial manipulation systems. Small size rotorcraft can indeed access to hard-to-reach places more easily than human operators, avoiding unnecessary risks for industrial workers and allowing inspection and maintenance operations without shut-downs of the facilities (the mandatory safety policy in case of human operation) and the use of scaffolding, cranes and other means.

These new promising applications of aerial robotic systems for manipulation tasks bring also new challenges that are still unresolved. On the one hand, it is necessary to develop new manipulation tools such as adapted arms or grippers that can be seamlessly integrated into the airframe to provide manipulation capabilities to UAVs. The existing algorithms for operating independently the UAV and the manipulators should be extended for achieving autonomous operation of the integrated system. In this respect, one of the most challenging issues is the development of new methods that consider both the UAV and the manipulator when planning the motion of the complete system. When moving between different locations inside a dense industrial installation, this planning will be essential for the generation of accurate movements close to obstacles. In addition, it will also enable rapid, agile manoeuvres (e.g., using the arm to let the aerial manipulator turn quickly) aiming to approach to the goal location avoiding waste of battery.

Many research works about aerial manipulation have been recently published. [1] presents the design of several lightweight, low-complexity grippers that allow quadrotors to grasp and perch on branches or beams and pick up and transport payloads. In a very different system scale, [2] proposes a system for aerial manipulation, composed of a helicopter and an industrial manipulator. The usage of an industrial manipulator is motivated by practical applications which were identified in different cooperation projects with the industry.

However, among the different contributions focused on aerial manipulation very few of them consider configurations with more than one arm. In [3] a human-size and lightweight dual arm manipulator is integrated in a multirotor platform and tested in outdoor flights. On the other hand, [4] proposes a dual arm aerial manipulator to turn a valve that requires a tightly integrated control scheme between aircraft and both manipulators. The arm-aircraft system for valve turning is validated through flight tests. Concerning theoretical contributions, [5] introduces a generic planar aerial manipulator with any number of arms attached at the center of mass of a UAV. The authors prove that this kind of systems are differentially flat regardless the number of joints of each arm and their kinematic and dynamic parameters. This theory is validated by simulating object grasping and transportation tasks.

On the other hand, although a large amount of works have been focused on the development of control techniques for the system integrating the aerial vehicle and the manipulator devices, not many of them deal with the associated motion planning problem. Furthermore, the existing contributions like [6] usually assume a strong simplification by addressing the planning problem in a decoupled way, i.e. adopting independent planners for the UAV and the manipulators that

[1] A. Caballero (e-mail: mbejdom@upo.es), A. Rodriguez-Castaño (e-mail: castano@us.es) and A. Ollero (e-mail: aollero@us.es) are with the University of Seville, Seville (Spain).

[2] M. Bejar (e-mail: mbejdom@upo.es) is with the University Pablo de Olavide, Seville (Spain).

switch their operation according to the mission phase. This means that during the navigation phase, the arm configuration is assumed to be fixed and hence the UAV planner is in charge of planning the motion. In contrast, the manipulation phase is resolved by using the manipulators planners and assuming that the aerial platform is not moving.

This work explores dual arm configurations that guarantee long reach manipulation in those scenarios where the target is far from the operation area of the UAV. In order to meet these requirements a new aerial robotic system with two arms for long reach manipulation (ARS-LRM) is proposed. More precisely, the system consists of a multirotor with a long bar extension that incorporates a lightweight dual arm in the tip (see Fig. 1). Thus, the long bar extension increases considerably the safety distance between rotors and manipulated objects while the dual arm offers extended manipulation capabilities with respect to the single arm configurations existing in the literature.

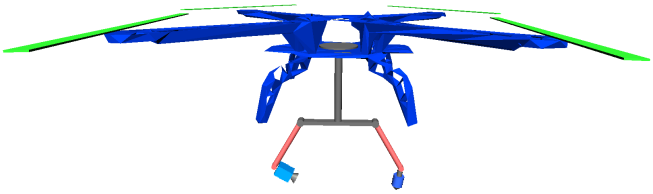


Figure 1: Aerial robotic system with two arms for long reach manipulation (ARS-LRM).

Concerning the motion planning problem, this paper investigates strategies for the ARS-LRM in cluttered environments in both navigation and manipulation tasks. For this purpose, the aerial platform and the dual arm device are considered jointly within the planner operation, which constitutes a remarkable difference to previous contributions where the planning problem was addressed in a decoupled way. This integrated strategy allows the consideration of a more complete set of system states that in turn will make it possible to achieve wider and safer operating conditions. Regarding the operation basis of the planner, an RRT*-based algorithm that optimizes energy and time performance has been developed.

This paper presents a first proof of concept of the ARS-LRM system described in previous paragraphs. The work begins with Section II presenting the structure of the integrated platform, the corresponding multi-body dynamical model and finally the distributed control approach derived for the system. Then, in Section III the proposed planning algorithm is explained in detail. In order to better illustrate its benefits, Section IV defines a realistic industrial scenario given by a riveting task. After presenting the complete system as well as the motivating scenario, Section V includes several simulations of the planning capabilities of the ARS-LRM system to endorse the validity of the proposed algorithm. Finally, Section VI is devoted to conclusions and future work.

II. MODELING AND CONTROL

A. System Description

As can be seen in Fig. 2, the proposed aerial robotic system for long reach manipulation (ARS-LRM) consists of a multirotor with a long bar extension that incorporates a lightweight dual arm in the tip. This configuration allows aerial manipulation tasks increasing considerably the safety distance between rotors and manipulated objects. Furthermore, the dual arm offers extended manipulation capabilities with respect to the single arm configurations existing in the literature. In this first prototype of the system each separate arm is composed of two links, corresponding the lower one to the end effector, but further extensions of the manipulation chain are considered in future work.

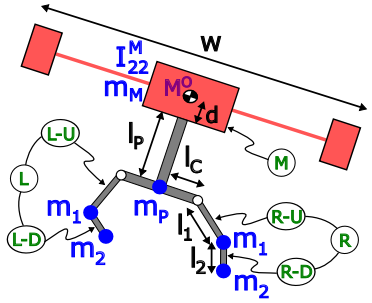


Figure 2: Geometry and mass distribution of the ARS-LRM system.

A planar characterization of the system will serve for establishing a first proof of concept for the ARS-LRM setup. This simplified approach eases the modeling and control developments while maintaining the operation basis of the system. Following this assumption, the multirotor is characterized by a mass m_M, a principal moment of inertia I^M_{22} and dimensions $2d \times w$. Regarding the long bar, its longitude is given by l_P and it is assumed to be aligned with the UAV center of mass M^O at a distance d. The cross-piece in the tip is defined by a length of $2l_C$. The total mass of the long bar and the cross-piece is m_P and will be treated as a punctual mass located where the long bar and the cross-piece intersects for simplicity purposes. Finally, the 2 arms are characterized respectively by the lengths of their links -l_1 for upper links and l_2 for lower links- and their masses -m_1 and m_2-, where again the masses will be treated as punctual masses located at the distal end of each link in order to derive more manageable expressions. The values of the aforementioned parameters are shown in Table I.

Table I: ARS-LRM Parameters

	Parameter	Value	Units
Mass and Inertia	m_M	6.5	kg
-	I^M_{22}	0.0933	$kg{\cdot}m^2$
-	m_P	0.15	kg
-	m_1	0.06	kg
-	m_2	0.03	kg
Geometry	d	0.1	m
-	w	0.9	m
-	l_P	0.2	m
-	l_C	0.1	m
-	l_1	0.15	m
-	l_2	0.05	m

B. Modeling

According to [7], the dynamics of a multirotor under 20kg is mostly determined by its mechanical model. This paper embraces the same assumption and consequently the behaviour of the ARS-LRM platform will be described by means of the mechanical model of the complete system. To this end, specific methodologies for multi-body systems will be applied below.

Several approaches can be found in the literature to derive equations of motion for mechanical systems. However, Kane's method [8] has proved in [9] to hold some unique advantages over other traditional approaches when addressing multi-body robotic systems like the ARS-LRM under study in this paper.

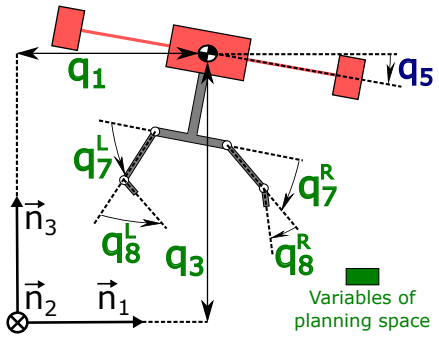


Figure 3: ARS-LRM model. Configuration variables. In green, the variables selected later for the planning space.

The configuration variables selected as system generalized coordinates are the longitudinal q_1 and vertical q_3 positions of the UAV center of mass M^O in the inertial reference frame N, the multirotor pitch angle q_5 and the joint angles both for left L and right R arms q_7^L, q_8^L, q_7^R and q_8^R (see Fig. 3). Generalized speeds u_i are defined as:

$$\begin{aligned} {}^N\mathbf{v}^{M^O} &= u_1\mathbf{n_1} + u_3\mathbf{n_3} & {}^{M}\boldsymbol{\omega}^{R-U} &= u_7^R\mathbf{n_2} \\ {}^N\boldsymbol{\omega}^{M} &= u_5\mathbf{n_2} & {}^{R-U}\boldsymbol{\omega}^{R-D} &= u_8^R\mathbf{n_2} \\ & & {}^{M}\boldsymbol{\omega}^{L-U} &= -u_7^L\mathbf{n_2} \\ & & {}^{L-U}\boldsymbol{\omega}^{L-D} &= -u_8^L\mathbf{n_2} \end{aligned} \quad (1)$$

which leads to the following kinematic differential equations:

$$\begin{aligned} \dot{q}_i &= u_i \quad (i = 1, 3, 5) \\ \dot{q}_j^k &= u_j^k \quad (j = 7, 8\ ;\ k = R, L) \end{aligned} \quad (2)$$

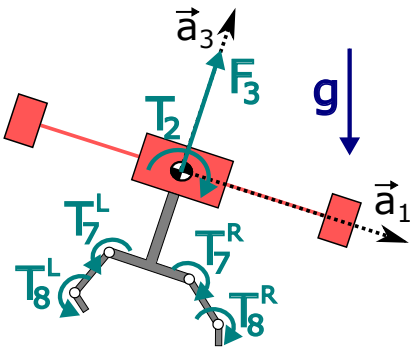


Figure 4: ARS-LRM model. Forces and torques applied to the system.

Regarding forces and torques exerted on the ARS-LRM system (see Fig. 4), the rotors generate a resultant lifting force $F_3\mathbf{a_3}$ applied at the multirotor center of mass M^O as well as a torque $T_2\mathbf{a_2}$ applied to rigid body M. On the other hand, control actions governing the manipulator are given by the torques applied to the arm joints $T_7^R\mathbf{a_2}$, $T_8^R\mathbf{a_2}$ $-T_7^L\mathbf{a_2}$ and $-T_8^L\mathbf{a_2}$.

Application of Kane's method through MotionGenesis software [10] leads to the following dynamic differential equations for translation and rotation, where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are dense matrices depending on the configuration variables q_5, q_7^R, q_8^R, q_7^L, q_8^L and the system parameters defined in Table I and g is the gravity acceleration.

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_3 \\ \dot{u}_5 \\ \dot{u}_7^R \\ \dot{u}_8^R \\ \dot{u}_7^L \\ \dot{u}_8^L \end{bmatrix} = \mathbf{A}\begin{bmatrix} F_3 \\ T_2 \\ T_7^R \\ T_8^R \\ T_7^L \\ T_8^L \end{bmatrix} + \mathbf{B}\begin{bmatrix} (u_5)^2 \\ (u_7^R)^2 \\ (u_8^R)^2 \\ (u_7^L)^2 \\ (u_8^L)^2 \end{bmatrix} + \mathbf{C}\begin{bmatrix} u_5u_7^R \\ u_5u_8^R \\ u_5u_7^L \\ u_5u_8^L \\ u_7^Ru_8^R \\ u_7^Lu_8^L \end{bmatrix} + \mathbf{D}g \quad (3)$$

C. Control

After modeling the ARS-LRM system, a distributed control scheme has been derived to provide the system with the capacity of executing navigation and manipulation manoeuvres. The objective is the completion of the simulation environment that will allow the investigation of new planning strategies to properly command the ARS-LRM platform. With this purpose, a basic control structure that makes use of nonlinear control strategies based on model inversion shall suffice to complete the testbed.

Regarding the multirotor, the control scheme is inspired by [7] and consists in linearizing the system through model inversion and applying PID control laws to the resultant dynamics. The underlying principle of control will be the adjustment of the multirotor lifting force vector, in order to generate the translational accelerations required to reduce position error. A general overview of the control scheme is shown in Fig. 5, where D_{13}^{-1}, K_5^{-1} and D_5^{-1} blocks represent, respectively, the inversions of the translational dynamics, rotational kinematics and rotational dynamics.

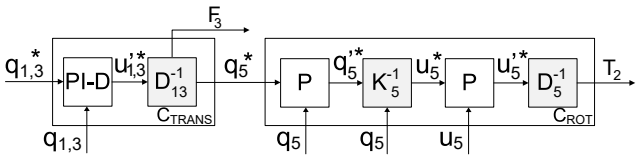


Figure 5: Scheme of the UAV controller.

The control strategy selected for each arm is again based on linearization through model inversion and PD control, which yields a nonlinear control law capable of commanding the link positions of both arms. The schematic representation of this approach is shown in Fig. 6 where D_{78}^{-1} represents the block in charge of inverting arm dynamics and torques T_7^R, T_8^R, T_7^L and T_8^L correspond to the output signals of the controller.

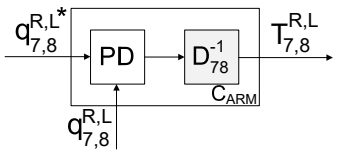


Figure 6: Scheme of each arm controller.

The parameters of the controller have been tuned by means of the classic pole assignment method. The selected values constitute a trade-off that guarantees a proper dynamics range while the common mechanical limitations of this kind of systems are not overreached.

III. MOTION PLANNER

Sampling based planners like the family of RRT algorithms [11] have demonstrated high potential in finding fast solutions for high-dimensional robots [12]. Furthermore, some of these methods bring the possibility of generating motion plans that optimize certain cost functions, as for the case of RRT* variations [13]. This makes it possible to find an optimal solution in terms of a specific metric. Taking all these considerations into account, an RRT*-based algorithm that optimizes energy and time performance has been selected for the ARS-LRM system.

Another determining factor for planner performance is the planning space considered when exploring the different possibilities of motion. In this work the planner explores jointly the configuration variables of the aerial platform (with the exception of pitch angle q_5) and the dual arm (variables in green color in Fig. 3). This integrated strategy allows the consideration of a more complete set of system states. In this way, it is possible to achieve wider and safer operating conditions since equivalent configurations in terms of final effector positions can be differentiated according to the positions of both the multirotor and the intermediate links.

The pseudocode of the common structure of an RRT* algorithm is shown in the code fragment 1. In order to apply this general structure to the ARS-LRM system, some of the intermediate functionalities have been customized for the problem under study. These particular developments will be dealt with in detail hereafter.

Algorithm 1 RRT* algorithm

Input: $map, param$
Output: $trajectory$
1: $Tree \leftarrow INITIALIZATION(map, param)$
2: **for** $i = 1$ **to** $iter_{max}$ **do**
3: $x_{rand} \leftarrow SAMPLE()$
4: $x_{nearest} \leftarrow NEAREST(Tree, x_{rand})$
5: $x_{new} \leftarrow STEER(x_{nearest}, x_{rand})$
6: **if** $\sim COLLISION(x_{nearest}, x_{new}, map)$ **then**
7: $x_{near} \leftarrow NEAR(Tree, x_{new})$
8: $Tree \leftarrow ADD(x_{nearest}, x_{near}, x_{new})$
9: $Tree \leftarrow REWIRE(x_{near}, x_{new})$
10: **end if**
11: **end for**
12: $trajectory \leftarrow TRAJECTORY(Tree)$

A. Computation of the Nearest Node

The $NEAREST(Tree, x_{rand})$ function finds the nearest node $x_{nearest}$ to the random state x_{rand} generated in the sampling-based exploration of the planning space. Since nodes include state information both for multirotor and dual arm accordingly with the integrated operation basis of the planner, there will be two different measurements for calculating the nearest node: the difference in position for the multirotor and the difference in angle for the arm joints. Thus, there appears the need of defining a homogenizing metric. The reference velocities u_{ref} (for the UAV) and w_{ref} (for the joints) have been defined with this purpose of transforming the heterogeneous measurements into a common metric given by the time magnitude required for each system component to move between the configurations associated with the nodes under analysis. The equations corresponding to this normalization approach are presented below:

$$\begin{aligned} t_{UAV} &= \frac{\sqrt{(\Delta q_1)^2 + (\Delta q_3)^2}}{u_{ref}} \\ t_{ARMS} &= \frac{\max\left(\left|\Delta q_7^R\right|, \left|\Delta q_8^R\right|, \left|\Delta q_7^L\right|, \left|\Delta q_8^L\right|\right)}{w_{ref}} \\ x_{nearest} &= \min_{x \in Tree} \left(\max\left(t_{UAV}|_x, t_{ARMS}|_x\right)\right) \end{aligned} \quad (4)$$

where Δq_i denotes the increment in variable q_i when going from the tree node x to the sampled node x_{rand}, that is, $\Delta q_i = q_i^{rand} - q_i^x$.

B. Collision Checking

The $COLLISION(x_{nearest}, x_{new}, map)$ function checks if the branch that would link two nodes produces some collision with the obstacles included in the map. To this end, a representative set of intermediate configurations between the nodes is generated using interpolation. Then, each intermediate configuration is investigated to see if any part of the system collides with the obstacles defined in the scenario.

This operation deserves special attention since it plays an important role in the advanced functionality of the ARS-LRM planner that allows differentiating equivalent configurations in terms of final effector positions according to the positions of both the multirotor and the intermediate links. The consideration of the different geometries of the system components, together with joint exploration of the planning space for both system components, are crucial features in this respect. Concerning the former, simplified models that alleviate the computational burden of collision checking but maintaining at the same time their capability to express the heterogeneity existing in the geometry of the different parts, are the desirable option. To this end, the multirotor has been considered rectangularly shaped while the dual arm and the long bar extension are modeled by rectilinear bars with negligible section. Regarding the obstacles, all of them have been considered round.

Another aspect that requires further consideration is the algorithm selected for detecting the collisions. In the case of

the multirotor, the approach is straightforward since it only requires checking whether the position of the center of mass is within the limits of the rectangular region that produces collisions with the obstacle (see Fig. 7).

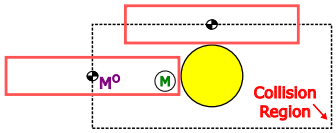


Figure 7: Scheme of the collision checking for the multirotor.

In contrast, the collision management for the extension bar and the dual arm consists of translating the collision condition to the angular space as shown in Fig. 8. In this way, the obstacle is characterized in terms of the minimum and maximum link angle that may produce a collision. Then, taking into account also the distance to the obstacle, it is possible to check the collision with a considerably reduction in the computational load with respect to standard procedures.

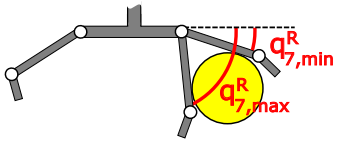


Figure 8: Scheme of the collision checking for the right upper link.

C. Computation of the Set of Near Nodes

The $NEAR(Tree, x_{new})$ function finds the set of tree nodes x_{near} that satisfy simultaneously the following conditions with respect to their distances to the new candidate node x_{new}: the difference in multirotor position is less than threshold γ_{UAV} and the differences in link orientations are all less than threshold γ_{ARMS}. This definition can be expressed mathematically as follows:

$$\begin{aligned} \rho_{UAV} &= \sqrt{(\Delta q_1)^2 + (\Delta q_3)^2} \\ \rho_{ARMS} &= \max\left(\left|\Delta q_7^R\right|, \left|\Delta q_8^R\right|, \left|\Delta q_7^L\right|, \left|\Delta q_8^L\right|\right) \end{aligned}$$

$$x_{near} = x \in Tree \,/\, \begin{cases} \rho_{UAV}|_x \leq \gamma_{UAV} \\ \rho_{ARMS}|_x \leq \gamma_{ARMS} \end{cases} \quad (5)$$

where Δq_i denotes the increment in variable q_i when going from the tree node x to the new candidate node x_{new}, that is, $\Delta q_i = q_i^{new} - q_i^x$.

D. Cost Functions

In order to apply the RRT* optimization sequence within the $ADD(x_{nearest}, x_{near}, x_{new})$ and $REWIRE(x_{near}, x_{new})$ functions, two different cost indices have been defined: the operation time of the complete system (CF_T), and the linear and angular displacements produced in the multirotor and the arm joints respectively (CF_E). These cost indices can be formulated as follows:

$$\begin{aligned} CF_T &= \max\left(t_{UAV}, t_{ARMS}\right) \\ CF_E &= p_1 \cdot \rho_{UAV} + p_2 \cdot \sigma_{ARMS} \end{aligned} \quad (6)$$

where t_{UAV} and t_{ARMS} were defined in equations (4); ρ_{UAV} was defined in equations (5); $\sigma_{ARMS} = \left|\Delta q_7^R\right| + \left|\Delta q_8^R\right| + \left|\Delta q_7^L\right| + \left|\Delta q_8^L\right|$ with Δq_i denoting the increment in variable q_i between the nodes in which the cost function is being evaluated ($\Delta q_i = q_i^{to} - q_i^{from}$); and $p_{1,2}$ are two weighting parameters that allow the prioritization of movements with minimum displacements in the multirotor or the dual arm.

IV. APPLICATION SCENARIO: RIVETING TASK

In order to demonstrate the validity of the motion planning strategy presented in previous section, the algorithm will be tested in a realistic industrial scenario given by a riveting task. The schematic description of the scenario is shown in Fig. 9, where coloured circles correspond to pipes existing in the industrial facility and surrounding circumferences denote the safety regions whose violation would be treated as a collision. As can be seen, the ARS-LRM system will be commanded to place two rivets with its right arm (target points marked in red) while the left arm provides visual feedback by pointing a visual camera integrated as end effector (see Fig. 1). In this first proof of concept, the riveting operations will assume ideal conditions, i.e. absence of interaction forces, since force control has not been implemented yet in the ARS-LRM system to facilitate the analysis of the planner results.

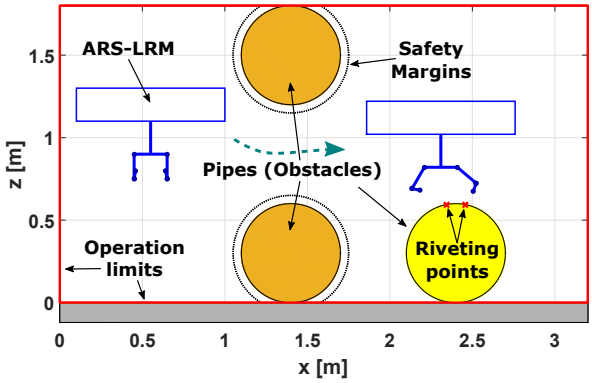


Figure 9: Application scenario given by a riveting task.

The achievement of the riveting objectives defined previously requires the execution of certain intermediate operations that include both navigation and manipulation manoeuvres:

1) **Navigation phase**: this phase corresponds to the system displacement required to reach an observation position over the riveting area. After this, a short transition phase not requiring planner execution will enforce a ready-to-go configuration for the first riveting manoeuvre that will be accomplished during the manipulation phase.
2) **Manipulation phase**: this phase covers the different manoeuvres involved in the manipulation task under consideration, the riveting operation.
    a) **Rivet placement:** approaching to the target point in the perpendicular direction to the target point plane by the riveting effector integrated in the right arm.

b) **Release:** opposite manoeuvre to the rivet placement in which the riveting effector leaves the target point, again following the perpendicular direction to the target point plane.
c) **Switching:** manoeuvre of the complete ARS-LRM system to switch between the ready-to-go configurations for riveting points 1 and 2.

V. SIMULATION RESULTS

In this section the RRT*-based algorithm derived for the ARS-LRM system has been used to calculate a motion plan that commands the riveting task presented in the previous section. The index selected for optimization has been the cost function CF_E defined in Section III-D. Furthermore, not only the planned trajectory will be a matter of study but also the trajectory executed by the controlled ARS-LRM model when receiving the former as control reference.

Due to the high dimension of the planning space considered by the ARS-LRM planner (see Fig. 3), a continuous treatment of the variable ranges considered in the sampling operation would lead to significantly elevated values of the execution times required for achieving convergent solutions. The former suggests the adoption of discretization patterns that guarantee bounded execution times for the planner. Table II shows the discretization patterns adopted for the navigation and manipulation phases in the riveting scenario.

Table II: Discretization of the planning-space

Phase	Navigation	Manipulation
Variable	Discretization Pattern	
q_1	$[45\,,\,50\,,\,\ldots\,,\,275]\,cm$	$[225\,,\,230\,,\,\ldots\,,\,255]\,cm$
q_3	$[30\,,\,35\,,\,\ldots\,,\,170]\,cm$	$[80\,,\,85\,,\,\ldots\,,\,120]\,cm$
q_7^R	$[0\,,\,45\,,\,90]^o$	$[30\,,\,40\,,\,\ldots\,,\,140\,,\,150]^o$
q_8^R	Fixed value of 170°	$[-45\,,\,0\,,\,45]^o$
q_7^L	$[0\,,\,45\,,\,90]^o$	Fixed value of 60°
q_8^L	Fixed value of 170°	Fixed value of 110°

Considering all the above information, the motion planner of the ARS-LRM system has been executed for the riveting scenario. As advanced before, the resultant plan (represented with green lines in the figures) has been also provided to the controlled ARS-LRM system in order to analyze the close-loop behavior of the system (represented with blue lines in the figures) when following the planned trajectory. This simulation work has been carried out in a Matlab-Simulink framework that represents the graphical evolution of the system variables and the corresponding virtual reality animation [14].

The results corresponding to the navigation phase are presented in Fig. 10 and Fig. 11. In Fig. 10 the trajectory followed by the ARS-LRM system is illustrated by the dotted line representing the movement of the UAV center of mass M^O. In Fig. 11 the evolution of the planning-space variables, both for the planned trajectory (green line) and the close-loop executed trajectory (blue line), has been shown. As can be observed, the planned trajectory succeeded in commanding efficiently the controlled ARS-LRM system through the navigation phase without producing collisions with the obstacles existing in the scenario.

Regarding the manipulation phase, Fig. 12 and Fig. 13 illustrate the achieved results. As in the navigation phase, Fig. 12 shows a schematic representation of the manoeuvres associated with the manipulation phase. Similarly, in Fig. 13 the evolution of the planning-space variables, both for the planned trajectory (green line) and the close-loop executed trajectory (blue line), has been represented for this phase. Once again the planned trajectory succeeded in commanding efficiently the controlled ARS-LRM system through the different manipulation manoeuvres involved in the riveting task. It is worth highlighting that Fig. 12 (switching) illustrates how the jointly consideration of the planning space for the multirotor and the dual arm allows the optimization of the switching manoeuvre between the riveting points. More precisely, the motion planner takes advantage of the multirotor vertical displacement to carry out the switching manoeuvre of the riveting effector in a more efficient way.

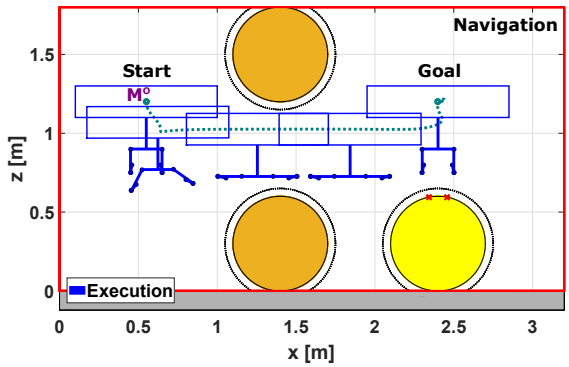


Figure 10: Navigation phase. The ARS-LRM system navigates through the obstacles. The dotted line represents the simulated center of mass position of the multirotor M^O.

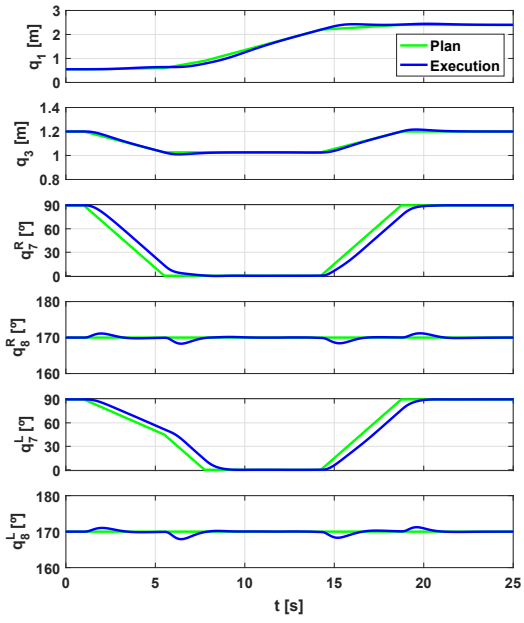


Figure 11: Navigation phase. Representation of the planning-space variables: planned trajectory (green) and executed trajectory (blue).

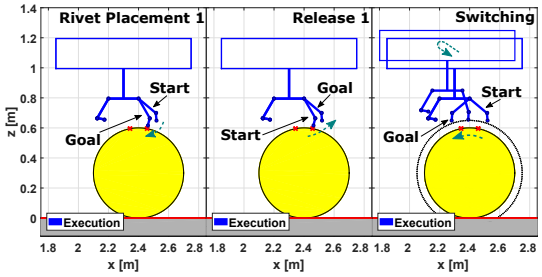


Figure 12: Manipulation phase. The ARS-LRM system places the first rivet and then switches between the ready-to-go configurations.

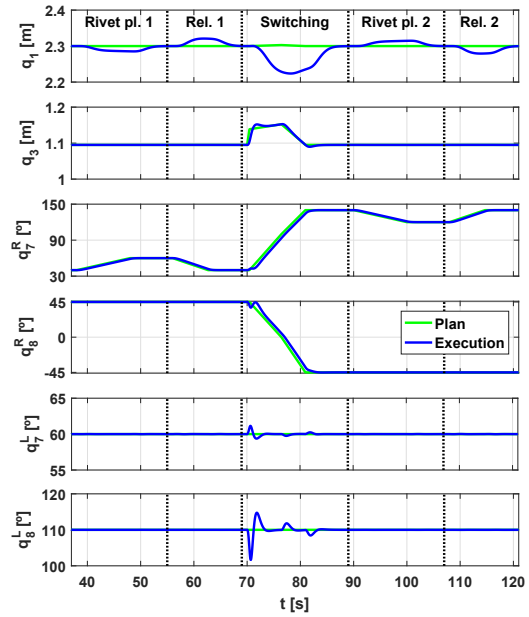


Figure 13: Manipulation phase. Representation of the planning-space variables: planned trajectory (green) and executed trajectory (blue).

VI. CONCLUSIONS

This paper has presented motion planning strategies for an aerial robotic system with two arms for long reach manipulation (ARS-LRM). In order to evaluate the algorithms under consideration, a simulation environment that characterizes the system behaviours that are relevant for planner operation was required. Consistently with this requirement, the ARS-LRM platform has been described in detail together with its potential benefits: a considerable increment in the safety distance between rotors and manipulated objects, and the extended manipulation capabilities offered by the dual arm. Taking this description as reference, the dynamics of the system has been modeled with specific methodologies for multi-body systems. Furthermore, a distributed control scheme that makes use of nonlinear control strategies based on model inversion has been derived to complete the testbed.

With respect to the planning approach, several features justify the relevance of this contribution. The aerial platform and the dual arm device have been considered jointly within the planner operation. In this way, it is possible to achieve wider and safer operating conditions since equivalent configurations in terms of final effector positions can be differentiated according to the positions of both the multirotor and the intermediate links. On the other hand, the planner operation is driven by an RRT*-based algorithm that optimizes energy and time performance in cluttered environments for both navigation and manipulation tasks.

In order to demonstrate the validity of the motion planning strategy presented, the algorithm has been tested in a realistic industrial scenario given by a riveting task. As was discussed in the simulation section, the planned trajectory succeeded in commanding efficiently the controlled ARS-LRM system through navigation and manipulation phases without producing collisions with the obstacles existing in the scenario. These satisfactory results are presented as a first validation of the proposed approach.

ACKNOWLEDGMENTS

This work has been supported by the European Project AEROARMS, funded by the Horizon 2020 research and innovation programme of the European Commission under grant agreement No 644271 and by the AEROMAIN project of the Spanish RETOS Programme DPI2014-59383-C2-1-R.

REFERENCES

[1] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2668–2673, 2011.

[2] K. Kondak, F. Huber, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero, "Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2107–2112, 2014.

[3] A. Suarez, A. E. Jimenez-Cano, V. Vega, G. Heredia, A. Rodriguez-Castaño, and A. Ollero, "Lightweight and human-size dual arm aerial manipulator," *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1778–1784, 2017.

[4] C. Korpela, M. Orsag, and P. Oh, "Towards valve turning using a dual-arm aerial manipulator," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3411–3416, 2014.

[5] B. Yüksel, G. Buondonno, and A. Franchi, "Differential flatness and control of protocentric aerial manipulators with any number of arms and mixed rigid-/elastic-joints," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 561–566, 2016.

[6] R. Ragel, I. Maza, F. Caballero, and A. Ollero, "Comparison of motion planning techniques for a multi-rotor UAS equipped with a multi-joint manipulator arm," *Workshop In Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pp. 133–141, 2015.

[7] K. Kondak, M. Bernard, N. Meyer, and G. Hommel, "Autonomously flying VTOL-robots: Modeling and control," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 736–741, 2007.

[8] T. Kane and D. Levinson, *Dynamics. Theory and Applications*. McGraw-Hill, 1985.

[9] L. A. Sandino, M. Bejar, and A. Ollero, "A survey on methods for elaborated modeling of the mechanics of a small-size helicopter. Analysis and comparison," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 219–238, 2013.

[10] Motiongenesis Kane 5.x. *http://www.motiongenesis.com/*.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] S. H. Tang, W. Khaksar, N. Ismail, M. Ariffin, and M. K. Anuar, "A review on robot motion planning approaches," *IEEE Access*, pp. 15–29, 2012.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[14] Motion planning for manipulation in aerial systems with two arms. Simulation results. *https://youtu.be/vN8Jf7G2QR8* and *https://youtu.be/PUT3r0TUQeQ*.

Multi-robot human scene observation based on hybrid metric-topological mapping

Laetitia Matignon[1,2], Stephane d'Alu[1,3] and Olivier Simonin[1,3]

***Abstract*— This paper presents an hybrid metric-topological mapping for multi-robot observation of a human scene. The scene is defined as a set of body joints. Mobile robots have to cooperate to find a position around the scene that maximizes the number of observed joints. It is assumed that the robots can communicate but have no map of the environment. The map is updated cooperatively by exchanging only high-level data, thereby reducing the communication payload. The mapping is also realized in an incremental way to explore promising areas of the environment while keeping state-space complexity reasonable. We proposed an on-line distributed heuristic search combined to this hybrid mapping. We showed the efficiency of the approach on a fleet of three real robots, in particular its ability to quickly explore and find the team position maximizing the joint observation quality.**

I. INTRODUCTION

Many robotic applications require to observe or recognize activities of one or more humans. A network of static cameras cannot deal with complex perturbations as dynamic occlusions or lighting changes. Recent works on **active distributed perception** are interested in using mobile cameras potentially embedded on mobile robots. These robots cooperate and merge information to move to adequate places to cover blind spots or react to changing conditions. In [1], active perception is performed between one mobile robot and a set of fixed cameras to assist and guide people in urban settings. But they do not consider the issue of coordinating multiple mobile robots to improve the perceptual information. In [2], distributed visual recognition of hand gestures is realized with a group of mobile robots. Hand images are classified by each robot from its point of view. A distributed consensus protocol allows the swarm to reach as a whole a final decision about the issued gesture. However, the navigation and coordination of the robots are very simple and are not used to improve the recognition.

In this article, we are interested in **human activity recognition with a fleet of mobile robots**. The objective is to exploit the mobility of the robots so that they adapt their position to find the spatial configuration that maximizes the joint recognition of a human activity (cf. Fig. 1). The joint recognition is done by integrating information collected by the fleet. The main challenge is to provide cooperation between robots when each individual point of view does not allow a satisfactory recognition, *e.g.* because of the presence of occlusions. Robots must communicate and coordinate to obtain the most complementary observations.

We propose an **on-line distributed solution** to the problem of positioning a team of mobile robots for an observation task of a human scene. We assume that the environment is unknown and has obstacles and occlusions that can prevent some displacements and observations of the scene (cf. Fig. 1). Robots only know the relative location of the scene so they have to explore and build a map of the environment. This map must gather various data concerning obstacles and human scene perception so as to be used by the distributed control strategy. To this end, **a cooperative hybrid metric-topological mapping** using the information of multiple mobile sensors is presented in this article.

In the following, section II proposes a formalization of the problem and the criterion we want to maximize. Then section III introduces an hybrid metric-topological mapping. In section IV, the on-line distributed control strategy is detailed. Finally, we present experimental results with a multi-robot team in section V.

II. PROBLEM SETTINGS

We are interested in **multi-robot observation of a human scene**, defined as the activity of a person in a bounded area. In a first stage, we limit our objective to detect the human body pose, that can be characterized by a set of skeleton joints. The objective is to optimize the configuration of the robot team around the scene so that the number of skeleton joints monitored by the team is maximized. In this article, we consider that skeleton joints are relatively static during one activity. Thus the task of human skeleton observation consists

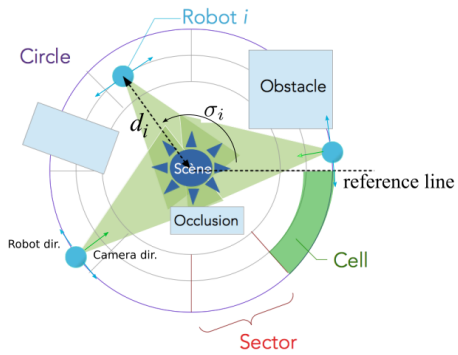


Fig. 1. Joint observation of a scene (human activity) with a fleet of $m = 3$ robots. The navigation model is based on circles with a spatial discretization in cells, where $C = 2$ circles and $D = 8$ sectors in this example.

[1]CITI Lab. Inria Chroma team, INSA Lyon 6 avenue des Arts, 69680 Villeurbanne cedex, France firstname.lastname@inria.fr

[2]Univ Lyon, Université Lyon 1, LIRIS, CNRS, UMR5205 Villeurbanne, F-69622, France

[3]INSA Lyon, Université de Lyon, 20 Avenue Albert Einstein, 69100, Villeurbanne, France

in obtaining a complete view of the targets (skeleton joints) at each new activity.

Several approaches have been proposed concerning the deployment of multiple robots to achieve the observation of moving targets. Most of these approaches are classified and discussed in a recent review [3], based on four major control techniques. Among these, the CMOMMT[1] framework [4] aims to dynamically position robots to maximize the number of targets under observation and the duration of observation of each target.

In the following, we use the CMOMMT formulation to formalize our problem of multi-robot observation of a human scene and the criterion we want to maximize.

Definition 1. *The **CMOMMT** model [4] is defined as a tuple* $< S, \mathcal{V}, K >$ *where:*

- S *a two-dimensional, bounded spatial region;*
- $\mathcal{V}$ *a team of* m *mobile robots, where* $\nu_{i,i=1,...,m}$ *is a robot with observation sensors that are potentially noisy and of limited range;*
- $K(t)$ *a set of* n *targets, where* $\kappa_{j,j=1,...,n}(t)$ *is a target, that is located within region* S *at time* t*.*

A robot ν_i is observing a target when the target is within ν_i's sensing range. In our case, a target is a skeleton joint. It is observed when a robot is able to track the joint with its sensor.

Definition 2. *The **observation matrix** $O(t)$ [4] is defined as* $O(t) = [o_{ij}(t)]_{m \times n}$ *such that:*

$$o_{ij}(t) = \begin{cases} 1 & \text{if robot } \nu_i \text{ is observing target } \kappa_j(t) \text{ at time } t \\ 0 & \text{otherwise.} \end{cases}$$

Definition 3. *The **observation** $o_i(t)$ of a robot ν_i at time t is defined as a binary vector of size n such that:*

$$o_i(t) = [o_{i1}(t), ..., o_{in}(t)]. \quad (1)$$

To evaluate the observations made by each robot and by the team, we define the individual and joint observation qualities.

Definition 4. *The **individual observation quality** $q_i(t)$ made by a robot ν_i at time t is defined as:*

$$q_i(t) = \frac{1}{n} \sum_{j=1}^{n} o_{ij}(t). \quad (2)$$

The quality $q_i(t)$ is the average number of skeleton joints accurately tracked by the robot ν_i at t.

Definition 5. *The **joint observation quality** $Q(t)$ made by the team $\mathcal{V}$ at time t is defined as:*

$$Q(t) = \frac{1}{n} \sum_{j=1}^{n} g_j(O(t), \mathcal{V}). \quad (3)$$

$$\text{where } g_j(O(t), \mathcal{V}) = \begin{cases} 1 & \text{if } \exists i \in \mathcal{V} \text{ such that } o_{ij}(t) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

[1]Cooperative Multi-robot Observation of Multiple Moving Targets

To quantify the individual contribution of each robot to the joint observation, we introduce the notion of **marginal contribution**. This refers to the marginal contribution of a player to a coalition in the Shapley value [5].

Definition 6. *The **marginal contribution** w_i of a robot ν_i in the joint observation of the team, at time t, is defined as:*

$$w_i(t) = \frac{1}{n} \sum_{j=1}^{n} o_{ij}(t) \wedge (o_{ij}(t) \oplus g_j(O(t), \mathcal{V} \setminus i)) \quad (4)$$

with $\oplus$ the exclusive disjonction.

The marginal contribution corresponds to the part of the observation that robot ν_i is the only one to see.

The objective is to maximize the joint observation quality Q, i.e. the number of targets that are being observed by the team. Notice that maximizing the joint observation quality is not decomposable into maximizing the individual observation quality, which could lead to redundant information. For instance, consider the following example with $m = 3$ robots and $n = 7$ targets. Observations, qualities and marginal contributions of each robots at time t are:

$$o_1(t) = [0, 0, 0, 1, 1, 0, 0],\ q_1(t) = 0.29,\ w_1(t) = 0.29$$
$$o_2(t) = [1, 1, 0, 0, 0, 1, 1],\ q_2(t) = 0.57,\ w_2(t) = 0$$
$$o_3(t) = [1, 1, 1, 0, 0, 1, 1],\ q_3(t) = 0.71,\ w_3(t) = 0.14$$

Even if Robot 2 has a high individual observation quality, its contribution is low because it observes the same targets as another robot. However, Robot 1 has a low individual quality, but it is the only one to observe some targets, so its contribution is the highest. This illustrates that the objective of maximizing Q requires to find the most complementary information.

III. Hybrid metric-topological mapping

In this work we assume the environment is unknown. The robots only know the relative location of the scene. So they have to explore the environment around the scene to find the positions from which Q is maximized. The exploration goal is to collect data about obstacles that can interfere with robots' navigation ; and about the observation qualities from each point of view. The quality varies depending on various unknown and dynamic parameters: the current pose of the human body, the lighting changes, the occlusions[2] and obstacles that can partially or fully hide the scene.

We propose in this work that each robot carries out a hybrid metric-topological mapping to gather all these data. This mapping is based on two different local maps (cf. Fig. 2) and on a concentric navigation model.

A. Concentric navigation model

We propose to consider a limited navigation space around the scene. It is composed of C **concentric circles**, at spaced radius, and centered on the scene. Robots can move by

[2]Occlusions are not in the navigation space, i.e. they are in the space between the scene and the most inner circle of the navigation model.

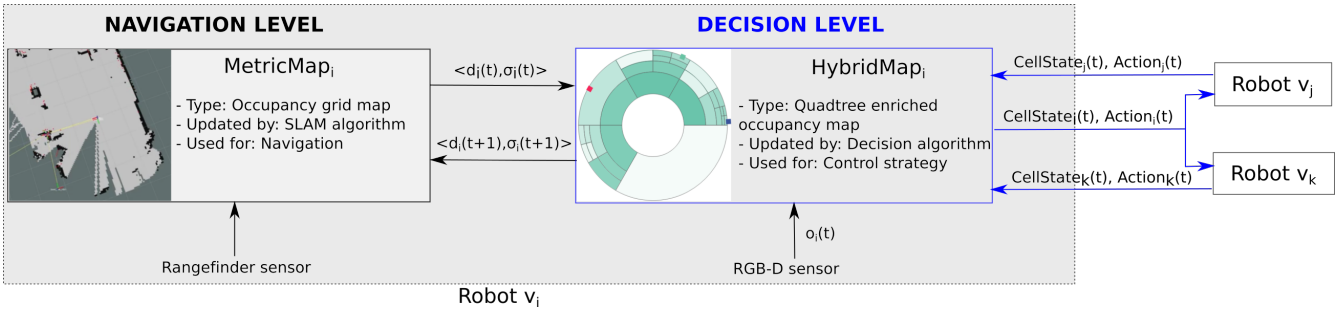


Fig. 2. Scheme of the different local maps constructed by one robot ν_i and of the data exchanged by the robots.

following these circles in two directions (forward/backward) as illustrated in Fig. 1. This navigation model allows to resolve several navigation constraints. First by fixing the camera orientation perpendicular to the movement direction of the robot, the scene is constantly in the robot's field of view. Secondly, moving along a circle trajectory is well adapted to the navigation of two-wheeled non-holonomic robots. Finally, it simplifies the coordination of the robots and reduces the risks of collisions. Indeed, no collision can arise between robots navigating on different circles, and collisions on a circle are easy to predict. Concerning obstacles avoidance, robots have only to detect obstacles that are on circle trajectories.

B. Local metric map

Occupancy grid maps, introduced by Elfes [6], represent the environment with a set of cells (usually squares or hexagons), each with an occupancy probability that determines the probability that the cell will be occupied by an obstacle.

In this paper, each robot builds a local occupancy grid map by using a SLAM[3] algorithm (cf. §V). The local metric map is used for robot navigation around the scene and obstacle avoidance. The position of a robot in its local map is defined as following.

Definition 7. *The **position** of a robot ν_i at time t is defined by $(d_i(t), \sigma_i(t))$ where d_i is the distance of the robot ν_i to the scene, and σ_i is the angle between a reference line and the line connecting the scene to the robot (cf. Fig. 1).*

We assume that each robot knows the position of the scene and of the reference line in its local map, creating a global reference frame for all the robots. Note this assumption could be partially released by using multi-robot SLAM techniques [7]. For instance in [8], we apply the merging of local occupancy grid maps [9] to compute a joint, global representation of the environment. **Then, robots only need to know the position of the scene in their local maps.**

C. Incremental cell mapping

Robots have to explore and gather data about the observation quality from each position around the scene. To handle the complexity of the state space to explore, we use **a discrete representation of the robots' positions**.

We define a set of D **sectors** that divide the circular space centered on the scene into slices with identical central angle of $\frac{2\pi}{D}$. Then we can determine a set of $D \times C$ contiguous cells where the robots are moving (cf. Fig. 1).

Definition 8. *At any position $(d_i(t), \sigma_i(t))$ of a robot ν_i at time t is associated a unique **cell** $c_i(t) = \langle [d_a, d_b]; [\sigma_a, \sigma_b] \rangle$ such that $d_i(t) \in [d_a, d_b[$ and $\sigma_i(t) \in [\sigma_a, \sigma_b[$. As well, at any cell $c = \langle [d_a, d_b]; [\sigma_a, \sigma_b] \rangle$ is associated a unique position $(d_a, \frac{\sigma_a + \sigma_b}{2})$.*

To gather data about the observation from each point of view, we define mean observation and quality for a cell.

Definition 9. *The **mean observation** $\phi_c(t) = [\phi_{c1}(t), ..., \phi_{cn}(t)]$ from a cell c at time t is defined as a vector of size n such that:*

$$\phi_{cj}(t) = \frac{1}{|\Delta(t)|} \sum_{\tau \in \Delta(t)} o_{N(\tau,c),j}(\tau) \tag{5}$$

where $\Delta(t) = \{\delta_x | \delta_x \leq t\}_{x=1,...,X}$ is the set of time steps where the cell c was visited by a robot. At time δ_x, the cell c was visited by the robot $N(\delta_x, c) \in \mathcal{V}$.

Definition 10. *The **quality of a cell** $\rho_c(t)$, associated to the cell c at time t, is defined as:*

$$\rho_c(t) = \frac{1}{n} \sum_{j=1}^{n} \phi_{cj}(t). \tag{6}$$

The quality of a cell is the mean observation quality made by all the robots from the position associated to that cell.

The state space to explore is then directly affected by the number of circles and sectors. This set is bounded[4] by $(C \times D)^m$ if robots are heterogeneous. If robots are identical[5], the size of the state space goes down to $\binom{C \times D}{m}$. Given the obstacles, it does not reduce to $\binom{D}{m}$ as it is not sufficient for the robots to explore the cells situated the closest to the scene as they are not guaranteed to be reachable.

[3]Simultaneous Localization And Mapping.

[4]It can be reduced if some cells are inaccessible because of obstacles and if only one robot per cell is allowed.

[5]Two robots can be arbitrarily swapped without changing the joint observation of the scene.

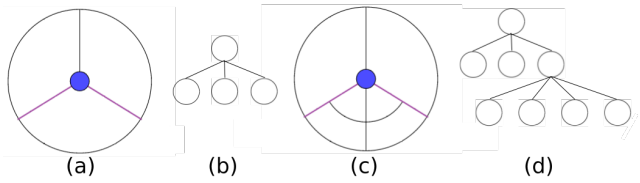


Fig. 3. Example of (a) a representation with three initial cells (b) corresponding to the initial tree structure where each of the three initial cells is a root of a quadtree. (c) One cell was split. (d) The node corresponding to the split cell has four children nodes in the quadtree. The blue circle at the center represents the scene.

To handle the space complexity and the time to explore the environment, we propose an **incremental division** of cells. The idea is to have at the beginning a coarse representation of the environment with few initial cells. Thus each initial cell covers a large area of the space around the scene, as illustrated on fig. 3(a) where three initial cells are defined. A cell can be split into sub-cells, and recursively, sub-cells can be split (cf. fig. 3). As each cell is associated with a robot position, this will increase the number of accessible positions. The objective is to **refine the discretization only in interesting areas of the environment**. The robots will split cells only where an accurate exploration could improve the joint observation quality. Conversely, some cells must not be split too finely, *e.g.* because they are behind obstacles. The observation from any positions in such a cell is occluded. Thus the number of cells to explore is reduced while refining the joint observation quality over time as the robots split interesting cells.

D. Local cooperative hybrid quadtree map

To manage this incremental mapping of the environment, we propose an hybrid metric-topological map based on a **quadtree**. A quadtree is a kind of tree in which each non-leaf node has four children [10]. In our structure, each initial cell is a root of a quadtree. Each node of the quadtree is associated to a cell and a position in the local metric map; and each cell of the navigation model is a leaf node of the quadtree. When a cell is split, its corresponding node gets four children nodes (cf. fig. 3).

The local hybrid map of each robot is **enriched** with various data (cf. Fig. 2):

- data coming from local sensors (e.g. rangefinder, depth camera);
- information from the local metric map of the robot;
- high level data sent by the other robots of the team.

These data are used to maintain for each leaf-node of the quadtree at each time t the following information:

- an obstacle occupancy probability coming from the local metric occupancy grid map;
- the mean observation $\phi_c(t)$ and quality $\rho_c(t)$ of the cell c corresponding to the node.

This information is updated following probabilistic quadtree principle [11]. When a cell is split, all the data of the children nodes are initialized with the data of their parent node. When a cell is visited, the corresponding leaf-node in its local quadtree map is updated.

The mean observation of a cell requires to know the observations made by the other robots. Thus the local hybrid map of a robot is **constructed in a cooperative manner** with high-level data sent by other robots. Communication architecture is detailed in the next section, as well as the distributed decision algorithm that uses the local hybrid map.

Algorithm 1: Choice of robot ν_i's action at time step t

1 $\forall k \in \mathcal{V}$ $w_k(t)$=ComputeContributions
2 **If** $w_i(t)$ is the best **then** Do not move
3 **Else** With probability ϵ Do an exploration action
4 With probability $1-\epsilon$ Do an exploitation action

IV. Multi-robot Decision

In [4], a distributed heuristic approach is proposed to solve the CMOMMT problem using weighted local force vector control. However this approach assumes uncluttered environments with either no or simple obstacles. Moreover it mainly focuses on maintaining a tracking mode on the moving targets under observation, and the search mode is restricted to the regions that may contain more targets [3]. So this approach is not adapted to our context where the objective is to quickly explore and find the position maximizing the number of targets observed by the team.

In a previous work [12], we have proposed and compared different heuristics to guide the exploration of the state space in an incremental quadtree map. This approach was based on two heuristics to decide the action of each robot, as detailed in Algorithm 1. First, only the robots with the lowest marginal contributions were moving at each time steps. Second, the action to be done by these robots was chosen between exploration and exploitation action according to meta-heuristics as Simulated Annealing [13]. **Exploration** consists in moving to the less visited adjacent cell to gain new information and to prevent the team from remaining in a local optimum; while **exploitation** consists in moving to the best adjacent cell or to split the current cell if it is the best one. The best adjacent cell is the cell that maximizes the joint quality given the current observations of the other robots. Mean observations of cells are used to infer the observations from the adjacent cells. Thus the cells that are potentially interesting for the observation are divided and explored more accurately. If a move is not possible[6], the current cell is split.

We demonstrated in [12] the efficiency of this approach, in particular its ability to quickly produce efficient or optimal solutions (compared to the number of possible joint positions), its anytime property, and its robustness to noisy observations. However, this was only tested in a simplified simulator where communication and robots' navigation were perfect and immediate, and without using metric information. The control strategy was centralized with a

[6]A move is not possible if the target cell is an obstacle or is already occupied by another robot.

Algorithm 2: Decision at time step t for one robot ν_i

1 Wait $\texttt{Action}_j(t)$ $\forall j < i$
2 $\forall j < i$ UpdateQuatree($\texttt{Action}_j(t)$)
3 $\texttt{Action}_i(t) =$ Choose my action (heuristics)
4 UpdateQuatree($\texttt{Action}_i(t)$)
5 Communicate $\texttt{Action}_i(t)$ to all $j \neq i$
6 Do $\texttt{Action}_i(t)$
7 Recover the observation $o_i(t)$ in the reached cell $c_i(t)$
8 UpdateQuatree($\texttt{CellState}_i(t)$)
9 Communicate $\texttt{CellState}_i(t)$ to all $j \neq i$
10 Wait $\texttt{Action}_j(t)$ $\forall j > i$
11 $\forall j > i$ UpdateQuatree($\texttt{Action}_j(t)$)
12 Wait $\texttt{CellState}_j(t)$ $\forall j \neq i$
13 $\forall j \neq i$ UpdateQuatree($\texttt{CellState}_j(t)$)

unique topological quadtree map shared by all the robots.

In this article, we introduce an **on-line distributed architecture of this approach** relying on asynchronous communication and topological quadtree maps cooperatively constructed by each robot. To maintain the consistency of local hybrid maps of each robot and of the heuristic search, the distributed implementation used a predefined order between the robots of the team. In addition, two types of messages are exchanged (cf. Fig2). At each time step t, messages sent by a robot ν_i to all the other robots $\nu_{j,j\neq i}$ are:

- its current cell and the associated observation: $\texttt{CellState}_i(t) =< i, t, o_i(t), c_i(t) >$
- its action : $\texttt{Action}_i(t) =< act, i, t, c_i(t) >$ where $act \in \{NONE, GOTO, SPLIT\}$ and $c_i(t)$ is the goal cell.

Algorithm 2 details one decision step for one robot. To avoid that two robots choose the same goal cell, each robot waits to receive the `Action` message of its preceding robots (line 1). Then it decides its action according to a meta-heuristic (line 3, cf. Algo. 1). Once it has chosen its action, a robot sends an `Action` message to the others (line 5) and waits until the end of this action to recover the observation from its reached cell (line 7). Then, the leaf-node corresponding to the reached cell is updated (line 8) and the robot sends a `CellState` message (line 9). Finally, each robot waits the `Action` message of its following robots (line 10) and the `CellState` message of all other robots to update its local hybrid map.

This ensures the consistency between the local hybrid maps and the distributed decision. In addition, it also **reduces the communication payload** as messages are exchanged between robots only at the decision level (cf. Fig. 2). Messages contain high-level and already processed data rather than heavy data coming directly from the sensors.

V. Experimental Results

In this section, we present results obtained with a team of real robots observing the activity of a person in a cluttered environment.



Fig. 4. On the left, one experiment with three robots around a human scene (*reading phone*) and a cluttered environment. On the right, Kinect views of one robot enriched with the skeleton data. Joints are represented with circles where filled ones are observed targets (confidence value of 1).

A. Experimental settings

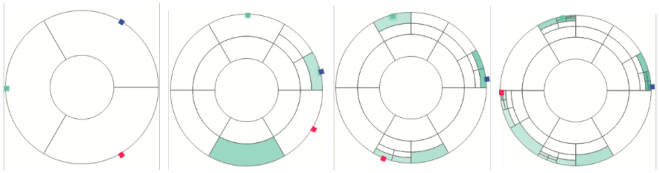
We used 3 Turtlebot2 robots equipped with a RGB-D camera (Kinect1) for the skeleton tracking, a low-cost 360° and 4 meters laser rangefinder (RP-Lidar) for local metric mapping and navigation, a netbook with Ubuntu and ROS connected to the mobile base and to the rangefinder for the decision and navigation part, and an Intel NUC mini-PC with Windows connected to the Kinect for the human observation part. ROS *gmapping*-package is used as a common particle filter SLAM algorithm. The decision algorithm presented in §IV was implemented as a ROS node. It requests skeleton data to a server running on the NUC. Data exchanged between robots use TCP/IP socket between netbooks.

Skeleton data are obtained using Microsoft Kinect SDK[7] that can track a set of $n = 20$ skeleton joints as targets. Data for each joint are the positions and orientations in the referenced frame of the camera, and a **confidence value**. The confidence value is 1 when the tracking of the joint seems to work, 0 if the tracking fails, and 0.5 if skeleton heuristics are able to adjust data for some lost or occluded joints. However, these are often far from the reality and can have negative side-effects for human activity recognition systems. So we do not use skeleton heuristics. Thus the observation value o_{ij} of a skeleton joint κ_j for a robot ν_i is the binary confidence value of the joint.

We perform a set of experiments in various cluttered environments and different human scenes, as illustrated in Fig. 4. Environments were designed so that it was impossible for one robot to find a cell from which it can see the full joint skeleton. We used Simulated Annealing [13] as a meta-heuristic with an exploration probability of ϵ. ϵ is initialized to 0.6 and reduced at each decision step according to $\epsilon = \epsilon(1 - 0.1)$. At each decision step, the two robots with the lowest marginal contributions are moving. A comparison in simulation of the performance of different heuristics and number of moving robots at each step can be found in [12].

For these experiments, the quadtree map was started with 3 initial zones (cf. Fig. 5a) and all the 3 robots on the same circle. Robots moved only along one circle. Even if a cell was split into 4 children nodes, each robot considered only the two adjacent cells on the same circle. A video

[7] `https://msdn.microsoft.com/en-us/library/jj131025.aspx`



(a) (b) (c) (d)

Fig. 5. Quadtree map cooperatively computed by $m = 3$ robots at different time steps of one experiment. Robots are represented by colored squares. Qualities of cells are represented by different shades of green: the greener the cell is, the better the quality is.

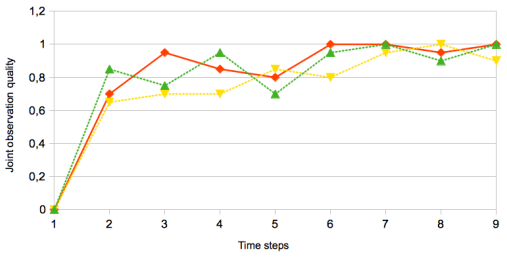


Fig. 6. Joint observation quality at each time step for 3 different experiments, with $m = 3$ robots using Simulated Annealing.

presenting different experiments can be found at `http://liris.cnrs.fr/lmatigno/videoCROME4.html`.

B. Results

Fig. 5 presents the quadtree map constructed by one robot at different decision steps during one experience. Fig. 5d is the quadtree map obtained when the maximal joint quality is found for the first time by the team. About half of the visited cells have a very low quality (white color) which shows the cluttered aspect of the environment with fully or partially occluded views of the scene. No visited cells has the best quality (dark green) so the robots have to coordinate to obtain the most complementary observations. This figure also illustrates how the robots are refining the discretization only in interesting areas around the scene.

Fig. 6 plots the current joint observation quality at each time step for different experiments. It shows that the optimal observation ($Q = 1$) was quickly found (8 time steps in the worst-case). The algorithm is also able to maintain a stable view of the scene as the joint quality still ranges between 0.8 and 1 after 6 steps.

VI. Conclusion

In this article, we presented an original mapping of the environment to deal with multi-robot observation of a human scene. Each robot carries out an hybrid metric-topological mapping to gather information about obstacles, occlusions and observation qualities of the scene. The map is updated cooperatively by exchanging only high-level data, thereby reducing the communication payload. The mapping is also realized in an incremental way to explore promising areas of the environment while keeping state-space complexity reasonable. We proposed an on-line distributed heuristic search combined to this hybrid mapping. Experiments with a fleet of robots showed the ability of the approach to quickly explore and find the team position maximizing the joint observation quality, of a person who is standing in a cluttered environment.

These results reveal many perspectives. First, we would like to test this approach when the person is doing a sequence of activities. In this case, the team will have to adapt its position at each new activity to obtain a complete view of the targets. To realize this fast adaptation, keeping some map information about obstacles and occluded cells could be helpful. We also intend to test other distributed optimization methods that could be used with our hybrid mapping, as distributed stochastic optimization approaches [14]. The main interests of these approaches are their speed of convergence, adaptability and scalability, which is highly appropriate to our context of real robots and dynamic human scenes. Finally, our approach could be extended to soft confidence values for body joints. This would require to redefine observation and marginal contribution in a probabilistic framework, and allow our approach to be tested with multi-camera human activity recognition systems that output probability distribution over classes of activity.

Acknowledgment

The authors would like to thank C. Wolf and E. Lombardi from LIRIS Lab. for the computer vision platform.

References

[1] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, "Active cooperative perception in network robot systems using POMDPs," in *Proc. of IROS*, 2010, pp. 4800–4805.

[2] A. Giusti, J. Nagi, L. Gambardella, and G. A. Di Caro, "Cooperative sensing and recognition by a swarm of mobile robots," in *Proceedings of the 25th IROS*, 2012, pp. 551–558.

[3] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2016.

[4] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Auton. Robots*, vol. 3:12, pp. 231–255, 2002.

[5] S. Hart, "Shapley value," in *The New Palgrave Dictionary of Economics*, S. N. Durlauf and L. E. Blume, Eds., 2008.

[6] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.

[7] S. S. G., M. Trentini, M. L. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *J. Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[8] S. Bultmann, L. Matignon, and O. Simonin, "Multi-robot navigation and cooperative mapping in a circular topology," INRIA Grenoble - Rhone-Alpes - CHROMA Team, Tech. Rep., 2016.

[9] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE Special issue on Multi-Robot Systems*, vol. 94, no. 7, pp. 1384–1397, 2006.

[10] R. A. Finkel, J. L. Bentley, R. A. Finkel, and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, Mar. 1974.

[11] G. K. Kraetzschmar, G. P. Gassull, K. Uhl, G. Pags, and G. K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Eds., Proceedings of the 5th IFAC/EURON*, 2004.

[12] J. Cohen, L. Matignon, and O. Simonin, "Incremental and adaptive multi-robot mapping for human scene observation," in *Int. Conf. on Tools with Artificial Intelligence (ICTAI)*, 2016, pp. 678–685.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[14] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos, "Multi-robot three-dimensional coverage of unknown areas," *I. J. Robotics Res.*, vol. 31, no. 6, pp. 738–752, 2012.

TIGRE: Topological Graph based Robotic Exploration

L. Fermin-Leon, J. Neira and J. A. Castellanos

***Abstract*— In this work we address the problem of autonomous robotic exploration and map building supported by high level information provided by online topological segmentation that incrementally generates an undirected connected graph of the environment. We formulate the exploration problem as the traversal of all the edges of the online graph, being the optimal solution an Eulerian path, if it exist. We propose an integrated approach, the TIGRE algorithm, that combines Graph-SLAM procedures with online Constrained Depth-First Search based graph traversal algorithms to efficiently explore the navigation area. Simulation results show that the online TIGRE algorithm provides similar error estimates for the robot poses and distance travelled than well-known off-line algorithms that use the full graph of the environment. Also, it outperforms other online and off-line algorithms in terms of estimated error.**

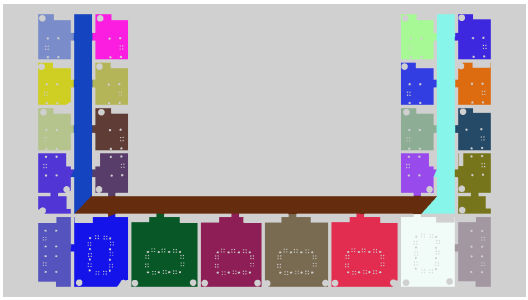


Fig. 1. Example of a topological segmentation [7] of an office-like environment [2] in approximate convex regions. The approach is robust to the presence of small obstacles, like furniture, and its low processing time allows to be used in exploration tasks.

I. Introduction

Autonomous robotic exploration for map building is the problem of finding the commands to be applied to the robot in order to efficiently (ideally, optimally) build a map of the environment. The map's posterior use determines its design and construction requirements, e.g. in maps aimed at the reconstruction of small areas, the goal would be attaining the highest precision in the reconstruction, or in maps aimed at subsequent path planning tasks the goal would be that every traversable path in the real world be represented in the resulting map.

The optimal set of commands describe a path that should optimize two conflicting criteria: (i) minimize robot and map uncertainty, and (ii) minimize distance traveled. In the context of feature-based SLAM, the map is represented as the position of a set of landmarks, and uncertainty is represented by the covariance of the landmark location estimates, which is known to decrease as successive observations or loop-closures are made [5], with the penalty of an increase in the distance traveled by the vehicle.

Typical approaches to autonomous exploration usually consist in determining some "interesting point" and then moving towards it. This point can be determined according to different criteria being the most popular the Frontier-based approach by Yamauchi [20], a greedy approach in which the robot always moves towards the closest unexplored region. An alternative group of works rely on decision-theoretic approaches to define the exploration policy [4], [3].

*This work has been supported by the MINECO-FEDER project DPI2015-68905-P, Grupo DGA T04-FSE, and the research grant Fundación Carolina

Leonardo Fermín-León, José Neira and José A. Castellanos are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain lfermin@unizar.es, jneira@unizar.es, jacaste@unizar.es

In this work we focus on the exploration task supported by high level information provided by a online topological segmentation of the navigation environment (figure 1). Thus, we model the environment as an undirected connected graph and we consider the real robot exploration problem of the graph of the environment, initially unknown, that has to be incrementally constructed and traversed by adequate decision-making strategies. We formulate the exploration problem as the traversal of all the edges of the online graph, being the optimal solution an Eulerian path, if it exist.

We propose the TIGRE algorithm, an online Topological Graph-based Robotic Exploration strategy, that, starting from an initially unknown environment: (i) it integrates a graph-SLAM front-end and back-end to reconstruct the navigation environment from the observations provided by a 2D laser scanner and odometry readings; (ii) it provides accurate loop-closures, up to reasonable performance bounds, to recognize previously visited areas; (iii) it constructs an undirected connected graph of the environment by means of an incremental contour-based topological segmentation algorithm; and (iv) it determines the graph exploration policy by using a constrained depth-first search based graph traversal algorithm.

The rest of the paper is structured as follows. Section II reviews the related work to our approach. Section III justifies the goal of casting the exploration problem as the traversal of all the edges of the graph. Section IV recalls the classical Tarry's algorithm, a depth-first search based graph traversal algorithm adopted from maze-searching problems. Section V reports the TIGRE algorithm and finally, section VI reports the simulation results where the performance of the TIGRE algorithm is compared to other full-graph and online-graph approaches.

II. Related Work

The problem of autonomous vehicle exploration of an unknown environment modeled as a graph can be traced back to the early works of Dudek et al. [6], where portable markers were used to recognize previously visited nodes of the graph, a method later improved by Rekleitis et al. [17]. Practical graph-based exploration approaches have been reported in the literature more recently. In [19] they use a hybrid (topological - metric) representation of the environment based on corners and openings for indoor environments. Exploration consists simply in traversing the hallway and then backtracking to every single room. An interesting work on practical implementation is reported in [18], where a robotic platform is used to explore abandoned mines and they are able to detect automatically the nodes and match them with previously visited nodes to construct the topological map. Also, in [1] they report an approach where both exploration and map building can be done with topological information without requiring additional representations is proposed, however, they rely on external place recognition capabilities.

In the graph theory community the exploration of unknown graphs has even a larger history, dating back to the times of Euler [9] who stated the well-known Königsberg bridges problem. Assuming that the full-graph of the environment is known a priori, a two-fold interpretation of the exploration problem as graph traversal is possible. First, a Travelling Salesman Problem (TSP) perspective could be adopted, where the exploration problem consists in searching for the shortest path, starting at the initial node, that travels through all the nodes of the graph, a well-known NP-complete problem. Recently, an efficient exploration strategy [15] has been proposed that considers the off-line solution of the TSP problem for a rough a priori known graph of the navigation area.

Second, a Chinese Postman Problem (CPP) perspective could be adopted, where the exploration problem consists in searching for the shortest path, starting at the initial node, that travels through all the edges of the graph, a polynomial complexity problem. In this latter case, if an *Eulerian path* exist, the lowest bound on the complexity of the algorithm is achieved, at the number of edges of the graph. From the point of view of the theory of experimental optimal designs [16] it follows that the Eulerian path could be referred to as *optimal* because it maximizes the information of any path in the graph with the same number of traversals and, therefore, it maximizes any of the optimality criteria defined from its information matrix.

In a real robot exploration problem the graph of the environment is initially unknown and, therefore, an online incremental graph construction and graph traversal strategies are required.

Classical work on topological segmentation of indoor environment have been recently surveyed [2], where different approaches are compared, namely, Voronoi Diagrams, Morphological Segmentation, Distance Transform Segmentation and Feature based segmentation. In all of these approaches the map of the navigation area is represented by an occupancy grid, i.e. a 2D array of pixels. A different approach, proposed in the computer vision community [12], is refered to as contour-based segmentation where the map is represented as a set of closed contours. The decomposition consists in segmenting these contours into sub-contours done according to a convexity criterium. Among the advantages of the latter approach is that complexity does not depend on the image size but in the characteristics of the contour. More importantly, it provides an insight of the characteristics on the shape. Recently, an incremental version of the dual-space decomposition [7] has been reported to provide an online topological description of the environment of a mobile robot.

Casting the exploration problem as the traversal of all the edges of the online graph requires the availability of incremental algorithms aim at approximating the solution of the CPP, being the optimal solution an Eulerian path, if it exist. Early work on motion planning [14] pointed out the relationships between maze-searching algorithms and robot motion planning with incomplete information driving the attention of the community towards classical algorithms (e.g. Trèmaux, Tarry and Fraenkel algorithms) that could be suitably adapted to motion planning problems as the exploration task considered in this work. In particular, Tarry's algorithm [9], an efficient algorithm to explore mazes, could be easily adopted within an exploration task because, from the knowledge of a partially known graph, it guarantees that every edge of the still unknown graph will be traversed exactly twice, once in every direction, with the exception of the edges incident to the start and target nodes, which will be traversed once each.

III. Optimality of the Eulerian Path

Following [6] we represent the navigation environment of our autonomous vehicle by means of an undirected connected graph $G = (V, E)$, where V is the set of numbered nodes and E is the set numbered edges, with $|V|$ and $|E|$ the number of nodes and edges respectively. Thus, starting from an initially unknown graph, an exploration algorithm will sequentially traverse the edges of the graph until a stopping criterium is satisfied.

From the perspective of the theory of optimal design of experiments [16], different optimality criteria (e.g. A-Opt, D-Opt, E-opt) could be computed as different instances of the family of *information functions* of the information matrix $\mathbf{Y} \in \mathbb{R}^{|V|\times|V|}$ (i.e. the inverse of the covariance matrix) of the graph G. Mathematically any optimality criteria could be expressed as $\|\cdot\| : \mathbf{Y} \rightarrow \mathbb{R}$.

Let $\mathbf{Y}$ and $\mathbf{Z}$ be two information matrices, then the concavity property, $\|\mathbf{Y} + \mathbf{Z}\| \geq \|\mathbf{Y}\| + \|\mathbf{Z}\|$, of the information function guaranties that traversing a new edge during the exploration of the graph monotonically increases the available information, and thus, as intuition suggests, an increasing number of traversals results in more informed maps on the navigation area. Once every edge of the graph

is traversed the potential information gain due to the re-traversal of an edge will only add redundant information with minor contributions to the information matrix of the graph as the number of re-traversals increases.

From graph theory [9] it follows that the shortest path, if it exists, that traverses every edge of the graph G once, is called an *Eulerian path* and it could be referred to as *optimal* because it maximizes the information of any path in the graph with $|E|$ edges and, therefore, it maximizes any of the optimality criteria defined from its information matrix.

Therefore, our TIGRE algorithm aims at traversing all the edges of the graph G at least once, as a potential Eulerian path would achieve, minimizing the number of re-traversals of previously known edges with a worst-case of traversing twice each edge of the graph due to its depth-first search nature.

IV. Tarry's Maze-Searching Algorithm

Early work on motion planning [14] pointed out the relationships between maze-searching algorithms and robot motion planning with incomplete information driving the attention of the community towards classical algorithms (e.g. Trèmaux, Tarry and Fraenkel algorithms) that could be suitably adapted to motion planning problems as the exploration task considered in this work.

In particular, Tarry's algorithm [9], an efficient algorithm to explore mazes, could be easily adopted within an exploration task because, from the knowledge of a partially known graph, it guarantees that every edge of the still unknown graph will be traversed exactly twice, once in every direction, with the exception of the edges incident to the start and target nodes, which will be traversed once each.

The assumptions of the Tarry's algorithm are:

1) *Node Recognition*, it can be recognized when a node has been previously visited.
2) *Edges Traversed Recognition*, upon arrival to a node, the edges that have been previously traversed outwards the node are known.
3) *Entrance Edge Recognition*, the entrance edge, i.e. the edge we traversed the first time we arrived to the node, is known for every visited node.

Under these assumptions, the algorithm can be easily described as: *Arriving at any node continue via any edge which has not yet been traversed outward, but choose the entrance edge only as a last resort*.

It is worth noting that, adding an extra condition as *Every time you arrive to a previously visited node by a new path, return by the path you came*, then the Trèmaux method is obtained, which is the basis of the Depth First Search (DFS) algorithm for graph traversal.

V. Topological Graph-based Robotic Exploration

In this section we describe the TIGRE algorithm, a topological graph-based robotic exploration algorithm rooted on the previously mentioned Tarry's maze-searching algorithm and that allows a robot to autonomously explore its navigation environment modeled as an undirected connected graph. Starting from an empty graph the algorithm evolves until a stopping criterium is satisfied.

At each time step an incremental contour-based topological segmentation algorithm [7] extracts the graph-based representation from the output of a graph-SLAM algorithm [11]. Every region in the topological segmented map is represented by one node in the graph and one additional node is used for the unexplored area. The graph's edges represent every pair of regions connected by a physically traversable path, including regions driving towards unexplored area. Physical traversability of all the edges of the topological graph increases the reliability of the obtained model for subsequent robotic tasks.

A. Tarry's Assumptions in Graph-SLAM

The graph SLAM algorithm for the map reconstruction satisfies, within performance bounds, the assumptions of the Tarry's algorithm described in the previous section:

1) *Node Recognition*, recognizing a previously visited node relates to the loop-closing capabilities of the algorithm. In our work we enforce loop-closing behaviour by navigating the vehicle in the vicinity of previously stored robot's poses within the navigation region.
2) *Edges Traversed Recognition*, the edges between nodes of the topological graph are defined by the traversed edges, and its direction vector, of the graph-SLAM algorithm
3) *Entrance Edge Recognition*, because the time history of the edge traversals is stored, the entrance edge is just the first, in chronological order, edge traversed into the region.

B. On-line Exploration Terminating Conditions

Fraenkel's constraints [14] reduce the exploration path length introducing a counter associated with the number of nodes with unexplored edges, guarantying that every edge will be traversed at least once but never more than twice, once in every direction. In our case we keep track of the number of the remaining edges to be traversed, and once every edge is traversed the exploration is over. Consequently, our exploration path length lies within an analogous interval, i.e. between the minimum number traversals, achieved by the off-line CPP algorithm when the full graph is known, and twice the number of edges of the graph.

Additionally, we modify the Tarry's algorithm by a break-tie criteria when multiple edges could be chosen to be traversed at a given time step of the evolution of the algorithm. In the original Tarry's algorithm whenever a new node is reached and there are several edges that can be traversed, one edge is chosen randomly. We modify this criteria by choosing, in those situations, an edge leading to a previously visited node instead of an edge leading to the unexplored area, thus enforcing the loop-closing behaviour and increasing the precision of the underlying graph-SLAM solution.

C. The TIGRE Algorithm

Algorithm 1 describes the pseudo-code of the proposed topological graph-based exploration algorithm[1]. Initially, the algorithm builds on top of the ROS navigation stack for the functions `simulator` and `navigate`.

Then, the `Graph_SLAM` function is based on the implementation reported in [13] of g2o [11] as a back-end for the optimization, by using the odometry readings and the 2D laser scan to update the pose graph and the grid-map. The function `Topological_Segmentation` segments the input grid-map, associating a label to each segmented region by using the implementation reported in [7] of the contour-based segmentation algorithm [12].

Next, the function `Build_Topological_Graph` builds an annotated topological graph. The information of the nodes includes their order in the sequence. Using this information the function `Extract_Incident_Edges` finds the valid edges connected to the current region (edges not traversed outwards) and classify them into either $Frontier_Edges$ (leading to unexplored areas) or $Link_Edges$ (leading to previously visited areas).

Finally, from those subsets of the incident edges the next goal location for the vehicle is selected. The algorithm favors the selection of goals leading to previously visited areas to enforce the loop-closing behaviour in the case of ambiguity.

Algorithm 1: TIGRE Algorithm

```
Input     : Map, Topological_Map, Pose_Graph
Output    : Topo_Graph = (V, E)
Variables: Link_Edges = {}
           Frontier_Edges = {}
Exploration_Completed = FALSE
Commands = 0
while Exploration_Completed = FALSE do
    [Scan, Odometry] = simulator(Commands)
    [Pose_Graph, Map] = Graph_SLAM(Scan,
    Odometry)
    Regions_Set = Topological_Segmentation(Map)

    Topo_Graph = Build_Topological_Graph(Map,
     Regions_Set, Pose_Graph)
    [Frontier_Edges, Link_Edges] =
    Extract_Incident_Edges(Topo_Graph)
    if Link_Edges = {} & Frontier_Edges = {} then
        Exploration_Completed = TRUE
    else
        Remove_Entrance_Edge(Link_Edges)
        if Link_Edges = {} & Frontier_Edges = {} then
            Goal = Entrance_Edge(Topo_Graph)
        else
            if Link_Edges ≠ {} then
                Goal = Extract_Goal(Link_Edges)
            else
                Goal = Extract_Goal(Frontier_Edges)
    Commands = navigate(Goal)
```

VI. Experimental Results

In this section we report simulation results, in the Player/Stage simulation environment [8], to illustrate the behaviour of the topological graph-based robotic exploration algorithm proposed in previous sections when an autonomous vehicle is navigating within the 20m $\times$ 20m *Cave* environment [10]. A C++ implementation of the TIGRE algorithm has been programmed on top of both simulation and navigation functions of the ROS package.

Figure 2 shows the online reconstruction of the navigation environment both from the geometrical perspective of the grid-map provided by the graph-SLAM algorithm, and the online topological graph used for the robot exploration task derived for the incremental contour-based topological segmentation. The computation of segmented regions, and their representative nodes are sufficiently topologically stable for the execution of the TIGRE algorithm. In some cases, a region is over-segmented and 2-connected regions appear but the performance of the topological exploration algorithm is unaffected because when only one edge drives out of a node, no decision is required.

Different performance metrics, as the average of 10 replications of the experiment, are reported to compare the behaviour of the graph-SLAM algorithm when the topological graph-based model of the navigation is provided by: (i) an off-line full graph-based solution to the Travelling Salesman Problem (TSP); (ii) an off-line full graph-based solution to Chinese Postman Problem (CPP); (iii) an on-line greedy Frontier-based algorithm; and (iv) our on-line TIGRE algorithm.

Figure 3 plots the final grid-maps obtained by each algorithm with the overlaid exploration trajectory. Also the final topological segmentations and the topological graphs are shown. Additionally, figure 4 plots the evolution of the rate robot pose error over distance for the complete exploration path length. From the computation of the mean error of the poses of the vehicle along its trajectory (recall that ground-truth is available in the simulation tool) we conclude that the best performance corresponds to the CPP-algorithm, the worst performance corresponds to the TSP-algorithm and that the TIGRE algorithm outperforms both the TSP and Frontier-based algorithms. The results agree with the intuition that both TSP and Frontier-based search for the shortest exploration path faster at the expense of reducing the number of loop-closing during the graph-SLAM execution. Both CPP and TIGRE, visiting all the edges of the graph, result in larger path lengths but in improved estimation errors.

Finally, figure 5 describes the evolution of the area coverage (percentage of the explored cells of the grid-map) versus the path length. Clearly the TSP algorithm results,

[1]The source-code of the algorithm is available at `https://github.com/lfermin77/TIGRE`

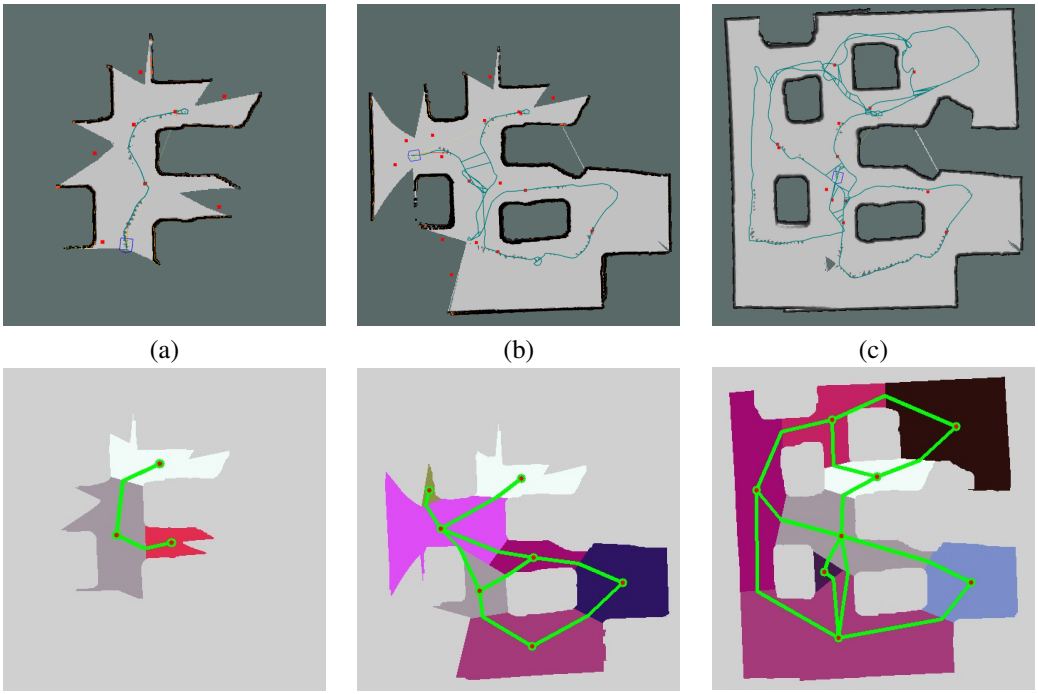
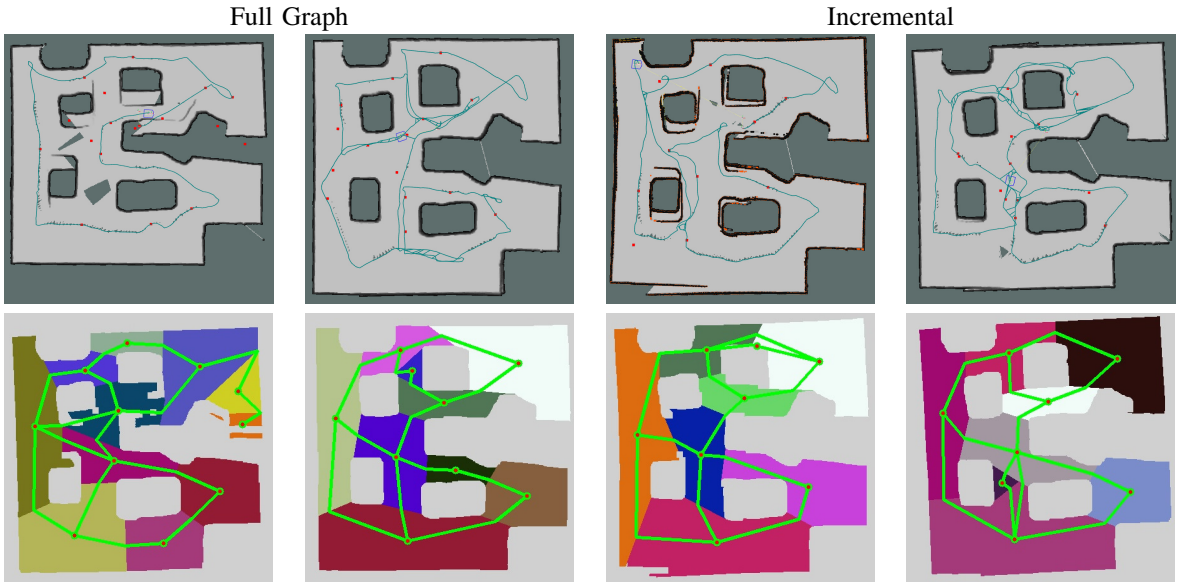


Fig. 2. Snapshots of the exploration task: (a) and (d) represent early stages of exploration task and its decomposition with the current topological graph (green); in (b) the robot chooses to traverse the edge presented in (e) first, rather than exploring the nearby frontier. The exploration is considered completed in (c) and (f).



(a) TSP $\bar{\epsilon} \simeq 0.72 \pm 0.46m$

(b) CPP $\bar{\epsilon} \simeq 0.21 \pm 0.07m$

(c) Frontier-based $\bar{\epsilon} \simeq 0.61 \pm 0.37m$

(d) TIGRE $\bar{\epsilon} \simeq 0.24 \pm 0.03m$

Fig. 3. Comparison of the performance of different off-line, that use the full-graph, and on-line, that incrementally build the graph, algorithms. The average estimated error $\bar{\epsilon}$ of the robot poses is shown together with its standard deviation computed from the ten replications of the experiment.

by definition, in the shortest exploration path length due to its inherent feature of driving the vehicle always to unexplored terrain visiting all the nodes of the graph but only a subset of the edges. Similarly, the Frontier-based, with its greedy approach towards unexplored terrain reports short exploration path length in this case-study. In both cases, the number of edge re-traversals (and therefore loop closures) is very low, thus a similar performance, in terms of estimation error is obtained for both of them. On the contrary, the behavior of the TIGRE and the CPP bear similarities because

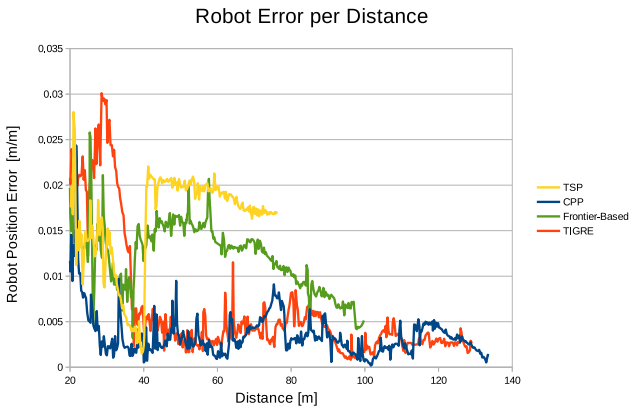


Fig. 4. Evolution of the estimated error of the robot pose divided by the distance traveled versus the distance traveled. Relevant behaviour appears as the traveled distance increases.

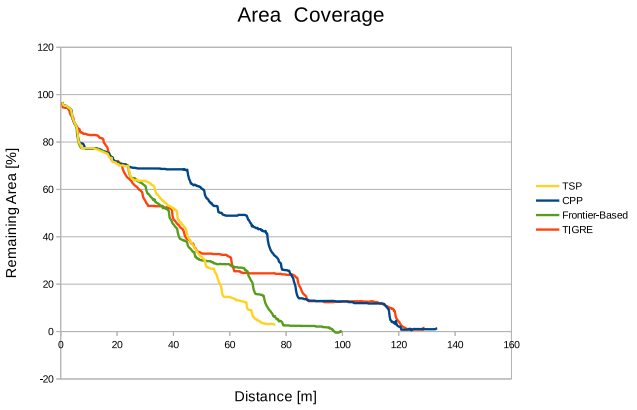


Fig. 5. Map area coverage versus the distance traveled by the robot.

they force the traversal of every edge of the graph, at least once, as mentioned in the previous sections, and therefore the distance travelled by the vehicle is larger, with frequent re-traversals of the edges of the graph (see the plateaus in the figure), and therefore achieving a better estimation error.

VII. Conclusions

This paper focused on the problem of autonomous robotic exploration and map building supported by high level information provided by online topological segmentation that incrementally generates an undirected connected graph of the environment. The exploration problem is formulated as the traversal of all the edges of the online graph.

The online TIGRE algorithm is proposed that integrates Graph-SLAM features, contour-based topological segmentation, incremental graph construction and online decision-making by an adaptation of a constrained depth-first search based graph traversal algorithm. Simulation results suggested a close behaviour of the online TIGRE algorithm with off-line algorithms that require the knowledge of the full graph of the environment (CPP), and that it outperfoms, in terms of error estimation of the robot poses other online (Frontier-based) and off-line (TSP) algorithms.

Further work is aimed at thoroughly evaluating the TIGRE algorithm is more complex scenarios and its comparison against other exploration methods reported in the literature.

References

[1] B. C. Akdeniz and I. Bozma, H. Exploration and topological map building in unknown environments. In *IEEE/RSJ International Conference on Robotic and Automation*, pages 1079–1084, 2015.

[2] R. Bormann, F. Jordan, W. Li, J. Hampp, and M Hagele. Room segmentation: Survey, implementation, and analysis. In *IEEE International Conference on Robotics and Automation*, pages 1019–1026, 2016.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[4] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *IEEE International Conference on Robotics and Automation*, pages 487–494, 2015.

[5] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[6] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

[7] L. Fermin-Leon, J. Neira, and J. A. Castellanos. Incremental contour-based topological segmentation for robot exploration. In *IEEE/RSJ International Conference on Robotic and Automation*, pages 2554–2561, 2017.

[8] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, volume 1, pages 317–323, 2003.

[9] J. L. Gross and J. Yellen. *Graph Theory and Its Applications*. Chapman and Hall/CRC, 2006.

[10] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[12] Guilin L., Zhonghua X., and Jyh-Ming L. Dual-space decomposition of 2d complex shapes. In *27th IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, Jun. 2014.

[13] M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos, and G. Grisetti. Multi-robot SLAM using condensed measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1076, 2013.

[14] V. J. Lumelsky. A comparative study on the path length performance of maze-searching and robot motion planning algorithms. *IEEE Transactions on Robotics and Automation*, 7(1):57–66, 1991.

[15] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss. Speeding-up robot exploration by exploiting background information. *IEEE Robotics and Automation Letters*, 1(2):716–723, 2016.

[16] F. Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

[17] I. M. Rekleitis, V. Dujmovic, and G. Dudek. Efficient topological exploration. In *IEEE International Conference on Robotics and Automation*, pages 676–681, 1999.

[18] D. Silver, D. Ferguson, A. Morris, and S. Thayer. Topological exploration of subterranean environments. *Journal of Field Robotics*, 23(6-7):395–415, 2006.

[19] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3–14, 2003.

[20] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.

Collision Avoidance for Safe Structure Inspection with Multirotor UAV

F. Azevedo[1], A. Oliveira[1], A. Dias[1], J. Almeida[1], M. Moreira[1], T. Santos[1], A. Ferreira[1], A. Martins[1] and E. Silva[1]

***Abstract*— The multirotor UAVs are being integrated into a wide range of application scenarios due to maneuverability in 3D, versatility and reasonable payload of sensors. One of the application scenarios is the inspection of structures where the human intervention is difficult or unsafe and the UAV can provide an improvement of the collected data. At the same time introduce challenges due to low altitude missions and also the fact of being manually operated without line of sight. In order to overcome these issues, this paper presents a LiDAR-based real-time collision avoidance algorithm, denoted by Escape Elliptical Search Point with the ability to be integrated into autonomous and manned modes of operation. The algorithm was validated in a simulation environment developed in Gazebo and also in a mixed environment composed by a real robot in an outdoor scenario and simulated obstacle and LiDAR.**

I. INTRODUCTION

In recent years, there has been an increasing research effort with multirotor Unmanned Aerial Vehicle (UAV) in a wider range of scenarios, such as search rescue missions, surveillance and inspection tasks. One of the reasons is the fact that this type of vehicles provides the required maneuverability to navigate through complex three-dimensional scenarios with a reasonable payload of sensors. Considering the application scenarios of inspection, the multirotor UAV provides the ability to collect data from different positions, angles and distances, and at the same time reduce the cost and the human risk. Most of this operations are performed through an operator or more recently in fully autonomous missions. In both cases, and due to low altitude operation and the existence of structures like buildings, power lines or even natural obstacles like trees, the risk of crashing, damage structures and injury surrounding people has been increased.

Therefore, this paper proposes to address the research area of real-time obstacle avoidance for manned and autonomous multirotor UAVs. Based on the work developed by [1] for rotorcraft UAVs and the evaluation performed of the advantages of LiDAR solutions [2] for obstacle avoidance, we propose to extend the method with a reactive obstacle avoidance algorithm based on LiDAR and applied to multirotor, denoted by *Escape Elliptical Search Point - LiDAR-based Collision Avoidance (E^{2}SP-LCA)*.

The E^{2}SP-LCA algorithm bounds the map and searches for any point that lies inside a safety volume (obstacle). Due to the dynamics of the vehicle, the safety volume will be proportional to the vehicle speed, taking also into account the direction of the velocity and the position of the *waypoint*. When an obstacle is found, the algorithm evaluates a set of candidate points to avoid the collision (escape points). As the UAV is normally blind above and below its position, the algorithm proposes to avoid the obstacle performing a path as horizontal as possible. If no valid escape point is found, it will intentionally deviate from the original trajectory to try to overpass the obstacle.

The paper is outlined as follows: Section II presents the related work starting with the obstacle detection and map representation followed by real-time obstacle avoidance techniques. In Section III is detailed the E^{2}SP-LCA, followed by its implementation and the obtained results, in Section IV. In Section V are exposed some algorithm remarks and considerations, followed by Section VI that presents the conclusions and the proposed future work to improve this project.

II. RELATED WORK

This section presents the research works related to map representation and obstacle detection, as well as real-time obstacle avoidance algorithms.

A. Obstacle Detection and Map Representation

In the field of multirotor UAVs, the detection of an obstacle is mainly performed through monocular cameras[3], LiDAR[2][4], stereo cameras[2] or their combination[2], depending on the application scenario. Their advantages and drawbacks have been evaluated in [2].

Being active sensors, LiDARs are typically insensitive to the light of the environment, having more accuracy and better performance for far obstacles. The low processing power required makes them more efficient for real-time applications. However, the collected data is produced sequentially, the maximum range is limited and requires more electrical power. Approaches like [4] and [5] are some of the examples that use LiDAR-based detection systems.

On the other side, the stereo cameras provide a snapshot of the environment at one instant (global shutter cameras), producing a dense 3D range information, with color correspondence and capable of detecting objects from long distances (depending on the baseline between the cameras). However, this system highly depends on the visual environment conditions and requires a significant processing power. Besides that, its range accuracy decreases with range squared. In [6] is used a stereo vision system for obstacle detection.

[1]INESC Technology and Science, ISEP - School of Engineering, Porto, Portugal fabio.a.azevedo, alexandre.a.oliveira, andre.dias, jose.m.almeida, miguel.m.moreira, tiago.a.santos, andre.f.ferreira, alfredo.martins, eduardo.silva@inesctec.pt

In addition to the systems referred above, there are other solutions like [7] and [3] that uses a monocular technique to avoid collision with structures.

In the most basic way, the obstacles can be represented on a map by a simple point cloud with the measures given by a LiDAR or the features extracted from images. However, this is computationally costly and can compromise the real-time requirement. For reducing this cost, the data can be clustered, resulting in a sparse representation. Another disadvantage of this method is that is not possible to distinguish between free and unmapped spaces.

Using a point cloud as input, the memory space required for storing the map information can be reduced using techniques like the representation by means of *octrees*, *Octomaps* or *Voxel Grids* [8].

Other ways of representing occupancy maps are analyzed and summarized in [9].

B. Real-time Obstacle Avoidance Algorithms

In [4] is presented a solution with a helicopter to perform infrastructure inspection with collision avoidance, using a fixed 2D LiDAR and two flight modes. The *pirouette descent* mode creates a spinning LiDAR with a cylindrical field of view by rotating the helicopter around its yaw axis while descending vertically. The *waggle cruise* flight mode performs a horizontal sweep while flying forward, allowing to scan a corridor-shaped space. It provides two solutions to avoid obstacles and reach the goal, however, it does not have a global map, which implies some constraints for the avoidance maneuver to be completed successfully, as not having obstacles above the vehicle. Besides that, as it only uses a fixed 2D LiDAR, the quality of the generated map is strongly dependent on the quality of its position estimation.

Another obstacle avoidance maneuver is presented in [1] that considers the vehicle as a sphere and constructs a safety volume around it. Whenever an obstacle enters the safety volume, it constructs an ellipse around the obstacle and searches for a point that allows a free path from the current position to the *escape point* and that also ensures a collision-free path through a defined distance from the *escape point*, on the direction to the *waypoint*. If no clear path is found, it extends the ellipse radius (a certain number of times) and performs another search. If no free path is found with the maximum ellipse radius, the UAV will hover until a pilot takes control of it.

The *escape point* has the advantage of allowing an uninterrupted flight for avoiding the obstacle, otherwise, the vehicle would need to stop (hover) and recalculate the trajectory considering arbitrary avoidance points.

Although it is applied to aircraft, Sabatini et al. [10] have implemented an obstacle avoidance ellipsoid-shaped safety zone around obstacles. The planning algorithm for the obstacle avoidance takes into account the aircraft dynamics, velocity, acceleration and distance to the obstacle. In a case of high velocities and/or accelerations, the time to find an alternative path and the distance to the obstacle are the major inputs of the cost function, as they are the main parameters to be considered in critical situations (an aircraft cannot hover).

III. E^{2}SP-LCA - Escape Elliptical Search Point - LiDAR-based Collision Avoidance

The E^{2}SP-LCA can be resumed as an algorithm that follows the classical architecture of mobile robots navigation. Whenever it gets an update of the occupancy map, performs a search for potential obstacles between the UAV position and the *waypoint*, that can cause a damage on the vehicle or blocking it from reaching the desired position (algorithm 1), and tries to avoid them.

Figure 1 illustrates the algorithm behavior. On every map update, it starts to search for obstacles inside a safety zone that is created around the UAV, and propagated through a certain direction, by means of spheres of variable radius s_{rad} and centers distance d_{center}.

The direction of the safety zone is given by the weighted sum (parameters n and m) of the vector from the vehicle position to the *waypoint* ($\vec{u}$) and the UAV velocity (${}^W\mathrm{P}_{\vec{velocity}}$). For calculating the center's separation is used the equation 1, presented in [1], where V is the voxel size, however, the radius of the spheres is a defined parameter that is proportional to the vehicle's velocity norm. This approach ensures that some vehicle dynamics (at every moment) is taken into account in the search for potential obstacles, as the velocity affects the safety volume size and propagation direction.

Algorithm 1 ${}^W\mathrm{U}$ = E^{2}SP(${}^W\mathrm{O}_b$, ${}^W\mathrm{P}$, waypoint)

$obstacle \leftarrow false$
$s_{rad} \leftarrow k \cdot \left(\left\| {}^W\mathrm{P}_{\vec{velocity}} \right\| + 1) \right)$
$\vec{u} \leftarrow waypoint - {}^W\mathrm{P}_{position}$
$\vec{d} \leftarrow n \cdot \hat{u} + m \cdot {}^W\mathrm{P}_{velocity}$
$d_{center} \leftarrow 2\sqrt{s_{rad} \cdot V - \frac{V^2}{4}}$
for $i = 0 \rightarrow n_spheres$ **do**
  $center[i] = {}^W\mathrm{P} + i \cdot d_{center} \cdot \frac{\vec{d}}{\|d\|}$
end for
for each $cell \in {}^W\mathrm{O}_b$ **do**
  for each $center$ **do**
    $distance \leftarrow \|cell - center\|$
    if $distance < s_{rad}$ **then**
      $obstacle \leftarrow true$
      if $distance < closest_dist$ **then**
        ${}^W\mathrm{O}_c \leftarrow cell$
        $closest_dist \leftarrow distance$
      end if
    end if
  end for
end for
if $obstacle$ **then**
  ${}^W\mathrm{U} \leftarrow sch_esc({}^W\mathrm{O}_b, {}^W\mathrm{P}, waypoint, {}^W\mathrm{O}_c)$
end if
return ${}^W\mathrm{U}$

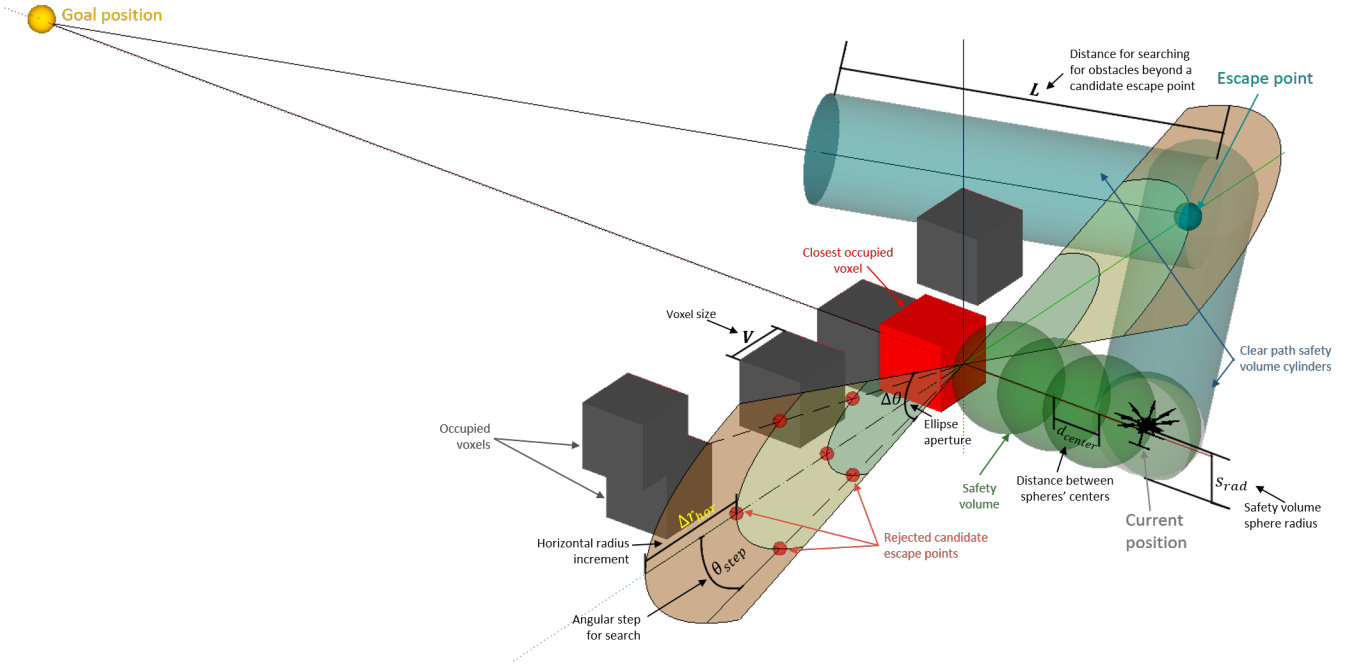


Fig. 1. E^{2}SP-LCA algorithm representation.

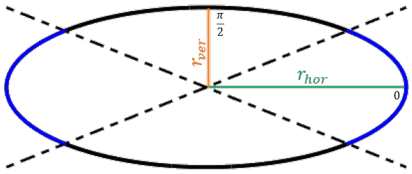


Fig. 2. Ellipse aperture. Valid zone in blue. Ellipse defined by a horizontal radius r_{hor} and a vertical radius r_{ver}.

$$d_{center} = 2\sqrt{s_{rad}V - \frac{V^2}{4}} \tag{1}$$

If any occupied cell is inside the spheres, the path is considered obstructed and is called the function to search for an alternative path (sch_esc). If more than one obstacle is found, for searching an escape will be only considered the closest obstacle ${}^{W}\mathrm{O}_c$, as it is the one with the greatest probability of causing an UAV crash.

For search for an alternative path, it is placed an ellipse centered on the closest obstacle, with a horizontal direction $\vec{e_{hor}}$ normal to the vector $\vec{w}$, defined by the vehicle position and the closest obstacle, and a vertical direction $\vec{e_{ver}}$ parallel to the vertical axes of the world frame ($\hat{k} = (0, 0, 1)$).

With an ellipse defined by equation 2, θ can be limited to a $\Delta\theta$ value. For example, if $\Delta\theta = \pi/2$, $\theta \in [-\pi; -3\pi/4] \cup [-\pi/4; \pi/4] \cup [3\pi/4; \pi[$, will result a valid search marked in blue on figure 2.

$$\begin{aligned} x &= r_{hor} \cdot cos(\theta) \\ y &= r_{ver} \cdot sin(\theta) \end{aligned} \tag{2}$$

Defining the valid angular aperture of the ellipse will constrain the avoidance path, not allowing the avoidance from above or below the obstacle. This angular aperture can be configured taking into account the application and the sensor in use. For example, if it is used a LiDAR sensor with a low vertical aperture, it is interesting to keep the vehicle as horizontal as possible.

For optimizing the avoidance path, the distance to travel should be as small as possible, so the direction of search (right to left or left to right) of a valid escape point will depend on the values of the distance from the left and right horizontal limit edges of the ellipse to the *waypoint* ($dist_l$ and $dist_r$, respectively). Those limit edges are obtained by setting θ to 0 (zero) or π on the ellipse equation 6 with the parameters represented on figure 2.

After been chosen the first escape point to be considered, the algorithm will search candidates on the edge of the ellipse (limited by an angular aperture $\Delta\theta$), by incrementing (or decrementing) an angular step θ_{step}. For each candidate escape point, the following conditions are evaluated:

- Clear path from current position to the candidate escape point;
- Clear path between the escape point and the waypoint along a predefined distance (L);

If both conditions are verified, the candidate escape point is considered valid, otherwise, the ellipse size is increased by Δr_hor and the procedure is repeated for the new ellipse.

A clear path is determined by verifying if there is any point/obstacle p_t that lies inside a cylinder between two points p_1 and p_2 with radius s_{rad}. For that, two vectors are generated, $\vec{d_{12}} = p_2 - p_1$ and $\vec{d_{1t}} = p_t - p_1$. Calculating the

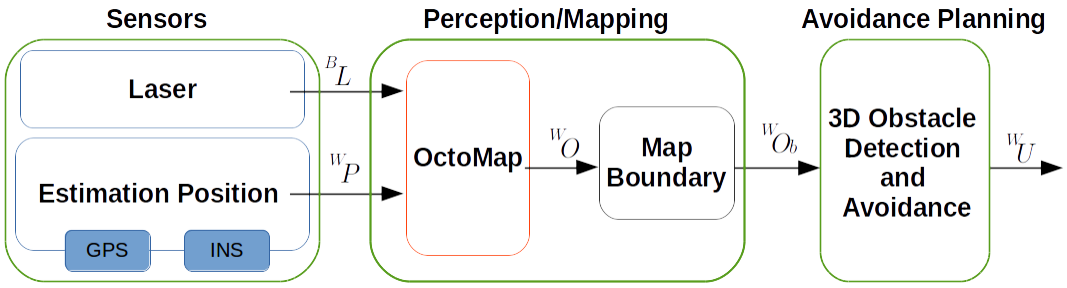


Fig. 3. Obstacle avoidance high level architecture.

dot product:

$$D = \vec{d_{12}} \cdot \vec{d_{1t}} \tag{3}$$

If $D < 0$ or $D > \left\|\vec{d_{12}}\right\|^2$, the point is outside the cylinder limits, otherwise, it has to be tested the closest distance from p_t to the line segment defined by p_1 and p_2.

Assuming α as the angular difference between vectors d_{12} and d_{1t}, and considering the fact that $\sin^2 + \cos^2 = 1$ and the dot product $D = \cos\alpha \cdot \left\|\vec{d_{12}}\right\| \cdot \left\|\vec{d_{1t}}\right\|$ which is equivalent to equation 3. Considering the distance from a point to the line segment, defined by $\sin\alpha \cdot \left\|\vec{d_{1t}}\right\|$ and the squared distance to the cylinder center given by $d_{cyl}^2 = (1 - \cos^2\alpha) \cdot \|d_{1t}\|^2$ we obtain

$$d_{cyl}^2 = \left(1 - (\vec{d_{1t}} \cdot \vec{d_{12}})^2 / \left(\left\|\vec{d_{1t}}\right\|^2 \cdot \left\|\vec{d_{12}}\right\|^2\right)\right) \cdot \left\|\vec{d_{1t}}\right\|^2 \tag{4}$$

by applying the dot product D. Therefore, considering equation 3, d_{cyl} can be rewrite as:

$$d_{cyl}^2 = \left\|\vec{d_{1t}}\right\|^2 - \left(\frac{D}{\left\|\vec{d_{12}}\right\|}\right)^2 \tag{5}$$

Having equation 5, if $d_{cyl}^2 > s_{rad}^2$, the path is clear, otherwise, there is an obstacle on the evaluated path and the candidate escape point is not valid.

Once a valid candidate escape point is found (using equation 6 with θ constrained), it is passed for the navigation through (${}^W\mathrm{U}$).

$$\begin{aligned} {}^W\mathrm{U} = {}&{}^W\mathrm{O}_c + r_{hor} \cdot \hat{e_{hor}} \cdot \cos\theta \\ &+ r_{ver} \cdot \hat{e_{ver}} \cdot \sin\theta \end{aligned} \tag{6}$$

If no valid escape point is found, after a predefined number of increases of the ellipse radius, the vehicle is commanded to move side-to-side through parameterized distance. If in that movement a clear path is found, the normal operation is returned, if not, the vehicle return to the point where the obstacle was detected, generates a warning message and waits for a manual control.

IV. Implementation and Results

In order to evaluate the E^{2}SP-LCA algorithm, we divided the validation into two phases: the first one was performed with the support of the simulation environment Gazebo[11][12] and the second one in a mixed environment composed by a real multirotor UAV in an outdoor scenario and a simulated obstacle and LiDAR sensor. Both approaches were implemented to validate the robustness of the E^{2}SP-LCA algorithm into different scenarios and in the particular case of the second test, also to ensure that the first tests of the algorithm were performed without risking a UAV with higher payload and more costly sensors like the LiDAR *Velodyne VLP-16*.

The algorithm was implemented in the framework ROS (Robotic Operating System)[13] in order to ensure a more straightforward integration between the simulation environment and the real multirotor UAV. The high level architecture is despited in figure 3 and is composed by three layers: **Sensors**, responsible for the sensor data acquisition, providing the LiDAR output in body frame reference ${}^B\mathrm{L}$ and an estimated vehicle pose, velocity and acceleration in global frame ${}^W\mathrm{P}$ through an Extended Kalman Filter (EKF) data fusion block of GPS and INS; **Perception/Mapping**, responsible for building a list of obstacles with the support of the Octomap toolbox that will generate unbounded voxels ${}^W\mathrm{O}$ with a predefine resolution of 0.4 meters. To improve the CPU performance and ensure real-time requirements we introduce a new feature to the Octomap to create a bounded voxels ${}^W\mathrm{O}_b$ that will be passed through a topic to the avoidance planning layer; **Avoidance Planning**, implements the E^{2}SP-LCA algorithm detail in section III based on the bounded voxels ${}^W\mathrm{O}_b$ and provides an output action denoted by ${}^W\mathrm{U}$ with a collision avoidance path planning escape point expressed in equation 6.

A. Simulation

The simulation environment chosen to benchmark E^{2}SP-LCA algorithm was Gazebo. Other simulators were considered, like MORSE[14] but the Gazebo was the one that provides a feasible integration with the autopilot PX4 project[15] through the *Software In The Loop* (SITL) and a simulated multirotor UAV model with a LiDAR payload sensor[16].

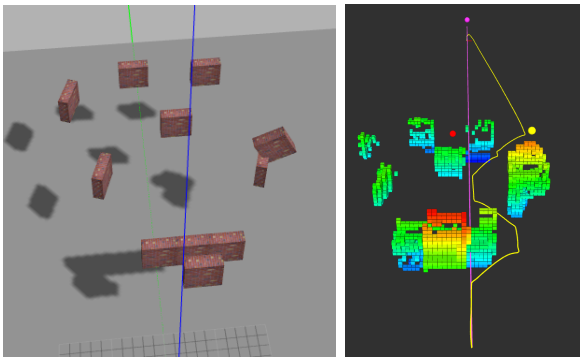


Fig. 4. Avoidance path into a complex scenario, with the purple line as ideal path, the yellow line as the UAV trajectory, red dot as the left extreme of the ellipse and the yellow dot as the chosen escape point

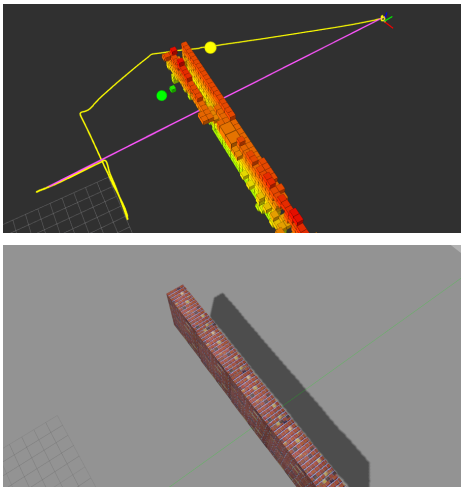


Fig. 5. Avoidance of a large obstacle, with the purple line as ideal path, the yellow line as the UAV trajectory, green dot as the right extreme of the ellipse and the yellow dot as the chosen escape point

The simulation environment is depicted in figure 4, is composed by several walls and a path defined by a purple line in the right figure. The obstacle avoidance trajectory is represented by the yellow line and is possible to observe that the UAV was able to overcome the obstacles and reach in a safe manner the desired position.

Figure 5 presents a situation where the UAV was not able to detect an escape point based on the predefined angular constrain ($\Delta\theta = \pi/2$) in order to avoid the UAV pass the obstacle from above (figure 2). This figure also represents a situation where the vehicle is capable of finding an escape after performing a movement parallel to the wall.

In order to evaluate the contribution of the bounded voxels ${}^{W}\mathrm{O}_b$ method against the unbounded voxel map, it was created a simulated environment, depicted in figure 6. The UAV navigate through it and the processing time for the obstacle search algorithm took an average of 7 ms with a standard deviation of 4.43 ms. For a fixed volume of 20 meters around the UAV position, the processing time decrease to an average of 1 ms with a standard deviation of 0.36 ms. Once the map representation is completed, the

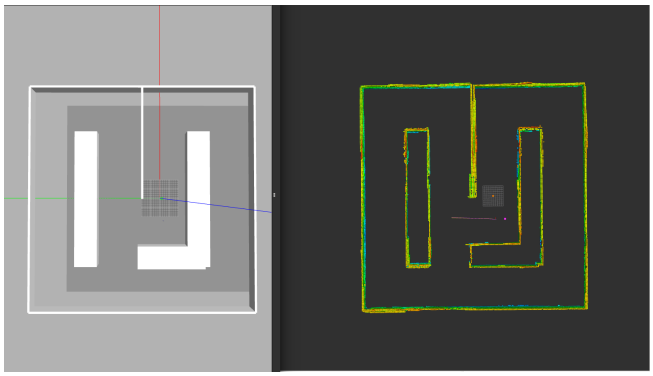


Fig. 6. Map for testing bounding method

unbounded method will stabilize in processing time while the bounded method keeps a low and constant time of processing during the UAV navigation. This allows us to conclude that this approach is more feasible in unstructured scenarios where the vehicles must navigate and keep the real-time constraints independent of the scenario.

With respect to the avoidance escape point, the required time processing in the simulation environment was 1.41 ms with a standard deviation of 0.97 ms for a scenario detailed in figure 4 composed by $\sim$ 760 voxels (bounded voxels). The simulations were performed with a CPU i7-740QM @ 1.73GHz, 8 GB of DDR3 RAM, NVIDIA GeForce GTX 460M, running the Ubuntu 14.04 LTS.

B. Field tests with a real UAV and a simulated sensor and obstacle

Based on the results obtained in the simulation environment, the second phase was the validation with a real UAV in an outdoor scenario (ISEP Campus) mixed with a simulated obstacle and payload LiDAR. The LiDAR used during field tests and the obstacle were the one that has been used for the simulation tests detail in section IV-A.

The implemented architecture is detailed in figure 7. The UAV is running internally the obstacle avoidance describe in figure 3 and receives remotely the simulated data from the LiDAR.

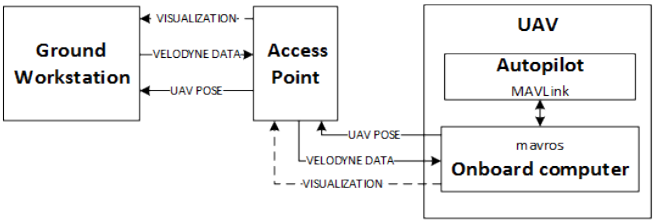


Fig. 7. Implemented architecture for real UAV in an outdoor scenario (ISEP Campus) mixed with a simulated obstacle and payload LiDAR.

The UAV is a customized hexacopter, depicted in figure 8, equipped with an open-source autopilot, Pixhawk board running PX4 Firmware and an embedded onboard computer, Odroid XU3, running Ubuntu 14.04 with Robot Operating System (ROS) Indigo.

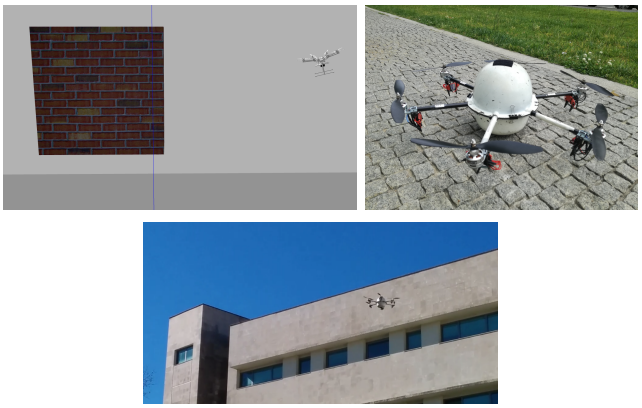


Fig. 8. Real UAV in an outdoor scenario (ISEP Campus) with virtual wall.

The outdoor field test was composed by a simulated obstacle (3x3x1 meters) and the real UAV perform a trajectory towards a position that requires an avoidance maneuver. The trajectory and the avoidance path is depicted in figure 8, with the yellow line being the UAV avoidance trajectory ${}^{W}\mathrm{U}$.

For the field test scenario, the embedded CPU average time processing for the obstacle search algorithm was 0.397 ms with a standard deviation of 0.1041 ms ($\sim$ 83 bounded voxels), with the avoidance escape point requiring 8.63 ms with a standard deviation of 0.136 ms

V. Remarks

The E^{2}SP-LCA is a collision avoidance algorithm that is capable of performing a safe inspection with low computational cost. This is obtained by considering as potential obstacles only the ones that lie inside a bounded volume, around the UAV position. The safety volume (volume inside which any occupied cell will be treated as an obstacle) has a dynamic behavior, once it is clearly dependent on the UAV's velocity, both in size (that depends on the velocity module) and direction of propagation, that depends both on the vector that connects the UAV current position and the desired one, and on the direction of its velocity vector). This dependence on the velocity vector can be tuned using the parameters n and m (algorithm 1), which makes E^{2}SP-LCA an algorithm that takes into account some vehicle dynamics and suitable for any multirotor UAV.

Another parameter that can be tuned is the aperture of the search ellipse, as well as the valid zone, meaning that it can be configured to work on a wide set of cases. For example, considering the figure 2, if a vehicle is able to detect what is above him and is operating on an environment where the obstacles are wide and have low height, the algorithm can be adapted to accept the top part of the ellipse as a valid zone and give a greater value to r_{ver} than to r_{hor} (this will set a preference to overpass the obstacles from above).

Adding to this, this approach tries to find a solution whenever a valid escape point is not found, moving parallel to the obstacle and trying to find a clear path from a different position (figure 5). However, the algorithm will perform this maneuver only a limited number of times, not ensuring that the desired point will be reached. If no valid path is found, the algorithm will ask to the pilot to take control of the vehicle, hovering on the position where it first detected the obstacle.

As all the obstacles are referenced to a global frame, the E^{2}SP-LCA relies on a good navigation and estimation of the vehicle's position. Another drawback of this approach is that the algorithm can enter on an infinite loop mode. This case might happen on an environment with many obstacles, if it keeps finding an obstacle while avoiding another (previously detected), entering on a mode of constant avoidance that might lead to a deviation from the desired point.

VI. Conclusions and Future Work

The paper presents a LiDAR-based real-time collision avoidance method for multirotor UAVs with the ability to ensure an autonomous structure inspection mission without a predefined out of bounds areas. The collision avoidance method was validated in a simulation environment developed in Gazebo and also in a mixed environment composed by a real UAV performing a mission in an outdoor scenario and a simulated obstacle and LiDAR. This approach provides a safe method to validate the vehicle behavior without the possibility of damage the sensors like LiDAR and also the ability to test in a small-scale UAV (low payload). In both scenarios, campus ISEP, and simulation environment, the vehicle was able to detect the obstacle and generate a collision avoidance safe path. For future work, we intend to validate the algorithm with an UAV with payload capability for a LiDAR Velodyne VLP-16 and perform the validation with natural obstacles like trees and also in the presence of structure obstacles, for instance, power lines, bridges and electricity poles. Another line of work will be the integration of the vision-based power line detection method denoted by PLineD[17] with the E^{2}SP - Escape Elliptical Search Point. The expected output of this future research work is the ability to combine the LiDAR information with the monocular vision system required by the PLineD algorithm and ensure an, even more, robustness UAV autonomous inspection procedure.

Acknowledgment

This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

[1] S. Hrabar, "Reactive obstacle avoidance for Rotorcraft UAVs," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011, pp. 4967–4974. [Online]. Available: http://ieeexplore.ieee.org/document/6094629/

[2] ——, "An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance," *Journal of Field Robotics*, vol. 29, no. 2, pp. 215–239, mar 2012. [Online]. Available: http://doi.wiley.com/10.1002/rob.21404



Fig. 9. UAV trajectory in RVIZ and in the Google Earth image. The yellow line represents the real UAV avoidance trajectory ${}^{W}\mathrm{U}$.

[3] L. Kovacs, "Visual Monocular Obstacle Avoidance for Small Unmanned Vehicles," *Conference on Computer Vision and Pattern Recognition Workshops*, pp. 59–66, 2016.

[4] T. Merz and F. Kendoul, "Beyond visual range obstacle avoidance and infrastructure inspection by an autonomous helicopter," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011, pp. 4953–4960. [Online]. Available: http://ieeexplore.ieee.org/document/6094584/

[5] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu, "LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid," *Aerospace Science and Technology*, vol. 55, pp. 344–358, aug 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1270963816301900

[6] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2008, pp. 807–814. [Online]. Available: http://ieeexplore.ieee.org/document/4650775/

[7] D. Magree, J. G. Mooney, and E. N. Johnson, "Monocular visual mapping for obstacle avoidance on UAVs," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2013, pp. 471–479. [Online]. Available: http://ieeexplore.ieee.org/document/6564722/

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[9] Wolfram Burgard, Maren Bennewitz, Diego Tipaldi, and Luciano Spinello, "Techniques for 3D Mapping." [Online]. Available: http://ais.informatik.uni-freiburg.de/teaching/ss14/robotics/slides/17-3dmapping.pdf

[10] R. Sabatini, A. Gardi, and M. A. Richardson, "Lidar obstacle warning and avoidance system for unmanned aircraft," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 8, no. 4, pp. 711 – 722, 2014. [Online]. Available: http://waset.org/Publications?p=88

[11] Open Source Robotics Foundation, "Gazebo," 2014. [Online]. Available: http://gazebosim.org/

[12] M. Zhang, H. Qin, M. Lan, J. Lin, S. Wang, K. Liu, F. Lin, and B. M. Chen, "A high fidelity simulator for a quadrotor uav using ros and gazebo," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov 2015, pp. 002 846–002 851.

[13] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[14] A. Dias, J. Almeida, N. Dias, E. Silva, and P. Lima, "Simulation environment for multi-robot cooperative 3d target perception," *SIMPAR 2014 4th International Conference on Simulation, Modeling , and Programming for Autonomous Robots, Springer-Verlag Lecture Notes in Computer Science*, 2014.

[15] "Gazebo Simulation · PX4 Devguide," 2016. [Online]. Available: https://dev.px4.io/simulation-gazebo.html

[16] K. Hallenbeck, "DataspeedInc / velodyne_simulator Bitbucket," 2015. [Online]. Available: https://bitbucket.org/DataspeedInc/velodyne_simulator

[17] T. Santos, M. Moreira, J. Almeida, A. Dias, A. Martins, J. Dinis, J. Formiga, and E. Silva, "PLineD: Vision-based Power Lines Detection for Unmanned Aerial Vehicles," in *17th International Conference on Autonomous Robot Systems and Competitions ICARSC*. IEEE, 2017.

Gas Source Localization Strategies for Teleoperated Mobile Robots. An Experimental Analysis

Andres Gongora, Javier Monroy, and Javier Gonzalez-Jimenez

***Abstract*— Gas source localization (GSL) is one of the most important and direct applications of a gas sensitive mobile robot, and consists in searching for one or multiple volatile emission sources with a mobile robot that has improved sensing capabilities (i.e. olfaction, wind flow, etc.). This work adresses GSL by employing a teleoperated mobile robot, and focuses on which search strategy is the most suitable for this teleoperated approach. Four different search strategies, namely chemotaxis, anemotaxis, gas-mapping, and visual-aided search, are analyzed and evaluated according to a set of proposed indicators (e.g. accuracy, efficiency, success rate, etc.) to determine the most suitable one for a human-teleoperated mobile robot. Experimental validation is carried out employing a large dataset composed of over 150 trials where volunteer operators had to locate a gas-leak in a virtual environment under various and realistic environmental conditions (i.e. different wind flow patterns and gas source locations). We report different findings, from which we highlight that, against intuition, visual-aided search is not always the best strategy, but depends on the environmental conditions and the operator's ability to understand how gas distributes.**

I. Introduction

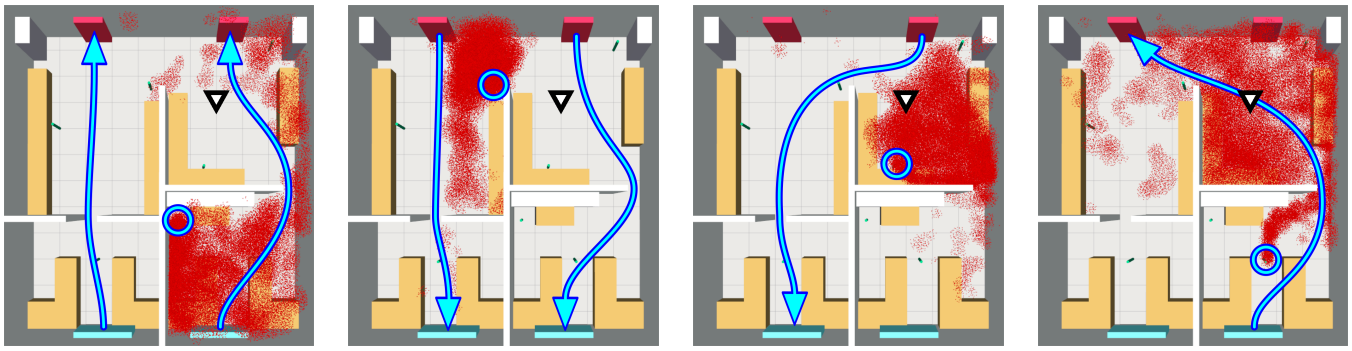
Robot teleoperation (also called telerobotics) is the remote operation of a robot to perceive and interact with the world at a distance [1]. A particular case is that of teleoperating a mobile robot, for instance, to work in hazardous environments (e.g. remote bomb disarming [2]), or to inspect difficult to reach sites (e.g collapsed buildings [3]) among others. In this context, another interesting application is the localization of volatile chemical sources, commonly addressed in literature as gas source localization (GSL). Specifically, the use of a teleoperated gas-sensitive mobile robot to remotely locate one or multiple gas emission sources.

Traditionally, GSL has been tackled with autonomous mobile robots in an attempt to automate the search process. Different approaches, ranging from bio-inspired techniques [4] to engineering solutions [5] have been proposed. Yet, due to the still limited capabilities of autonomous robots and the complex mechanism that rule gas dispersion [6], most works in this field have only been validated under laboratory conditions (i.e. unidirectional and laminar wind fields [7], absence of obstacles in the environment [8], etc.) far from the complexity of real-world settings. Hereof, a teleoperation approach introducing the human intuition and reasoning seems a natural solution to alleviate said drawbacks, something that to the best of the authors' knowledge, has not been tried for GSL to date. An important issue to address in this case is the "definition" of the teleoperation interface. More concretely, what information needs to be provided to the operator for a successful and efficient resolution. Most certainly, it will involve gas identity and concentration measurements of the gas that is being tracked, which could be provided by carrying an electronic nose [9] (e-nose) on the robot. But the sense of smell alone might not always suffice, meaning that other sensory inputs, like the wind flow or visual clues, may also be needed to maximize the search performance. In this work we analyze the influence of these sensory inputs for olfactory telerobotics, specifically, those that match the following GSL approaches:

- **Chemotaxis**: it relies solely on chemical measurements from an e-nose to find the emission source, typically by travelling along the gradient of sensed gas towards where the concentration is highest [7], [10]. It has the advantage of being the simplest approach, but performs poorly for low concentration gas profiles or in turbulent environments [11].
- **Anemotaxis**: in addition to an e-nose, the robot is equipped with an anemometer to track the airborne gas plume [12]. Algorithms implementing this strategy commonly receive the name of "plume tracking methods", and involve following the sensed gas up-wind to its origin [13], [14].
- **Gas Mapping**: it relies on spatio-temporal memory of the aforementioned variables to create a map of how the gases distribute in the environment [15], [16] from which to infer the location of the source [17]. This approach performs well in extremely unstructured and disordered airflow fields, where plumes of well-defined shapes are not likely to be formed. In such cases, mapping the gases of the entire inspection area might be the most reliable way to find the source, although certainly not the most efficient.
- **Visual-aided search**: this approach encompasses any GSL strategy that exploits prior knowledge about the appearance of the gas source to improve the search process [18]–[20]. This entails two important aspects, the proper recognition of objects in the scene, and the semantic inference to correlate the shape of an object with its smell and vice versa. Both of which should pose no challenge to a human operator.

This work has been funded by the Governments of Spain and Andalusia, and the European Regional Development Fund under project TEP530.

Andres Gongora, Javier Monroy, and Javier Gonzalez-Jimenez are with with the Machine Perception and Intelligent Robotics (MAPIR) research group, University of Malaga, Spain.
andresgongora@uma.es
jgmonroy@uma.es
javiergonzalez@uma.es



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 1: The four scenarios of the simulated experiments showing the wind's main flow (arrows), the location of the active gas-leak (circles), other gas-leak candidates shown during visual-aided search (green cylinders), the distribution of the emitted gas after 60 seconds (red point-cloud), and the robot's starting position (black triangle).

The aim of this work is then to seek which of the above strategies is the most suitable for teleoperated GSL tasks. In pursuit of a fair comparison, we propose evaluating their average effectiveness and accuracy, how they affect the operator's search behaviour, and in terms of expended energy, their efficiency. The analysis is conducted on an extensive dataset composed over 150 experiments, gathered during a previous campaign [21], where users had to locate a gas source in a simulated environment with a teleoperated mobile robot. More specifically, and in accordance with the aforementioned GSL strategies, the experiments are classified in four different configurations: (i) pure chemotaxis, (ii) anemotaxis, (iii) gas mapping with chemical and wind flow data, and (iv) the latter plus visual clues (i.e. all sensors combined).

Next, Section II provides a description of the experimental setup and the collected dataset, and Section III proposes different magnitudes to compare the studied GSL strategies. Then, Section IV presents the results and discussion about our study, and finally, Section V offers the most relevant conclusions and suggests future research.

II. Acquisition of the Experimental Data

In this study we use a dataset for teleoperated GSL composed of more than 150 experiments presented in a previous work [21], which is briefly described next to make this paper self contained.

The experiments were performed on a simulator where volunteer participants had to locate as accurately as possible a gas source with a virtual telepresence robot. The simulator was set up with GADEN [22], a gas dispersion simulation framework capable of generating complex 3D environments for mobile robotic olfaction. Four realistic and dynamic gas release scenarios were considered by varying the wind flow conditions and the emission location within an office-like indoor environment (see Fig. 1). Users operated the mobile robot via a web-based interface [23], [24], and declared the gas source by terminating the experiment at the desired location.

Simulation trials were chosen instead of real-world experiments because it allowed us to repeat an experiment several times under identical test conditions (wind flow, gas release, source location, etc.), something hardly achievable in real-world scenarios [25]. Nevertheless, we must stress out that the data gathered during the experiments, and exploited along this study, was not that of the simulator, but those parameters and variables related to user activity, including the search path, navigation commands and the execution time.

Finally, related to the different sensors and algorithms involved in the four GSL strategies to be analyzed, we simulated a photo ionization gas detector (PID), a 2D ultrasonic anemometer for wind flow sensing, implemented the GMRF gas distribution mapping algorithm [15], and visually displayed gas source candidates as green cylinders.

III. Evaluation Methodology

Direct comparison between the different GSL strategies is not a trivial task because of the differences in the test scenarios and in the environmental conditions contained in the dataset. That is, one of the tested strategies could be advantageous when dealing with a certain type of gas distribution, yet fail for the others. Moreover, the evaluation criteria selected for such comparison has also a strong influence in the results. Notice that basic magnitudes commonly found in literature, like the robot's final distance to the source [26], or the time spent searching [12], might not be indicative of true performance differences when comparing GSL strategies under different environmental conditions.

In this work we analyze each combination of scenario and GSL strategy separately, and propose three indices to compare them:

A. Success Ratio (S)

We evaluate the success of a GSL trial by measuring the distance (d) between the user estimated source location (the final position of the robot), and its actual location. A threshold distance D_{th} of 50 cm is set (also used by other works [26], [27]) to label an experiment as successful ($d \leq D_{th}$) or not ($d > D_{th}$).

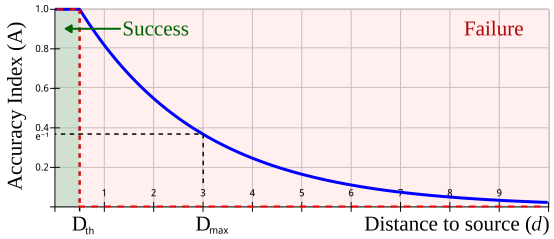


Fig. 2: Comparison between Accuracy Index (blue-solid line) and the success/failure binary label (red-dashed line) for increasing distances to the emission source (d). For both indices Dth = 0.5m and $Dmax$ = 3m.

Based on this measure, we calculate the success ratio (S) for each strategy as the ratio between successful attempts and the total number of trials [28].

B. Accuracy Index (A)

Although the success ratio gives us an insight into the average effectiveness of each strategy, it is too strict to assess its accuracy properly. For example, it does not distinguish between an experiment with a final error close to the specified D_{th}, to another with a very large error (in any case both are marked as failures). For that reason we also compute a continuous accuracy index based on the final distance to the source as:

$$A = \begin{cases} 1 & \text{if } d \leq D_{th} \\ e^{\frac{D_{th}-d}{D_{max}-D_{th}}} & \text{otherwise,} \end{cases} \quad (1)$$

where D_{max} is a constant that controls the decrease rate by establishing a distance upon which the localization is considered to have failed. For our particular scenarios we chose $D_{max} = 3m$, as greater distances mean that the user most probably declared the source location at the wrong room.

Note that A is a better indicator of search accuracy than d alone because it penalizes exponentially unsuccessful experiments instead of discarding them (see Fig. 2), and because can be adjusted to compensate for the environment dimensions.

C. Efficiency Index (η)

Besides accuracy, efficiency is the second most important aspect to consider when developing a GSL system for mobile robots. Because the energy a robot consumes depends on the duration of the experiment and the traveled distance, we must also evaluate how much the operators' path differ from optimal, understanding as optimal path the shortest possible one that goes from the starting position directly to the gas-leak (accounting for obstacles), at the robot's maximum (safe) speed.

To measure efficiency, we use Eq. (2) as proposed by Hayes et al. [28]:

$$\eta = \frac{E_{min}}{E} = \frac{\alpha T_{min} + \beta P_{min}}{\alpha t + \beta p} \quad (2)$$

which can be interpreted as the relation between the minimum required energy to locate the source (e.g. shortest possible path P_{min} and time T_{min}) with the energy E employed during the trial, where α and β denote the energy cost per unit of employed time (t) and travelled path length (p), respectively. Hence, the efficiency index η is a dimensionless quantity defined so that $\eta \in [0, 1]$, with 1 being the optimal path solution.

Note that although α and β can take any positive real value, we follow Hayes et al.'s suggestion and set them such that the minimum distance and time energy costs are equal, that is, $\alpha T_{min} = \beta P_{min}$. Also, because T_{min} is determined by the maximum safe operating speed V_{max} at which the robot can travel the path P_{min} in a specific environment, we know that $T_{min} = P_{min}/V_{max}$. Introducing both considerations into Eq. (2) we get:

$$\eta = \frac{2P_{min}}{V_{max} \cdot t + p} \quad (3)$$

D. Average Occupation Map

Finally, and despite not being a quantifiable magnitude as such, we generate heat maps displaying the proportion of time the users spent, in average, at each location of the environment. These maps facilitate judging the influence of each strategy on the user's search behaviour to determine, for example, whether anemotaxis allowed to discard the exploration of rooms that exhausted clean air.

IV. Analysis and Discussion

In this section we present the results of comparing each of the four GSL strategies according to the aforementioned indices.

A. Search Success

Table I summarizes the success ratios for all combinations of search strategy and scenario, as well as the total successes separately. Keep in mind that an experiment counts only as successful if the operator got within 50 cm of the emission source (refer to Section III-A).

Visual-aided search has generally the best success ratio of them all. However, S drops to 50% for Scenario 4, which, as we will later discuss in Section IV-D, might be because visual clues can be confusing if the environment contains

TABLE I: Percentual success ratios (S) of the tested GSL strategies, individual and overal results for each scenario.

	Chemotaxis	Anemotaxis	Gas Mapping	Visual Search	Summary
Scenario 1	22.2	12.5	11.1	87.5	32.4
Scenario 2	22.2	37.5	22.2	90.9	45.9
Scenario 3	11.1	0.0	0.0	100.0	28.9
Scenario 4	27.3	60.0	38.5	50.0	42.9
All Scenarios	**21.1**	**28.6**	**19.5**	**83.8**	**37.8**

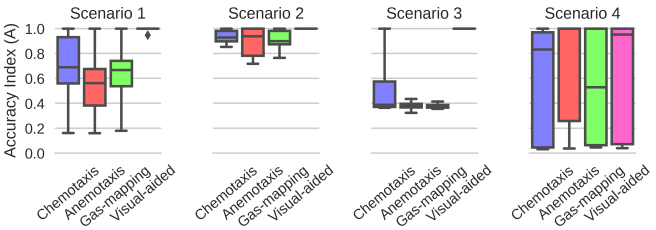


Fig. 3: Accuracy Index for each scenario and GSL strategy (Dth = 0.5m and Dmax = 3m). This index represents how close the operator got to the emission source.

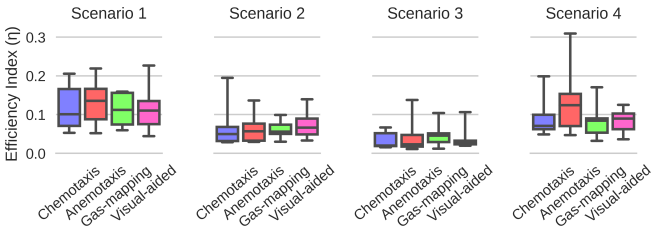


Fig. 4: Efficiency index for each scenario and GSL strategy. Efficiency measures the relation between the shortest possible path and the user's path.

very complex gas distributions. In this case, anemotaxis or gas-mapping appear more advantageous despite their usually low effectiveness.

As for chemotaxis, it does not seem to be specially successful for any of the considered scenarios. Still, it is a robust option with consistent results, even in Scenario 3 where anemotaxis and gas-mapping failed completely.

B. Search Accuracy

As mentioned in Section III-B, a more descriptive and fair comparison involves the use of the Accuracy Index (A). As can be observed in Fig. 3, accuracy retains some similarities with the success ratio (e.g. general effectiveness for each scenario), but now also reveals information about those experiments that failed locating the emission source.

Visual aided search remains in general the most accurate strategy, while anemotaxis and gas-mapping stay approximately on par to each other with similar medians and extremum in most cases.

Also, notice that in terms of accuracy chemotaxis seems to overtake anemotaxis and gas-mapping (except in Scenario 4, where anemotaxis performs best), suggesting that it might be a good choice for teleoperated GSL that requires reliability and simplicity over absolute precision.

C. Search Efficiency

The search efficiency η is an index that should reveal if a low success ratio (S) really means being unable to properly locate the gas source, or on the other hand, indicates that a particular search strategy trades accuracy for energy saving (i.e. coarser localization estimations but in shorter times).

Fig. 4 illustrates in a boxplot the values of this index for each scenario and GSL strategy. It can be seen that for a given scenario all strategies behave similarly (close medians and quartiles), which is a quite noteworthy result: there is no apparent correlation between accuracy and efficiency. This is better noticeable in Fig. 5, where a scatter plot shows a comparison between this index (efficiency) and the accuracy index seen in the previous subsection. In any case, this absence of correlation contradicts our initial intuition, which was that strategies that lead the operator to invest more search effort (i.e more time and therefore less efficiency) would have gotten much closer to the emission source (i.e higher accuracy). As can be seen, none of the compared search strategies shows this tendency, implying that the maximum accuracy of a given search strategy is limited by the environmental conditions.

D. Search Behaviour

Regarding how GSL strategies influence how the operators moved the robot while searching, we analyze next the most visited locations for the tested strategies and scenarios.

Fig. 6 depicts this information by plotting a heat map of where the operators spent their search time. For convenience, it also shows the gas distribution after the experiments' initial 60 seconds (first column), and the user declared gas source locations (i.e. the robot's end positions). As can be observed in all scenarios, the users tended to move along the center of the rooms and doorways for most part of the experiment, and expanded their search in proximity of the emission source or places with high gas concentrations. Still, there appear to be characteristic differences for each configuration:

- Chemotaxis appears to be the most exploratory of all strategies, probably because the users had to locate an initial trail of the gas distribution, and once they had found it, they also needed to determine its gradient.
- Anemotaxis reduces exploration by discarding rooms with clean air currents, and by instantly revealing the gross travel direction of gas plumes. However, the wind information appeared to confuse the users once in proximity of the emission source (particularly for Scenario 1), as vortices and turbulences were constantly stirring gas patches around, with exception of Scenario 4. Here the closeness of the emissions to an ideal gas plume around the source favoured anemotaxis in terms of accuracy (Fig. 3) and efficiency (Fig. 4).
- Gas mapping shows no notable differences with anemotaxis, despite that the employed predictive gas-mapping tool should have aided establishing a coarse location of the emission source.
- Visual-aided search clearly diverges from the other strategies in that it does not start with a search for a hint of the gas plume, but a direct approach towards the visible gas source candidates. As somehow expected, visual clues predominate remarkably over all other information sources, encouraging most operators to visit many false candidates and only stopping their search once they measured a high gas concentration in the proximity of one of them. As illustrated in Table I, this behaviour is not always optimum (e.g. Scenario 4), as it leads to the

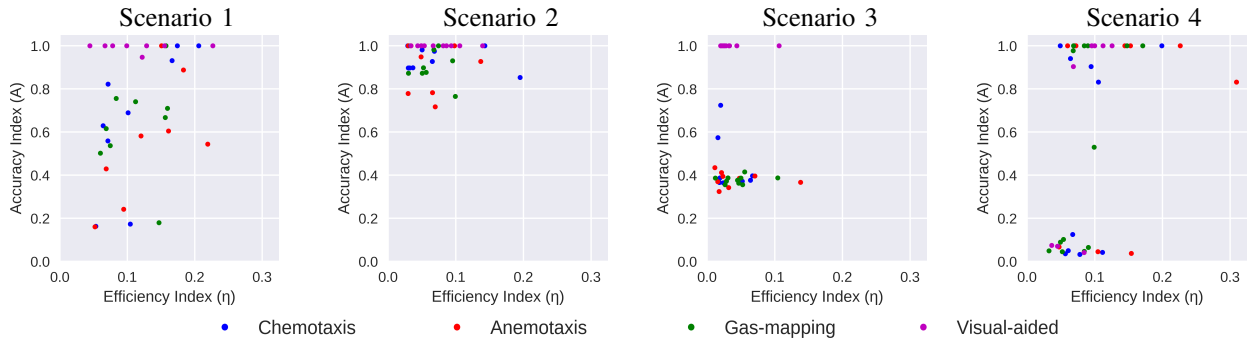


Fig. 5: Scatter plot comparing each experiment's accuracy index (A) against if performance index (η), grouped by test scenario and color coded to indicate the employed GSL strategy. For both axes, higher values are better (i.e. closer to the top right corner), indicating that the search was more efficient and accurate.

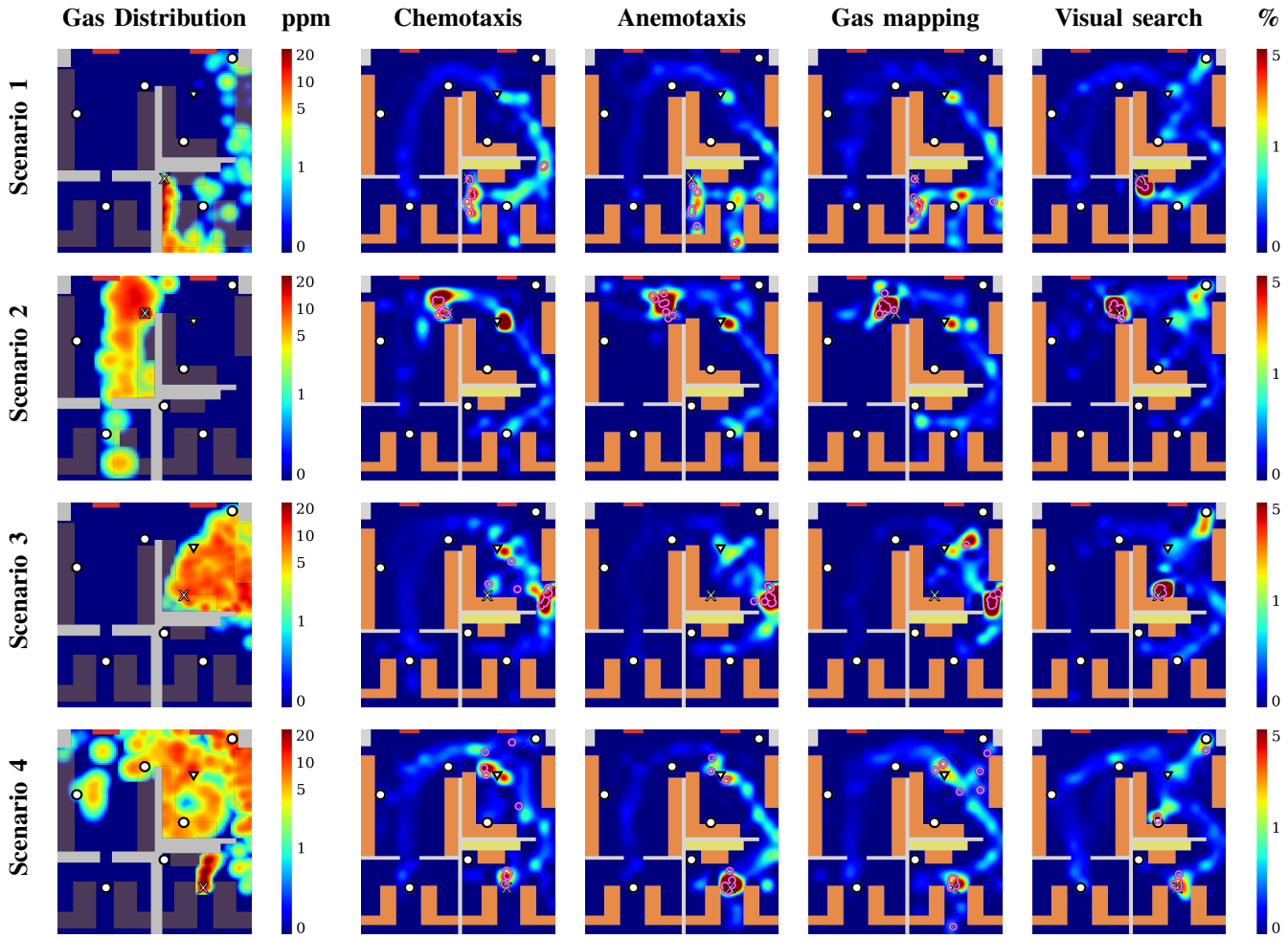


Fig. 6: Heat maps showing where the users (i.e. the robots) searched during the experiments, plotted as percent of the amount of time spent at any location for each strategy and scenario, and smoothed with a 0.5m radius for visualization purposes. The robot's starting position is depicted as a triangle, the gas sources are shown as white crosses (active gas-leaks) and white circles (possible candidates shown during visual search), and the user declared gas source locations (i.e. the robot's end position) as pink circles. The environment's inner walls are shown in gray, and the furniture (not passable by the robot) in brown. For convenience, the left column also shows the gas distribution after the experiments' initial 60 seconds. Please note that the gas distributions in the experiment were dynamic, and therefore differed over time from those shown here.

declaration of false emission candidates with high gas concentrations, just because the wind conditions where such that gas accumulated around them.

Conclusively, a GSL teleoperator would certainly benefit from training to take all information sources into account without being overwhelmed, and get the most out of wind and gas-map information even when presented with visual clues.

V. Conclusions and Future Work

In this work we have compared chemotaxis, anemotaxis, gas-mapping, and visual-aided GSL strategies for telerobotics by analyzing a previously gathered dataset that contains more than 150 test-runs under different environmental conditions and gas distributions. We have evaluated their average effectiveness, accuracy, efficiency and overall impact on the operators with several performance indices, specifically proposed to deal with data coming from experiments with different test conditions, and resolved the following:

- Visual-aided search, which we implemented as a combination of the other approaches plus the representation of visual clues, is the most effective GSL strategy. However, it offers no advantages in terms of search efficiency and can confuse the operator if various candidates have measurable gas around them.
- Chemotaxis is the best choice for applications that benefit from reliability and simplicity rather than precision.
- Anemotaxis and gas-mapping stand-out similarly at the beginning of the GSL process, but require trained operators to be effective once in close proximity to the emission source.
- There is no apparent correlation between accuracy and efficiency. How close the operators can get to the emission source only depends on the environmental conditions and the employed strategy.

For future research we plan to perform similar experiments in a real environment, despite a more limited control over the distribution of the emitted gas, in order to verify the results we have obtained so far.

References

[1] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous robots*, vol. 11, no. 1, pp. 9–18, 2001.

[2] L. Srinivasavaradhan, G. Chandramouli, and A. Maniprashanna, "7 th sense. a multipurpose robot for military," in *Perspective Technologies and Methods in MEMS Design, 2009. MEMSTECH 2009. 2009 5th International Conference on*. IEEE, 2009, pp. 158–160.

[3] T. Kamegawa, T. Yarnasaki, H. Igarashi, and F. Matsuno, "Development of the snake-like rescue robot" kohga"," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 5081–5086.

[4] P. Pyk, S. B. i Badia, U. Bernardet, P. Knüsel, M. Carlsson, J. Gu, E. Chanie, B. S. Hansson, T. C. Pearce, and P. F. Verschure, "An artificial moth: Chemical source localization using a robot based neuronal model of moth optomotor anemotactic search," *Autonomous Robots*, vol. 20, no. 3, pp. 197–213, 2006.

[5] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'infotaxis' as a strategy for searching without gradients," *Nature*, vol. 445, no. 7126, pp. 406–409, 2007.

[6] B. I. Shraiman and E. D. Siggia, "Scalar turbulence," *Nature*, vol. 405, no. 6787, pp. 639–646, 2000.

[7] G. Ferri, E. Caselli, V. Mattoli, A. Mondini, B. Mazzolai, and P. Dario, "SPIRAL: A novel biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 393–402, 2009.

[8] Y. Wada, H. Matsukura, and H. Ishida, "Estimation of gas source location from fluctuating readings of gas sensors and anemometer on mobile robot in outdoor environment," *ECS Transactions*, vol. 75, no. 16, pp. 99–106, 2016.

[9] C. Sanchez-Garrido, J. Monroy, and J. Gonzalez-Jimenez, "A configurable smart e-nose for spatio-temporal olfactory analysis," in *IEEE Sensors*, 2014, pp. 1968–1971.

[10] O. Rochel, D. Martinez, E. Hugues, and F. Sarry, "Stereo-olfaction with a sniffing neuromorphic robot using spiking neurons," in *16th European Conf. on Solid-State Transducers*, 2002, pp. 4–p.

[11] H. Ishida, "Robotic systems for gas/odor source localization: Gap between experiments and real-life situations," in *Proc. IEEE Int. Conf. on Robotics and Automation, 2007. (ICRA 2007)*, 2007, pp. 3–8.

[12] L. Marques, U. Nunes, and A. T. de Almeida, "Olfaction-based mobile robot navigation," *Thin solid films*, vol. 418, no. 1, pp. 51–58, 2002.

[13] J.-G. Li, Q.-H. Meng, Y. Wang, and M. Zeng, "Odor source localization using a mobile robot in outdoor airflow environments with a particle filter algorithm," *Autonomous Robots*, vol. 30, no. 3, pp. 281–292, 2011.

[14] P. P. Neumann, V. Hernandez Bennetts, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, "Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms," *Advanced Robotics*, vol. 27, no. 9, pp. 725–738, 2013.

[15] J. Monroy, J.-L. Blanco, and J. Gonzalez-Jimenez, "Time-variant gas distribution mapping with obstacle information," *Autonomous Robots*, vol. 40, no. 1, pp. 1–16, 2016.

[16] J. Monroy, J. Gonzalez-Jimenez, and C. Sanchez-Garrido, "Monitoring household garbage odors in urban areas through distribution maps," in *IEEE Sensors*, 2014, pp. 1364–1367.

[17] M. Saska, J. Langr, and L. Přeučil, "Plume tracking by a self-stabilized group of micro aerial vehicles," in *International Workshop on Modelling and Simulation for Autonomous Systems*. Springer, 2014, pp. 44–55.

[18] H. Ishida, H. Tanaka, H. Taniguchi, and T. Moriizumi, "Mobile robot navigation using vision and olfaction to search for a gas/odor source," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proc. 2004 IEEE/RSJ Int. Conf.*, vol. 1. IEEE, 2004, pp. 313–318.

[19] P. Jiang, Q.-H. Meng, and M. Zeng, "Mobile robot gas source localization via top-down visual attention mechanism and shape analysis," in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*. IEEE, 2010, pp. 1818–1823.

[20] H. Ishida, T. Ushiku, S. Toyama, H. Taniguchi, and T. Moriizumi, "Mobile robot path planning using vision and olfaction to search for a gas source," in *Sensors, 2005 IEEE*. IEEE, 2005, pp. 4–pp.

[21] A. Gongora, J. Monroy, and J. Gonzalez-Jimenez, "A robotic experiment toward understanding human gas-source localization strategies," in *Int. Symposium on Olfaction and Electronic Nose (ISOEN)*, 2017.

[22] J. Monroy, V. Hernandez-Bennetts, H. Fan, A. Lilienthal, and J. Gonzalez-Jimenez, "Gaden: A 3d gas dispersion simulator for mobile robot olfaction in realistic environments," *MDPI Sensors*, vol. 17, no. 7: 1479, pp. 1–16, 2017. [Online]. Available: http://mapir.isa.uma.es/mapirwebsite/index.php/mapir-downloads/papers/259

[23] F. Melendez-Fernandez, C. Galindo, and J. Gonzalez-Jimenez, "A web-based solution for robotic telepresence," *Submitted*, 2017.

[24] J. Monroy, F. Melendez-Fernandez, A. Gongora, and J. Gonzalez-Jimenez, "Integrating olfaction in a robotic telepresence loop," in *Int. Symposium on Robot and Human Interactive Communication*, 2017.

[25] Y. Zhang, *Indoor air quality engineering*. CRC press Boca Raton, FL, 2005.

[26] A. Lilienthal, H. Ulmer, H. Frohlich, A. Stutzle, F. Werner, and A. Zell, "Gas source declaration with a mobile robot," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 1430–1435.

[27] A. J. Lilienthal, T. Duckett, H. Ishida, and F. Werner, "Indicators of gas source proximity using metal oxide sensors in a turbulent environment," in *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*. IEEE, 2006, pp. 733–738.

[28] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 260–271, 2002.

Constrained-covisibility marginalization for efficient on-board stereo SLAM

Matías A. Nitsche[1] and Gastón I. Castro[1] and Taihú Pire[2] and Thomas Fischer[1] and Pablo De Cristóforis[1]

***Abstract*— When targeting embedded applications such as on-board visual localization for small Unmanned Air Vehicles (UAV), available hardware generally becomes a limiting factor. For this reason, the usual strategy is to rely on pure motion integration and/or restricting the size of the map, i.e. performing visual odometry. Moreover, if monocular vision is employed, due to the additional computational cost of stereo vision, this requires dealing with the problem of unknown scale.**

In this work we discuss how the cost of the tracking task can be reduced without limiting the size of the global map. To do so, the notion of covisibility is strongly used which allows choosing a fixed and optimal set of points to be tracked. Moreover, this work delves into the concept of parallel tracking and mapping and presents some finer parallelization opportunities.

Finally, we show how these strategies improve the computational times of a stereo visual SLAM framework called S-PTAM running on-board an embedded computer, close to camera frame-rates and with negligible precision loss.

I. Introduction

With the growing interest in UAVs, there has been an increasing need for localization methods capable of operating on-board and in real-time. Designing systems that provide accurate pose estimation in challenging environment while running on platforms with limited computational resources is thus a key problem in mobile robotics. For this reason, in GPS-denied scenarios such as indoors or outdoors in areas with poor reception, vision-based approaches have been widely used.

However, efficient vision-based localization solutions generally still require considerable computational power. This is particularly difficult when targeting low-payload robots, where only small and low-resource processing hardware can be employed. It is thus worth considering new strategies for reducing computational requirements of vision-based localization methods and allow meeting real-time constraints.

From a methodological point of view, vision-based localization methods can be classified as visual odometry (VO) or Simultaneous Localization and Mapping (SLAM) approaches. VO techniques focus on ego-motion integration to get a camera pose estimate, while SLAM approaches build a global map against which the robot can localize. One of the main drawbacks of VO approaches is that accumulated pose drift is never corrected due to the absence of global map information. In contrast SLAM approaches are able to localize against the map without requiring motion integration.

For the case of SLAM (Simultaneous Localization and Mapping), one widely adopted strategy was one proposed with PTAM [1] (Parallel Tracking and Mapping), where both tracking and mapping tasks are decoupled as separate computing threads. Many recent feature-based visual SLAM approaches embraced this approach and added also a loop-closing thread [2], [3].

Both tracking and mapping tasks have costly operations (such as point-matching and Bundle-Adjustment) which are largely dependent on the number of map-points. Since the map can grow up to thousands of map-points and keyframes, these tasks would not be able to run in real-time if all map is considered at each step. While one simple approach could be to restrict the size of the map and discard old information, this increases pose drift and eliminates the possibility of loop-closure.

Therefore, an efficient method to determine which part of the map is relevant is required. For tracking, the subset of points expected to be observed by the current camera-frame is needed. While for mapping, a subset of related keyframes and their corresponding observations need to be selected to perform local optimization. This selection becomes critical since not only it determines the efficiency of tracking and mapping tasks but also their accuracy and robustness.

The notion of covisibility [4] can be used to determine the relevant keyframes and map points (i.e. mutually observable) to a given camera pose and the currently tracked map. A pair of keyframes are said to be co-visible when they observe at least one visible map point in common. However, it is not yet clear how to best select these keyframes and map points using covisibility.

In this regard, the main contribution of this paper is to present a map marginalization strategy based on covisibility information that can be employed to solve more efficiently some of the most computationally demanding tasks in a SLAM system. We analyze how precision is affected and how this can help running full SLAM on low-resource hardware platforms. Furthermore, this work presents several approaches to minimize contention points and improve parallelism.

II. Related Work

In terms of on-board vision-based localization, a number of recent works employing either visual-odometry or SLAM-based approaches are presented.

[1]Matías Nistche, Gastón Castro, Thomas Fischer and Pablo De Cristóforis are with the ICC, Computer Science Research Institute (CONICET-UBA), Argentina. {mnitsche, gcastro, pdecris}@dc.uba.ar

[2]Taihú Pire is with the CIFASIS, French Argentine International Center for Information and Systems Sciences (CONICET-UNR), Argentina. pire@cifasis-conicet.gov.ar

Sanfourche et al. [5] proposes a stereo visual odometry suitable for UAVs. The method tracks features from successive camera frames while establishing 2D-3D associations with respect to a keyframe-based map. There is no optimization performed over the map, however pose drift grows slower compared to frame-to-frame visual odometry. They achieve 20Hz operation, however, while they claim their approach is suitable for embedded systems, experiments are performed using a relatively powerful Intel Core 2 Duo computer.

Weiss et al. [6] propose an on-board localization method equipped with a single camera that uses a inertial-optical flow approach for speed and IMU bias estimation. As speed integration is prone to position drift, an optimized version of PTAM is used to produce a 6DoF pose estimation. Combining visual and inertial measurements has proven effective for solving localization on UAVs.

Burri et al. [7] build upon visual-inertial odometry (VIO) obtaining dense environment reconstruction suitable for mission planning and exploration. Estimation drift derived from the VIO method is corrected by performing relocalization between local submaps. They obtain 20Hz operation time using a tailor made ARM-FPGA system [8].

Leutenegger et al. [9] proposes a novel visual-inertial SLAM system coined OKVIS. They formulate the problem as one joint optimization where an IMU measurement error term is considered along with the usual reprojection error within the minimization cost function. However, OKVIS is not conceived to work in low-resource hardware platforms and experiments are performed on a powerful laptop computer.

One of the main works on fully on-board stereo vision for UAV navigation was presented by Schauwecker et al. [10], where a visual odometry system based on PTAM is used to estimate the UAV pose. This work was extended in [11], where two stereo cameras were used: one facing forward, used to run a reduced SLAM system, and another facing downward, used for ground plane detection and tracking. Estimations obtained by each sensor are fused using EKF. The authors show that using two stereo cameras significantly increases pose estimation accuracy and robustness.

In terms of reducing the computational cost of SLAM approaches, some works focus on the problem of dealing with a large global map.

In [12] Lynen et al. present a framework for tracking the camera pose relative to a global map. They assert their method can be used for real time localization of mobile platforms with limited resources without the use of an external server. They achieved this by employing a covisibility strategy [13] to efficiently localize against the map.

Strasdat et al. [14] introduces an optimization framework that distinguishes two different keyframe windows. An inner window is composed by those keyframes that will be actively optimized while an outer window of close keyframes will act as fixed constrains. Optimization windows are defined based on the degree of covisibility between keyframes.The authors claim constant time operation when the maximum number of keyframes on each window is restricted.

Mur-Artal and Juan D. Tardós present a SLAM system called ORB-SLAM2 [3] that maintains a covisibility graph and a corresponding minimum spanning tree. These graphs are used to retrieve local windows of keyframes, so that tracking and mapping tasks operate locally while allowing to work on large environments and enabling for pose-graph optimization performed when closing a loop. This covisibility graph used in ORB-SLAM2 is similar to the one introduced by Strasdat et al. in [14].

III. Method

In the context of optimization-based SLAM, the tracking task is in charge of determining frame-to-frame camera pose. The tracking minimizes re-projection errors determined by map point to image-feature matches. Since this optimization requires an initial solution, it is usual to predict camera motion from previous poses and/or using additional proprioceptive sensors (e.g. IMU, wheel encoders, etc.). With this predicted pose, map-to-frame matches need to be established. To do so, map points are projected to the camera frame and then, by nearest-neighbor search in image-descriptor space, matches are obtained. These matches become observations under the optimization framework, representing a set of constraints. Finally, pose-optimization is performed, typically using Gauss-Newton or Levenberg-Mardquardt approaches.

From the previous steps performed by the tracking task, the most computationally demanding are typically: feature detection and descriptor extraction, point-to-feature matching, local map building and pose minimization. In particular, computational time of all these but feature extraction step strongly depend on the number of map points considered.

Moreover, for matching, in principle all map points need to be projected to the camera frame, which scales linearly with the size of the map. Also, this process is wasteful since many points could be actually invisible due to occlusions. Thus, a better approach is to use covisibility information in order to find map points seen by keyframes which share observations to the current camera frame. In other words, it is possible to build a *local map* of points which are highly likely to be currently visible.

When dealing with resource-constrained computing platforms, including all co-visible points to the local map may incur in excessive computational cost for tracking. Additionally, the size of this map can grow unbounded in certain conditions, which represents an undesirable situation in terms of real-time operation. For this reason, in these cases it is desirable to limit this local map.

A difficulty here appears since covisibility information needs to be built empirically during tracking from successful matches: whenever a point is matched to an image-feature in a given camera frame, this point is defined as *visible* from said frame. Covisibility can thus be represented as a graph between keyframes where each edge has a weight corresponding to the co-visibility degree, i.e. the number of shared observations.

Covisibility information built in this manner (via detected matches) cannot be guaranteed to match actual covisibility as would be obtained by exhaustive pair-wise frustum-culling between all keyframes. Thus, using covisibility information to build the local map may not necessarily retrieve the full set of points that could be observed by the current frame. For this reason, some works [3] propose to use not only directly co-visible keyframes but also a second level of keyframes co-visible to the first. This increases the possibility of including points which should be marked as directly co-visible but where this information has not yet been discovered. However, this comes at the expense of a larger local map and thus higher tracking cost.

In the following section, a simpler and more effective strategy for covisibility based local map building is presented, which allows to reach a bounded computational cost of the tracking task. Furthermore, it let balance efficiency vs. tracking precision.

A. Proposed Local Map Building Strategy

The proposed strategy for local map building is outlined in algorithm 1.

This strategy defines how to obtain the set of points M_L and keyframes K_L defining the local map, based on the previous set of successfully tracked points M_T (i.e. matched to image-features). Moreover, a *reference keyframe* K_r is designated during this process. This keyframe is considered to be the closest to the current camera frame in terms of observed points and is later used to determine when a new keyframe should be added.

With this local map, the points contained are then projected to the current image and used for point-to-feature matching. The set of successfully matched points will define the set M_T used for the next iteration. In other words, M_T will always be a subset of M_L.

In general terms, the local map building strategy find points visible by a set of keyframes K_{cov}, co-visible to the reference frame K_r. This reference, in turn, is defined as the keyframe observing the highest number of points in M_T.

In particular, only the first N keyframes with highest covisibility degree with K_r are considered. Moreover, only up to M points observed by these keyframes are added to M_L. Finally, low covisibility keyframes can be ignored using a minimum threshold C_{min}. As a result, the size of the local map is bounded. Moreover, the cost of building this local map scales linearly w.r.t. the number of keyframes co-visible to K_r. This is due to the fact that this term dominates the number of observing keyframes of a given point in M_T. Moreover, both M_T and K_{cov} are bounded by M (in previous iteration) and N, respectively.

It should be noted that M_L is first initialized using M_T, since using the aforementioned limits does not guarantee that all points in M_T will be in the result. This is particularly important when tracking is bad and M_T is small, which would result in a too small M_L.

As a result of applying this local-map building strategy, M_L is bounded by M. Thus, the cost of subsequent matching and minimization operations are also bounded by M.

Algorithm 1: Local-Map building strategy

Input: M_T tracked map
Output: M_L local map, K_L local keyframes, K_r reference keyframe

```
/* initialize with previous tracked map                    */
M_L ← M_T
/* find K_r which observes most points in M_T               */
foreach p in M_T do
    foreach K_i : observingKeyframes(p) do
        count(K_i) ← count(K_i) + 1
K_r ← argmax_{K_i} count(K_i)
/* get N most covisible keyframes to K_r                    */
K_cov ← sort_n(covisible(K_r), N)
/* add up to M pts observed by KFs in K_cov to M_L          */
foreach K_i in K_cov do
    if count(K_i) < C_min then
        continue
    M_L ← M_L ∪ observedPoints(K_i)
    K_L ← K_L ∪ {K_i}
    if #M_L > M then
        break
```

IV. Efficient on-board stereo SLAM

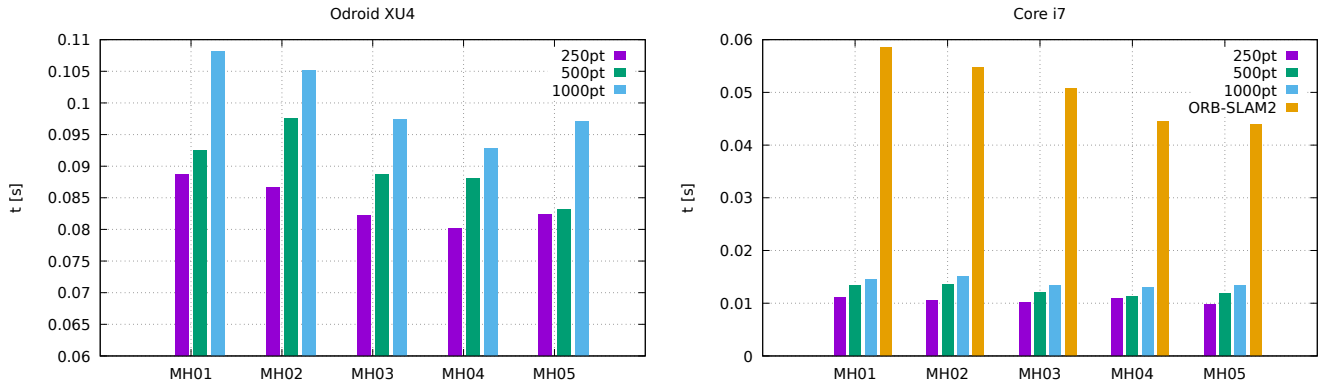
In order to verify the strategy proposed in this work, we build upon the stereo visual SLAM system S-PTAM [2], which has proven to be stable, accurate and suitable for large scale operation. In next sections, we describe some design and implementation considerations to better exploit parallelism in a optimization-based SLAM system. With these improvements S-PTAM is capable of running on-board on low-resource hardware platforms.

A. Efficient Map Access

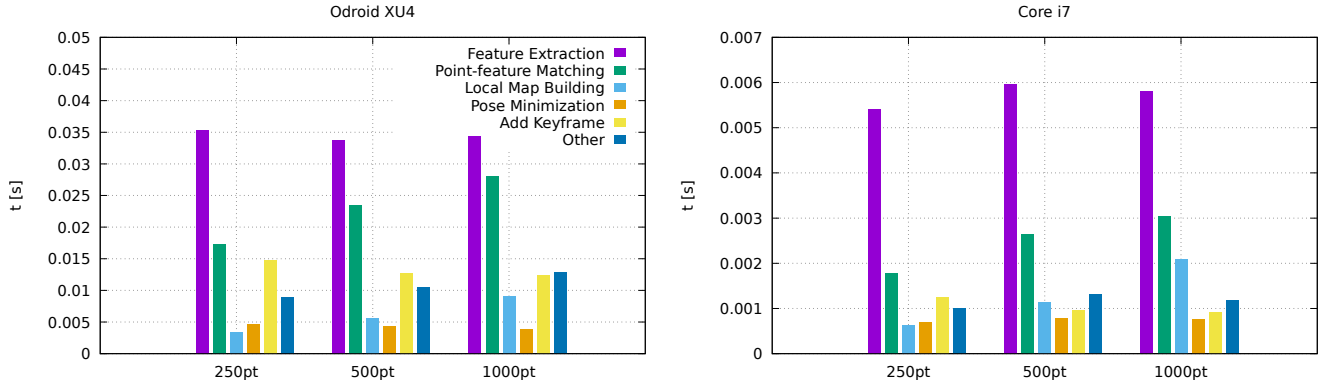
1) Access Requirements: As tracking and mapping tasks communicates through the map, contention points need to be minimized for the purpose of maximizing parallelization. The map is composed of two elements: map points and keyframes, which are related by two different graphs: a visibility bipartite graph between points and keyframes (where edges describe a measurement) and a covisibility graph between keyframes (where edges determine degree of covisibility).

Tracking requires frequent read access to keyframes and map points for local map definition (Section III) and feature-to-point matching. Occasionally, a frame is declared keyframe and its unmatched features are triangulated and added to the map as new points. Only in this case the tracking thread requires write access to maintain consistency of visibility and covisibility graphs.

On the other hand, the mapping task optimizes the map when a new keyframe is created. It requires both read and write access to keyframes and map points. It is also responsible for finding new matches between points and



(a) Execution time for the complete tracking task, for $M = 250, 500, 1000$ points.



(b) Execution time of each step of the tracking task, for $M = 250, 500, 1000$ points, averaged over all MH sequences.

Fig. 1: Mean execution times for different values of the local map size (M), for both processing platforms, over all MH sequences of EUROC dataset.

image features, updating the relations graphs with the new measurements.

Finally, a loop-closing task tries to detect loops upon new keyframe creation and, when detected, performs global pose-graph optimization. This involves write-access to all keyframes and map points. In principle this would involve halting mapping and tracking operations, however this is undesirable.

2) Synchronization strategy: In order to satisfy previous requirements while maximizing parallelism, instead of locking the whole map, map points and keyframes can be individually locked in order to access information such as point position and keyframe poses as well as other relational information such as keyframe covisibility.

In this scenario, when requiring write-access to multiple keyframes and map points simultaneously, such as when closing a loop, one approach could be to lock all of these simultaneously. However, a better choice is to arbitrate access to the map by defining required working regions beforehand. In other words, the mapping task can inform the region where it is currently working on and the loop-closing task can then avoid this region until it is freed.

B. *Feature extraction and matching*

Similarly to ORB-SLAM, features can be extracted on each image using a fixed-size grid. Over each cell, FAST [15] features are detected. If not enough are found in one cell, a lower feature response threshold is used. This process allows to obtain features throughout the whole image. However, this step results in a large number of features and filtering is required. To do so, we recursively divide the image area using a quad-tree, assigning features to each cell accordingly. When a given number of cells is reached, the feature with the strongest response of each cell is returned. In this way, it is possible to limit how many features are used for tracking and have an homogeneous distribution of feature in the image.

Also, to better exploit hardware parallelism, feature extraction (detection and description) and point-to-feature matching can be done concurrently amongst both images of the stereo pair.

V. Experiments

In this section we present the results obtained by the use of the local map building strategy in terms of the resulting performance improvement, particularly when running on low-resource hardware, and of its impact in localization precision. As a reference, we also run the modified S-PTAM

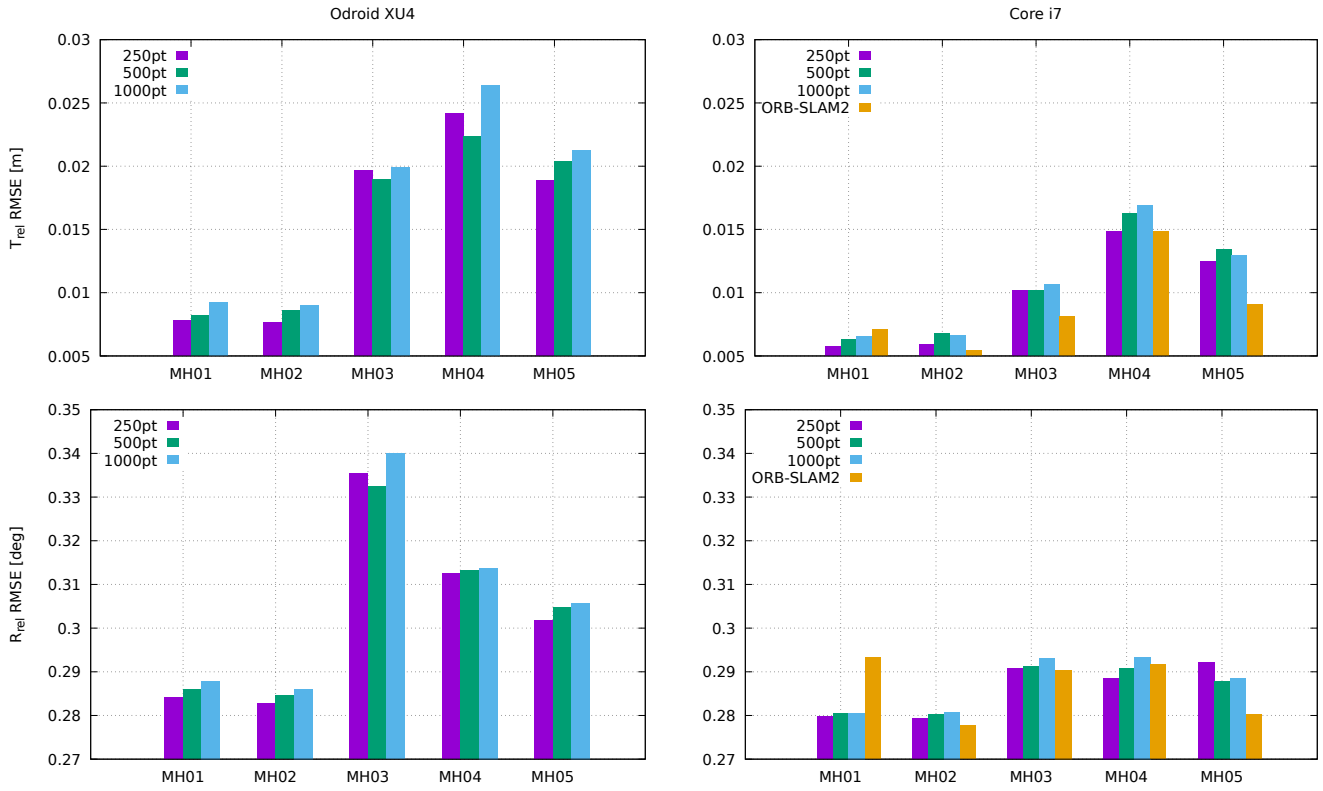


Fig. 2: Impact of different local map sizes on precision: RMSE values for relative translation and rotation errors, for each MH sequence of the EUROC dataset, for both processing platforms. Corresponding error values for ORB-SLAM2 are included.

on a powerful desktop computer and compare the obtained results and that of ORB-SLAM2 [3]. In this case we are not using loop-closing on S-PTAM, for fairness we disable this feature in ORB-SLAM2 as well.

Since the purpose of this work is to ultimately enable on-board and real-time execution of a Visual SLAM system for localization of UAVs, we test the S-PTAM system running on board an Odroid XU4 computer with four Cortex-A15 cores running at 2 GHz and four Cortex-A7 cores running at 1.4 GHz. For establishing a precision baseline without hardware constraints, we also run S-PTAM on an Intel Core i7-7700.

For a realistic and repeatable experimentation, we used the EUROC MAV dataset as input data. Since this is a challenging dataset with rapid camera-motion, we fuse IMU data using the MSF sensor fusion framework [16]. When running S-PTAM on the Odroid XU4, we replay the EUROC rosbag files on a desktop computer and feed data through an Ethernet connection to the Odroid, so as to remove any impact of I/O to the performance of the embedded system. Moreover, we only run MH sequences since V sequences present motion which is too fast for Odroid to follow.

In figure 1, we present the execution times for the tracking task of S-PTAM when using our proposed local map building strategy and that of ORB-SLAM2 (with author's parameters for this dataset), for different values of M (maximum size of local map). We also show the computationally most demanding steps of the tracking task in S-PTAM. We do not present execution of ORB-SLAM2 on-board the Odroid XU4 computer since tracking was quickly lost due to high processing time for each frame.

In figure 2 we present the relative translation and orientation errors of S-PTAM with different values of M, when running on each platform. Moreover, we also measure ORB-SLAM2 tracking precision for reference. It should be noted that since we are interested in using the SLAM system as a real-time localization source for autonomous navigation of UAVs, what is of importance when measuring precision is the error arising from camera pose reported after each tracking iteration, instead of the one obtained after local or even global bundle-adjustment. This is an important difference w.r.t. to other works where error is measured only after the complete dataset is replayed. For this reason, we run ORB-SLAM2 against EUROC dataset while measuring localization error in the same way, using instantaneous camera pose information.

When analyzing the results, a series of considerations can be made. First, it can be seen that the total tracking time (fig. 1a) of S-PTAM is much smaller than ORB-SLAM, around $4x$ to $6x$ faster. Also, the performance of S-PTAM on the Odroid XU4 is around and order of magnitude lower that on the Core i7. In any case, Odroid XU4 manages to track the camera at around 9 to 12 Hz in general, which is close to camera frame-rate. On the other hand, on Core i7, tracking rate is around 66 Hz, which is considerable higher than camera frame-rate. Second, it is possible to observe the

effect of the proposed local map building strategy, where limiting the size of this map reduces computational cost.

In order to better understand the performance improvement obtained by the use of the proposed local map building strategy, we also show the mean execution time of the main steps of the tracking task (fig. 1b). It can be seen that in all cases, the most demanding step corresponds to feature extraction (detection and description). The second most demanding step corresponds to the point to feature matching. Here it can be seen that lowering M has a positive impact on performance. Finally, as expected the cost of the local map building step itself is also lessened the less points are included in the output. On the other hand, lowering M has a slight negative impact on the keyframe creation step. This can be explained since a smaller local map implies that there is a higher chance of adding points which were not successfully matched.

In terms of tracking precision, in figure 2 it can be seen that, in general, reducing the number of points in the local map does not entail a significant impact on translation or rotation relative errors. Moreover, a difference can be observed between execution on Odroid XU4 and the Core i7 computers. This can be explained since on Odroid XU4 there is approximately a 50% frame-loss. Finally, when comparing to ORB-SLAM2 running on the Core i7, it can be seen that the localization performance of the S-PTAM system is quite similar. On the other hand, due to the high computational cost of ORB-SLAM2, measuring the localization precision running on Odroid XU4 was not possible.

VI. Conclusions

This work presents a local map building strategy based on constrained covisibility marginalization of the global map, in the context of an optimization-based SLAM. The purpose of this strategy is to reduce the computational cost of the tracking task in order to restrict the the size of the local map, aiming at on-board and real-time execution of the system on resource-constrained platforms, such as those present on small UAVs.

In order to prove the feasibility of the proposed approach, we implemented this strategy on the state-of-the-art S-PTAM system and performed a series of experiments on the challenging EUROC dataset. We also ran the ORB-SLAM2 system to establish a baseline for performance and precision.

Results show the reduction of computational time of the tracking task, which is of significant importance for on-board execution of the system on UAVs. Moreover, it can be seen how the S-PTAM system manages to track the camera at rates exceeding most standard cameras when running on a more powerful computer. On the other hand, on a resource-constrained platform, while performance is much lower, it is still possible to track the camera with rapid and challenging motions.

VII. Future Work

Analyzing the obtained results, we identify some future work areas. First, it can be seen that feature detection, description and matching are still some of the most demanding steps of a feature-based SLAM system. For this reason, it is interesting to consider finer optimization of these tasks using architecture-specific features, such as ARM's NEON instruction set. Second, we plan to perform closed-loop experiments by performing autonomous navigation of UAVs with the S-PTAM system running on-board and functioning as the main localization source. Finally, since S-PTAM already features loop-closing, we plan to evaluate performance of the system when running it on-board and to introduce optimization to allow for real-time execution.

References

[1] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.

[2] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-PTAM: Stereo Parallel Tracking and Mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27 – 42, 2017.

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *CoRR*, vol. abs/1610.06475, 2016.

[4] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 3738–3744.

[5] M. Sanfourche, V. Vittori, and G. L. Besnerais, "evo: A realtime embedded stereo odometry for mav applications," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2107–2114.

[6] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 957–964.

[7] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 1872–1878.

[8] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 431–437.

[9] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visualinertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[10] K. Schauwecker, N. Ke, S. Scherer, and A. Zell, "Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing," in *Autonomous Mobile Systems 2012*, ser. Informatik aktuell, P. Levi, O. Zweigle, K. Huermann, and B. Eckstein, Eds. Springer Berlin Heidelberg, 2012, pp. 11–20.

[11] K. Schauwecker and A. Zell, "On-board dual-stereo-vision for the navigation of an autonomous mav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 1–16, 2014.

[12] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization." in *Robotics: Science and Systems*, 2015.

[13] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *European conference on computer vision*. Springer, 2012, pp. 752–765.

[14] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011, pp. 2352–2359.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer vision–ECCV 2006*, pp. 430–443, 2006.

[16] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2013, pp. 3923–3929.

On the use of Unmanned Aerial Vehicles for Autonomous Object Modeling

Michael C. Welle*, Ludvig Ericson*, Rares Ambrus* and Patric Jensfelt*

***Abstract*— In this paper we present an end to end object modeling pipeline for an unmanned aerial vehicle (UAV). We contribute a UAV system which is able to autonomously plan a path, navigate, acquire views of an object in the environment from which a model is built. The UAV does collision checking of the path and navigates only to those areas deemed safe. The data acquired is sent to a registration system which segments out the object of interest and fuses the data. We also show a qualitative comparison of our results with previous work.**

I. INTRODUCTION

Mobile robots are slowly making their way into everyday life, with robustness and operational up-time increasing every year. [1] report deploying mobile robots in unstructured environments (offices and elderly care homes) for durations of up to 6 months. Such experiments are pushing the boundaries of what these systems are capable of, and widen the frontier to the next set of issues to be addressed, such as exposure to large amounts of data, learning patterns about the environment, life-long robust localization and navigation, etc.

For robots to operate successfully for extended periods of time, their understanding of the environment needs to adapt as new data becomes available. In our work we are interested in analysing changes in the environment, and in building predictive models of where objects or people are likely to be at some future time. The basis for this is a robust perception system, able to reliably segment, model and re-identify objects of interest in the environment.

In previous work [2] we have looked at autonomously navigating around objects of interest and acquiring multiple views, which are fused into canonical models of the objects. We have seen that recognition of these objects in future observations increases with the number of views initially acquired by the robot. Unfortunately, when deploying such systems in unstructured environments (e.g. an office or a home) for extended periods of time, it quickly becomes apparent that a mobile robot's path is quite often obstructed by natural clutter (tables, chairs, etc.). This implies that in most situations a robot can potentially navigate to one or two additional vantage points within a room, which is often not enough.

In this work we address the first steps in solving this issue. Our aim is to augment the capabilities of an indoor mobile robot by pairing it with an Unmanned Aerial Vehicle (UAV). UAVs have a much wider reach as compared to wheeled mobile robots, and would thus allow the exploration of the environment from new angles which would otherwise be inaccessible. This would also allow the robot to overcome issues such as missing data due to occlusions or oblique surfaces, and would facilitate applications such as tracking or recognition. However, unlike mobile robot navigation, autonomous indoor flying is still very much a challenge, both technical and from a safety point of view. While a number of commercial standard mobile robot platforms are available with off-the-shelf navigation and localization capabilities, UAVs for indoor use are often built from scratch to fit the needs of the research in question. In addition, the flight time is limited to only a few minutes.

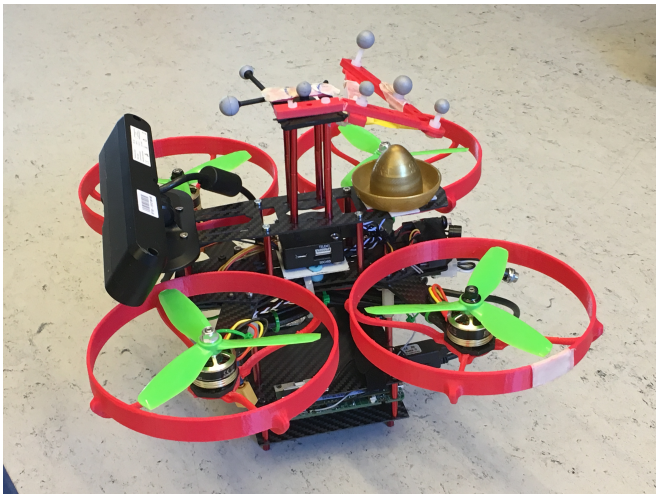


Fig. 1: The quadrotor drone used in the experiments. It is equipped with a Primesense RGBD camera looking forward and a set of reflective IR-markers for motion capture positioning.

In this paper we present an end to end object modeling pipeline with an emphasis on the part from where an object to be observed has been identified. To limit the scope and focus on the perception and object modeling part, we make a number of simplifying assumptions to account for the abovementioned challenges. We make use of a motion capture system for positioning the drone. In the past we have shown that a mobile robot can robustly segment out objects of interest in the environment through change detection [3]. This is what we envision as the means with which the objects to be observed are generated. In this work we use a simpler method, which facilitates running experiments with the UAV in a safer, more confined space (a cage with nets on all sides).

*The authors are with the Centre for Autonomous System at KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden {mwelle,ludv,raambrus,patric}@csc.kth.se

The contribution of this paper is a UAV system which is able to autonomously plan and follow a path, while keeping an object in the environment in view. The UAV does collision checking of the path and navigates only to those areas deemed safe. While navigating, the UAV collects RGBD views which are sent to a registration system that segments out the object of interest and fuses the data. We also show a qualitative comparison of our results with previous work.

II. RELATED WORK

This paper presents a system and hence the related work is rather broad in scope. We focus here on discussing related work for view planning and object modeling.

In [4] the view planing problem (VPP) is surveyed. In [5] the authors propose an information entropy-based approach to the VPP. With a metric that measures the information gain for a possible next view point of a camera, they are able to pick the next best view out of the set containing all possible next view points. Repeating this in succession gives a path that gives a solution to the view planning problem. In [6] autonomous multicopters are used to build a 3-D model of outdoor structures. [7] discusses the setup of a UAV outdoors but adding an obstacle avoiding system. Similarly, [8] employs a freely available 2-D map of buildings in order to construct a rough 3-D model and inspect and refine it with a UAV. Bircher et al. [9] focus on structural inspection in their work, and they employ a two-step optimization paradigm to find good viewpoints. In the context of GPS-denied environment [10] presents an online path planning approach in cluttered environments. An area that has many similarities to the UAV research is underwater robots. They also have more degrees of freedom compared to a regular wheeled robot on land. Work presented in [11] shows a sampling based design of an inspection route. Also from the underwater domain, [12] presents a "Next Best Underwater View" that considers the placement of a light source. The approach can deal more effectively with distortions during the mapping process.

Our work also deals with object modeling by fusing RGBD data. Closest to our work is that of Faulhammer el al [2], where object models are created autonomously with a mobile robot, by acquiring additional views. The biggest difference between our work and that of [2] is that we run our experiments on a UAV - a platform much more difficult to control. In addition, while [2] has access to the robot odometry and uses a camera tracker to obtain the camera poses, our registration method uses a minimization framework which exploits the structure of our problem, and which does not rely on an initial guess for the camera poses. Since our aim and that of [2] are similar, we show a qualitative comparison with their method in the results section. The work of Prank et al [13] also deals with object model creation, however the focus is on a user-friendly system which allows the easy and robust creation of such models on a turntable. We include the models we obtain of our objects using their method in the results section.



Fig. 2: An overview of the system presented in the paper. The input is a partial object to be observed and the output is a (more) complete object model. The main steps in the project are view planning, plan execution and object modeling data fusion.

Finally, we obtain the resulting object model through a scene differencing operation. Herbst et al. [14] and Finman et al. [15] also use scene differencing for segmenting out objects, however the emphasis is not on creating accurate 3D models, but either on SLAM or on improving segmentation. Moreover, neither method explicitly addresses the view planning problem in the context of an autonomous agent.

The related works mentioned here do solve parts of our problem but to the best of the authors knowledge we present the first end to end solution for autonomous object modeling using UAVs in indoor environments including view planning with collisions avoidance.

III. OVERVIEW

An overview of our system is shown in Fig. 2. The input is a cluster of points corresponding to a partial observation of an object and the output is an object model. The main steps in the pipeline are view planning, plan execution and object modeling data fusion. Sections IV-A, IV-B and IV-C describe these in more detail. Figure 3 shows a snapshot from one of the experiments. We can see to the left the UAV hovering near an object sequentially moving between the view points generated by the view planning component. The plan is checked for collisions, in this case typically caused by the walls of the cage. In the upper right corner the camera image at the same point in time is shown, with the object clearly in view. The lower right corner show the raw RGBD frame along with the view plan.

A. Experimental setup

To better understand some of the decisions made in the method section we present our experimental setup already here.

1) Hardware: The UAV we use is shown in Figure 1. It is a custom made platform with a 250mm base (distance between motor axes) and a PX4 PixRacer flight controller. The flight controller is connected via a serial link to an onboard Intel i7 NUC computer where the computations are performed. The main sensing modality is a Primesense 1.08 RGBD camera. We experimented with a pan-tilt-unit initially to actuate the camera and thereby increase the degrees of freedom available for the view planing. However, due to weight limitations we had to settle for a staic mounting where the camera is tilted down an angle $\tau = 37°$. The vertical fields of view of the camera is $\alpha = 45°$. The total weight of the UAV including a 5000mAh battery is approximately 1.7kg. We use a an Optitrack motion capture

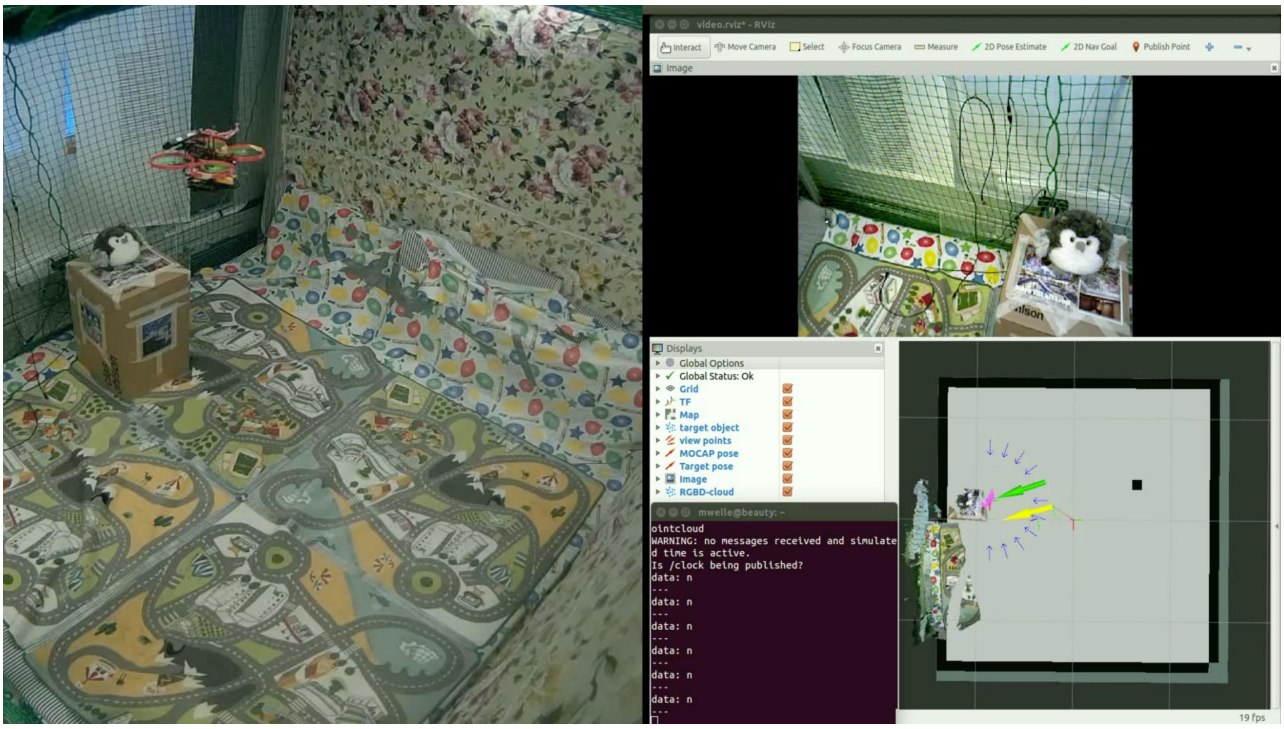


Fig. 3: A snapshot from one of the experiments. Left: A view of the UAV and the scene it looks at. Upper right: The current view from the camera. Lower right: The view plan and the raw data from the camera.

system. All experiments are performed inside a 3x3x3m cage with protective nets.

B. Generating the initial partial object

In [2], [3] we have shown how objects can be segmented out from a static background. We consider this output to be the input for our system. Because of the location for the experiments in this paper, a cage with mattresses on the floor, we are not able to let our wheeled robot generate this input. Instead we implemented a simple method for obtaining the input cluster. We first obtain the static background by executing a flight path with the UAV inside the cage and collecting RGBD data. Next, we place a new object in the environment. We use the UAV to acquire an RGBD snapshot of the environment. We segment the point cloud by removing all points below a certain height threshold, and we select the cluster of points closest to the camera. This simple segmentation method requires user input (i.e. the height at which to filter the point cloud). However, by placing the object on the same supporting object throughout the experiments this input is only needed once, while allowing us to remove the dependency on the mobile robot and focus on flying the drone, acquiring data and filtering it.

IV. METHOD

In this section we describe our approach to view planning and object modeling.

A. View planning

We approximate the object, which is only partially known, using a vertical cylinder. This is clearly a simplification but experiments show that it is a good initial model. Let r and h be the radius and height of this cylinder respectively. Figure 4 illustrates the cylinder and the way we calculate the view point candidates in cylindrical coordinates. From before we have that the vertical field of view is α and the downward tilt angle of the camera is τ. We want to keep the objects in the center of the view. We furthermore want to provide some margin of error in the positioning. We define this in terms af an angle β with which the field of view is reduced at each end. In our experiments we use $\beta = 10°$. We can calculate Z and R from the figure according to

$$Z = \frac{t_+(2rt_- + h)}{t_+ - t_-} \tag{1}$$

$$R = \frac{Z}{t_+} + r \tag{2}$$

where

$$t_- = tan(\tau - (\frac{\alpha}{2} - \beta)) \tag{3}$$

$$t_+ = tan(\tau + (\frac{\alpha}{2} - \beta)) \tag{4}$$

This defines a circle with radius R and located $Z + Z_0$ above the ground. We sample N view points on this circle evenly spread out along the circle. Given a 2-D map of the test environment, a simple collision check based on proximity is employed in order to eliminate view points that are not reachable for the UAV.

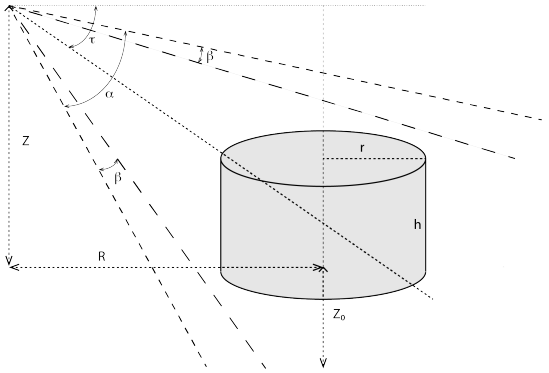


Fig. 4: View point calculation based on a cylindrical approximation of the object and trying to keep this cylinder in the center of the image with a minimum angular margin of β.

B. Plan execution

To execute the plan, the UAV checks its current position and moves autonomously to the nearest way point. The UAV reaches the way point when it is within some tolerance, T_p^1, in X, Y and Z as well as a heading tolerance in yaw, T_a^1. This is the starting point for the view acquisition, unless some of view points have been removed, in which case it moves from way point to way point to the closest end of the circle segment broken by obstacles. Then it starts to position itself with a tolerance of T_p^2 in X,Y,Z and T_a^2 tolerance in yaw. The UAV holds this position for up to $5sec$ before autonomously taking a snapshot of the RGBD-pointcloud. This procedure is repeated until snapshots from all reachable viewpoints are taken. The UAV continues to traverse the viewpoints back in a safe manner until it reaches its starting view-point after which it lands at the initial position. In our experiments we use $T_p^1 = 0.4m$, $T_a^1 = 50°$, $T_p^2 = 0.1m$ and $T_a^2 = 10°$,

C. Object modeling

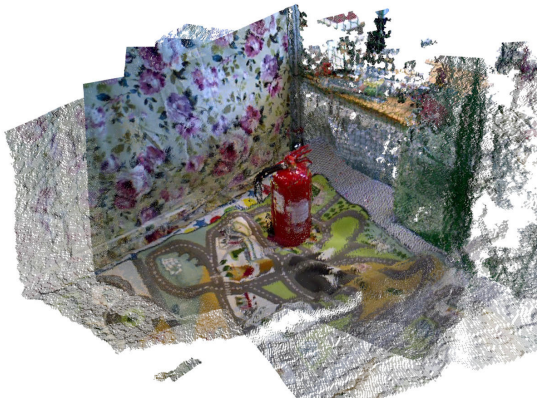
Following the view planning and plan execution (data acquisition step), a number of RGBD views $V = V_i$ containing the object of interest have been collected autonomously by the UAV. In addition, we also have access to the RGBD views $R = R_i$ collected before the object was introduced in the environment as reference. We perform a number of registration steps to align the object views and the reference views, after which we segment out the object from the registered point clouds. We use image features for the registration, with the assumption that enough texture can be found in the environment or on the object such that salient features can be extracted successfully. Note that we do not use the position fix from the motion capture system as initial guess for the registration, and instead start with the identity matrix as the initial solution. The motion capture system has very high accuracy, which is an unrealistic assumption for an initial solution in a real world environment (i.e. mobile robot odometry tends to be noisier) .

We first register the reference views $R = R_i$. For each image R_i, we extract SIFT images features [16], and for every pair R_i and R_j we compute feature correspondences using the SIFT descriptors. We augment each remaining feature with the depth value from the corresponding RGBD frame, and we perform a RANSAC [17] step to remove spatially inconsistent matches. The remaining matches are of the form (P_i, P_j) with $P_i \in R_i$ and $P_j \in R_j$, $P_i = (X, Y, Z)$ - coordinates in the camera frame of reference. Registering these views in a common frame of reference is done using image features in a least squares minimization framework. We define the transformations T_{Ri} and we solve the following minimization problem:

$$\min_{T_{Ri}, T_{Rj}} \sum_i \sum_j \|T_{Ri} \cdot P_i - T_{Rj} \cdot P_j\|^2 \tag{5}$$



(a)



(b)



(c)

Fig. 5: Object modeling results: a, b and c) show the registered frames V_i for three experiments.

Note that when acquiring the reference views R, the UAV is free to collect data without aiming at a specific object, which ensures a uniform and complete coverage of the



Fig. 6: Qualitative results of the object models built: (a) Ground truth models built on a turntable using [13]; (b) Results from the controlled experiments of [2]; (c) Uncontrolled experiments of [2]; and (d) Our object models.

environment. This coverage ensures that enough features and enough overlap is present between frames for a successful registration.

The next step is to register the object views $V = V_i$. This is more challenging, as the drone's path can be obstructed by obstacles such as walls, resulting in fewer frames and less information for the registration. Moreover, depending on the object introduced in the scene, the depth information can sometimes be unreliable, especially if the camera view axis incidental to the surface of the object is too oblique. As above, we are interested in finding the transformations T_{Vi} which bring the views $V = V_i$ into a common frame of reference. From our experiments, solving Eq. 5 for the set of views $V = V_i$, where less information is available sometimes results in failure. Instead we use the set of views R_i as reference, and keep the transforms computed earlier T_{Ri} fixed.

As before, we compute SIFT feature correspondences between image pairs V_i and V_j, augmented with depth information. Next, we compute SIFT feature correspondences between image pairs V_k and R_l. The minimization problem becomes:

$$\min_{T_{Vi}, T_{Vj}} \sum_i \sum_j \|T_{Vi} \cdot P_i - T_{Vj} \cdot P_j\|^2 + \\ + \sum_k \sum_l \|T_{Vk} \cdot P_k - T_{Rl} \cdot P_l\|^2 \quad (6)$$

In Eq. 6, we force the views V_i to register with each other and against the reference views R_i. Keeping the transformations T_{Ri} fixed is required, otherwise the registration of R_i might be diverge if the quality of data in V_i is too poor.

We perform a final registration step where we allow only small changes to the transforms T_{Vi} and T_{Ri}. This step fine-tunes the registration, and has the same shape as 6, except both T_{Vi} and T_{Ri} are varied by the optimizer. The results of the registration can be seen in Fig. 5.

For all registration steps we use the Ceres optimization engine [18], and the transforms (unless previously computed) are initialized with the identity matrix. Note that because we estimate transforms between all pairs of frames, the notion of "loop-closure" is implicitly taken into account in our pipeline.

Finally, we segment out the object model by taking a point cloud difference between the view point clouds V_i and the reference point cloud R_i. The results of the segmentation are shown in Fig. 6. We also perform a voxel-grid downsampling operation of the resulting object point cloud, thus removing some of the clutter and noise.

V. RESULTS

We perform experiments using five household items. Before each run, we first perform a flight pass with the UAV and stop at specified way points in the cage. This gives us the map we compare against, which would normally be provided by the mobile robot. We then place one of the objects in the

environment (in various positions, including corners), and we run the segmentation procedure described in Sec. III-B. Further, we collect data at the viewpoints computed and we extract one object model for each experiment.[1]

We compare our results qualitatively with the results obtained by [2], where a mobile robot was used to acquire views and build models of the objects autonomously. Two types of experiments were performed in [2]: *controlled* and *uncontrolled*. In the controlled experiments, the object was placed in an accessible area, such that the robot was able to navigate all around and collect additional views from all sides. In the uncontrolled experiments, the object was placed in more natural but less accessible locations, and the robot was only able to collect a few additional views. Finally, we also show ground truth point cloud models of the objects built using a turntable with the method described in [13].

The results are show in Fig. 6. We notice that the ground truth models are the sharpest, with the least amount of blur or noise, and with the textures clearly visible. Fig. 6 b) and c) show the progression when the objects are placed from more to less accessible locations. The degradation in quality can readily be observed when the robot can only take a few snapshots of the objects from further away. In contrast, our results in Fig. 6 d) are closer in quality to the ones of [2] in the controlled experiments, even though we conduct our experiments on a platform which is much more difficult to control. We note that our registration and modeling step performs successfully, and that the resulting model contains much more information than a single scan of the object would.

As the drone is not limited to the floor for navigation, we conclude that the pipeline proposed has the capability of augmenting a mobile robot system by safely navigating to previously inaccessible locations, collecting data reliably and using the data to create object models autonomously.

VI. CONCLUSION

In this paper we presented an end to end object modeling pipeline for an unmanned aerial vehicle (UAV). We explained how to generate a view plan, execute it to acquire data and how to use this data to build object models. Our experiments show that the quality of our models are close to those generated by a mobile robot in situations where the objects have been placed in easy to access locations as presented in current state of the art in object modeling [2]. To the best of our knowledge this is the first end to end solution for autonomous object modeling using a drone in indoor environments including view planning with collision avoidance.

In future work we want to relax the assumption of having positioning information from the motion capture system and we want to form the team between the wheeled mobile robot and the UAV which was the motivation for the work in the first place. We also want to investigate replacing the wheeled mobile robot all together with one or several drones performing modeling of indoor spaces.

[1]The data can be found online at goo.gl/gLn8D2

ACKNOWLEDGMENT

The work presented in this papers has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 (STRANDS), the Swedish Foundation for Strategic Research (SSF) through its Centre for Autonomous Systems and the Swedish Research Council (VR) under grant C0475401.

REFERENCES

[1] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J. Wyatt, D. Hebesberger, T. Körtner, *et al.*, "The strands project: Long-term autonomy in everyday environments," *arXiv preprint arXiv:1604.04384*, 2016.

[2] T. Fäulhammer, R. Ambruş, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 26–33, 2016.

[3] R. Ambruş, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 1854–1861, IEEE, 2014.

[4] W. R. Scott and G. Roth, "View Planning for Automated Three-Dimensional Object Reconstruction and Inspection," vol. 35, no. 1, pp. 64–96, 2003.

[5] Y. F. Li and Z. G. Liu, "Information entropy-based viewpoint planning for 3-D object reconstruction," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 324–337, 2005.

[6] K. Schmid, H. Hirschmueller, A. Domel, I. Grixa, M. Suppa, and G. Hirzinger, "View planning for multi-view stereo 3D Reconstruction using an autonomous multicopter," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, no. 1-4, pp. 309–323, 2012.

[7] X. Z. Peng, H. Y. Lin, and J. M. Dai, "Path planning and obstacle avoidance for vision guided quadrotor UAV navigation," *IEEE International Conference on Control and Automation, ICCA*, vol. 2016-July, pp. 984–989, 2016.

[8] W. Jing, J. Polden, P. Y. Tao, W. Lin, and K. Shimada, "View Planning for 3D Shape Reconstruction of Buildings with Unmanned Aerial Vehicles," vol. 2016, no. November, pp. 13–15, 2016.

[9] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics,"

[10] S. Lai, K. Wang, H. Qin, J. Q. Cui, and B. M. Chen, "A robust online path planning approach in cluttered environments for micro rotorcraft drones," *Control Theory and Technology*, vol. 14, no. 1, pp. 83–96, 2016.

[11] B. Englot and F. S. Hover, "Three-dimensional coverage planning for an underwater inspection robot," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1048–1073, 2013.

[12] M. Sheinin and Y. Y. Schechner, "The Next Best Underwater View," p. 32000, 2015.

[13] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "Rgb-d object modelling for object recognition and tracking," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 96–103, IEEE, 2015.

[14] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward object discovery and modeling via 3-d scene comparison," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2623–2629, IEEE, 2011.

[15] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense rgb-d maps," in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 178–185, IEEE, 2013.

[16] C. Wu, "Siftgpu: A gpu implementation of scale invariant feature transform (sift)," 2007.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[18] S. Agarwal, K. Mierle, *et al.*, "Ceres solver," 2013.

Deconvolutional Networks for Point-Cloud Vehicle Detection and Tracking in Driving Scenarios

Víctor Vaquero*, Ivan del Pino*, Francesc Moreno-Noguer, Joan Solà, Alberto Sanfeliu and Juan Andrade-Cetto

***Abstract*— Vehicle detection and tracking is a core ingredient for developing autonomous driving applications in urban scenarios. Recent image-based Deep Learning (DL) techniques are obtaining breakthrough results in these perceptive tasks. However, DL research has not yet advanced much towards processing 3D point clouds from lidar range-finders. These sensors are very common in autonomous vehicles since, despite not providing as semantically rich information as images, their performance is more robust under harsh weather conditions than vision sensors. In this paper we present a full vehicle detection and tracking system that works with 3D lidar information only. Our detection step uses a Convolutional Neural Network (CNN) that receives as input a featured representation of the 3D information provided by a Velodyne HDL-64 sensor and returns a per-point classification of whether it belongs to a vehicle or not. The classified point cloud is then geometrically processed to generate observations for a multi-object tracking system implemented via a number of Multi-Hypothesis Extended Kalman Filters (MH-EKF) that estimate the position and velocity of the surrounding vehicles. The system is thoroughly evaluated on the KITTI tracking dataset, and we show the performance boost provided by our CNN-based vehicle detector over a standard geometric approach. Our lidar-based approach uses about a 4% of the data needed for an image-based detector with similarly competitive results.**

I. INTRODUCTION

Autonomous driving (AD) is nowadays a reality. The main reasons for this success are twofold. On the one hand, research advances in related areas such as machine learning and computer vision are obtaining a high level of scene comprehension of the vehicle surroundings. On the other hand, new hardware and on-vehicle sensors are providing the community with enough data to develop new and robust perception algorithms as well as the ability to process them in real time.

However, there is still a long road until fully autonomous vehicles (AV) drive along totally free in our cities. Urban traffic is very challenging and dynamic, with numerous intervening elements like pedestrians, cyclists, other vehicles and even street furniture. To this end, accurate detection and tracking algorithms are elements of vital importance in AVs. These systems must be robust enough to recognise, understand, and act in response to any possible situation while assuring the safety of drivers, pedestrians, and other elements in our roads.

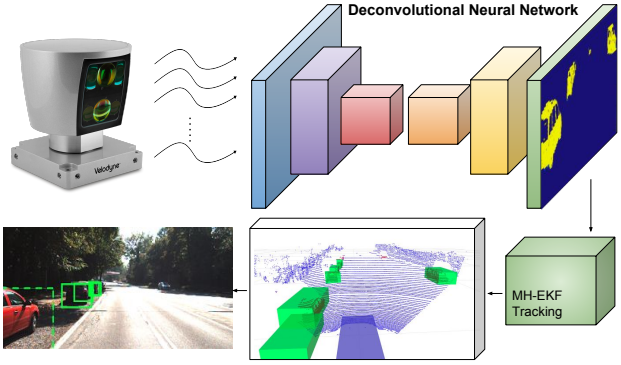


Fig. 1. We introduce a novel CNN-based vehicle detector on 3D range data. The proposed model is fed with an encoded representation of the point cloud and computes for 3D each point its probability of belonging to a vehicle. The classified points are then clustered generating trustworthy observations that are fed to our MH-EKF based tracker. Note: Bottom left RGB image is shown here only for visualization purposes.

With the advent of deep learning technologies, image-based scene understanding involving tasks such as object detection, semantic segmentation or motion capture have experienced an impressive performance and accuracy boost [1], [2], [3], [4]. However, image-based methods may suffer a high performance decrease in real driving scenarios under harsh environmental conditions e.g. heavy rain, snow, fog, or even night scenes. To avoid such situations and increase robustness, redundancy must be included in AVs. This is commonly tackled by creating autonomous perception systems that rely on other sensors such as radar or 3D lidar range-scanners.

Lidar sensors are specially suitable for AD purposes since they provide very accurate spatial information of the environment, are robust to hard climate conditions and their performance is almost independent on the illumination of the scene. Yet, deep learning methods deployed over 3D-lidar point clouds are far from the successful performance achieved on 2D-RGB images. This is mainly due to the computational burden introduced by the change in problem dimensionality as well as lack of annotated training data.

We present a robust and accurate vehicle detection and

*These authors contributed equally to this work. {vvaquero,idelpino}@iri.upc.edu.

The authors are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, 08028 Barcelona, Spain.

This work has been supported by the Spanish Ministry of Economy and Competitiveness projects ROBINSTRUCT (TIN2014-58178-R) and COLROBTRANSP (DPI2016-78957-R), by the Spanish Ministry of Education FPU grant (FPU15/04446), the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656) and by the EU H2020 project LOGIMATIC (H2020-Galileo-2015-1-687534). The authors also thank Nvidia for hardware donation under the GPU grant program.

tracking system that uses solely 3D lidar information as input. A sketch of the developed system is shown in Fig. 1. The main core of the presented approach is a tailored Fully Convolutional Network (FCN) [3] trained to detect vehicles from featured range and reflectivity representations of the 3D point cloud provided by a Velodyne 64-HDL sensor. Our FCN fulfils this task by performing a point-wise classification of whether each 3D point belongs to a vehicle or not. Positive samples are then clustered and 2D vehicle poses are obtained. This is performed by choosing the best fitting oriented bounding boxes (x, y, θ) over the external perimeter resulting after projecting the clusters to the ground plane. This 2D information is finally fed to a tracker based on a Multi-Hypothesis Extended Kalman Filter (MH-EKF) which, along with extracted 3D features such as the heigh of the corresponding cluster, provide the final results on 3D tracking.

We test our system over the Kitti tracking benchmark [5], where lidar-only methods are heavily penalised due to the image-based 2D evaluation measurements. However, we show the competitiveness of our approach and validate the hypothesis of using CNN-based lidar detectors against other geometrical methods.

II. RELATED WORK

Deep learning techniques, and more specifically Convolutional Neural Networks (CNNs), have demonstrated an outstanding performance in classical computer vision problems such as object classification [1], [6], detection [2], [7], and semantic segmentation [3]. However, CNNs have not yet deployed its potential over range lidar point clouds. We next review some approaches proposed to detect objects in such 3D sparse point clouds, and how Deep Learning methods are approaching this task.

Classical Object Detection in Lidar Point Clouds. Extensive literature exists about detecting objects in lidar-generated point clouds. Most common, are segmentation approaches that try to cluster closer points together and classify the resulting groups [8], [9], [10], [11]. These methods typically hold for both single (2D) and multi-layer (3D) lidars. For the latter, voting schemes are also used to vertically fuse single-layer clusters, obtaining part-based models of the objects [12], [13]. In autonomous driving applications, segmentation approaches previously tend to remove the ground-plane [14], [15], easing the clustering and classification steps. This heuristic is useful for the computation of the bounding box of the detected object, as will be shown in Section III-C. Other subtle clustering methods create graphs over pre-processed 3D voxels, exploiting their connectivity later in the classification step [16], [17], [18].

Recent methods scrutinize directly the 3D range scan space with sliding window approaches. *Vote3D* [19] for example encode the sparse lidar point cloud in a grid with different features such as mean and variance of intensity, a grade of occupancy, and other three different shape factors. The resulting representation is scanned in a sliding manner with 3D windows of different sizes and orientations, classifying the final candidates using SVMs and a voting scheme.

For the classification of point cloud clusters, the standard approach inherited from RGB algorithms, is to hand-craft features such as spin images, shape models or geometric statistics. Details of the most commonly used 3D features can be found in [20]. Learning procedures have also been used to obtain useful features via sparse coding, such as the work in [21].

Deep Learning for 3D Lidar Object Detection. Following the feature learning tendency, and aware of the success of CNN models, a few authors are applying convolutions over 3D lidar point clouds. For example, 3D convolutions, which are commonly used for video analysis (devoting its third dimension to the time variable) have been applied for 3D vehicle detection in [22]. However, due to the high dimensionality and sparsity of the data, deploying this methods over point clouds implies a high computational burden, which is not yet practical for on-line applications. Reformulating convolutions is a solution. In this way, *Vote3D* has been very recently extended in [23] by replacing SVMs with novel sparse 3D convolutions that act as voting weights for predicting the detection scores. Other methods design and apply sparse convolutions, such as [24], [25].

Another adopted approach is to obtain equivalent 2D representations of the 3D point cloud to apply the well know and optimized 2D convolution tools. In this way, [26] built a front view representation in which each element encodes a ground-measured distance and height of the 3D point. On top of this representation they apply a Fully Convolutional Network trained to predict the objectness of each point, and simultaneously perform a regression of the 3D bounding box of each vehicle. Similarly, we classify 3D points as belonging to vehicles or background although our image-like lidar representation includes direct information about range and reflectivity of the points and we use a more advanced deconvolutional architecture, as it will be shown in Sections III-A and III-B.

The very recent evolution of [26] combines their front view representation of the lidar information with a bird's eye view to generate accurate 3D bounding box proposals [27]. These are later fused with RGB images in a region-based fusion network, obtaining state of the art results in the detection challenge of the Kitti dataset. However, this method does not fulfil the lidar-only requirement that we impose in our work.

III. VEHICLE DETECTION & TRACKING

We reformulate the task of detecting vehicles in lidar point clouds as a per-point classification problem in which we want to obtain the probability of each sample to be a vehicle, therefore: $p(k|p_i)$, where $k \in \{\text{vehicle, no-vehicle}\}$; $i \in 1, ...N$, and $p_i \in \mathbb{R}^3$ represents each point of the point cloud $\mathcal{P}$ in the Euclidean space.

A. 2D Representation of Range Data

To efficiently exploit the successful deep convolutional architectures, we project our point cloud to an image-like representation through $G(\mathcal{P}) \in \mathbb{R}^{H \times W}$. This process is sketched in Fig. 2.

To obtain the transformation $G(\cdot)$, we first project the 3D Cartesian point cloud to spherical coordinates $sph(p_i) = \{\phi_i, \theta_i, \rho_i\}$. According to the Velodyne HDL-64 specifications, elevation angles θ are represented as a $H \in \mathbb{R}^{64}$ vector with a resolution $\Delta\theta$ of $1/3$ degrees for the upper laser rays and $1/2$ degrees the lower half respectively. Moreover, $G(\cdot)$ needs to restrict the azimuth field of view, $\phi \in [-40.5, 40.5]$ to avoid the presence of unlabelled vehicle points, as the Kitti tracking benchmark has labels only for the front camera viewed elements. The azimuth resolution was set to a value of $\Delta\phi = 0.18$ degrees according to the manufacturer, and hence lying in $W \in \mathbb{R}^{451}$. Each H, W pair encodes the range (ρ) and reflectivity of each projected point, so finally our input data representation lies in an image-like space $G(\mathcal{P}) \in \mathbb{R}^{64 \times 451 \times 2}$.

To get the equivalent ground-truth representation needed for the learning process, we use the Kitti tracklets, which are given in the RGB space. These tracklets are converted to 3D and $\mathcal{P}^*$ is obtained after labelling the inlier points. These ground-truth 3D class labels are also encoded as one channel in the $G(\cdot)$ image-like space, with pixels taking values of 1 for background and 2 for vehicles. Within the 'vehicle' class, we consider the associated Kitti classes for *car*, *van* and *truck*. Yet, the evaluation methods of the benchmark do not have into account *truck* classes, which may penalize our Kitti measured performance.

B. Deconvolutional Networks for Vehicle Detection

For the per-pixel vehicle classification task, we propose the deconvolutional architecture shown in Fig. 3, having in mind the recent success of these architectures. Here we disclose some of the key insights of our design.

To reduce the imbalance in the vertical and horizontal dimensions of the Velodyne representations and obtain more tractable intermediate feature maps, in the first convolutional layer we impose twice horizontal than vertical. Initial convolutional filter sizes are also designed according to the shape of the vehicles observed in the new representation, so that to obtain a receptive field consistent with it. Moreover, to address the disproportion between the number of samples of each class, we penalize misclassification of the positive vehicle samples with ω as seen in Eq. 2.

Since our contractive-expansive design could suppose an information bottleneck in the narrow layers, we introduce skip connections concatenating equivalent feature maps from both parts. These links help the learning process of the lower layers by back-propagating purer gradients from the upper parts. Finally, we state the classification problem at different resolutions of the network in order to obtain a direct and finer control of the learning process. This is done by including intermediate losses that guide the network faster to a correct

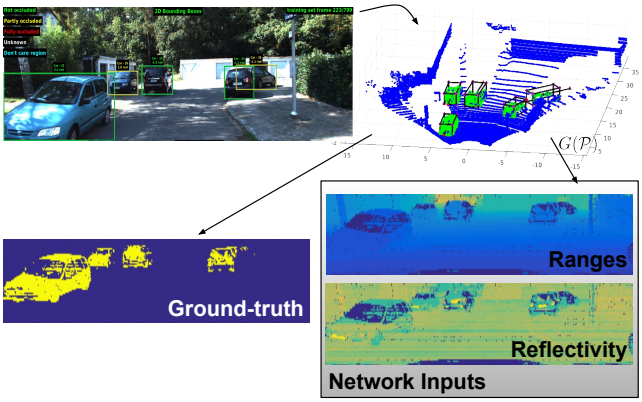


Fig. 2. To obtain a useful input for our CNN-based vehicle detector, we project the 3D point cloud raw data to a featured image-like representation containing ranges and reflectivity information by means of $\mathbb{G}(\cdot)$. Ground-truth for learning the proposed classification task is obtained by first projecting the image-based Kitti tracklets over the 3D Velodyne information, and then applying again $\mathbb{G}(\cdot)$ over the selected points.

solution, introducing new valuable gradients at these middle levels.

Hence, the network is trained via end-to-end back-propagation guided by the following loss function:

$$\mathcal{L}(\hat{\mathcal{Y}}, \mathcal{Y}) = \sum_{r=1}^{3} \lambda_r \mathcal{L}_r(\hat{\mathcal{Y}}_r, \mathcal{Y}_r), \tag{1}$$

where r represents the intermediate loss-control positions, λ_r are regularization weights for the loss at each resolution, and $\hat{\mathcal{Y}}, \mathcal{Y}$ are respectively the predictions and ground-truth classes at those resolutions.

In our approach, $\mathcal{L}_r$ is a multi-class Weighted Cross Entropy loss (WCE) [28], defined as:

$$\mathcal{L}_r^{WCE} = - \sum_{i,j,k}^{H_r, W_r, K_r} \omega(\mathcal{Y}_{i,j}) Id_{[\mathcal{Y}_{i,j}]} \log(\hat{\mathcal{Y}}_{i,j,k}), \tag{2}$$

where $Id_{[x']}(x)$ is an index function that selects the probability associated to the expected ground truth class and $\omega(k)$ is the previously mentioned class-imbalance regularization weight computed from the training set statistics.

C. Obtaining the Vehicle Bounding Boxes

The output $\hat{\mathcal{Y}}$ of the designed network is a matrix laying in the $G(\cdot) \in \mathbb{R}^{64x451}$ space where each pixel represents the probability of the corresponding 3D point to belong to a vehicle. In order to obtain the vehicle bounding boxes needed for the tracking and evaluation steps, we first apply the inverse transformation $G^{-1}(\hat{\mathcal{Y}})$ to the network output. In this way, we obtain the equivalent classified 3D point cloud $\hat{\mathbb{P}} \in \mathbb{R}^3$ that is finally clustered by means of an euclidean threshold. For each resulting cluster a set of features is extracted to build the EKF observation vector, such as the centroid, height and 'vehicleness' (calculated as the mean of the classification scores given by the network to that cluster). Clusters are then projected over a polar grid on the ground plane, accounting for each azimuth angle only the closest

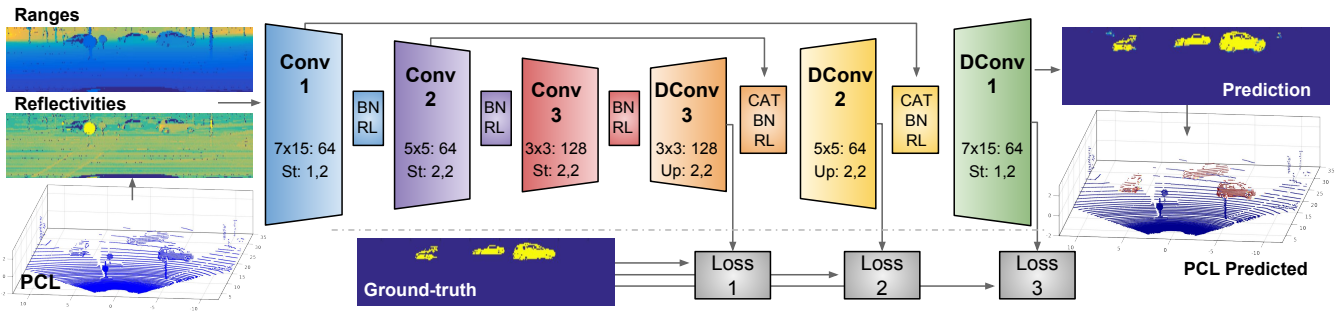


Fig. 3. Our network encompasses only convolutional and deconvolutional blocks followed by Batch Normalization (BN) and ReLu (RL) non-linearities. The first three blocks conduct the feature extraction step controlling, according to our vehicle detection objective, the size of the receptive fields and the feature maps generated. The next three deconvolutional blocks expanse the information enabling the point-wise classification. After each deconvolution, feature maps from the lower part of the network are concatenated (CAT) before applying the normalization and non-linearities, providing richer information and better performance. During training, three losses are calculated at different network points, as shown in the bottom part of the graph.

point to the sensor. The aim of this process is to get the external perimeter of each vehicle, which will ease the task of fitting a 2D oriented rectangular model to any of them.

The oriented bounding box fitting process consist on first performing an angular swept of bounding boxes in the $[-\frac{\pi}{4}, \frac{\pi}{4}]$ interval. For each one we cast simulated 2D Velodyne rays to obtain the geometrically equivalent impact over the boxes. The best 2D fitting box is then chosen as the one with the minimum mean square distance between the real vehicle detected points and the simulated ones. Finally we extract useful features for the vehicle observation vector of the tracker, such as width, length and centre of the 2D bounding box, as well as recover the 3D box using the previously calculated cluster height.

D. Lidar-based Vehicle Tracking

We implemented a multi-hypothesis extended Kalman filter (MH-EKF) for tracking bounding boxes according to a realistic motion model suited for wheeled vehicles in road environments. As vehicles transit on the road plane, the 2D bounding boxes (BB) are considered for tracking. Since the true vehicle dimension and centroid are not measurable through simple detections, we locate the BB origin at the closest visible corner, which is indeed measurable. For each BB we start a MH-EKF, which tracks its 2D position, orientation and velocity. The state vector is

$$\mathbf{x} = \begin{bmatrix}\mathbf{p}^\top & \theta & v & \rho\end{bmatrix}^\top = \begin{bmatrix}x & y & \theta & v & \rho\end{bmatrix}^\top \quad (3)$$

where $\boldsymbol{p} \triangleq (x, y)$ and $\boldsymbol{\pi} \triangleq (x, y, \theta)$ are the BB's position and pose, v is the linear velocity in the local x direction, and ρ is the inverse of the curvature radius (so that the angular velocity is $\omega = v\rho$).

Due to the limited geometrical information of the detected bounding boxes, we establish multiple hypotheses for the box motion: i.e, one moves along the main horizontal axis, and another one across it. Initially we assign uniform weights to all hypotheses $w_i = 1/N,\ i \in [1, \cdots, N]$. Each EKF estimation $\mathbf{x}_i$ evolves normally according to the motion model $\mathbf{x} \leftarrow f(\mathbf{x}, \mathbf{w}, \Delta t)$, described as

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + v_i \begin{bmatrix}cos(\theta_i)\\ sin(\theta_i)\end{bmatrix} \Delta t \quad (4)$$

$$\theta_i \leftarrow \theta_i + v_i \rho_i \Delta t \quad (5)$$

$$v_i \leftarrow v_i + w_v \quad (6)$$

$$\rho_i \leftarrow \rho_i + w_\rho\ , \quad (7)$$

where $\leftarrow$ represents a time-update; and the measurement model $\mathbf{y} = h(\mathbf{x}_i) + \mathbf{v}$, described as

$$\mathbf{y} = (\boldsymbol{\pi}_i \ominus \boldsymbol{\pi}_V) \ominus \boldsymbol{\pi}_S + \mathbf{v}\ . \quad (8)$$

In these models, $\mathbf{w} = (w_v, w_\rho)$ and $\mathbf{v}$ are white Gaussian processes, $\ominus$ is the subtractive frame composition, $\boldsymbol{\pi}_V$ is the pose of the own vehicle, which is considered known through simple odometry, and $\boldsymbol{\pi}_S$ is the sensor's mounting pose in the vehicle. The measurement $\mathbf{y} = (x_S, y_S, \theta_S)$ matches the result of the detection algorithm in sensor frame. At each new observation, the weights are updated according to the current hypothesis likelihood λ_i, that is,

$$\lambda_i = \exp(-\frac{1}{2}\mathbf{z}_i^\top \mathbf{Z}_i^{-1} \mathbf{z}_i) \quad (9)$$

$$w_i \leftarrow w_i \lambda_i, \quad (10)$$

where $\mathbf{z}_i = \mathbf{y} - h(\mathbf{x}_i)$ is the current measurement's innovation, and $\mathbf{Z}_i$ its covariances matrix. Weights are systematically normalized so that $\sum w_i = 1$. Finally, when a weight drops below a threshold τ, its hypothesis is discarded. A few observations after the initial detection only one hypothesis remains for each filter.

This basic scheme is modified with the management of the visible corners: in cases of partial occlusion or vehicle overtake, we may have to switch the initially detected corner (which may has gone out of sight) by the closest currently visible one. This is done by trivially updating the (x, y) states to the new visible corner, leaving all other states untouched.

IV. RESULTS

We measure the performance increase provided by our CNN-based lidar detector over the presented MH-EKF tracker in the Kitti Tracking benchmark. Additionally, we provide insights of the precision/recall obtained by our DeepLidar detector and a qualitative evaluation of the full system that can be seen in Fig. 4.

The Kitti tracking benchmark is composed of a training set of $8,000$ Velodyne scans grouped into 21 different sequences covering diverse urban environments. For these, 3D tracklets defined over the corresponding RGB images are provided. In addition, $11,095$ scans are given in a test set grouped into 29 sequences with no annotations provided. Velodyne data timestamps are not given for any of the sequences in the tracking benchmark. As our tracker integrates the observations with the vehicle odometry, we therefore had to create synthetic timestamps simulating the Velodyne data at 10Hz as specified by the manufactures.

A. Full Working System

We designed and trained our Deep Learning models using MatConvNet. Networks are initialized with the He's method [29] and use Adam optimization with the standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Data augmentation is done with a 50% chance by performing only horizontal flips in order to preserve the geometry properties of the lidar information. The training process is performed on a single NVIDIA K40 GPU for $200,000$ iterations with a batch size of 20 Velodyne scans per iteration. The learning rate is fixed to 10^{-3} during the first $150,000$ iterations after which, is halved. We select the imbalance regularizator ω as 25 and the loss regularizers λ_r as 1, 0.7 and 0.5 respectively.

For the clustering step, we group 3D points imposing a maximum distance of $1m$. After that, clusters with less than 25 points or with a radio below than $50cms$ are discarded. The remaining clusters and its respective bounding boxes are then converted to ROS format and serve as input observations for our tracker.

Each detected vehicle is assigned a bi-hypothesis MH-EKF: one hypothesis assumes motion along the longest rectangle dimension; the other across it. Each MH-EKF is set up as summarized in Tab. I, which shows (in order) the number of hypotheses, the pruning threshold, the initial means and sigmas of each hypothesis, and the process and measurement noises' sigmas. The orientation observation noise is set to the maximum possible error for a rectangle, $\frac{\pi}{2}$, and dynamically adjusted by a factor that depends on the model fitting error and the cluster dimensions:

$$c = k\frac{\sum (r_p - r_v)^2}{n(w+l)^2} \tag{11}$$

where r_p and r_v are the ranges of the real points and the virtual ones, n is the number of points, w and l are the width and length of the virtual rectangle, and k is a tuning parameter experimentally set to 100. All metric units (r_p, r_v, w, l) are expressed in meters.

TABLE I
PARAMETER SETUP FOR ALL MH-EKFS

param	value	units / comment
N	2	along and across
τ	0.001	
$\mathbf{x}_1$	$x_S, y_S, \theta_S, 0, 0$	m, m, rad, m/s, 1/m
σ_1	$2, 2, \pi/2, 20, 0.2$	m, m, rad, m/s, 1/m
$\mathbf{x}_2$	$x_S, y_S, \theta_S + \pi/2, 0, 0$	m, m, rad, m/s, 1/m
σ_2	$2, 2, \pi/2, 20, 0.2$	m, m, rad, m/s, 1/m
$\sigma_{\mathbf{w}}$	$0.5, 0.01$	m/s, 1/m
$\sigma_{\mathbf{v}}$	$0.9, 0.9, c\,\pi/2$	m, m, rad

B. Experiments

We first evaluate the performance of our point-wise convolutional vehicle detector. In order to avoid over-fitting and audit the generalization capacities of the proposed architecture, we perform a 4-fold cross-validation step during training. In this way, we train the same architecture selecting each time different sequences to compose a validation set of around $1,000$ samples in a manner that the *vehicle* vs *non-vehicle* points ratio in the resulting sets remains similar. Averaged results show that our detector is able to classify Velodyne 3D points from the validation sets with a mean precision of 82.3% and a recall of 87.6%. Notice that this measures are point-wise, and do not refer to the number of vehicles, but to the mean amount of points correctly classified for each scan. However this results demonstrates the capacity of the trained model to retain generalized information of vehicles according to our input representations, which enables to train the final model with the full $8,000$ samples of the Kitti training set.

In addition, we measure the contributions of our DeepLidar vehicle detector applied to the tracking task. For this purpose, we evaluate the full system performance with three different detection modules:

- *Geometric:* is our baseline detection approach which uses the full raw Velodyne information as input. It initially performs a ground floor removal, according to [14] and applies a clustering algorithm over the remaining points. Bounding boxes are then extracted as described in Section III-C, to obtain the final detections. However, as there is not trustworthy information about the vehicleness of the created clusters, additional geometric constraints are introduced, e.g. no track is created until a corner of the vehicle is identified.
- *DeepLidar:* is the proposed deep model trained over the full training set. We therefore show the results obtained over the testing dataset, which are provided by the Kitti evaluation server.
- *CNN-GT:* its aim is to set the upper bounds of the tracker capacities under ideally lidar vehicle detections. For that, it simulates the perfect output of our convolutional detector by using the ground-truth of our data representation as predictions. As no noise is introduced on the detection step with this approach, the threshold discarding clusters with less than 25 points is lowered to 4. It is only evaluated in the training set, as ground-truth is not provided for the test sequences.

TABLE II
QUANTITATIVE EVALUATION ON THE KITTI TRACKING BENCHMARK

	Geometric		DeepLidar		CNN-GT
	Train	Test	Train	Test	Train
Mostly Tracked (%)	7.4	10.6	-	18.5	44.5
Partly Tracked (%)	56.5	45.1	-	52.2	47.7
Mostly Lost (%)	35.9	44.3	-	29.4	7.8
Recall (%)	46.4	42.1	-	55.4	79.0
Precision (%)	44.1	37.5	-	63.8	73.9
False Alarm Rate	1.97	2.35	-	1.06	1.00
MOTA	-25.7	-38.9	-	15.5	41.9

The quantitative tracking results of the proposed system are shown in Table II. Since there is no single ranking criteria to evaluate the tracking task, we follow the Mostly-Tracked (MT), Partly-Tracked (PT) and Mostly-Lost (ML) evaluation measurements from [30], as we consider that it reflects better the contributions of the different detector schemas over the final tracking results. This criteria accounts as MT targets those that are successfully tracked for at least the 80% of its life span, whereas as ML the ones with less than 20%, and PT the rest. In addition we include the CLEARMOT *MOTA* metric [31]. It is commonly used due to its expressiveness, as it combines in one single criteria three sources of errors (False Negatives, False Positives and ID-Switches) over the number of ground truth objects. It reports a percentage between $(-\inf, 100]$, which takes negative values when the number of errors made by the tracker exceeds the number of total objects in the scene.

The importance of our detector is stated through the noticeable improvements with respect to the geometric baseline method in all the metrics. Our CNN configuration is able to reduce by almost 15% the ML targets, providing better target tracks, which is reflected as an increase of the MT and PT values. The difference of our *DeepLidar* approach with respect to the ideal *CNN-GT* detector is in fact understandable. Considering that there is no noise introduced by the ideal detector, there is no need for setting a minimum cluster size and therefore farther vehicles can be tracked, which directly reflects in a better MT and MOTA. On the other hand, the fact that this ideal system does not achieve perfect results is explained by the own Lidar technology (very far vehicles are impacted by very few points or not even impacted).

It is noteworthy to mention at this point that the evaluation measurements for the Kitty tracking benchmark are performed on 2D bounding boxes over RGB images. When working only with Lidar information we need to project our 3D tracked bounding boxes to 2D images, which results in lower image pixel-level accuracy and therefore penalise our Lidar-only systems. As a fact, Kitti RGB images contains almost $1.4M$ $(375 \times 1242 \times 3)$ colour samples. When compared to image tracking methods our CNN inputs only have $64 \times 451 \times 2$, which means that we perform the full vehicle detection and tracking pipeline with just a fraction of 4.13% over the total values of the RGB methods.

V. CONCLUSIONS

In this work we presented a full vehicle detection and tracking system based only on 3D lidar information. It combines a convolutional neural network performing a point-wise vehicle detection, with a multi-object tracker. Our CNN-based detector classifies each 3D laser point as belonging to a vehicle or not by using a featured 2D-lidar representation which involves both range and reflectivity information. The resulting positively classified points are then grouped together to create identification hypotheses, and fed to a multi-hypothesis Extended Kalman Filter to track their motion. We evaluated our system on the KITTI tracking dataset, showing that the inclusion of the CNN-based detection module improves systematically the whole system performance.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 91–99.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3431–3440.

[4] V. Vaquero, G. Ros, F. Moreno-Noguer, A. M. Lopez, and A. Sanfeliu, "Joint coarse-and-fine reasoning for deep optical flow."

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 3354–3361.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[7] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Int. Conf. Learning Representations (ICLR)*, 2014.

[8] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2007, pp. 3402–3407.

[9] V. Vaquero, E. Repiso, A. Sanfeliu, J. Vissers, and M. Kwakkernaat, "Low cost, robust and real time system for detecting and tracking moving objects to automate cargo handling in port terminals," in *2nd Iberian Robotics Conf.*, ser. Adv. Intell. Syst. Comput., vol. 418, 2015, pp. 491–502.

[10] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2011, pp. 2798–2805.

[11] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppe, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe, *et al.*, "Moving object detection with laser scanners," *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.

[12] O. M. Mozos, R. Kurazume, and T. Hasegawa, "Multi-part people detection using 2D range data," *Intl. Journal of Social Robotics*, vol. 2, no. 1, pp. 31–40, 2010.

[13] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3D range data," in *AAAI Conf. Artif. Intell. (AAAI)*, 2010, pp. 1625–1630.

[14] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[15] A. Teichman, J. Levinson, and S. Thrun, "Towards 3D object recognition via classification of arbitrary object tracks," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2011, pp. 4034–4041.

[16] D. Z. Wang, I. Posner, and P. Newman, "What could move? finding cars, pedestrians and bicyclists in 3D laser data," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2012, pp. 4038–4044.

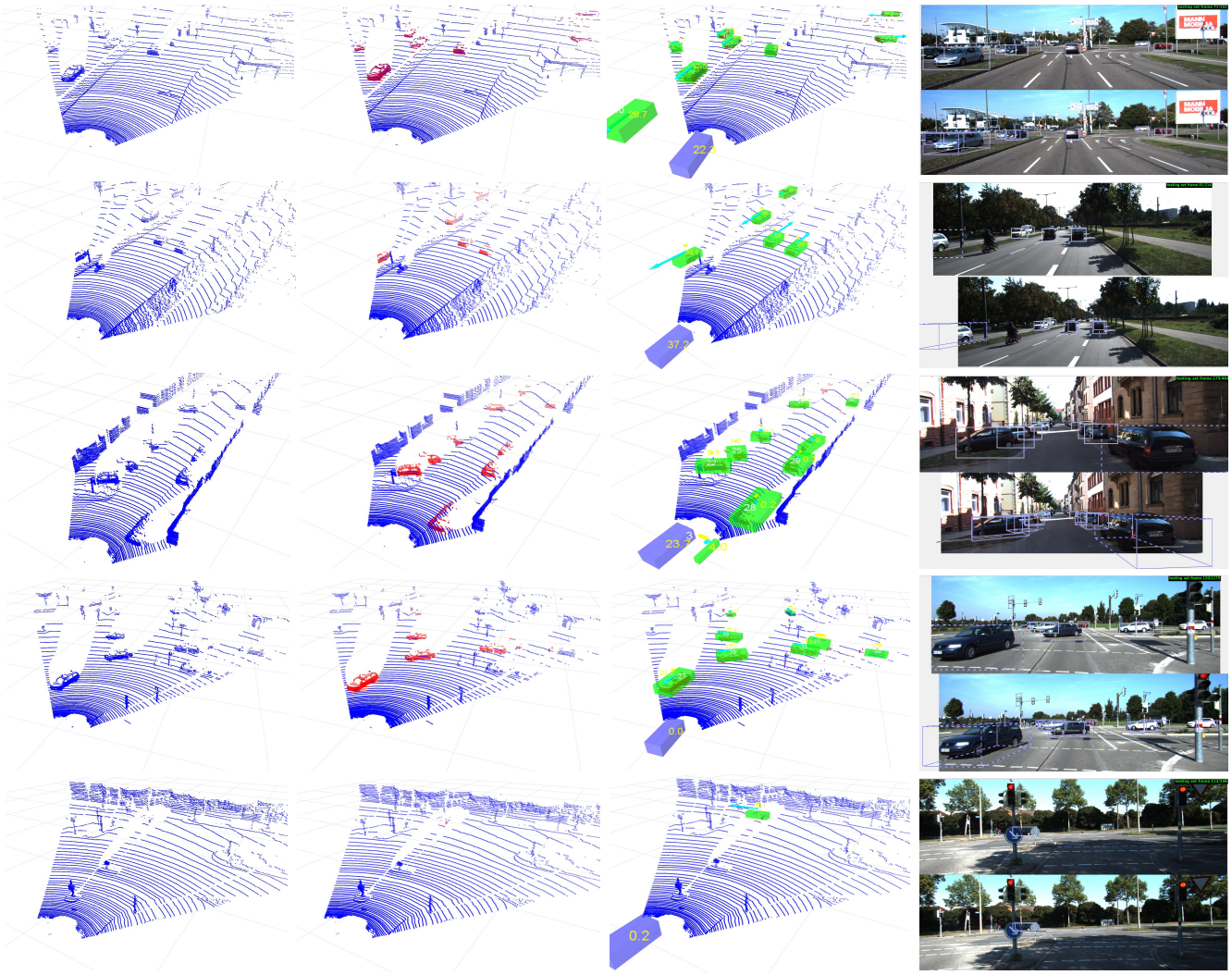


Fig. 4. Qualitative results of our system. All images are taken from the testing set of the Kitti Tracking benchmark, so none of them were previously seen by our Deep Lidar detector. In columns, images show the raw input point cloud, the Deep detector output, the final tracked vehicles and the RGB projected bounding boxes submitted for evaluation. Note how despite the scarce information provided by the lidar, our system is able to detect (red coloured points) and track (green boxes) vehicles in complex urban environments, even when they are partially occluded (bottom row).

[17] R. Triebel, J. Shin, and R. Siegwart, "Segmentation and unsupervised part-based discovery of repetitive objects," in *Robotics: Science and Systems (RSS)*, 2010, pp. 65–72.

[18] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 2027–2034.

[19] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection." in *Robotics: Science and Systems (RSS)*, 2015.

[20] J. Behley, V. Steinhage, and A. B. Cremers, "Performance of histogram descriptors for the classification of 3D laser range data in urban environments," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2012, pp. 4391–4398.

[21] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for classification of outdoor 3D scans," in *Australasian Conf. Robotics Automat. (ACRA)*, 2013.

[22] B. Li, "3D fully convolutional network for vehicle detection in point cloud," *arXiv preprint arXiv:1611.08069*, 2016.

[23] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE Int. Conf. Robotics Autom. (ICRA)*, 2017.

[24] V. Jampani, M. Kiefel, and P. V. Gehler, "Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 4452–2261.

[25] B. Graham, "Sparse 3D convolutional neural networks," in *British Machine Vision Conf.*, 2015, pp. 150.1–150.9.

[26] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Robotics: Science and Systems (RSS)*, 2016.

[27] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[28] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe, "Training constrained deconvolutional networks for road scene semantic segmentation," *arXiv preprint abs/1604.01545*, 2016.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[30] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid-boosted multi-target tracker for crowded scene," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2953–2960.

[31] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, pp. 1–10, 2008.

Acting Thoughts: Towards a Mobile Robotic Service Assistant for Users with Limited Communication Skills

F. Burget* L.D.J. Fiederer* D. Kuhner* M. Völker* J. Aldinger* R.T. Schirrmeister C. Do
J. Boedecker B. Nebel T. Ball W. Burgard

***Abstract*— As autonomous service robots become more affordable and thus available also for the general public, there is a growing need for user friendly interfaces to control the robotic system. Currently available control modalities typically expect users to be able to express their desire through either touch, speech or gesture commands. While this requirement is fulfilled for the majority of users, paralyzed users may not be able to use such systems. In this paper, we present a novel framework, that allows these users to interact with a robotic service assistant in a closed-loop fashion, using only thoughts. The brain-computer interface (BCI) system is composed of several interacting components, i.e., non-invasive neuronal signal recording and decoding, high-level task planning, motion and manipulation planning as well as environment perception. In various experiments, we demonstrate its applicability and robustness in real world scenarios, considering fetch-and-carry tasks and tasks involving human-robot interaction. As our results demonstrate, our system is capable of adapting to frequent changes in the environment and reliably completing given tasks within a reasonable amount of time. Combined with high-level planning and autonomous robotic systems, interesting new perspectives open up for non-invasive BCI-based human-robot interactions.**

I. Introduction

For patients with heavily impaired communication capabilities, such as severly paralyzed patients, their condition forces them to constantly rely on the help of human caretakers. Robotic service assistants can re-establish some autonomy for these patients, if they offer adequate interfaces and possess a sufficient level of intelligence. Generally, an intelligent system requires adaptive task and motion planning modules to determine appropriate task plans and motion trajectories for the robot, that implement a task in the real world. Moreover, it requires a perception component, e.g., to detect objects of interest or to avoid accidental collisions with obstacles. Typically used interfaces, such as haptic (buttons), audio (speech) or visual (gesture) interfaces, to command the robotic system are intuitive and easy options for healthy users, but difficult to impossible to use for paralyzed individuals.

In this paper, we present a novel framework, schematically depicted in Fig. 1, that allows closed-loop interaction between users with minimal communication capabilities and a robotic service assistant. To do so, we record neuronal activity elicited in the human brain, the common origin of all types of communication, with an electroencephalography (EEG) system. Furthermore, we adopt a convolutional neural network approach for online decoding of neuronal activity, in order to allow users to navigate through a graphical user interface (GUI) provided by a high-level task planner. The set of feasible actions displayed in the GUI, depends in turn on the current state of the world, which is stored in a central knowledge base and continuously updated with information provided by the robot and a camera perception system. Once a task has been selected, it is decomposed into a sequence of atomic actions by the high-level planner. Subsequently, each action is resolved to a motion for the mobile manipulator using low-level motion and manipulation planning techniques. In the following, the individual components shown in Fig. 1 will be described in detail, before presenting a quantitative evaluation of the overall system regarding its performance.

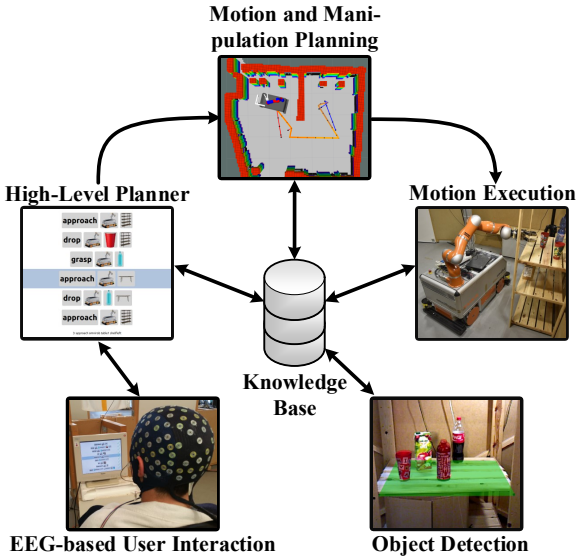


Fig. 1. Framework unifying decoding of neuronal signals, high-level task planning, low-level motion and manipulation planning, scene perception with a centralized knowledge base at its core. Intuitive goal selection is provided through an adaptive graphical user interface.

II. Related Work

Multiple previous studies have focused on robotic systems assisting people with disabilities. For example, Park *et al.* [1] implemented a system for the autonomous feeding of yogurt to a person. Chung *et al.* [2] focused on autonomous drinking which involved locating the drink, picking it up and bringing

* These authors contributed equally to the work. Authors are with the Department of Computer Science and Faculty of Medicine, University of Freiburg, Germany. {burgetf, kuhnerd, do, aldinger, jboedeck, nebel, burgard}@informatik.uni-freiburg.de and {lukas.fiederer, martin.voelker, robin.schirrmeister, tonio.ball}@uniklinik-freiburg.de. This research was supported by the German Research Foundation (DFG, grant number EXC 1086) and grant BMI-Bot by the Baden-Württemberg Stiftung.

it to the person's mouth. Using a hybrid BCI and head movement control, Achic *et al.* [3] studied a setup with a moving wheelchair and an attached robotic arm. None of these systems used pure BCI control. In contrast, Wang *et al.* [4] used a motor imagery BCI with three classes to achieve low-level control of a robotic arm. More relevant, Schröer *et al.* [5] developed a robotic system which receives a BCI command from a user and autonomously assists the user in drinking from a cup. However, this approach only considers a single object and a fixed-base manipulator. More recently, Muelling *et al.* [6] presented a shared-control approach to assistive robotics, albeit focused on invasive BCIs. Nonetheless, their approach could be combined with the high-level planning approach presented in our work.

In these applications, robust decoding of brain signals is required. Inspired by the successes of deep convolutional neural networks (ConvNets) in computer vision [7], [8] and speech recognition [9], [10], deep ConvNets have recently been applied more frequently to EEG brain-signal decoding. Deep ConvNets were already applied to decoding tasks useful for building brain-computer interfaces. Lawhern *et al.* [11] used a deep ConvNet to decode P300 oddball signals, feedback error-related negativity and two movement-related tasks. When evaluated cross-subject, i.e., trained on some subjects and evaluated on others, their ConvNet yields competitive accuracies compared with widely-used traditional brain-signal decoding algorithms. Tabar and Halici [12] used a ConvNet combined with a convolutional stacked autoencoder to decode motor imagery within-subject, yielding better accuracies than several non-ConvNet decoding algorithms. Schirrmeister *et al.* [13] used a shallow and a deep ConvNet to decode both motor imagery and motor execution within-subject, reaching or slightly surpassing the accuracies of the widely used EEG motor-decoding algorithm *filter bank common spatial patterns* [14]. Bashivan *et al.* [15] used a ConvNet trained on fourier-transformed inputs to estimate mental workload. In addition to the work on evaluating ConvNet decoding accuracies, ConvNet visualization methods allow us to get a sense of what brain-signal features the network is using [13], [15], [16]. Taken together, these advances make deep ConvNets a viable alternative for brain-signal decoding in brain-computer interfaces. Still, to our knowledge, online control with deep ConvNets has not yet been reported for an EEG-based brain-computer interface.

III. Online Decoding of Neuronal Signals

The system at hand is developed to control more complex scenarios than the ones considered in previous work. Particularly, we consider scenarios involving manipulation of objects as well as human-robot interaction. Feasible goals are determined by our GUI which is controlled by directional commands. As reliable classification of brain signals into navigation directions cannot yet be achieved directly with non-invasive BCIs, we used a deep ConvNet approach for decoding of multiple mental tasks from EEG (Schirrmeister *et al.* [13]). This approach introduces a hybrid network, combining a deep ConvNet with a shallower ConvNet architecture. The deep part consists of 4 convolution-pooling blocks using exponential linear units (ELU) [17] and max pooling, whereas the shallow part uses a single convolution-pooling block with squaring non-linearities and mean pooling. Both parts use a final convolution with ELU to produce output features. These features are then concatenated and fed to a final classification layer. We trained the ConvNet to decode five mental tasks: right hand finger and both feet toe movements, object rotation, word generation and rest. These mental tasks evoke discernible brain patterns and are used as surrogate signals to control the GUI. *Offline* training was done with a cropped training strategy using shifted time windows within the trials as input data [13].

From our experience it is important to train the BCI decoder and subjects in an environment that is as close as possible to the real application environment to avoid pronounced performance drops. Therefore, we designed a gradual training paradigm within the high-level planner GUI where the displayed environment, timing and actions are identical to those of the real control task. The training paradigm proceeds as follows: We first train each subject *offline* using simulated feedback. Subjects are aware of not being in control of the GUI. The mental tasks are cued using grayscale images presented for 0.5 s in the center of the display. At all times a fixation circle is displayed at the center of the GUI and the subject is instructed to fixate on it to minimize eye movements. After a random time interval of 1-7 s the fixation circle is switched to a disk for 0.2 s, which indicates the end of the mental task. At the same time the GUI action (*go up, go down, select, go back, rest*) corresponding to the cued mental task is performed to update the GUI. To keep training realistic we include a 20 % error rate, i.e., on average every fifth action is erroneous. We instruct the subjects to count the error occurrences to assert their vigilancy. This offline data is used to train the individual deep ConvNets. Then, the subjects do *online* training by performing the decoded mental tasks in the GUI. Finally, we stop cueing the mental tasks. To evaluate the performance of the BCI control, we let the subjects create instructed high-level plans in the GUI. These tasks are then executed by a simulated robot or the real mobile manipulator, when available. To provide more control over the mobile manipulator and enhance the feeling of agency, subjects have to confirm the execution of every planned action and can interrupt the chain of actions at any moment during their execution. BCI decoding accuracies for the label-less instructed tasks are assessed by manually rating each decoding based on the instructed task steps. Statistical significance of the decoding accuracies were tested using a conventional permutation test with 100 k random permutations of the labels (i.e., p-value is the fraction of label permutations that would have led to better or equal accuracies than the accuracy for the original labels).

IV. High-Level Goal Formulation Planning

We use domain independent planning to derive the required steps for reaching a desired high-level goal in a

complex task. The user can formulate a high-level goal without knowledge of the internal representation of objects in the planning system and the exact capabilities of the robot. This is achieved by an intuitive graphical user interface, where the object parameters of the goal are specified by incrementally refining the objects by referring to their *type*, e.g., "cup" or *attributes*, e.g., "content = apple-juice".

Domain independent planning identifies a sequence of actions that transforms the current world state into a state satisfying a goal condition. A planning task consists of: (i) a planning domain describing static components such as the object type hierarchy and the available actions and (ii) a problem instance describing the objects present in the world and their current state, as well as a goal description. While the current state of the objects can be extracted from the knowledge base, the goal has to be chosen in the GUI.

A restricted vocabulary is shared between the user and the planning system. Objects or sets of objects are identified by creating *referring expressions* to them composed of *shared references* built on this vocabluary [18]. We briefly describe the relevant aspects of our previous work in this area [19]. In general, a *referring expression* ϕ is a logical formula with a single free variable. ϕ *refers* to an object o if $\phi(o)$ is valid. E.g., the reference $\phi(x) \equiv cup(x) \wedge contains(x, water)$ refers to all cups containing water. We restrict ourselves to references that are simple conjunctions of facts, which is not only preferable for computational reasons, but also allows us to incrementally refine references by adding constraints. For example, adding $contains(x, water)$ to $cup(x)$, restricts the set of all cups to the set of cups containing water.

We distinguish between three types of fundamental object references: *individual references*, *typename references* and *relational references*. *Individual references* are identified by name, such as the "omniRob" robot. *Typename references* can be identified by the name of their type. While we cannot refer to the cups in our scenario directly, we can refer to an unspecific cup. *Relational references* are encountered when objects can be referred to via predicates in which they occur as an argument. The relations in our scenario are object attributes. For example, the *content* of the cup is used to clarify which cup is meant. These object references are used to create references to goals. We start defining goals with the action that achieves it as we found that this is most natural to the user, e.g., $put(x, y) \wedge cup(x) \wedge shelf(y)$. After the initial selection of a goal type (e.g., *drop*) it is necessary to determine the objects for all parameters of the goal predicate or action. These parameters are refined by constraining the previous choice until the argument is either determined uniquely (i.e., it is impossible to constrain the argument further) or the user declares that any remaining option is acceptable. We exclude unreachable goals, but we allow for goals that can only be achieved after a sequence of preceding actions (e.g., *drinking water* could require to fetch a cup, bring it to the patient, fetch a bottle and pour the water into the cup in order to be executed). The goal that is determined by the selection process of the GUI is then passed to a custom domain independent planner.

V. Robot Motion Generation

For generating paths for the mobile base, we apply the sampling-based planning framework *BI²RRT** [20]. Given a pair of terminal configurations, it performs a bidirectional search using uniform sampling in the configuration space until an initial sub-optimal solution path is found. This path is subsequently refined for the remaining planning time, adopting an informed sampling strategy, which yields a higher rate of convergence towards the optimal solution. Execution of paths is implemented via a closed-loop joint trajectory tracking algorithm using robot localization feedback.

To realize pick, place, pour and drink motions efficiently, we adopt a probabilistic roadmap planner approach [21]. The planner uses a graph of randomly sampled task poses (end-effector poses), which are connected by edges. To find a plan between two poses, the planner connects both poses with the roadmap graph and uses the A* algorithm to find an optimal path between the start and goal pose. The execution of the plan maps the task space velocity commands to joint velocity commands by employing a task space motion controller. We sample random poses around the object to determine grasp motions. For dropping objects we extract horizontal planes from the camera's point cloud and sample poses above those planes to find a suitable drop location.

VI. Implementation Details

Implementation of our framework in the real world requires several components, such as neuronal signal decoding, scene perception, knowledge base operations as well as symbolic and motion planning, to run in parallel. Therefore, we distributed the computation across a network of 7 computers, communicating among each other via ROS. The decoding of neuronal signals has four components. EEG measurements are performed using *Waveguard EEG* caps with 64 electrodes and a *NeurOne* amplifier in AC mode. Additionally, vertical and horizontal EOGs, EMGs of the four extremities and ECG's are recorded. For recording and online-preprocessing, we used BCI2000 and Matlab. We then transferred the data to a GPU server where our deep ConvNet classified the data into 5 classes. The high-level planner GUI consists of a back- and front-end. The back-end of the GUI uses the *Fast Downward* planner [22] to iteratively build goal references and to find symbolic plans for the selected goal. As the planning time is not crucial for the performance of our system, we used *Fast Downward* with a basic configuration in our experiments. The central knowledge base is implemented as a ROS node, which is able to store objects with arbitrary attribute information. All changes in the knowledge base automatically trigger updates of the front-end, unexpected ones interrupt the current motion trajectory execution. Finally, we used *SimTrack* [23] for object pose detection and tracking.

VII. Experiments

To evaluate our framework, we consider the environment schematically depicted in Fig. 2, containing two shelves and a table as potential locations for manipulation actions. The

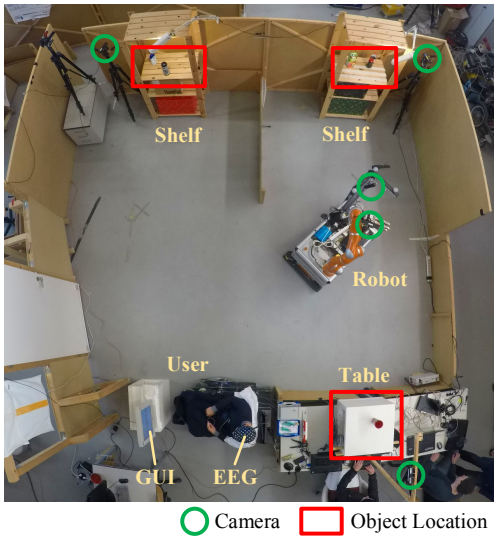


Fig. 2. Experimental environment: Two shelves and a table can be considered by the robot for performing manipulation actions. Five RGBD sensors observe the environment. A human operator selects a goal using EEG control and the high-level planner GUI.

user sits in a wheelchair in front of a screen, displaying the graphical interface of the high-level planner. The robot used in the experiments is the *omniRob* omni-directional mobile manipulator platform by KUKA Robotics, which is composed of 10 degrees of freedom (DOF), i.e., 3 DOF for the mobile base and 7 DOF for the manipulator. Additionally, the *Dexterous Hand 2.0* by Schunk is attached to the manipulator's flange and used to perform grasping and manipulation actions. The tasks we considered in our experiments required the robotic system to autonomously perform the following actions: *drive from one location to another*, *pick up an object*, *drop an object (on a shelf or table)*, *pour liquid from a bottle into a cup*, *supply a user with a drink*. Moreover, we use a perception system composed of five RGBD cameras. Three of them are statically mounted at the shelves and the table, in order to observe the scene and to report captured information to the knowledge base. The other two cameras are carried by the robot on-board. The first one is located at the mobile base and used to perform collision checks in manipulation planning. The second camera is mounted at the robot's end-effector and used for tasks involving physical human-robot interaction. A demonstration of our framework can be found in the accompanying video: http://www.informatik.uni-freiburg.de/~burgetf/ecmr17/.

A. Online Decoding of Neuronal Signals

We evaluated the BCI control setup with four healthy subjects (S1-4, all right-handed, three females, aged 26.75±5.9). At the time of writing the validation, S4 was still in progress and no validation with the mobile manipulator was performed. In total, 52 runs have been recorded (20 with the real robot) where the subjects executed various instructed high-level plans. For 32 runs, we used simulated feedback from the GUI in order to generate a significant amount of data for the evaluation. The performance of the BCI decoding during these runs was assessed using video

TABLE I

AGGREGATED MEAN±STD RESULTS FOR 52 BCI CONTROL RUNS (EXP. VII-A), * P-VALUE $< 10^{-6}$

	Runs #	Accuracy* [%]	Time [s]	Steps #	Path Optimality [%]	Time/Step [s]
S1	18	84.1±6.1	125±84	13.0±7.8	70.1±22.3	9±2
S2	14	76.8±14.1	150±32	10.1±2.8	91.3±12.0	9±3
S3	17	82.0±7.4	200±159	17.6±11.4	65.7±28.9	11±4
S4	3	63.8±15.6	176±102	26.3±11.2	34.5±1.2	6±2
	52	76.7±9.1	148±50	16.7±7.1	65.4±23.4	9±2

recordings of interactions with the GUI. We rated GUI actions as correct if they correspond to the instructed path and incorrect otherwise. Actions which are necessary to remediate a previous error are interpreted as correct if the correction is intentionally clear. Finally, we rated *rest* actions as correct during the (simulated) robot executions, incorrect if the next robotic action had to be initialized and ignored them during high-level plan creation. For evaluation, five metrics have been extracted from the video recordings: (i) the accuracy of the control, (ii) the time it took the subjects to execute a high-level plan, (iii) the number of steps used to execute a high-level plan, (iv) the path optimality, i.e., the ratio of the steps used to the minimally possible number of steps, and (v) the average time per step. We summarized the results in Table I. In total, a 76.67 % correct BCI control was achieved, which required 9 s per step. Selecting a plan using the GUI took on average 148 s and required the user to perform on average 16.74 steps in the GUI of the high-level planner. The path formed by these steps is on average 34.6 % away from the optimal path. The decoding accuracy of every subject is significantly above chance ($p < 10^{-6}$).

The subject-averaged EEG data used to train the hybrid ConvNets and the decoding results of the train/test transfer are visualized in Fig. 3. In Fig. 3(a) we show the signal-to-noise ratio (SNR) of all 5 classes $\mathcal{C}$ of the labeled datasets. We define the SNR for a given frequency f, time t and electrode e as

$$\mathrm{SNR}_{f,t,e} = \frac{\mathrm{IQR}\left(\{\mathrm{median}\left(\mathcal{M}_i\right)\}\right)}{\mathrm{median}\left(\{\mathrm{IQR}\left(\mathcal{M}_i\right)\}\right)} \quad i \in \mathcal{C},$$

where $\mathcal{M}_i$ corresponds to the set of values at position (f, t, e) of the i-th task, with $|\mathcal{M}_i|$ being the number of repetitions. median$(\cdot)$ and IQR$(\cdot)$ is the median and interquartile range (IQR), respectively. The upper part describes the variance of the class medians, i.e., a large variance means more distinguishable class clusters and a higher SNR. The denominator describes the variance of values in each class, i.e., a lower variance of values results in a higher SNR. The low SNR in EMG channels shows that the subjects did not move during the tasks.

The decoding accuracies achieved on the test data after initial training of the ConvNets are visualized in Fig. 3(b). To further support the neural origin of the BCI control signals, Fig. 3(c) shows physiologically plausible input-perturbation network-prediction correlation results (see [13] for methods).

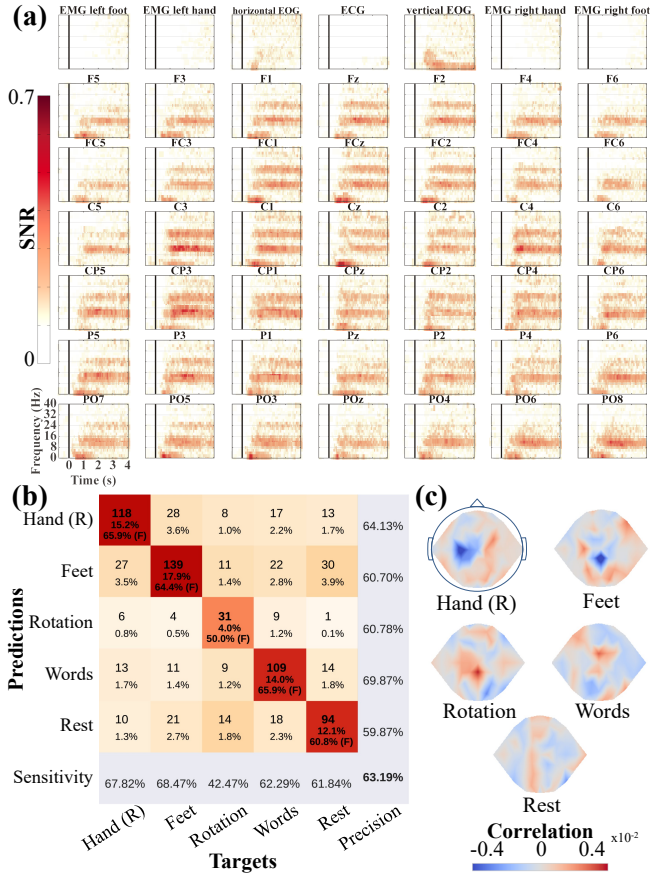


Fig. 3. EEG data and decoding results. **(a)** SNR of the first 4 s of data used to train the hybrid ConvNet. Highest SNR can be observed in the alpha (7-14 Hz) and lower beta (16-26 Hz) bands. These frequency bands are robust markers of task related mental tasks. Note that the non-EEG channels (top row) were withheld from the ConvNets at any time and are displayed as negative control. Not all channels are displayed because of space constraints. **(b)** Confusion matrix of decoding accuracies for the train/test transfer. Accuracies are well above the theoretical chance level of 20 %. **(c)** Topographically plausible input-perturbation network-prediction correlation maps in the alpha (7-13 Hz) frequency band. For details on the visualization technique we refer the reader to [13].

B. Fetch and Carry Task

The first experiment, considering the use of the real robot, evaluates the complete system in fetch-and-carry tasks. The goal was to transfer an object from one location to another, e.g., from a shelf to the table, using the robot. To fulfill such tasks the robot typically needs to execute four subtasks: *approach* object location, *grasp* object, *approach* other location, *drop* object. The user was instructed to select a predefined goal using the EEG-controlled high-level planner. Moreover, we selected a random initial placement for the objects in each run, in order to cover different environment states. The experiment was repeated ten times by the user. Table II shows the averaged results for the experiment. The second column indicates the overall number of desired action calls, as scheduled by the high-level planner, as well as the number of calls actually performed. The third to fifth columns represent the success rate, mean and standard deviation for the runtime of actions, respectively. Note, that the number of scheduled and actually executed actions

TABLE II

AGGREGATED RESULTS FOR 10 RUNS (EXP. VII-B)

Actions	# Executions (# Scheduled)	Success Executions [%]	Runtime [s] Mean	Std
Grasp	10 (10)	90.0	37.56	4.62
Drop	9 (10)	89.0	34.13	5.75
Approach	19 (20)	100.00	33.05	18.48
Total	38 (40)	94.74	34.42	14.02

TABLE III

AGGREGATED RESULTS FOR 10 RUNS (EXP. VII-C)

Actions	# Executions (# Scheduled)	Success Executions [%]	Runtime [s] Mean	Std
Grasp	34 (30)	91.0	40.42	10.31
Drop	30 (30)	97.0	37.59	4.83
Approach	80 (80)	100.0	20.91	7.68
Pour	10 (10)	100.0	62.90	7.19
Drink	13 (10)	77.0	57.10	8.20
Total	167 (160)	95.86	32.46	15.51

might differ for two reasons. A number of executed calls, lower than the scheduled ones, indicates that a previous action step has failed to succeed and plan recovery was not possible. On the other hand, a higher number of executed calls indicates that the user was able to achieve plan recovery by commanding a repetition of the failed action. Moreover, we recorded the largest standard deviation for the *approach* action, which can be attributed to the diverse complexity of the planning problem for the mobile base and the distance to travel between the selected grasp and drop location. In total, our system achieved a success rate of 80% for the entire task. Planning and execution required on average 140.63±36.7 s. Errors were mainly caused by object detection issues, i.e., the system was not able to detect the object or the detection was not precise enough to be able to successfully grasp or drop an object.

C. Drinking Task

The last experiment evaluates the direct interaction between user and robot. Therefore, we implemented an autonomous robotic drinking assistant. Our approach enables the robot to fill a cup with a liquid, move the robot to the user and finally provide the drink to the user by execution of the corresponding drinking motion in front of the user's mouth. In addition to the techniques described above, successful pouring and drinking using a robot requires the detection of the liquid level in the cup and a reliable detection and localization of the user's mouth.

To detect the liquid level while pouring, we follow a vision-based approach introduced by Do *et al.* [24]. Given the camera's viewing angle and the liquid's index of refraction, the liquid height is determined from the depth measurement using a relationship based on Snell's law (see [25] for more details). Using this knowledge, we first detect the cup, extract the depth values for the liquid and finally estimate the real liquid height. The type of liquid and hence the index of refraction is assumed to be given beforehand. The viewing

angle can be determined from the depth data. A Kalman filter is then used to track the liquid level and compensate for noise. Once it is detected that the liquid level has exceeded a user defined value, a stop signal is sent to terminate the pouring motion.

For detection and localizing of the user's mouth, we adopt a two step approach. In the first step, we segment the image based on the output of a face detection algorithm in order to extract the image region containing the user's mouth and eyes. Afterwards, we detect the position of the mouth of the user, considering only the obtained image patch. Regarding the mouth orientation, we additionally consider the position of the eyes in order to obtain a more robust estimation of the face orientation, hence compensating for slightly changing angles of the head. The face, mouth and eye detectors are implemented in OpenCV by applying an algorithm that uses Haar cascades [26], [27].

Table III shows the averaged results for the experiment. Here, only 3.75% of the 160 scheduled actions had to be repeated in order to complete the task successfully. In one run, plan recovery was not possible leading to abortion of the task. Thus, our system achieved in total a success rate of 90% for the drinking task. Planning and execution required on average 545.56 ± 67.38 s. For the evaluation of the liquid level detection approach, we specified a desired fill level and executed 10 runs of the pour action. The resulting mean error and standard deviation is 6.9 ± 8.9 mm. In some instances the bottle obstructed the camera view, resulting in poor liquid level detection and a higher error.

VIII. Conclusions

In this paper, we presented a thought-controlled mobile robotic service assistant, capable of successfully performing complex tasks, including close range interaction with the user, in a continuously changing environment to increase the independence of severely paralyzed patients. Through the use of a high-level planner as an intermediate layer between user and autonomous mobile robotic service assistant, we overcome the curse of dimensionality typically encountered in non-invasive BCI control schemes, thus opening up new perspectives for human-robot interaction scenarios.

References

[1] D. Park, Y. K. Kim, Z. M. Erickson, and C. C. Kemp, "Towards assistive feeding with a general-purpose mobile manipulator," *arXiv:1605.07996*, 2016.

[2] C.-S. Chung, H. Wang, and R. A. Cooper, "Autonomous function of wheelchair-mounted robotic manipulators to perform daily activities," in *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–6.

[3] F. Achic, J. Montero, C. Penaloza, and F. Cuellar, "Hybrid bci system to operate an electric wheelchair and a robotic arm for navigation and manipulation tasks," in *Advanced Robotics and its Social Impacts (ARSO), 2016 IEEE Workshop on*. IEEE, 2016, pp. 249–254.

[4] C. Wang, B. Xia, J. Li, W. Yang, D. Xiao, A. C. Velez, and H. Yang, "Motor imagery bci-based robot arm system," in *Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 1. IEEE, 2011, pp. 181–184.

[5] S. Schröer, I. Killmann, B. Frank, M. Völker, L. D. J. Fiederer, T. Ball, and W. Burgard, "An autonomous robotic assistant for drinking," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 6482–6487.

[6] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, pp. 1–22, 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385*, 2015.

[9] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-r. Mohamed, G. Dahl, and B. Ramabhadran, "Deep Convolutional Neural Networks for Large-scale Speech Tasks," *Neural Networks*, vol. 64, pp. 39–48, 2015.

[10] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for LVCSR," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4955–4959.

[11] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: A compact convolutional network for eeg-based brain-computer interfaces," *CoRR*, vol. abs/1611.08024, 2016.

[12] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of EEG motor imagery signals," *Journal of Neural Engineering*, vol. 14, no. 1, p. 016003, 2017.

[13] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human eeg," *arXiv:1703.05051*, 2017.

[14] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan, "Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2008, pp. 2390–2397.

[15] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks," in *arXiv: 1511.06448*, 2016.

[16] S. Stober, "Learning Discriminative Features from Electroencephalography Recordings by Encoding Similarity Constraints," in *Bernstein Conference 2016*, 2016.

[17] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *ArXiv e-prints*, vol. 1511, 2016, p. arXiv:1511.07289.

[18] R. Dale and E. Reiter, "Computational interpretations of the gricean maxims in the generation of referring expressions," *Cognitive science*, vol. 19, no. 2, pp. 233–263, 1995.

[19] M. Göbelbecker, "Assisting with Goal Formulation for Domain Independent Planning," in *KI 2015: Advances in Artificial Intelligence*. Springer, 2015, pp. 87–99.

[20] F. Burget, M. Bennewitz, and W. Burgard, "BI^{2}RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016.

[21] L. E. Kavraki and J.-C. Latombe, *Probabilistic Roadmaps for Robot Path Planning*. John Wiley, 1998, pp. 33–53.

[22] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research 26 (JAIR 2006)*, pp. 191–246, 2006.

[23] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1300–1307.

[24] C. Do, T. Schubert, and W. Burgard, "A probabilistic approach to liquid level detection in cups using an RGB-D camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016.

[25] Y. Hara, F. Honda, T. Tsubouchi, and A. Ohya, "Detection of Liquids in Cups Based on the Refraction of Light with a Depth Camera Using Triangulation," in *IROS*, 2014.

[26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[27] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. I–I.

Multi range Real-time depth inference from a monocular stabilized footage using a Fully Convolutional Neural Network

Clément Pinard[a,b] , Laure Chevalley[a], Antoine Manzanera[b], David Filliat[b]

***Abstract*— We propose a neural network architecture for depth map inference from monocular stabilized videos with application to UAV videos in rigid scenes. Training is based on a novel synthetic dataset for navigation that mimics aerial footage from gimbal stabilized monocular camera in rigid scenes.**

Based on this network, we propose a multi-range architecture for unconstrained UAV flight, leveraging flight data from sensors to make accurate depth maps for uncluttered outdoor environment.

We try our algorithm on both synthetic scenes and real UAV flight data. Quantitative results are given for synthetic scenes with a slightly noisy orientation, and show that our multi-range architecture improves depth inference.

Along with this article is a video that present our results more thoroughly.

I. Introduction

Scene understanding from vision is a core problem for autonomous vehicles and for UAVs in particular. In this paper we are specifically interested in computing the depth of each pixel from image sequences captured by a camera. We assume our camera's velocity (and thus displacement between two frames) is known, as most UAV flight systems include a speed estimator, allowing to settle the scale invariance ambiguity of the depth map.

Solving this problem could be beneficial for several problems such as environment scanning or applying depth-based sense and avoid algorithms for lightweight embedded systems that only have a monocular camera. Not relying on depth Sensors such as stereo vision, ToF camera, LiDar or Infra Red emitter/receiver allows to free the UAV from their weight, cost and limitations. Specifically, along with some RGB-D sensors being unable to operate under sunlight (e.g. IR and ToF), most of them suffer from range limitations and can be inefficient in case we need long-range information such as trajectory planning [7]. Unlike RGB-D sensors, depth from motion is flexible w.r.t. displacement and thus robust to high speeds or high distances as choosing among previous frames gives us a wide range of different displacements. For estimating such depth maps, we designed an end-to-end learning architecture, based on a synthetic dataset and a fully convolutional neural network that takes as input an image pair taken at different times. No preprocessing such as optical flow computation, nor visual odometry is applied to the input, while the depth is directly provided as an output.

[a]Parrot, Paris, France
(clement.pinard, laure.chevalley)@parrot.com
[b]U2IS, ENSTA ParisTech, Université Paris-Saclay, Palaiseau, France
(clement.pinard, antoine.manzanera, david.filliat)@ensta-paristech.fr



a) b) c)

Fig. 1. Camera stabilization can be done via a) mechanic gimbal or b) dynamic cropping from fish-eye camera, for drones or c) hand-held cameras

We created a dataset of image pairs with random translation movements, with no rotation, and a constant displacement magnitude applied during the whole training.

The assumption about videos without rotation appears realistic for two reasons:

- Hardware rotation compensation is mainly a solved problem, even for consumer products, with IMU-stabilized cameras on consumer drones or hand-held steady-cam (Fig 1).
- this movement is somewhat related to human vision and vestibulo-ocular reflex (VOR) [2]. Our eyes orientation is not induced by head rotation, our inner ear among other biological sensors allows us to compensate parasite rotation when looking at a particular direction.

Using the trained network, we propose an algorithm for real condition depth inference from a stabilized UAV. Displacement from sensors is used to compute real depth map, as it only differs from the synthetic constant displacement images by a scale factor. Our network output also allows us to *a posteriori* optimize the depth inference. By adjusting frame shift to get a displacement that would make the network get the same disparity distribution as during its training, we lower the depth error for next inference. For example, with large distances, ideal displacement between two frames is higher, and thus the shift is also higher. Moreover, we use multiple batch inference to compute multiple depth maps centered around a particular range, and fuse them to get a high precision for both close and far objects, no matter the distance, given a sufficient displacement from the UAV.

II. Related Work

Deep Learning and Convolutional Neural Networks have recently been widely used for numerous kinds of vision problem such as classification [13] and hand-written digits recognition [14].

Depth from vision is one of the problems studied with neural network, and has been addressed with a wide range

of training solution. Some datasets [6], [19] allow a neural network to learn end-to-end depth or disparity [15], [22], [4]. Reprojection error has also been used for unsupervised training for depth from a single image [20], [23] or for disparity between two frames of a stereo rig [12], [5].

Depth from a single image, although interesting, suffers from a major drawback which is overfitting. No motion is given to the network during inference, and the resulting depth is inferred from context, whereas they can be decorrelated. This technique can be sufficient for road driving context with an obvious road in front of the camera, but for a UAV flight usage, we may have to deal with very heterogeneous scenes. On the other hand, depth from a stereo pair is only implying a single lateral movement, and lacks a forward component to appear realistic for any aerial stabilized footage.

For depth from more complex movement from a monocular camera, current state of the art methods tend to use motion, and especially structure from motion, and most algorithm do not rely on deep learning [1], [17], [11]. Prior knowledge w.r.t. scene is used to infer a sparse depth map with its density usually growing over time. These techniques also called SLAM are typically used with unstructured movement (translation and rotation with varying magnitudes), produce very sparse point-cloud based 3D maps and require heavy calculation to keep track of the scene structure and align newly detected 3D points to the existing ones.

Our goal is to compute a dense depth map (where every point has a valid depth) using only two frames from the same camera, at different times, and without prior knowledge on the scene and movement, apart from the lack of rotation and the scale factor.

III. End-to-end learning of Depth Inference

Inspired by flow estimation and disparity (which is essentially magnitude of optical flow vectors), a problem to which exist a lot of very convincing methods [8], [10], we set up an end-to-end learning workflow, by training a neural network to explicitly predict the depth of every pixel in a scene, from an image pair with constant displacement value.

A. Still Box Dataset

We design our own synthetic dataset, using the rendering software *Blender*, to generate an arbitrary number of random rigid scenes, composed of basic 3d primitives (cubes, spheres, cones and tores) randomly textured from an image set scrapped from *Flickr* (see Fig 2).

These objects are randomly placed and sized in the scene, and walls are added at large distances as if the camera was inside a box (hence the name). The camera is moving at a fixed speed value, but to an uniformly distributed random direction, which is constant for each scene. It can be anything from forward/backward movement to lateral movement (which is then equivalent to stereo vision).

B. Dataset augmentation

In our dataset, we store data in 10 images long videos, with each frame paired with its ground truth depth. This allows us to set *a posteriori* distances distribution with a variable temporal shift between two frames. If we use a baseline shift of 3 frames, we can e.g. assume a depth three times as great as for two consecutive frames (shift of 1). In addition, we can also consider negative shift, which will only change displacement direction without changing speed value. This allows us, given a fixed dataset size, to get more evenly distributed depth values to learn, and also to de-correlate images from depth, preventing over-fitting during training, that would result in a scene recognition algorithm and would poorly perform on a validation set.

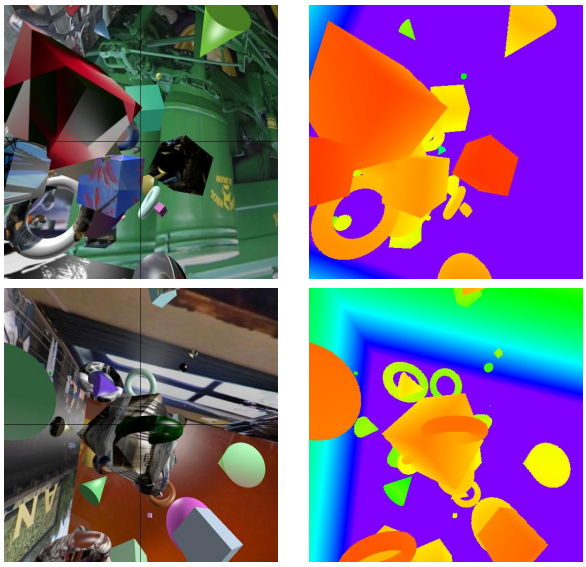


Fig. 2. Some examples of our renderings with associated depth maps (red is close, purple is far)

C. Depth Inference training

Our network is broadly inspired from FlowNetS [3] (initially used for flow inference) and called DepthNet. It is described in details in [18], we provide here a summary of its structure (Fig 3) and performances. Each convolution (apart from depth modules) is followed by a Spatial Batch Normalization and ReLU activation layer. Batch normalization helps convergence and stability during training by normalizing a convolution's output (0 mean and standard deviation of 1) over a batch of multiple inputs [9], and Rectified Linear Unit (ReLU) is the typical activation layer [21]. Depth Module are convolution modules, reducing the input to 1 feature map, which is expected to be the depth map, at a given scale. One should note that FlowNetS initially used LeakyReLU which has a non-null slope for negative values, but tests showed that ReLU performed better for our problem.

The main idea behind this network is that upsampled feature maps are concatenated with corresponding earlier convolution outputs (e.g. Conv2 output with Deconv5 output). Higher semantic information is then associated with information more closely linked to pixels (since it went through less downsampling convolutions) which is then used for reconstruction.

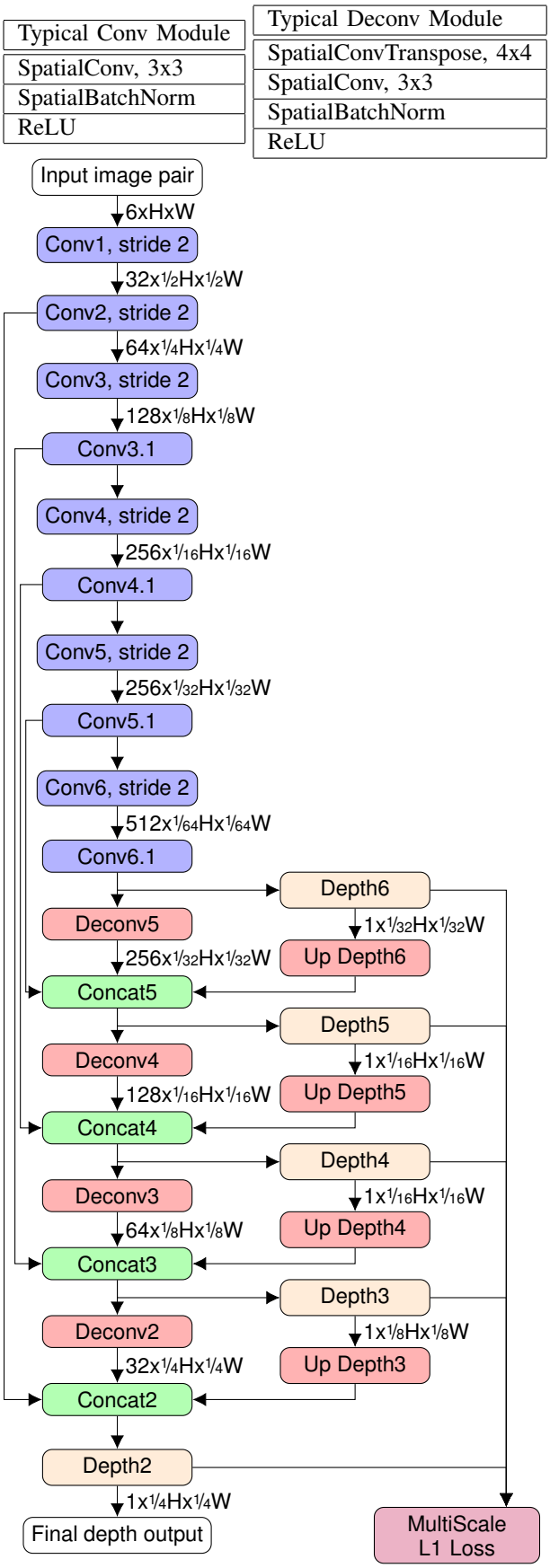


Fig. 3. DepthNet structure parameters, Conv and Deconv modules detailed above

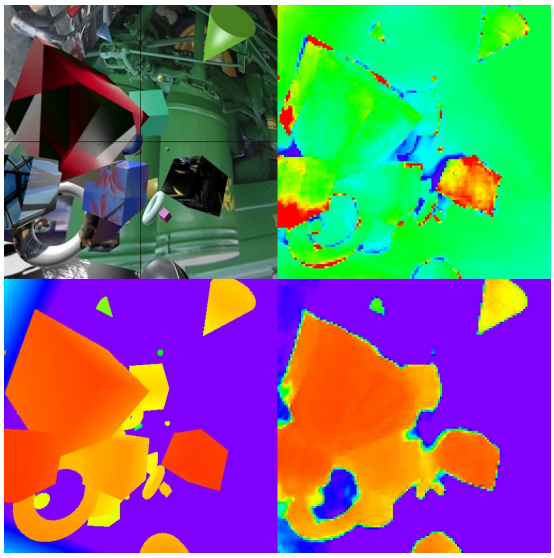


Fig. 4. Result on 512x512 images from $\text{DepthNet}_{64\to128\to256\to512}$. Upper-left: input, lower-left: Ground Truth depth, lower-right: our network output (128x128), upper-right: error, green is no error, red is overestimated depth, blue is underestimated

This multi-scale architecture has been proven very efficient for flow and disparity computing while keeping a very simple supervised learning process.

The main point of this experimentation is to show that direct depth estimation can be efficient regarding unknown translation. Like FlowNetS, we use a multi-scale criterion, with a L1 reconstruction error for each scale:

$$Loss = \sum_{s \in scales} \gamma_s \frac{1}{H_s W_s} \sum_i \sum_j |\beta_s(i,j) - \zeta_s(i,j)| \quad (1)$$

where

- γ_s is the weight of the scale, arbitrarily chosen.
- $(H_s, W_s) = (1/2^s H, 1/2^s W)$ are the height and width of the output.
- ζ_s is the scaled depth groundtruth, using average pooling.
- β_s is the ouput of the network at scale s.

As said earlier, we apply data augmentation to the dataset using different shifts, along with classic methods such a flips and rotations. We also clamp depth to a maximum of 100m, and provide sample pairs without shift, assuming its depth is 100m everywhere. As a consequence, the trained network will only be able to infer depth lower than 100m.

We applied training on several input size images, from 64x64 to 512x512. Fig 4 shows training results for mean L1 reconstruction error. Like FlowNetS, network output are downsampled by a factor of 4 with reference to the input size. As Table I shows, best results are obtained with multiple fine-tuning, with intermediate scales 64, 128, 256, and finally 512 pixels. Subscript values indicate finetuning processes. FlowNetS is performing better than DepthNet but by a fairly

Network	L1Error		RMSE	
	train	test	train	test
FlowNetS_{64}	1.69	4.16	4.25	7.97
DepthNet_{64}	2.26	4.49	5.55	8.44
$\text{FlowNetS}_{64\to128\to256\to512}$	0.658	**2.44**	1.99	**4.77**
$\text{DepthNet}_{64\to128}$	1.20	3.07	3.43	6.30
$\text{DepthNet}_{64\to128\to256}$	0.876	**2.44**	2.69	4.99
$\text{DepthNet}_{64\to128\to256\to512}$	1.09	2.48	2.86	**4.90**
$\text{DepthNet}_{64\to512}$	1.02	2.57	2.81	5.13
DepthNet_{512}	1.74	4.59	4.91	8.62

TABLE I. Quantitative results for depth inference networks. FlowNetS is modified with 1 channel outputs (instead of 2 for flow), trained from scratch for depth with Still Box, subscript indicates fine tuning process.

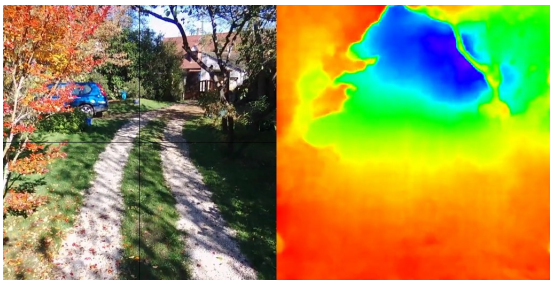


Fig. 5. Result on 512x512 real images input from a Bebop drone footage

light margin while being 5 times heavier and most of the time much slower.

IV. UAV NAVIGATION USE-CASE

A. Optimal frame shift determination

We learned depth inference from a moving camera, assuming its velocity is always the same. Results from real condition drone footage, on which we were careful to avoid camera rotation can be seen Fig 5. These results did not benefit from any fine-tuning from real footage, indicating that our Still Box Dataset, although not realistic in its scenes structures and rendering, appears to be sufficiently heterogeneous for learning to produce decent depth maps in real conditions. When running during flight, such a system can deduce the real depth map ζ from the network output and the drone displacement, knowing that the training displacement was D_0 (here $0.3m$)

$$\begin{aligned} \zeta(t) &= DepthNet(I_t, I_{t-\Delta t})\frac{D(t,\Delta t)}{D_0} \\ D(t,\Delta t) &= \left\| \int_{t-\Delta t}^{t} V(\tau) d\tau \right\| \end{aligned} \tag{2}$$

The actual correct interpretation of the output of DepthNet is rather a percentage than a distance. 100% meaning max distance for a given displacement D. We can introduce a function $\beta = \frac{DepthNet(I_t, I_{t-\Delta t})}{maxDistance}$ and a dimension-less parameter $\alpha = \frac{maxDistance}{D_0}$ for computing actual depth using the displacement D as the only distance related factor.

$$\zeta(t) = \alpha\beta(I_t, I_{t-\Delta t})D(t,\Delta t) \tag{3}$$

Depending of the depth distribution of the ground-truth depth map, it may be useful to adjust frame shift Δt. For example, when flying high above the ground with low speed, big structure detection and avoidance requires knowing precise distance values that are outside the typical range of any RGB-D sensor. The logical strategy would then be to increase the temporal shift between the frame pairs provided to DepthNet as inputs. More generally, one must provide inputs to DepthNet in order to ensure a well distributed depth output within its typical range. Depth-wise normalized error which is the essential quality measurement for values that we want to rescale, will diverge when ground truth depth approaches 0. Indeed, in addition to being equivalent to an infinite optical flow, the depth-wise error cannot tend to 0, which will make the expression ${}^{error}/_{depth}$ tend to $+\infty$ at 0 We thus need to choose the optimal spatial displacement and corresponding temporal shift to minimize error on the next inference, assuming the same depth distribution, to avoid too low or too high equivalent ground-truth. We chose the space displacement as:

$$D_{optimal}(t+1) = \frac{E(\zeta(t))}{\alpha\beta_{mean}} \tag{4}$$

With $E(\zeta(t))$ the mean of depth values and β_{mean} the optimal mean output of β, e.g. 0.5. $\Delta(t)$ is then computed numerically to get the frame shift with the closest corresponding displacement possible.

B. Multiple shifts inference

As neural network are traditionally computed within massively parallel architectures such as GPUs, multiple depth maps can be computed efficiently at the same time in a batch, especially for low resolution. Batch inference can then be used to compute depth with multiple shifts $\Delta(t,i)$. These multiple depth maps can then be combined to construct a higher quality depth map, with high precision for both long and short range. We propose a dynamic range algorithm, described Fig 6 to compute an combine different depth maps.

Instead of only one optimal displacement $D(t)$ from $E(\zeta)$, we use K-mean clustering algorithm [16] on the depth map to find a list of clusters on which each shift will focus. The clustering outputs a list of n centroids $C_i(\zeta)$ and corresponding $D_i(t)$ and $\Delta(t,i)$. n is an arbitrary chosen value, usually ranging from 1 to 4.

Final DepthMap is then computed from fusing these outputs using a weighted mean for each pixel. Each weight is actually a linear interpolation from 0 to 1 according to distance of depth from a target value β_{mean}. That way, fusion will favor values that are closer to this optimal value. An ϵ value is added to solve fusion when every depth map is off its wanted range.

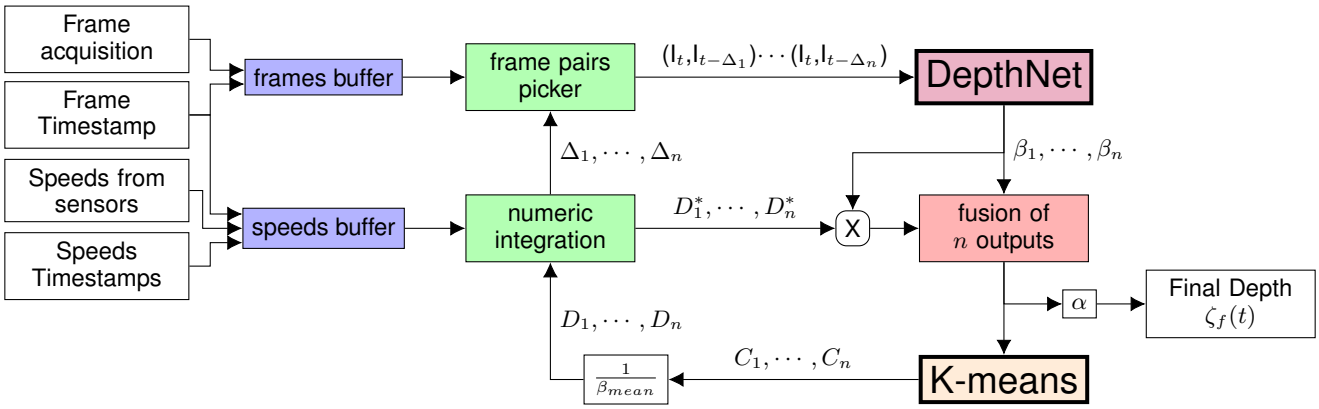


Fig. 6. Multiple shifts architecture. We used n different planes. Numeric integration, given a desired displacement D gives the closest possible displacement between frames D^* , along with corresponding shift Δ. As discussed in part IV, the fusion block computes pixel-wise weights from $\beta_1, \cdots, \beta_n$ to make a weighted mean of $\beta_1 D_1, \cdots, \beta_n D_n$

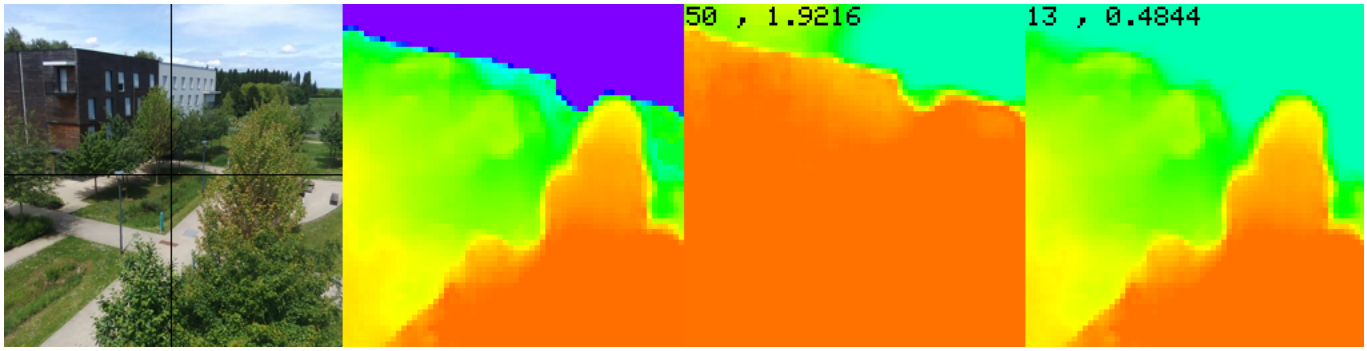


Fig. 7. real condition application of the multi-shift algorithm with Tiny DepthNet Clamped. First image is input. Last two are outputs of the network, for shifts of 50 and 13 with a drone flying forward at $1m.s^{-1}$ and at an altitude of $12m$, with corresponding displacements from sensors. Second is fused output, capped to $100m$ up

$$
\begin{aligned}
w_{ijk} &= \epsilon + f(\beta(img_t, img_{t-\Delta(t,i)})) \\
f : x \mapsto &\begin{cases} 0 & \text{if } x < \beta_{min} \\ \frac{x-\beta_{min}}{\beta_{mean}-\beta_{min}} & \text{if } \beta_{min} \leq x < \beta_{mean} \\ \frac{\beta_{max}-x}{\beta_{max}-\beta_{mean}} & \text{if } \beta_{mean} \leq x < \beta_{max} \\ 0 & \text{if } x \geq \beta_{max} \end{cases} \\
\zeta_i(t) &= \alpha D_i(t) \beta(img_t, img_{t-\Delta(t,i)})
\end{aligned} \tag{5}
$$

$$
\forall (j,k) \in [\![0, W]\!] \times [\![0, H]\!], \zeta_f(t)_{jk} = \frac{\sum_i w_{ijk} \zeta_{ijk}(t)}{\sum_i w_{ijk}} \tag{6}
$$

For our use-case, we set $\beta_{min} = 0.1$, $\beta_{mean} = 0.4$, $\beta_{max} = 0.9$ and $\epsilon = 10^{-3}$. i is the index of frame shift, j, k are the spatial indices. Fig 7 shows a result of the proposed algorithm for a batch size of 2. Notice how the high shift detects buildings while low shift detects trees.

C. Clamped DepthNet

Our proposed algorithm is actually suffering a problem for real condition videos, because we assume a perfect stabilization. Therefore, on very far objects (e.g. the sky), any minor optical flow caused by a default in stabilization will result in a massive error in depth. Moreover, our network being very good at recognizing shapes and giving it the same depth everywhere, this can result in the whole sky being computed as relatively close. We thus propose a network designed for a simpler problem: during training on still box, we clamp depth from $10m$ to $60m$, with a shift of 5 images (instead of 3 for DepthNet). These new parameters allow the network to only focus on mid range objects, dismissing close and far objects with respectively a too large and too small optical flow. This training workflow is very well suited for multiple shift depth inference. Every image pair will have a dedicated depth to analyze, allowing the fusion to not be bothered with redundant data, because of the high initial range of DepthNet.

Figure 8 shows results for multiples synthetic 256x256 scenes with ground truth, along with inference speed and a small noise added to camera initial orientation at each frame. $R(t) = R_0 + Euler(N_0 \mu(t))$, with $\mu(t)$ being a 3-dimensional random unit vector and N_0 a constant fixed to 0.001. We also report performance a thin version of our clamped network, that shows better results than DepthNet with 1 plane only in this noisy setup. The thin network has the same depth, but every convolution has an output half the number of feature maps of the original DepthNet. These

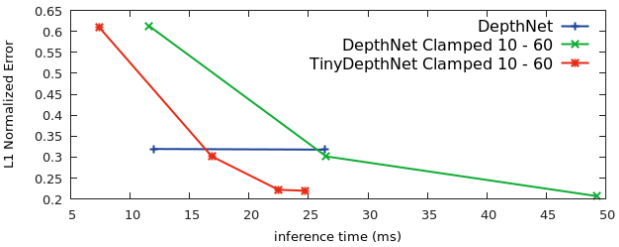


Fig. 8. results for synthetic 256x256 scenes with noisy orientation. DepthNet has been tested with 1 and 2 planes, DepthNet Clamped with 1 to 3 planes and Tiny DepthNet Clamped with 1 to 4 planes. Y axis is Absolute mean error (m) divided by ground-truth depth, X axis is inference speed, in ms

results have been obtained on a Quadro K2200m powered laptop.

V. Conclusion and Future Work

We proposed a novel way of computing dense depth maps from motion, along with a very comprehensive dataset for stabilized footage analysis and a technique for dynamic range real flight computing. This algorithm can then be used for depth-based sense and avoid algorithms in a very flexible way, in order to cover all kinds of path planning, from collision avoidance to long range obstacle bypassing.

A more thorough presentation of the results can be viewed in this video. `http://perso.ensta-paristech.fr/~manzaner/Download/ECMR2017/DepthNetResults.mp4`

Future works include implementation of such a path planning algorithm, and construction of a real condition fine tuning dataset, using UAVs footages and a preliminary thorough 3D offline scan. This would allow us to measure quantitative quality of our network for real footages and not only subjective as for now. We could also use unsupervised techniques, using re-projection errors as in [23].

We also believe that our network can be extended to re-inforcement learning applications that will potentially result in a complete end-to-end sense and avoid neural network for monocular cameras.

The major drawback of our algorithm is however the necessity for a scene to be rigid. This is obviously never the case, and even though UAV footage are less prone to moving objects like in autonomous driving problems, we will have this issue whenever a moving target is to be followed. To solve this problem, an explicit movement equation for both the camera and the moving targets may have to be computed, as in [20]. In any case, this problem will be a challenge and may not be solvable with fully Convolutional networks only as we did in this article.

References

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[2] Lorente De Nó, R. Vestibulo-ocular reflex arc. *Archives of Neurology & Psychiatry*, 30(2):245–291, 1933.

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[5] Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, abs/1603.04992, 2016.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[7] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[8] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. *ArXiv e-prints*, March 2017.

[11] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[12] Kishore Reddy Konda and Roland Memisevic. Unsupervised learning of depth and motion. *CoRR*, abs/1312.3429, 2013.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[17] Raul Mur-Artal and Juan D Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[18] Clément Pinard, Laure Chevalley, Antoine Manzanera, and David Filliat. End-to-end depth from motion with stabilized monocular videos. In *submitted to the International Conference on Unmanned Aerial Vehicles in Geomatics (UAV-G)*, 2017.

[19] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[20] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfM-Net: Learning of Structure and Motion from Video. *ArXiv e-prints*, April 2017.

[21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[22] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[23] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

Robust Submap-Based Probabilistic Inconsistency Detection for Multi-Robot Mapping

Yufeng Yue[1], Danwei Wang[1], P.G.C.N. Senarathne[2] and Chule Yang[1]

***Abstract*— The primary goal of employing multiple robots in active mapping tasks is to generate a globally consistent map efficiently. However, detecting the inconsistency of the generated global map is still an open problem. In this paper, a novel multi-level approach is introduced to measure the full 3D map inconsistency in which submap-based tests are performed at both single robot and multi-robot level. The conformance test based on submaps is done by modeling the histogram of the misalignment error metric into a truncated Gaussian distribution. Besides, the detected inconsistency is further validated through a 3D map registration process. The accuracy of the proposed method is evaluated using submaps from challenging environments in both indoor and outdoor, which illustrates its usefulness and robustness for multi-robot mapping tasks.**

I. Introduction

With the maturity of the single robot technology, the use of a group of coordinated robots [1] in the past decade has been gradually taken seriously. The use of multiple robots significantly improve the efficiency and robustness in search and rescue tasks [2] and collaborative mapping [3]. One of the key challenges is to generate a globally consistent map of the environment. Global map generation is generally achieved by fusing the local maps generated by individual robots [4]. However, this is based on the assumption that the created local maps are consistent with no significant errors. Fusing erroneous local maps results in an inconsistent global map, which renders the accurate execution of autonomous tasks infeasible. Therefore detecting inconsistency in both local maps and fused global maps is vital for multi-robot missions.

The development of map inconsistency detection strategies has not received much attention, where the majority of the research is focused on generating accurate maps and post-processed optimization. However, map generated does not guaranteed to be foolproof and may generate inconsistent maps, especially in large environments and over long operating times. In addition, post-processed optimization is computationally expensive, where a proper triggering time and quantitative inconsistency measurements are required. Hence, there is a gap between the map generation and the map optimization, which is the inconsistency detection of the map. The capability to detect inconsistency in the generated maps allows robots either to perform an efficient optimization algorithm in order to continue the mission, or to return to a safe location for recovery without getting trapped.



(a) Consistent map (b) Inconsistent map

Fig. 1. An example of consistent and inconsistent map

This paper presents a novel strategy that detects the inconsistency of map generated from two levels, i.e., single robot and multi-robot level. The inconsistency of the system is modeled as a probabilistic distribution, where submap-based inconsistency testing is performed on three types of inconsistency defined on both single robot and multi-robot level. In both levels, map inconsistency measurements are computed by modeling the misalignment errors between submap pairs into a truncated Gaussian distribution, and the resultant mean and variance are used to detect the inconsistency. Besides, the detected inconsistency is further validated through a 3D map registration process. The fused global map is deemed consistent only if all submap pairs of all robots pass the inconsistency checks.

Main contributions of this work are listed below:

- A novel multi-level topological framework is proposed to model the inconsistency of full 3D maps on both single robot and multi-robot level.
- A submap-based probabilistic algorithm is developed to efficiently detect and verify the three types of inconsistency (i.e., sequential, local and global).

The rest of this paper is organized as follows: Section II reviews the related literature. Section III presents the topological probabilistic structure. Section IV details submap-based inconsistency modeling and detection. Section V presents the experiments. Section VI concludes the paper.

II. Related Work

Various mapping algorithms have been proposed in the past decades [5] [6], however, these methods can't guarantee to produce globally consistent maps all the time. To increase the convergence of mapping, graph SLAM back-end has been proposed to minimize the accumulated error [7]. Based on that work, variants using dynamic covariance scaling [8] and switchable constraints [9] have been addressed.

*The research was partially supported by the ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme.

[1] Yufeng Yue, Danwei Wang and Chule Yang are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

[2] P.G.C.N. Senarathne is with ST Engineering-NTU Corporate Laboratory, Nanyang Technological University, Singapore

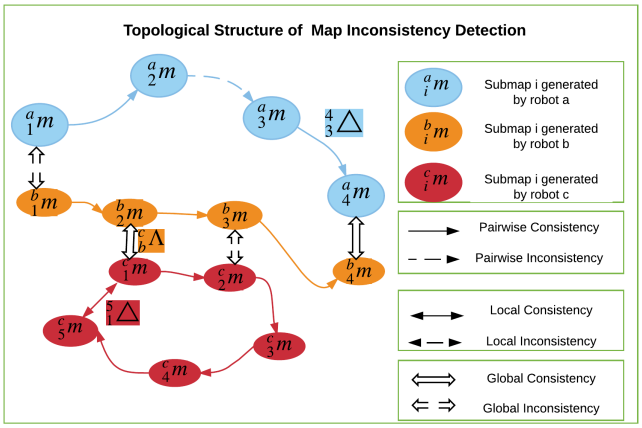


Fig. 2. Topological structure of map inconsistency detection. The nodes with different colors represents the submaps of different robots. The three types of edges show the inconsistency at different levels.

Some researchers present vision based place recognition to trigger back-end [10], which is sensor-dependent and needs to process huge amount of raw sensor data. Under the condition of limited communication and memory of multi-robot system, a compact map generated by compressing raw sensor data is an alternative choice. However, very few research has been conducted on detecting the inconsistency of generated maps, except the works proposed in [11] [12] [13]. In [11], a Bayesian representation of map posterior is used to detect the wrong data associations. Then it is extended to a multi-scan scenario with a cascaded map inconsistency statistical test in [12]. Based on that work, [13] developed a robot homing strategy whenever the map is detected to be inconsistent. These works pioneer the map inconsistency detection problem, and the use of this knowledge to improve autonomous mapping missions. However, these methods focus on sensor data level in a 2D environment with a single robot, which is infeasible to be extended to a multi-robot scenario in full 3D mapping missions.

In large environment, submap-based approach provides incremental [14] and efficient [15] approach for robot mapping. For multiple robots coordination [16], submap is even more suitable under the constraints of limited communication bandwidth and coverage. As submaps are generated over a short motion window of the robot, it is reasonable to assume that they are locally consistent [17]. To the extent of our knowledge, this is the first application of robust submap-based inconsistency detection for multi-robot mapping.

III. Topological Structure of Map Inconsistency Detection

This section presents the structure of submap-based inconsistency measurement, which is shown in Fig. 2. The submap-based inconsistency measurement is defined as a topological structure, where the node is defined as the submap and the edge is defined as the submap-wise inconsistency. Then we further explain the three types of inconsistency on single robot and multi-robot level.

A. Definition of Node

The node is defined as a submap and notation is denoted in Eq.(1)

$$ {}^{x}_{i}V = \{{}^{x}_{i}m\}_{i\in n_x, x\in\mathcal{R}} \tag{1}$$

where n_x stands for the set of known submaps generated by robot x, and $\mathcal{R}$ represents the set of robots $\{a,b,c,\cdots\}$. Submap $\{{}^{x}_{i}m\}$ is generated when certain conditions have met and is added to the topology as a new node. In this paper, a submap is represented as a 3D probabilistic occupancy voxel map [5]. Considering the limited size of submap will lead to less drift, while still providing sufficient submap size for map registration. The criterion that triggers the creation of a new submap is based on the condition that the robot moved for a certain distance or rotated for a certain angle.

Based on submaps generated, the local map and global map on a higher level are defined below.

1) Local map generated by a single robot: Local map ${}^{x}m$ generated by a single robot x consists of submaps $\{{}^{x}_{i}m\}$ with $i \in n_x$, which is shown as the chain of node with the same color in Fig. 2 and is denoted as:

$$ {}^{x}m = \{{}^{x}_{i}m, i \in n_x\} \tag{2}$$

2) Global map generated by multi-robot: Global map M that integrates 3D occupancy local maps ${}^{x}m$ with $x \in \mathcal{R}$ is defined as:

$$ M = \{{}^{x}m, x \in \mathcal{R}\} \tag{3}$$

B. Definition of Consistency

Map inconsistency is defined as a measurement of alignment between the submaps. On that basis, consistency indicates that submaps have been registered properly, while inconsistency implies a large offset between two submaps. The inconsistency between two submaps is represented by the edge, which defines the inconsistency between two submaps on single robot level and multi-robot level.

1) single robot level: The submaps i, j within the local map ${}^{a}m$ connected by an edge ${}^{i}_{j}\triangle$ is denoted as single robot level inconsistency, which has two types: sequential inconsistency and local inconsistency.

a) Sequential inconsistency: Sequential inconsistency is defined as the probability between consecutive submaps ${}^{a}_{i}m$ and ${}^{a}_{i-1}m$ and is defined in Eq.(4). Sequential inconsistency always happens when two consecutive submaps didn't align properly due to SLAM error like a sharp turn occurs or a sudden drift in odometry sensor.

$$ {}^{i}_{i-1}\triangle = p({}^{a}_{i}m|{}^{a}_{i-1}m)_{\{i=2:n_a\}} \tag{4}$$

b) Local inconsistency: local inconsistency is defined in Eq.(5), which calculates the probability of ${}^{a}_{i}m$ over the past submaps ${}^{a}_{j}m$ generated by robot a.

$$ {}^{i}_{j}\triangle = \ p({}^{a}_{i}m|{}^{a}_{j}m)_{\{i=k+1:n_a, j=1:i-k\}} \tag{5}$$

The calculation is not performed from $i-k$ to $i-1$ submaps because ${}^{a}_{i}m$ always has small overlapping with those submaps. Local inconsistency always happens when the robot in current submap ${}^{a}_{i}m$ didn't recognize the previously traveled submap ${}^{a}_{j}m$.

2) Multi-robot level: The local maps ${}^{a}m$, ${}^{b}m$ connected by an edge ${}^{a}_{b}\Lambda$ is denoted as multi-robot level inconsistency. Global inconsistency is defined as probability of submap ${}^{b}_{j}m$ in robot b over all the submaps ${}^{a}_{i}m$ in robot a, which is defined in Eq.(6). Global inconsistency always happens when two robots can not recognize the same place they visited.

$$ {}^{a}_{b}\Lambda = p({}^{b}m|{}^{a}m) \propto \prod_{i=1:n_a, j=1:n_b} p({}^{b}_{j}m|{}^{a}_{i}m) \tag{6} $$

3) Overall system: With the three types of inconsistency defined above, we can write the inconsistency measurement for the whole system. Here, we denote the inconsistency of the system in Eq.(7).

$$ \begin{aligned} &\prod_{a} \underbrace{p({}^{a}_{i}m|{}^{a}_{j}m)}_{\text{single robot level}} \cdot \prod_{a,b} \underbrace{p({}^{b}_{j}m|{}^{a}_{i}m)}_{\text{multi-robot level}} \\ =&\prod_{a} \left(\underbrace{p({}^{a}_{i}m|{}^{a}_{i-1}m)}_{\text{pairwise consistency}} \underbrace{p({}^{a}_{i}m|{}^{a}_{j}m)}_{\text{local consistency}} \right) \cdot \prod_{a,b} \underbrace{p({}^{b}_{j}m|{}^{a}_{i}m)}_{\text{multi-robot level}} \end{aligned} \tag{7} $$

Since the three types of inconsistency defined can be represented at submap-level, the inconsistency measurement can be modeled as a Gaussian distribution $f(x;\mu,\sigma^2)$. The modeling will be detailed in Sec.(IV).

IV. Submap-Based Inconsistency Detection

In this section, the submap-based inconsistency measurement is calculated by modeling the histogram of inconsistency distance into a truncated Gaussian distribution. Then submap-based inconsistency testing is performed and verified by applying 3D map registration.

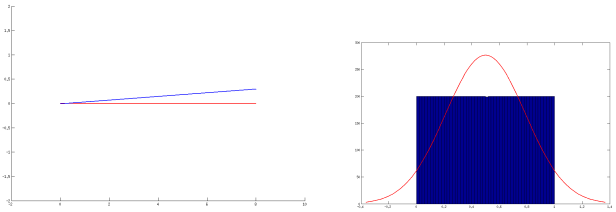
A. Submap-wise Inconsistency Distance

A proper inconsistency distance should be defined to describe the discrepancy between submaps. Here, we assume m and n as two submaps, where m_i and n_j are two voxels in submap m and n, respectively. The edge connects m and n can be any of the three types of inconsistency defined in Sec.III.

The distance applied here is the Occupancy Iterative Closet Point(OICP) distance defined in [4]. The OICP distance is an error metric specially designed for 3D occupancy grid map that combines the Euclidean distance $d_e(m_i,n_j)$ and occupancy probability distance $d_f(m_i,n_j)$. The inconsistency distance between a pair-wise matched voxels is defined in Eq.(8). The details of OICP distance can be found in [4].

$$ d(m_i,n_j) = d_e(m_i,n_j) + d_f(m_i,n_j) \tag{8} $$

Where $d(m_i,n_j)$ is the OICP distance between voxel m_i in submap m to the closest voxel n_j in submap n. The histogram of OICP distance over all the matched voxels (i,j) is computed to show the distribution of the inconsistency distances. Since inconsistency is defined as a probability distribution, the histogram is further modeled into probability distribution to describe the discrepancy between submaps.



(a) The pair-wise submaps are mis-aligned with an error (b) Uniform histogram is modeled into guassian distribution

Fig. 3. An example of misaligned submaps and the histogram of inconsistency distance

B. Modeling of Probability Distribution

For simplification, we assume the submaps m and n to be straight lines and the probability of each voxel equals to 1. Histograms of perfect alignment and misalignment are modeled into Gaussian distributions $f(x;\mu,\sigma^2)$.

1) Modeling of Perfect Alignment : Assuming two submaps are noise free and perfectly aligned, which means the distance metric $d(m_i,n_j)$ between all corresponding voxels equals to zero. Then the histogram is subject to a Dirac delta distribution, which is the limit the normal distribution when $\sigma^2 \to 0$.

$$ f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}};\ \mu = 0,\ \sigma^2 \to 0, \tag{9} $$

However, in real environment the map will be noisy and alignment will have a distance residual, the σ^2 will always greater than zero.

2) Modeling of Misalignment: Considering the inconsistency situation for straight lines, the two lines will intersect at a point. Here, we assume the two lines intersect at the start point, as shown in Fig. 3a. The parallel lines are not considered since submap overlapping is a basic requirement for inconsistency detection.

The histogram of distance distribution complies a uniform distribution and is shown in Fig. 3b.

$$ f(x) = \begin{cases} \frac{1}{b-a} & for\ a \le x \le b \\ 0 & for\ a \le 0\ or\ x \ge b \end{cases} \tag{10} $$

The uniform histogram can also be modeled by a Gaussian distribution with mean μ and variance σ^2 as shown in Fig. 3b.

$$ f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{-(x-\mu)^2}{2\sigma^2}};\ \mu = \frac{a+b}{2},\ \sigma^2 = \frac{(b-a)^2}{12} \tag{11} $$

Based on the observations, the simplification reveals that μ and σ^2 can be used to describe the discrepancy. In general, large μ and σ^2 indicate a large misalignment error.

C. Truncated Gaussian Distribution

As shown above, the histogram can be modeled into Gaussian distribution $X \sim N(\mu,\sigma^2)$. Since the distance is absolute value, the Gaussian distribution can be modified to truncated Gaussian distribution in the interval of $[a,b]$. Given



(a) Indoor experiment (b) Outdoor experiment

Fig. 4. The environment of the experiment

a normal distribution $X \sim N(\mu, \sigma^2)$ with the probability density function(pdf) $\Phi(x)$ and cumulative density function(cdf) $\Psi(x)$. Then, $X \sim N(\mu, \sigma^2)$ conditioned on interval $a \leq x \leq b$ is derived:

$$f(x; \mu, \sigma^2; a, b) = f_t(x; \mu_t, \sigma_t^2) = \frac{\Phi(x)}{\Psi(b) - \Psi(a)} \tag{12}$$

Let $\alpha = \frac{a-\mu}{\sigma}$, $\beta = \frac{b-\mu}{\sigma}$where truncated mean μ_t and truncated variance σ_t^2 are calculated as follows:

$$\mu_t = \mu + \sigma \frac{\Phi(\alpha) - \Phi(\beta)}{\Psi(\beta) - \Psi(\alpha)} \tag{13}$$

$$\sigma_t^2 = \sigma^2 [1 + \frac{\alpha\Phi(\alpha) - \beta\Phi(\beta)}{\Psi(\beta) - \Psi(\alpha)} - (\frac{\Phi(\alpha) - \Phi(\beta)}{\Psi(\beta) - \Psi(\alpha)})^2] \tag{14}$$

Here we set $a = 0$, as distance is always no less than 0, and $b = d$, where d is the maximum search distance of OICP registration algorithm. On the interval of $[0, d]$, μ_t describes the average misalignment distance, and σ_t^2 describes the degree of misalignment divergence.

D. Inconsistency Testing

In this part, a test is performed to detect the three types of inconsistency defined in Sec.III.

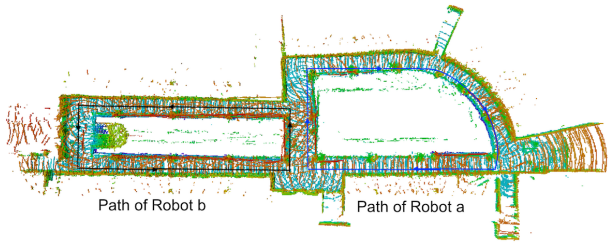
Given all the submaps $\{{}_i^x m\}_{i \in n_x, x \in \mathcal{R}}$, the overlapping between submaps is firstly tested. Then the histogram of misalignment distances is modeled into truncated Gaussian distribution. An $N \times N$ square matrix Γ is generated that contains Γ_{ij} value defined in Eq.(15), where $N = \sum_{x \in \mathcal{R}} n_x$ is the total number of submaps. Here, the sparsity of matrix Γ depends on the pairs of overlapping submaps.

$$\Gamma_{i,j} = \frac{1}{\mu_t \cdot \sigma_t^2} \tag{15}$$

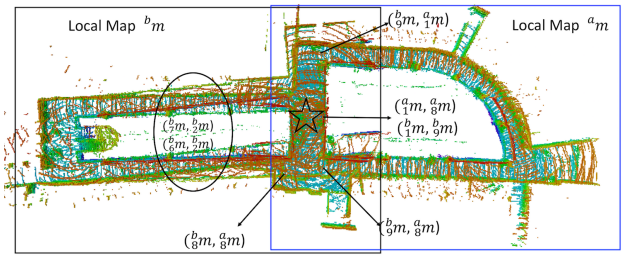
Inconsistency testing is performed for each $\Gamma_{i,j}$, a large value of $\Gamma_{i,j}$ indicates consistency, while a low value implies that large misalignment compared with the perfect map alignment. A global map is consistent if all tests are successful. For inconsistent submaps, they share a common area, however, are not aligned properly. Hence, we apply a novel map registration algorithm [4] to align the inconsistent submaps and generate a $N \times N$ matrix Θ. By comparing matrix Γ with Θ, the large difference verifies the detected inconsistency . An overall algorithm is shown below.

Algorithm 1 Submap-Based Inconsistency Detection

Require: The generated submaps $\{{}_i^x m\}_{i \in n_x, x \in \mathcal{R}}$
Ensure: Detecting sequential, local and global inconsistency
  for $\{{}_i^x m\}_{i \in n_x, x \in \mathcal{R}}$ with overlapping pairs **do**
    for single robot level **do**
      Sequential inconsistency: ${}_{i-1}^{i}\triangle = p({}_i^a m | {}_{i-1}^a m)_{\{i=2:n_a\}}$
      Local inconsistency: ${}_{j}^{i}\triangle = p({}_i^a m | {}_j^a m)_{\{i=k+1:n_a, j=1:i-k\}}$
    end for
    for multi-robot level **do**
      Global inconsistency: ${}_{j}^{i}\Lambda = p({}_j^b m | {}_i^a m)_{\{i=1:n_a, j=1:n_b\}}$
    end for
  end for
  Generating inconsistency matrix $\Gamma_{N \times N}$
  for $\Gamma_{(i,j)} > 0$ **do**
    Generating matrix Θ after performing map registration
  end for
  Verified inconsistency matrix $\Gamma^v = \Theta - \Gamma$



(a) The ground truth of indoor 3D map

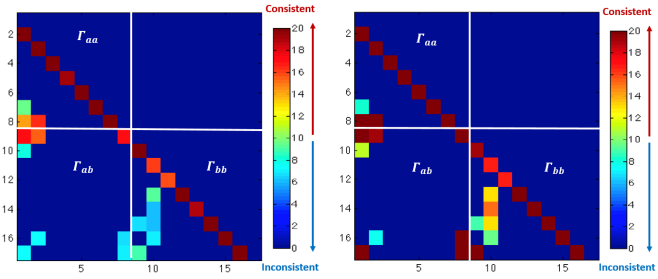


(b) The inconsistent indoor 3D map

Fig. 5. The ground truth and inconsistent map generated in indoor environment. For local map ${}^a m$ is consistent while the local map ${}^b m$ can't close the loop when it comes back to the starting point.

V. Experimental Results

Experiments performed using two robots in indoor and outdoor environments are presented in this section. Two robots were teleoperated in both indoor and outdoor environment at Nanyang Technological University. The robot was equipped with a Hokuyo Laser range finder for pose estimation using Gmapping [18] and a Velodyne VLP-16 for 3D perceptions, as shown in Fig. 4. The individually generated 3D grid maps are post-processed using C++ and Matlab to analyze their inconsistency. Submaps are generated at $10m$ intervals. The resolution of the 3D occupancy grid map in both experiments is set to be $0.1m$.



(a) Inconsistency matrix Γ (b) Inconsistency matrix Θ after map registration

Fig. 6. The deeper the color, the submap pair is more consistent. The large difference of the color in the same pixel verifies the inconsistency.

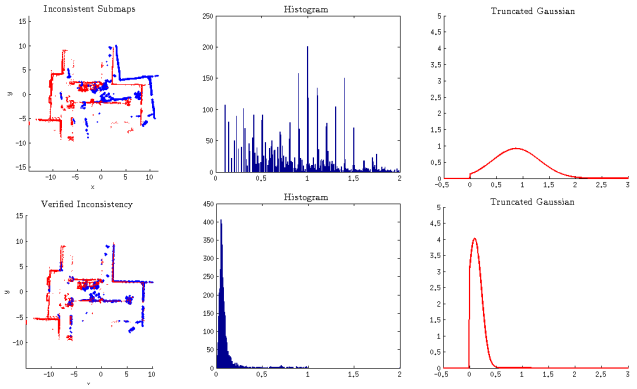


Fig. 7. The top row shows the detected global inconsistency between submap ${}^{b}_{9}m$(blue color) and ${}^{a}_{8}m$(red color), while the bottom row verifies the inconsistency after performing map registration

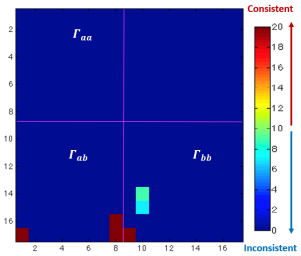


Fig. 8. The matrix Γ^{v} shows the verified inconsistent pairs by identifying the difference between Fig .6a and Fig. 6b in indoor scenario

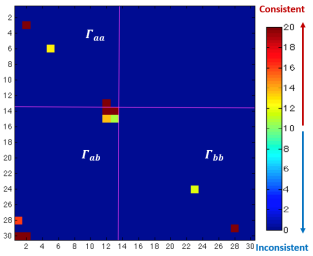


Fig. 9. Verified inconsistency matrix in outdoor environment. The deeper color represents a higher probability of inconsistency

A. Indoor Environment

The ground truth map and the inconsistent map are shown in Fig. 5. The ground truth of the map was generated with a very high number of particles and by moving the robot extremely slowly and smoothly. For experiments, the number of particles was decreased to the default value of 30. The two separated local maps are fused to generate the global map and the result is shown in Fig. 5b. For local map ${}^{a}m$, 8 submaps were generated. And 9 submaps were generated for the local map ${}^{b}m$.

Square matrix $\Gamma_{17\times17}$ is generated to describe the inconsistency as shown in Fig. 6a. As for single robot level inconsistency detection, $\Gamma_{[1:8,1:8]}(\Gamma_{aa})$ shows the inconsistency between submaps ${}^{a}_{1:8}m$, and $\Gamma_{[9:17,9:17]}(\Gamma_{bb})$ shows the inconsistency between submaps ${}^{b}_{1:9}m$. For multi-robot level inconsistency, $\Gamma_{[9:17,1:8]}(\Gamma_{ab})$ represents the global consistency between submaps ${}^{b}_{1:9}m$ and submaps ${}^{a}_{1:8}m$. The two map sessions started and ended at the same position and is marked with a star in Fig. 5b. Note that only sequential inconsistency detection is performed, so the matrix is not symmetric.

1) Single robot level: For local map ${}^{a}m$ and ${}^{b}m$, the sequential consistency ${}^{i}_{i-1}\triangle$ are accepted. As shown in matrix Γ, all the pixels representing sequential inconsistency is with high values. Local inconsistency for map ${}^{a}m$ is also accepted, which means the local loop is closed. For example, the high value of $\Gamma_{(8,1)}$ and $\Gamma_{(7,1)}$ indicates last two submaps ${}^{a}_{7}m$, ${}^{a}_{8}m$ aligned properly with first submap ${}^{a}_{1}m$. However, the low value for $\Gamma_{(13:16,10)}$, $\Gamma_{(15,9)}$, and $\Gamma_{(17,9)}$ indicate the local inconsistency detected in ${}^{b}m$. For example, $\Gamma_{(14,10)}$ shows the inconsistency between ${}^{b}_{6}m$ and ${}^{b}_{2}m$. And $\Gamma_{(17,9)}$ shows the misalignment between first submap ${}^{b}_{1}m$ and last submap ${}^{b}_{9}m$.

2) Multi-robot level: The measured global inconsistency is presented in lower left part in Fig. 6a, and is based on the overlapping area between ${}^{a}m$ and ${}^{b}m$ in Fig. 5b. Pixel $\Gamma_{(9,1)}$ is with a high value, because robot a and b started at the same position and aligned properly. Then $\Gamma_{(17,1)}$ and $\Gamma_{(16,2)}$ are with a low value, due to the misalignment between last two submaps ${}^{b}_{9}m$, ${}^{b}_{8}m$ and start position of ${}^{a}_{1}m$ and ${}^{a}_{2}m$. The low values of $\Gamma_{(16,8)}$ and $\Gamma_{(17,8)}$ also detects the inconsistency between submaps $({}^{b}_{8}m,{}^{a}_{8}m)$ and $({}^{b}_{9}m,{}^{a}_{8}m)$.

3) Map Registration: As mentioned before, inconsistency is caused by misalignment between submaps. Here, a map registration algorithm is applied to verify the potential inconsistency detected. Square matrix Θ is generated and is shown in Fig. 6b, which shows that the previously inconsistent submaps have been aligned after performing submap registration.

At single robot level, the sequential inconsistency remains unchanged, which validated our measurements. For ${}^{a}m$, the local inconsistency is also validated. For ${}^{b}m$, the local inconsistency value of $\Theta_{(14,10)}$, $\Theta_{(15,10)}$, $\Theta_{(17,9)}$ have increased sharply, which indicates these pairs have been aligned after map registration. At multi-robot level, the global inconsistency value $\Theta_{(16,8)}$, $\Theta_{(17,1)}$ and $\Theta_{(17,8)}$ have all been detected with a big change. An example of $\Theta_{(17,8)}$ is shown in Fig. 7, which is a detailed explanation of the inconsistency between submap ${}^{b}_{9}m$ and ${}^{a}_{8}m$.

4) Verified Inconsistency Matrix: The verified inconsistency matrix Γ_{v} is generated by identifying the large differences before and after map registration in corresponding consistency matrix Γ and Θ. As illustrated in Fig. 8, local inconsistency is verified in local map ${}^{b}m$ for $\Gamma^{v}_{(14,10)}$, $\Gamma^{v}_{(15,10)}$ and $\Gamma^{v}_{(17,9)}$. Global inconsistency between ${}^{a}m$ and ${}^{b}m$ is verified for $\Gamma^{v}_{(17,1)}$, $\Gamma^{v}_{(16,8)}$ and $\Gamma^{v}_{(17,8)}$. The shade of color indicates the confidence level of the inconsistency detected. Inconsistency detected in $\Gamma^{v}_{(13,10)}$, $\Gamma^{v}_{(16,10)}$ and $\Gamma^{v}_{(15,9)}$ have been rejected due to the slight change after performing map

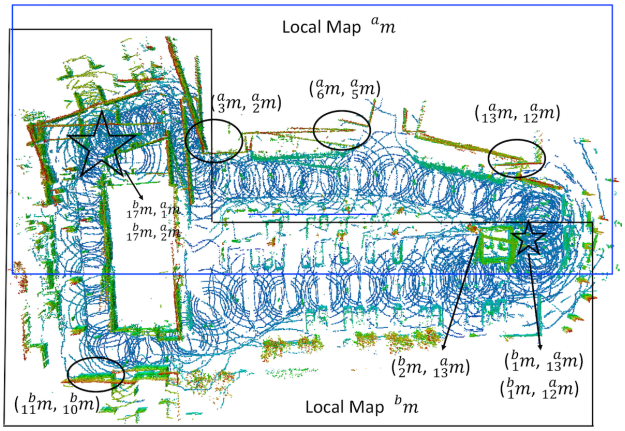


Fig. 10. The inconsistent map generated in a car park. Due to the unstructured environments and moving objects, the map is quite inconsistent. Our method is able to detect the inconsistency accurately.

registration. This is usually caused by low overlapping areas.

B. Outdoor Environment

The outdoor experiment was performed in an open car park in NTU. As can be seen from Fig. 4b, the environment was quite challenging with moving cars and people. Ground truth map was hard to generate in the environment, hence we only show the inconsistent map and inconsistency detected in Fig. 10. The start and end position are marked with stars. For local map ^{a}m, 13 submaps were generated. And 17 submaps were generated for the local map ^{b}m.

Since there is no local loop for the local maps ^{a}m and ^{b}m, only sequential inconsistency and global inconsistency is detected. The verified inconsistency matrix Γ^{v} is shown directly to describe the inconsistency in Fig. 9.

a) single robot Level: The inconsistency of robot a is shown in $\Gamma^{v}_{[1:13,1:13]}$. Pairwise inconsistency is detected in $\Gamma^{v}_{(3,2)}$, $\Gamma^{v}_{(6,5)}$ and $\Gamma^{v}_{(13,12)}$. For local map ^{b}m, the inconsistency is shown in $\Gamma^{v}_{[14:30,14:30]}$ for submaps $^{b}m_{1:17}$. Sequential inconsistency is detected in $\Gamma^{v}_{(24,23)}$ and $\Gamma^{v}_{(29,28)}$. Three pairs of inconsistent submaps are marked with ellipses in Fig. 10.

b) Multi-Robot Level: $\Gamma^{v}_{[14:30,1:13]}$ represents the global inconsistency between submaps $^{b}_{1:17}$ and $^{a}_{1:13}$. Global inconsistency is detected in $\Gamma^{v}_{(30,1)}$, $\Gamma^{v}_{(30,2)}$ and $\Gamma^{v}_{(28,1)}$. In addition, $\Gamma^{v}_{(14,12)}$, $\Gamma^{v}_{(14,13)}$, $\Gamma^{v}_{(15,12)}$ and $\Gamma^{v}_{(15,13)}$ represents the inconsistency detected between $(^{b}_{1}m,^{a}_{12}m)$, $(^{b}_{1}m,^{a}_{13}m)$ $(^{b}_{2}m,^{a}_{12}m)$ and $(^{b}_{2}m,^{a}_{13}m)$. The global inconsistency detected is shown in the overlapping area between ^{a}m and ^{b}m in Fig. 10.

VI. Conclusion

In this paper, the problem of measuring the inconsistency of maps generated by multi-robot mapping missions is addressed. We proposed a multi-robot map inconsistency detection strategy that evaluates the inconsistency on local and global levels. The inconsistency is measured by modeling the submap-wise misalignment error metric distribution into a truncated Gaussian distribution. In addition, a map registration algorithm is applied to verify the detected inconsistency. Our method successfully detects the inconsistency in the challenging indoor and outdoor environment. More importantly, as shown in Fig.(8) and Fig.(9), the inconsistency of different types(ie. sequential, local and global) are detected with a probability distribution, which can be the uncertainty of post-processed optimization or the signal of resetting mapping mission. The proposed submap-based framework is demonstrated to reduce the problem complexity and to accurately detect inconsistency in the global maps.

In future work, it can be integrated with a multi-robot exploration mission where it provides the ability to recover and continue the mission when the global map is inconsistent.

References

[1] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[2] D. Scaramuzza, M. C. Achtelik, L. Doitsidis *et al.*, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments," *IEEE Robotics Automation Magazine*, vol. 21, no. 3, pp. 26–40, Sept 2014.

[3] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[4] Y. Yue, D. Wang, P. Senarathne, and D. Moratuwage, "A hybrid probabilistic and point set registration approach for fusion of 3D occupancy grid maps," in *2016 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2016, pp. 1975–1980.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[6] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 957–962.

[7] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[8] P. Agarwal, G. D. Tipaldi, and L. Spinello, "Robust map optimization using dynamic covariance scaling," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 62–69.

[9] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1879–1884.

[10] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph slam," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[11] D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard, "Towards lazy data association in slam," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 421–431.

[12] M. Mazuran, G. D. Tipaldi, and L. Spinello, "A statistical measure for map consistency in slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3650–3655.

[13] I. Bogoslavskyi, M. Mazuran, and C. Stachniss, "Robust homing for autonomous robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2550–2556.

[14] J. L. Blanco, J. A. Fernndez-Madrigal, and J. Gonzez, "Toward a unified bayesian approach to hybrid metric–topological slam," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 259–270, April 2008.

[15] K. Ni and F. Dellaert, "Multi-level submap based slam using nested dissection," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2558–2565.

[16] M. J. Schuster, C. Brand, and b.-p. y. o. Hirschmüller, Heiko, "Multi-robot 6d graph slam connecting decoupled local reference filters."

[17] E. Rehder and A. Albrecht, "Submap-based slam for road markings," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1393–1398.

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.

Synthesized semantic views for mobile robot localization

Johannes Pöschmann, Peer Neubert, Stefan Schubert and Peter Protzel[1]

Abstract— **Localizing a mobile robot in a given map is a crucial task for autonomy. We present an approach to localize a robot equipped with a camera in a known 2D or 3D geometrical map that is augmented with semantic information (e.g., a floor plan with semantic labels). The approach uses semantic information to mediate between the visual information from the camera and the geometrical information in the map. Moreover, semantic information is robust to appearance changes like lighting conditions. Instead of solely relying on salient semantic landmarks (i.e., "things" like doors) we also exploit "stuff"-like semantic classes such as wall and floor. The presented localization approach builds upon the idea of computing a semantic segmentation of an incoming camera image using a Convolutional Neural Network and subsequent matching to semantic views synthesized from a map. We give details about the algorithmic approach on how to semantically segment images, synthesize images from the semantic 2D or 3D map, the matching between images from both sources, and the integration in Monte Carlo localization. Further, we provide a set of proof-of-concept experiments and evaluate the influence of the selected set of semantic classes. To work towards the usage of hand-drawn sketches as input map, we also evaluate the robustness of the presented approach to map distortions.**

I. INTRODUCTION

The ability to recover and maintain knowledge about the own position in the world is essential for mobile robots. The usage of a priori known maps can significantly facilitate this task, reducing the SLAM problem to a pure localization. Moreover, if we can use cameras for this localization task, the expected sensor costs are small and we can also transfer this capability to other mobile sensor devices than robots. But where do the maps for visual localization come from? Instead of mapping the environment with cameras in advance, in this paper, we work towards exploiting other sources of maps: known 2D floor plans and hand-drawn top-view sketches of the environment. To allow for comparison with the actual camera view, the maps are augmented with semantic information about walls, doors, and other objects.

The way we use the robot's current camera image and the map information for localization is partially inspired by the navigation system of desert ants presented by Möller [1]. As climatic conditions doesn't allow the usage of pheromone traces, desert ants developed a vision-based navigation approach: Before they leave their nest's location, a snapshot (home view) of the environment is taken. After an ant has finished the foraging, it has to find its way back to the nest. Therefore, it starts to compare the home view with *synthetic* views: distorted current views that are obtained by transforming the current image into possible motion directions – the most similar image pair then indicates the correct homing direction. Details on a technical realization of the *ant algorithm* on mobile robots are provided by Möller [2].

In the here presented approach to localization using the given quite different modalities (a 2D plan and a visual sensor), we exploit recent advances in Deep Learning to assign pixel-wise *semantic* information to images. Semantic information serves as an intermediate layer between a current camera image and a geometrical map in order to facilitate the matching between both modalities. Semantic labels represent the world in a rather abstract manner; this prevents a camera-map matching algorithm from being sensitive to appearance changes caused by lighting or rotation (e.g., swivel chair). For a 2D map, semantic information about walls, floor, windows, etc. can be automatically extracted from construction plans or feasibly drawn into 2D floor plans by humans. Humans can also easily integrate additional semantic classes like furniture or plants. Starting from this 2D map, we create a semantic 3D map using knowledge about the occurring classes. Localization is done in a Monte Carlo manner using synthesized semantic views from this map. A future use case would be a human operator sketching the path for a robot's delivery task on a tablet and adding some semantic information, encoding information like: "Follow that way and go left after the third tree".

In this paper, we

- present an approach to localization in given 2D semantic maps using synthesized semantic images and ConvNet-based semantic segmentations from a robot's camera
- describe the implementation in a Monte Carlo localization system
- provide proof-of-concept experiments as well as an evaluation of influences of chosen semantic classes and robustness to geometrical map errors

II. RELATED WORK

The task of localizing camera images within a given map by image synthesis can be addressed in different ways. Our previous work [3] uses the geometrical information of a map for a depth image synthesis at desired positions with a subsequent comparison to a current greyscale camera view. Instead of using distance information of a map directly, Wolcott and Eustice [4] use a map enhanced with intensity information to synthesize greyscale-similar images for a subsequent comparison to the current view, and the approach of Pascoe et al. [5] builds upon a map with colour information for image synthesis. Caselitz et al. [6] show an alternative approach for the matching of a given map with a current camera image: Instead of using the map to

[1]The authors are with Chemnitz University of Technology, Germany firstname.lastname@etit.tu-chemnitz.de

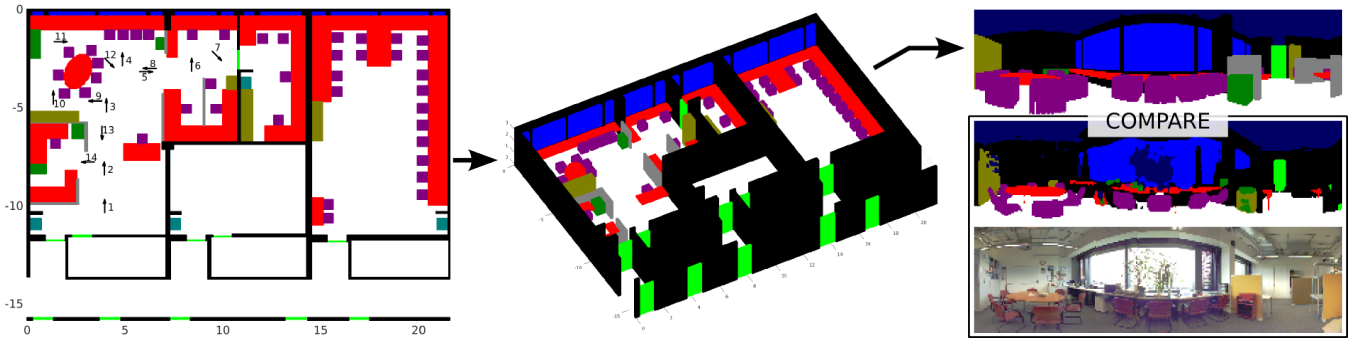


Fig. 1. Left: Accurately hand-drawn 2D semantic top-view map of an indoor environment, consisting of 10 different object classes. Annotated are the 14 ground truth positions of the robot during our real world experiments. This map is used to generate a 3D semantic pointcloud (centre), in which 2D semantic panorama images are synthesized (top right). Bottom right: Camera images are fed into a CNN for image segmentation, resulting in a semantic panorama image of a real world scene. Comparison of both semantic panoramas yields a image similarity and enables localization.

generate synthesized images, they used the camera image stream to reconstruct the environment's geometrical structure for a subsequent point cloud matching.

Another way of localization within a given map is to exploit the semantic information of the environment. Kuiper's spatial semantic hierarchy paradigm [7] uses semantic information as a topological map. Vasudevan et al. [8] add semantic information to observed objects (doors and household objects) in order to build a hierarchical topological-semantic map, with which they can annotate places (e.g., office, corridor). Another approach is landmark-based localization. Atanasov et al. [9] and Anati et al. [10] use object detectors to recognize landmarks, which are annotated in the given map, followed by particle filter localization. Our work is closest related to [10]. Through the application of a soft object detector, they generate a heatmap with semantic information about the occurring objects for a set of panoramic camera images and compare them with the expected heatmap at each particle location. In contrast, we propose a camera-based semantic localization algorithm which does not rely on specific, distinctive object classes, but rather works with non-expressive classes like wall and floor. Furthermore, we evaluate the robustness of our systems towards distortions of the given semantic map. This opens a variety of use cases like localization within a human-made sketch of the environment.

III. ALGORITHMIC APPROACH

Fig. 1 illustrates the involved steps. Starting from a 2D floor plan with semantic labels, a 3D map is created. For an assumed camera pose, we can synthesize a label image. Given a camera image taken from the real robot pose, we can compute a second label image using a semantic segmentation algorithm. By comparing the two label images, we can evaluate the assumed robot pose. This approach directly integrates in the well-know Monte Carlo localization (e.g., see [11] for an introduction): each sample provides an assumed robot pose that can be evaluated using the above approach.

A. Synthesizing images from a given 2D semantic map

The task is to localize a robot within a 2D semantic map. The map can be automatically generated (e.g. from floor plans), augmented by humans (e.g., with the semantic labels) or completely hand-drawn. In the later presented experiments, a simple, colour-coded 2D map of the environment is used (each class has a different colour). Given a 2D floor plan, such a map can be easily created by a human with some basic image editing software. All objects occurring in the real world can be freely drawn within the map. Each object class (e.g., wall, floor, table) is represented by a different colour and is associated with a class-label and a minimum and maximum height. With this information, the 2D semantic map can be converted into a 3D semantic point cloud. A 2D semantic map and the corresponding 3D point cloud of an indoor environment are shown in Fig. 1.

To synthesize an image given this 3D point cloud (with associated semantic class labels) and a requested camera pose, we follow the straight forward approach described in [3]: Each 3D point's distance is projected onto a unit sphere centred at the requested camera pose. The azimuthal and polar angles are discretized to the target image resolution. By keeping only the class label of the point with minimal distance for each direction, this spherical grid corresponds to the synthetic label image; pixels without projected 3D points are set to NaN values. In contrast to the representation used in [10], this allows for seeing multiple objects on the vertical image axis. Again, see Fig. 1 for an example.

This preliminary approach is easy to implement but its runtime is linear in the number of 3D points. Presumably, the runtime could be improved by the usage of computer graphics techniques including ray tracing algorithms and efficient data structures like k-d trees or octrees.

B. Semantic image segmentation

To be able to compare synthesized semantic images with real world images, we need to associate semantic information to each pixel in an image. This is a well known task, called semantic segmentation, for which the currently best performing methods build upon Deep Learning techniques. We use the "Pyramid Scene Parsing Network (PSPNet)" [12], which bases on a Convolutional Neural Network and achieved first rank in ImageNet scene parsing challenge 2016. We used an out of the box CNN, trained on the ADE20k dataset [13], which includes 150 object classes,

containing both indoor and outdoor class instances. Given an image as input, the PSPNet associates each pixel with one of the 150 object classes. Afterwards, the resulting semantically segmented images are transformed into spherical coordinates in order to correspond to the synthesized-semantic-image shape and thus enabling a holistic image comparison between both images (see Fig. 1).

C. Holistic image comparison

We decide to use a holistic image comparison instead of landmark based approaches, since we also want to exploit often occurring, less-expressive classes like walls or floor ("stuff" in contrast to "things" [14]) for localization. Consequently, a key component of our approach is an expressive similarity metric for a holistic comparison of semantic images. Therefore, a weight matrix is used that scores each assignment of pixels of the 150 ADE20k classes used by the PSPNet to pixels with classes present in our 2D semantic map. Currently, the values of the weight matrix are set empirically based on the following three key concepts:

1) Soft class matching: Full similarity between matching classes (e.g., desk and table) results in a score of 1, whereas partial similarity between related classes (e.g., chair and sofa) results in a score between 0 and 1.
2) Weighted class significance: Often occurring classes (e.g., wall or floor) are not expressive, leading to a smaller score between 0 and 1.
3) Class matching penalty: Overlapping of two contradicting classes leads to a score between -1 and 0. Penalization of, e.g., overlapping floors and walls results in a precise distance measurement to nearby walls.

Synthesized and segmented images are compared pixel-wise with the following score function:

$$similarity = \frac{\sum_{pixels} w_{i,j}}{\sum_{pixels}} \tag{1}$$

where w_{ij} is the weight/score for the linkage between class i from a segmented image and class j from a synthesized image. With a similarity of 1, the two images are identical. Negative similarities are set to 0.

D. Monte Carlo localization

Algorithm 1 gives details on how the described image synthesis and comparison approach integrates in Monte Carlo localization. The main loop in line 3 processes each image and applies the described image synthesis and comparison. A semantic segmentation J_t is computed for the current camera image I_t (line 4). We use two different strategies to generate a particle heading direction.

1) **Odometry heading**: Either we sample for each particle (the loop starting in line 6) odometry and compare a synthesized semantic image S with the semantic segmentation J_t using equation 1 (line 17).
2) **Visual compass**: Or we apply for each particle an image comparison between J_t and multiple S with different directions and use the direction with highest image similarity as the particles direction (similar to a visual compass).

Algorithm 1: Semantic Monte Carlo localization

Data: Semantic 3D map M, image sequence $I_{1:n}$, odometry measures $U_{1:n}$
Result: Particle set P, each $p_j \in P$ is a robot pose

```
// initialize particle set
1  P ←initParticles()
2  S_cache ← prepareSynthImageCache(M)
   // process each image
3  for t = 1 : n do
4      J_t ← computeSemanticSegmentation(I_t)
5      E_cache ← prepareSimilarityEvaluationCache(M)
6      foreach p_j ∈ P do
           // sample motion from odometry
7          p_j^u ← applyOdometry(p_j, U_t)
           // query cache for existing similarity
              evaluation E of this pose
8          if isInCache(E_cache, p_j^u) then
9              E ← queryCache(E_cache, p_j^u)
10         else
               // query cache for existing
                  synthesized image
11             if isInCache(S_cache, p_j^u) then
12                 S ← queryCache(S_cache, p_j^u)
13             else
14                 S ← synthesizeImage(M, p_j^u)
15                 S_cache ←insertInCache(S_cache, S, p_j^u)
16             end
               // compare images
17             E ← compareImages(J_t, S)
18             E_cache ←insertInCache(E_cache, p_j^u, E)
19         end
           // obtain sampling weight from E
20         w_j ← getWeight(E)
21     end
       // importance sampling
22     P ← importanceSampling(w, P)
23 end
```

Both strategies are described in detail in [3] and are evaluated in section IV. The resulting similarity is used to compute a weight w_j for the particle resampling in line 22. For resampling, we use the low variance sampler from [11].

A major bottleneck for runtime could be the image synthesis (particularly since we use the straight forward approach described in section III-A). To keep runtime feasible, the above algorithmic listing includes two caching stages to exploit the property that many particles are very close to each other: S_{cache} stores all already synthesized images, its lifetime corresponds to the lifetime of the map. The second cache E_{cache} holds the comparison result for a synthesized image and a current camera image. It is cleared with each new camera image. To effectively use these caches, we have to discretize the set of possible poses. Dependent on the application, the synthesized images could also be computed offline in advance. In our experiments, we precompute synthesized images at a spatial resolution of 10 cm in x and y direction and 3° in angular direction.

IV. EXPERIMENTAL SETUP

In order to evaluate the robustness of our algorithm against map distortions and the selected semantic classes, we applied our semantic Monte Carlo localization approach in a set of proof-of-concept experiments.

A. Experimental setup

An indoor **scene** with two labs, some smaller rooms and a corridor serves as experimental environment. Building upon an existing floor plan, we created a metrically exact semantic 2D map and manually augmented it with a set of basic object classes. With the idea of a hand-drawn map in mind, we constrained the number of classes in our semantic map to 10: wall, dividing wall, floor, door, window, table, chair, cupboard, sideboard, and sink. The resulting semantic map is shown in Fig. 1. For our proof-of-concept experiments, we collected camera images, odometry and ground truth pose data of our mobile robot at 14 different positions (see Fig. 1).

For **data acquisition**, we used a skid-steering mobile robot equipped with a *uEye UI-1240ML-C-HQ* RGB camera mounted on a pan-tilt unit; details on our mobile robot can be found in [15]. By using a wide-angle lens, the camera captures images with 76° vertical and 95° horizontal field of view. Since the pan-tilt unit can be moved in the range of approx. $-90°$ to $+90°$, we were able to capture multiple images at one location to enhance the field of view. For each robot position, 7 camera images are semantically segmented, transformed into spherical coordinates, and stitched into a single 268° panoramic image. A high number of camera images is used to avoid distortions at the image borders of raw images (without rectification).

The choice of the **weight matrix** in the holistic image matching is crucial. As the classes wall and floor appear more often in an indoor environment, the score for a correct match was reduced from 1 to 0.25, whereas, more expressive classes have higher impact on the similarity score. A match between wall/dividing wall and floor is scored with -1. Thereby the image similarity receives a huge penalty if the edges between floor and wall do not match in both images, leading to a localization with correct distances to nearby walls. Furthermore, we apply soft class matching as ADE20k contains many similar classes, e.g., chair, armchair, seat, swivel chair, stool, bench, sofa, etc.. All these classes need to be linked to our general class chair, with a matching score between 0 and 1. For our proof-of-concept experiments, the matching scores (and thereby the weight matrix w) are chosen empirically, following the design guidelines presented in section III-C. As a baseline, we use an identity matrix, in which all scores between classes are either 0 or 1. Complete weight matrices are available upon request.

We conduct the following **experiments**:

1) Local and global localization. We run all setups on two tasks: (1) "local" localization with known initial position and 500 particles in the particle filter and (2) "global" localization with unknown start position and 2000 particles. We choose a high number of particles for the known initial position case, since we need to apply high uncertainty (σ=0.5 m) to our robot's odometry due to the severe map distortions (see Fig. 2).
2) Visual compass vs. odometry heading.
3) Scaling of the semantic map with factors ranging between 1.1 and 1.5 in one direction and 0.91 and 0.67 in the other direction (see Fig. 2).
4) Shear of the semantic map in both directions with a factor between 0.1 and 0.5 (see Fig. 2).
5) Variation of classes in the map with three cases: (1) construction basic: containing wall, dividing wall and floor; (2) construction extended: containing case (1) and door and window; (3) furniture: containing table, chair, cupboard, sideboard and sink.
6) Variation of panoramic field of view with following steps: 50°, 100°, 150°, 200° and 268°.

We conduct 10 runs of each experiment to account for the stochastic nature of Monte Carlo localization. We use the Euclidean distance to the ground truth position for evaluation metric and compare against plain odometry measures of our mobile robot.

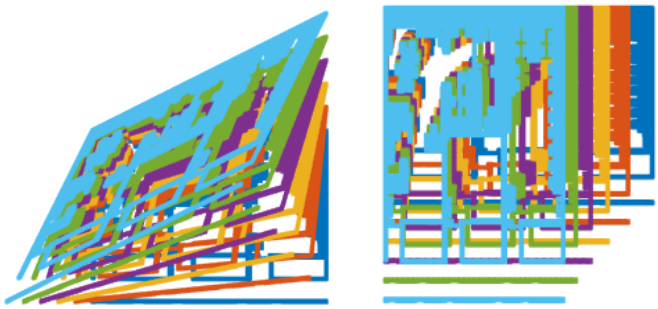
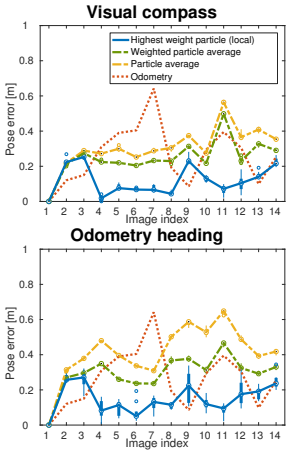


Fig. 2. Left: Original 2D semantic map (see Fig. 1) is shown dark blue. The five overlaid images depict a map sheared with factors ranging between 0.1 and 0.5. Right: The same for scaled maps with factors ranging from 0.91 to 0.67 in x direction and 1.1 to 1.5 in y direction.

B. Experimental results

Evaluation results of the presented approach using the **visual compass and odometry heading** strategy are shown in Fig. 3 (left). Both strategies result in lower maximum and average pose error than the plain odometry. The visual compass based strategy achieves high localization accuracies with distances to ground truth positions constantly below 30 cm, at some locations close to zero. Since our holistic image comparison depends on a reasonable alignment between synthesized image S and segmented image J_t, the multiple sampled directions in visual compass result in more precise and repeatable localizations. Comparison of average particle distance and average weighted distance shows, that our image similarity metric is feasible, since particles with a higher weight are closer to ground truth positions than particles with a lower weight. In some (not shown) experiments the odometry heading strategy provided better results, presumably due to visual aliasing that can occur for the visual compass method. In summary, the visual compass strategy achieves the overall better results and is therefore used in all following experimental results.

The influence of **map scaling** (see Fig. 2) is shown in the first row of Fig. 4. As expected, errors rise with growing scaling factors. Up to a scaling by factors 1.40 & 0.71, a reliable localization, with considerably smaller errors than robot's odometry, is possible. Global localization is achieved within the first three images for all cases.



Experiment	Identity matrix		Weighted matrix		
	highest weight	min dist. to GT	highest weight	min dist. to GT	odometry
Unmodified semantic map	0.160 m	(0.023 m)	**0.120 m**	(0.020 m)	0.260 m
Scaling by 1.10 & 0.91	0.267 m	(0.034 m)	**0.176 m**	(0.033 m)	0.488 m
Scaling by 1.20 & 0.83	3.036 m	(0.078 m)	**0.363 m**	(0.052 m)	1.052 m
Scaling by 1.30 & 0.77	6.836 m	(0.671 m)	**0.569 m**	(0.073 m)	1.625 m
Scaling by 1.40 & 0.71	8.376 m	(1.565 m)	**0.686 m**	(0.097 m)	2.197 m
Scaling by 1.50 & 0.67	8.352 m	(1.991 m)	**1.176 m**	(0.122 m)	2.768 m
Shearing by 0.10	0.287 m	(0.028 m)	**0.165 m**	(0.026 m)	0.632 m
Shearing by 0.20	1.396 m	(0.039 m)	**0.391 m**	(0.042 m)	1.213 m
Shearing by 0.30	2.956 m	(0.060 m)	**0.534 m**	(0.056 m)	1.808 m
Shearing by 0.40	3.441 m	(0.152 m)	**0.790 m**	(0.079 m)	2.407 m
Shearing by 0.50	4.063 m	(0.388 m)	**1.234 m**	(0.118 m)	3.006 m
200° field of view	0.160 m	(0.024 m)	**0.126 m**	(0.023 m)	0.260 m
150° field of view	0.852 m	(0.169 m)	**0.189 m**	(0.023 m)	0.260 m
100° field of view	0.969 m	(0.210 m)	0.284 m	(0.026 m)	**0.260 m**
50° field of view	1.712 m	(0.082 m)	0.367 m	(0.051 m)	**0.260 m**
Classes construction ext.	**0.174 m**	(0.024 m)	0.186 m	(0.024 m)	0.260 m
Classes construction basic	1.956 m	(0.025 m)	**0.252 m**	(0.027 m)	0.260 m
Classes furniture	2.674 m	(1.069 m)	0.678 m	(0.266 m)	**0.260 m**

Fig. 3. Left: Results for particle heading strategies visual compass and odometry heading (described in III-D) applied to the unmodified semantic map. Each blue (green, yellow) curve shows the median result of the (x,y) pose error of the particle with the highest weight (weighted average particle distance, average particle distance) over ten runs and is augmented with boxplots for each image of the sequence (25th and 75th percentiles, whisker scale is 1.5, outliers are shown as circles). Right: Comparison of highest weight particle, minimal distance particle and odometry by means of mean Euclidean distance to ground truth positions over ten runs under usage of the visual compass method for local localization.

The influence of **map shearing** (see Fig. 2) is shown in the second row of Fig. 4. Our semantic Monte Carlo localization achieves small localization errors and high repeatability up to a shearing with factor 0.4. A shearing factor of 0.5 leads to severe localization error fluctuations between single runs. Furthermore, localization gets lost in the last three images. This is caused by ambiguous image similarity results for image 12, leading to different resulting paths. Nevertheless, global localization is achieved for all cases within first three images. Summarizing, our semantic Monte Carlo localization achieves remarkably good localization results for map shearing with factors up to 0.4.

The influence of **horizontal field of view** is shown in the third row of Fig. 4. With only $50°$ field of view, a wide fluctuation between localization errors are observable, since information provided by the image is very limited. Still, our semantic Monte Carlo localization algorithm is able to localize from an unknown initial position, as in all other cases. With more than $150°$ the image contains enough information for a reliable localization. Summarizing, a large field of view panoramic image is crucial for reliable localization results.

The influence of **classes within the semantic map** is shown in the fourth row of Fig. 4. Localization with only furniture classes did not work in this experiment: Without walls, the synthesized images contain a lot of objects from adjacent rooms, which are not present in the real camera images (since there are walls, of course). Furthermore a lot of information is missing in scenes, for which our robot mainly observes walls and floor. On the other hand, walls, dividing walls and floor ("construction basic") offer enough information for a coarse but reliable localization in the margin of our robot's odometry. With additional information from classes window and door ("construction extended"), localization accuracies rise as expected. Summarizing, our semantic Monte Carlo localization depends on the classes wall and floor, which offer enough information for a rough localization and prevent confusing views into adjacent rooms. Additional classes facilitate a precise localization.

The achieved localization results are summarized in Fig. 3 (right). The baseline approach, an identity matrix for image comparison, is sensitive for map perturbations, thus reliable results are only achieved within slightly modified maps. The minimal particle distance shows, that the ground truth position is not lost in most cases, but the particles rather diverged into multiple clusters. Sometimes the highest weight particle is located in a cluster far from the ground truth position because the identity matrix leads to ambiguous similarity measurements. Therefore, the choice of the weight matrix is essential for our semantic Monte Carlo localization, since it considerably improves robustness against map perturbations.

V. Discussion

In this paper, we presented a semantic Monte Carlo localization algorithm based on holistic image comparison of a synthesized map view and the result of a semantic image segmentation. A set of prove-of-concept experiments showed its ability to localize a mobile robot, equipped only with a RGB camera and odometry sensors, within a given 2D semantic map. Furthermore, we applied a wide range of perturbations to the given semantic map, in order to investigate our system's robustness. Experimental results are promising, since reliable local and global localization is achieved in a wide range of map deformations. Additionally, our semantic Monte Carlo localization algorithm does not depend on distinctive classes or landmarks, but rather on common classes like walls and floor, with which it achieves a rough localization. Also, our system depends on a large field of view panoramic image.

Currently, a key part of our localization algorithm, holistic image comparison, is handcrafted and uses empirically chosen parameters (e.g. weight matrix w). Presumably, our system could greatly benefit from better image similarity measurements, e.g., using (supervised) learning of class

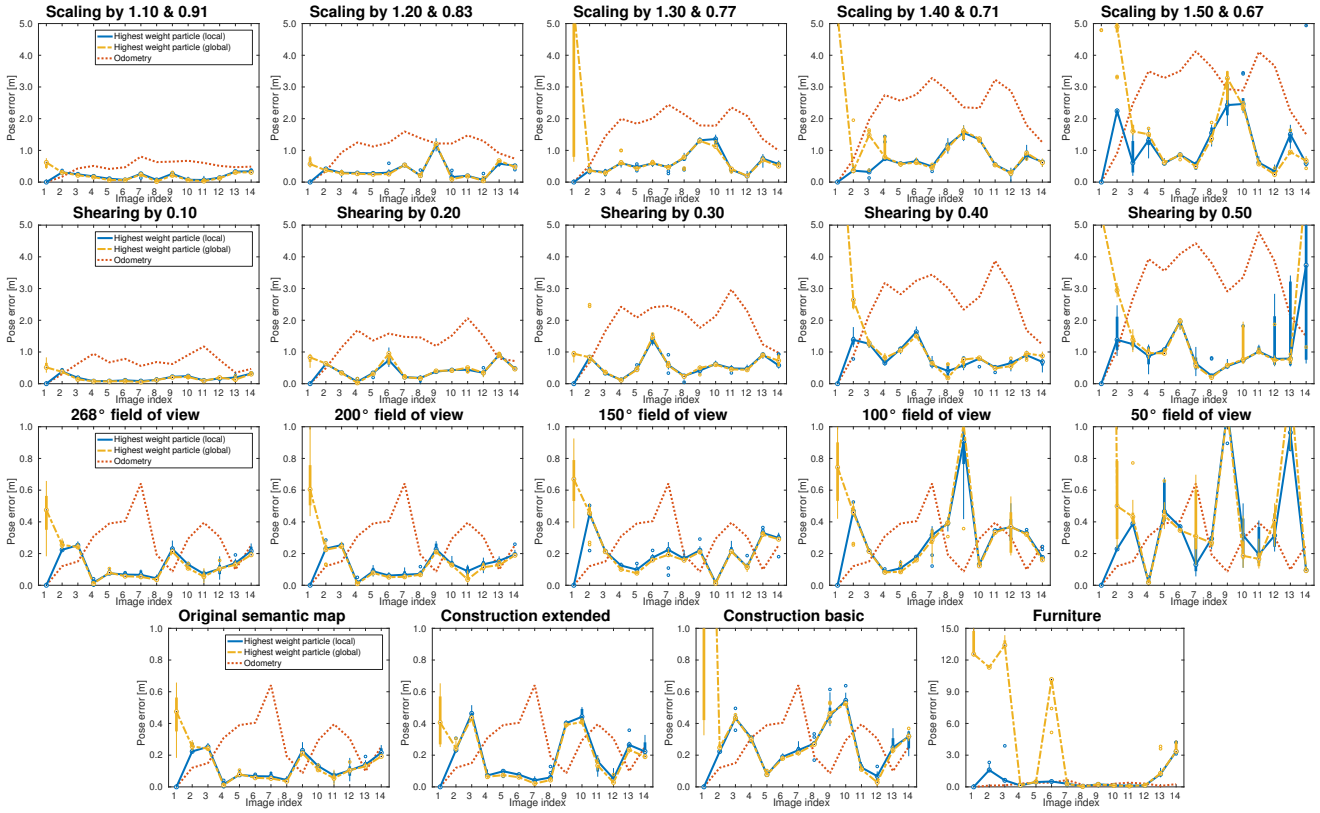


Fig. 4. First row: Results for synthetic scaling of the original semantic map (see Fig. 2). Second row: Results for synthetic shearing of the map (see Fig. 2). Third row: Results for different horizontal field of views. Fourth row: Results for variation of classes in the semantic map with three cases (see IV-A for more information). Each orange curve shows the (x,y) pose error of the robot's odometry. Each blue respectively yellow curve shows the median result of the (x,y) pose error of the particle with the highest weight over ten runs and is augmented with boxplots for each image of the sequence (25th and 75th percentiles, whisker scale is 1.5, outliers are shown as circles) in the case of local respectively global localization.

weights and mutual influences. Further, the evaluation of the different strategies for heading direction estimation (visual compass or odometry) motivates the development of a combined approach as direction for future work.

Based on the achieved results, our future work aims at applying our semantic Monte Carlo localization algorithm to real world scenarios. One promising use case is localization in truly hand drawn sketches, since our localization algorithm is robust against map deformations, enabling a variety of applications in human robot interaction, e.g. navigation of autonomous transportation robots in a construction site based on a sketch of the surroundings. Another reasonable use case is localization within a building based on fully automatic preprocessed floor plans.

References

[1] R. Möller, "A model of ant navigation based on visual prediction," *Journal of Theoretical Biology*, vol. 305, pp. 118 – 130, 2012.

[2] ——, "Local visual homing by warping of two-dimensional images," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 87 – 101, 2009.

[3] S. Schubert, P. Neubert, and P. Protzel, "Towards camera based navigation in 3d maps by synthesizing depth images," in *Proc. of. Towards Autonomous Robotic Systems (TAROS)*, 2017.

[4] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Sept 2014, pp. 176–183.

[5] G. Pascoe, W. Maddern, A. D. Stewart, and P. Newman, "Farlap: Fast robust localisation using appearance priors," in *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2015.

[6] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 1926–1931.

[7] B. Kuipers, "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, no. 1, pp. 191 – 233, 2000.

[8] S. Vasudevan, S. Gchter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robotsan object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359 – 371, 2007.

[9] N. A. Atanasov, M. Zhu, K. Daniilidis, and G. Pappas, "Semantic localization via the matrix permanent," in *Robotics: Science and Systems (RSS)*, July 2014. [Online]. Available: http://terraswarm.org/pubs/305.html

[10] R. Anati, D. Scaramuzza, K. G. Derpanis, and K. Daniilidis, "Robot localization using soft object detection," *2012 IEEE International Conference on Robotics and Automation*, pp. 4992–4999, 2012.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[12] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016.

[13] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] D. A. Forsyth, J. Malik, M. M. Fleck, H. Greenspan, T. K. Leung, S. Belongie, C. Carson, and C. Bregler, "Finding pictures of objects in large collections of images," in *Proceedings of the International Workshop on Object Representation in Computer Vision II*, ser. ECCV '96. London, UK, UK: Springer-Verlag, 1996, pp. 335–360. [Online]. Available: http://dl.acm.org/citation.cfm?id=645306.648641

[15] S. Lange, D. Wunschel, S. Schubert, T. Pfeifer, P. Weissig, A. Uhlig, M. Truschzinski, and P. Protzel, "Two autonomous robots for the dlr spacebot cup - lessons learned from 60 minutes on the moon," in *Proc. of ISR: International Symposium on Robotics*, 2016.

Adaptive Sampling-based View Planning under Time Constraints

Lars Kunze[1] and Mohan Sridharan[2] and Christos Dimitrakakis[3] and Jeremy Wyatt[4]

***Abstract*— Planning for object search requires the generation and sequencing of views in a continuous space. These plans need to consider the effect of overlapping views and a limit imposed on the time taken to compute and execute the plans. We formulate the problem of view planning in the presence of overlapping views and time constraints as an *Orienteering Problem* with history-dependent rewards. We consider two variants of this problem—in *variant (I)* only the plan execution time is constrained, whereas in *variant (II)* both planning and execution time are constrained. We abstract away the unreliability of perception, and present a sampling-based view planner that simultaneously selects a set of views and a route through them, and incorporates a prior over object locations. We show that our approach outperforms the state of the art methods for the orienteering problem by evaluating all algorithms in four environments that vary in size and complexity.**

I. Introduction

Planning of visual search for objects in large continuous spaces is an open problem with many aspects that make existing solutions inadequate. A set of views must be selected from an infinite number of possible views. To provide any preference ordering over this infinite set, it is necessary to have prior information about where objects might be found, and for that prior to be non-uniform. This formulation of the task of finding the best set of views is a sub-modular problem, and greedy solutions can be shown to have bounded sub-optimality. However, it does not take into account the costs of sequencing those views—the best set of views may, for instance, be time consuming to visit, and there may be another set of views that is slightly worse in terms of the information provided but vastly quicker for a robot to traverse.

The view planning problem is further complicated by four issues. First, some views will overlap and the value of a view, at any point in a sequence of views, will depend on the sequence of views taken so far. Since the values of views are history dependent, the problem ceases to be sub-modular. Second, the sensing process should ideally be modeled as being unreliable, transforming the problem from one of planning in a physical space, to one of planning in belief space. Third, in many practical applications, the time available to execute the planned trajectory of views is constrained. Fourth, existing view planning approaches have not considered the length of time used to plan, which will determine how much of the plan can be executed.

[1]Oxford Robotics Institute, Dept. of Engineering Science, University of Oxford, United Kingdom, lars@robots.ox.ac.uk

[2]Department of Electrical and Computer Engineering, The University of Auckland, New Zealand, m.sridharan@auckland.ac.nz

[3]Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden, chrdimi@chalmers.se

[4]Intelligent Robotics Lab, School of Computer Science, University of Birmingham, United Kingdom, jlw@cs.bham.ac.uk

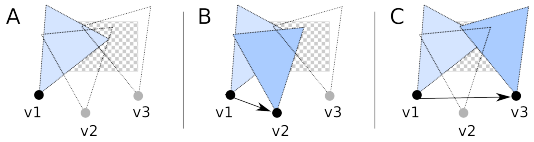


Fig. 1: View planning with overlapping views and time limit. Imagine three possible views onto a table, of which a robot can only observe two in the time available. If view $v1$ is taken first (A), then the robot can next take $v2$ or $v3$ (B or C). The reward for each of these two views depends on its overlap with $v1$. The robot must account for both history-dependent rewards and navigation costs.

We propose a solution to the first, third, and fourth of these issues related to the challenging problem of view planning with overlapping views and time constraints. We do not consider unreliable perception, which transforms view planning into a yet worse class of problems—we leave this for future consideration. In this paper, we refer to the problem of bounded execution time as *variant (I)*, and to the problem of bounded planning and execution time as *variant (II)*. We assume the ability to use sensor inputs to generate 2D and 3D maps of the environment, for navigation and object recognition; and assume prior knowledge about the locations of objects. We make the following key contributions:

- We show that the view selection and view sequencing problem can be posed as an orienteering problem (OP) with redundant views and history-dependent rewards.
- We present a Gibbs sampling-based view planning algorithm (GVP) that produces approximate solutions, but will provably converge in the limit to an optimal sequence given a finite set of views.
- We extend our algorithm (GVP) to learn, from a batch of example problems, how to divide the available time between thinking and acting.
- We evaluate our approaches on a range of environments under different time constraints and compare them to the state of the art.

To locate any given object, our algorithm (GVP) first generates a much larger number of candidate views than can possibly be searched in the available time, orders them by the probability of providing a view of the object, and selects the m best views for subsequent analysis. A sampler incrementally selects a sequence of views that simultaneously maximizes the likelihood of finding the object, considering the field of view overlap and the viewing history, and minimizes the time taken to do so—see Fig. 1. Many such sampling processes are run, each of which is terminated when the time limit is reached, and the robot chooses the best sequence of views.

For *variant (I)*, we compare GVP to a randomized OP solver, and to two solvers for the traveling salesperson problem (TSP) (1) a fast but sub-optimal greedy TSP solver (TSP-G); and (2) an optimal but slow TSP solver based on dynamic programming (TSP-DP). We show that GVP typically outperforms other algorithms for problems of different sizes and different bounds on execution time. On small problems, TSP-DP produces better solutions, but it is challenging to terminate in reasonable time for larger problems–GVP is thus an effective, if sub-optimal, solution. For *variant (II)*, we evaluate the performance of our algorithm in simulated environments, for object search tasks. We show that our adaptive algorithm outperforms fixed policies.

The remainder of the paper is organized as follows. We discuss related work in Sec. II, and the problem formulation in Sec. III. Sec. IV describes our sampling-based view planning algorithm and implementation details. Sec. V discusses experimental results, and we conclude in Sec. VI.

II. Related Work

Early work on object search discussed its intractability in a continuous space, even under some simplifying assumptions [1]. Subsequent approaches have employed different strategies to address the complexity, e.g., visual saliency [2]; planning at the level of rooms and views with associative knowledge [3], [4]; and search with qualitative spatial relations [5]. Some approaches assume reliable observations, whereas others reason under partial observability using a mix of qualitative and probabilistic reasoning for efficient search, e.g., for estimating target location at the level of rooms [6] or locations in rooms [7], [8].

The fundamental problem of view planning for object search can be viewed as a generalization of the *art gallery problem* and the *watchman problem*. The art gallery problem is NP-hard, and is a generalization of the guarding problem where sensors have infinite range, bounded only by obstacles [9]. A randomized algorithm developed for this problem provides a high probability of bounded error between its coverage and the optimal solution [10]. Approximate algorithms have been proposed for art gallery problem sequenced by a traveling salesman problem (TSP) [11]. With unreliable sensing, the joint art-gallery and watchman problem is a continuous space, continuous action POMDP, but even discrete state, discrete action POMDPs can become computationallu intractable. Researchers have partially addressed this intractability through hierarchical decompositions [4].

Our view planning problem is related to planning in belief space, e.g., for task and motion planning [12], although we do not plan in belief space. Also, unlike probabilistic roadmaps [13], we do not restrict the solution space too much, since we consider time constraints. There is also some relation to work in temporal logic planning [14], although existing approaches do not address the issues of interest and do not scale like our approach.

Our view planning problem is most related to the orienteering problem (OP) [15]. In an OP, a rewarding sequence of locations must be visited, where each location can have (in general) a different associated reward. We use a sampling-based algorithm for the OP as one of the baselines for comparison [16]. OPs, are however, typically stationary reward problems, whereas we must solve a varying-reward OP, also called general OP [17].

We assume reliable perception and work with a continuous state, continuous action MDP with a history-dependent reward function. Our novel contribution is an algorithm for the joint problem of selecting views (art-gallery) and planning a route in continuous space. We present a randomized algorithm that interleaves the selection of views with the selection of the route. Unlike previous work, including those that used stochastic branch and bound [18] or Monte-Carlo tree search [19], our method has very low space complexity.

III. Problem Formulation

We decompose the problem of view planning for object search with time constraints in a continuous environment into two parts: (i) transforming the continuous problem into a discrete problem; and (ii) solving the discrete problem using a sampling-based approach. We consider the discrete problem as an OP with history-dependent rewards, i.e., given a fully connected graph of locations $s \in S$, a cost function $C(s, s')$, and a time limit T, maximize the expected reward $R(s_0, s_1, \dots)$ obtained from a sequence of visited locations—the reward of a location depends on the locations that have been visited before. We investigate two variants of this problem formulation. *Variant (I)* considers only the execution time (T_E) when solving the view planning problem within a time limit, i.e., $T_E \leq T$. *Variant (II)* considers both planning time (T_P) and execution time (T_E), i.e., $(T_P + T_E) \leq T$.

IV. Sampling-based View Planning

This section provides an overview of our algorithm, followed by a detailed description (Sec. IV-A), an adaptive extension (Sec. IV-B), and the implementation (Sec. IV-C). We assume that we are given a: (1) 2D map (M_{2D}); (2) 3D map (M_{3D}); (3) probability distribution P of a robot at location s observing an object of a certain type; and (4) cost function $C(s, s')$ that provides the temporal cost of moving between locations s and s'.

To search for objects within time limit T, the robot generates a trajectory $\boldsymbol{s} = s_0, s_1, \dots, s_{t'}$ composed of a sequence of locations s_t ($t = 0, \dots, t'$). Here, t indexes waypoints in the trajectory—the time from the t-th to the $t{+}1$-th location is not fixed, and t' denotes the index of the last feasible waypoint given time limit T. We use $\phi : s \to \{0, 1\}$ to denote whether or not we can observe the object at location s. Then $P(\phi_t = 1 \mid \boldsymbol{\phi}_{1:t-1}, \boldsymbol{s}_{1:t})$, with $\phi_t = \phi(s_t)$, is the probability of a positive observation at the next time step given the trajectory history—Sec. IV-C describes how we compute these probabilities. Given T, P and C, the objective is to find a trajectory $\boldsymbol{s}$ that maximizes expected reward R:

$$R_T(\boldsymbol{s}) = R_T(s_{1:t'}) = \sum_{t=1}^{t'} P(\phi_t = 1 \wedge \phi_k = 0 \; \forall k < t); \quad (1)$$

with total cost not exceeding time limit T:

Algorithm 1: View planning (phase one): OP construction

1 Function GVP-OP-CONST (M_{2D}, P, tc)
Input : 2D map M_{2D}; prob. dist. of perceiving an object P (based on M_{3D}), target coverage tc $(0 < tc \leq 1)$
Output : Set of locations S; cost function C
2 **begin**
3 $S \leftarrow \emptyset$
4 $Cov(S) \leftarrow 0$
5 **while** $Cov(S) < tc$ **do**
6 $s \leftarrow sampleLocation(M_{2D})$
7 $S \leftarrow S \cap s$
8 $Cov(S) \leftarrow Cov(S) + P(s)$
9 /* Filter redundant locations $s \in S$ with zero/low probability (P) such that the coverage constraint holds*/
10 $S \leftarrow filterRedundantLocations(S)$
11 /* Calculate the cost $C(s, s')$ for all location pairs */
12 **for** $s \in S, s' \in S$ **do**
13 $C(s, s') \leftarrow computeCost(M_{2D}, s, s')$
14 **return** S, C

$$C(\boldsymbol{s}) = C(s_{1:t'}) = \sum_{t=1}^{t'} C(s_{t-1}, s_t) \leq T. \tag{2}$$

Instead of exhaustively exploring all possible trajectories $\boldsymbol{s}$, we generate them from a distribution, which prefers trajectories that look the best myopically, and puts a non-zero probability on every path.

A. The Sampling-based Algorithm

The core idea of our algorithm is a two-phase anytime sampling-based optimization of the reward function. The first phase (Alg. 1) transforms the continuous problem into an OP, and samples a set of possible locations S until the search area is covered to a certain degree (Lines 3-8):

$$Cov(S) = \sum_{s \in S} P(s) \geq tc \tag{3}$$

where tc denotes a given target coverage[1]. Locations with zero or low probability are filtered out to satisfy the coverage constraint (Line 9), and the cost $C(s, s')$ for all location pairs is calculated (Lines 12-13).

The second phase (Alg. 2), which solves the OP, first selects the m best locations from S (Line 4), and updates and normalizes P by taking view dependencies into account (Line 6). Next, it generates a series of trajectories defined as an ordered sequence of locations, i.e., $\boldsymbol{s}^k = (s_0^k, s_1^k, \ldots, s_{t^k}^k)$, within time limit T (Lines 8-20). All trajectories start from the current robot pose, i.e., $\forall k,\ s_0^k = s_0$. The tth location of the kth trajectory (s_t^k) is sampled from the following distribution *without replacement* (Line 14):

$$s_t^k \sim P(\phi_t^k = 1 \mid \boldsymbol{\phi}_{1:t-1}^k, \boldsymbol{s}_{1:t}^k) \frac{\mathrm{e}^{-\varrho C(s_{t-1}^k, s_t^k)}}{Z} \tag{4}$$

where $t = 1, \ldots, t^k$ and $\boldsymbol{\phi}_{1:t-1}^k = 0$ if we have not found the object yet. The exponential expression is the transition distribution of choosing the next location based on costs, and Z is a normalizing constant. The sampling procedure only stops when time limit T is exceeded, and discards the last location, i.e., all sampled trajectories account for the time constraint in Equation 2. However, the trajectories can be of different length. Finally, $\varrho \geq 0$ is a parameter used to adjust the influence of the cost function. If $\varrho = 0$, $\mathrm{e}^{-\varrho C(s_{l-1}^k, s_l^k)}/Z$ is a uniform distribution, and reward is only based on location and not costs—higher the value of ϱ, greater the preference for locations with lower costs, leading to more cost-effective trajectories. When a decision must be made, the trajectory with the highest reward (so far) is chosen for execution (Equation 1) (Lines 21-22). If there is a tie, the trajectory with the smallest expected cost is chosen. We experimentally analyzed the effect of the parameters in both algorithms. However, finding an optimal setting for the parameters is beyond the scope of this paper.

[1] $Cov(S)$ takes the dependency of views into account.

Algorithm 2: View planning (phase two): OP solving

1 Function GVP-OP-SOLVE $(S, P, C, T, n_{\boldsymbol{s}}, m, \varrho)$
Input : Set of locations S; prob. dist. of perceiving an object P; cost function C; time limit T; number of trajectories $n_{\boldsymbol{s}}$; number of locations to be considered m $(0 \leq m \leq |S|)$; regularization parameter ϱ
Output : Sequence of locations $\boldsymbol{s}$
2 **begin**
3 /*Select the m best locations form S according to P */
4 $S' \leftarrow selectMBestLocations(S, m, P)$
5 /* Update and normalize P according to S' */
6 $P \leftarrow updateAndNormalize(P, S')$
7 /* Generate a set of trajectories $\mathcal{S}$ (of size $n_{\boldsymbol{s}}$) */
8 $\mathcal{S} \leftarrow \emptyset$
9 **for** $k \leftarrow 1$ **to** $n_{\boldsymbol{s}}$ **do**
10 /* Initialize sequence with current robot location*/
11 $\boldsymbol{s}^k \leftarrow (s_0^k)$
12 $t \leftarrow 1$
13 **while** $C(\boldsymbol{s}^k) < T$ **do**
14 $s_t^k \leftarrow sampleNextLocation(P, C, \boldsymbol{s}^k, \varrho)$
15 $\boldsymbol{s}^k \leftarrow append(\boldsymbol{s}^k, s_t^k)$
16 $S' \leftarrow S' \setminus s_t^k$
17 /* Update and normalize P according to new S' */
18 $P \leftarrow updateAndNormalize(P, S')$
19 $t \leftarrow t + 1$
20 $\mathcal{S} \leftarrow \mathcal{S} \cap \boldsymbol{s}_{0:t-2}^k$
21 $\boldsymbol{s}^* \leftarrow \arg\max_{\boldsymbol{s}^k \in \mathcal{S}} R_T(\boldsymbol{s}^k)$
22 **return** $\boldsymbol{s}^*$

B. Adaptive View Planning

The algorithm for *variant (I)* do not include planning time within the time constraint[2]. To account for the planning time T_P in *variant (II)*, we revised the constraint as $T_P + T_E \leq T$, resulting in a trade-off between planning time and execution time. To maximize the expected reward R_T acquired during execution, the robot is required to minimize its planning time. As an algorithm's planning time is influenced by parameters $\widehat{T}$, $n_{\boldsymbol{s}}$, and m, their optimal values need to be determined[3]. Since they depend on T and on the problem size $|S|$, we determine them experimentally.

[2] We assume a complete plan should be produced before execution. This assumption can be relaxed by allowing concurrent planning and execution.

[3] In adaptive view planning, we use $\widehat{T}$ to refer to the algorithm's parameter, and T to refer to the given time constraint; whereby we assume that $\widehat{T} \leq T$.

Let us assume a robot is given a new search problem with time limit T' and size $|S'|$. For an optimal solution, the robot need to determine values of the parameters that maximize the reward function R_T. Since the planning time T_P is included in the time limit, the robot cannot search the parameter space exhaustively. We propose to approximate the surface of the reward function from known problems. We assume that the robot has computed the reward functions over the parameter space ($\widehat{T}$, $n_{\boldsymbol{s}}$, and m) for some (but not all) search problems, and uses this information to interpolate (or extrapolate) the reward function for new search problems. We index pre-computed reward functions by T and $|S|$, i.e. $r(T, |S|)$. To determine the parameters ($\widehat{T}$, $n_{\boldsymbol{s}}$, m) that maximize the reward for a new problem $\langle T', |S'| \rangle$, adaptive extension sampling performs the following steps:

1) Select two known problems $\langle T_1, |S_1| \rangle$ and $\langle T_2, |S_2| \rangle$
2) Interpolate (or extrapolate) $r(T', |S'|)$ from the pre-computed functions $r(T_1, |S_1|)$ and $r(T_2, |S_2|)$.
3) Find the maximum in $r(T', |S'|)$ and return the corresponding $\widehat{T}$, $n_{\boldsymbol{s}}$, and m.

In this work we have used a linear interpolation method.

C. Implementation

This section describes the integration of our algorithm with other components, and our algorithm's implementation.

a) Integration on a robot: We integrated our view planner with the perception and action components of a simulated SCITOS A5 robot (`http://metralabs.com`) equipped with a 2D laser range finder, a depth camera, and a *semantic* camera. The range finder is used to create M_{2D} for robot navigation. The cameras are mounted on a pan-tilt unit (PTU) and have the same field of view. The depth camera is used to generate M_{3D}. The semantic camera is used for object recognition—it returns *true* (*false*) when an object of a given type is (is not) in the field of view. Recall that we assume perfect perception as we are primarily interested in evaluating the planning algorithm—a more realistic sensor model can be included if needed. We also use the motion planning and navigation routines from the Robot Operating System (ROS). The robot and the camera can thus be controlled by specifying a target pose the PTU's angles respectively.

b) Sampling of locations (S): Step 1 of Alg. 1 samples locations $s \in S$ until a predefined area of M_{3D} is covered (based on P). Each location s is composed of robot pose ($x \in X$) and view pose $v \in V$ variables, i.e., $s = (x, v)$. We first sample a robot pose x from M_{2D} and verify that the robot can reach it. We then generate a number (n_v) of random pan and tilt angles for the PTU (can use fixed angles too), and use the SCITOS robot model's forward kinematics to compute the poses of the cameras on the PTU. At each pose $x \in X$, the robot takes several views $v \in V$. We repeat this process until the predefined space is covered.

c) Probability distribution $P(\phi_t = 1 | s_{1:t})$: The probability distribution P is based on the assumption that objects rest on surfaces. From a given M_{3D}, we first extract the supporting surfaces based on the estimated normal of each voxel. Then, we identify the supporting surfaces' voxels that would be observed at each generated pose, by counting the number of voxels that lie within a frustum projected to the pose s. This provides the initial distribution over views, i.e., $P(\phi_0 = 1)$. Since we sample views without replacement, we remove any selected views, update the probabilities of dependent views, and normalize the distribution. We treat overlapping views as mutually dependent; once a view is chosen, we update the probabilities of all dependent views. We do this until we reach a plan length t^k.

d) Cost function $C(s, s')$: The cost, represented as time in seconds, of moving between locations s and s' is the maximum of two sub-costs (1) navigation cost C_{nav}; and (2) pan-tilt unit cost C_{ptu}—we assume the robot can navigate and operate its PTU concurrently:

$$\begin{aligned} C(s, s') &= C((x, v), (x', v')) \\ &= max(C_{nav}(x, x'), C_{ptu}(v, v')). \end{aligned} \tag{5}$$

To compute the navigation cost for a pair of robot poses (x, x'), we call the motion planner in ROS, retrieve a trajectory of waypoints, calculate the linear and angular distances between all consecutive waypoints, multiply them by their respective velocities, take the maximum of the linear and angular durations, and sum them up. The PTU cost is calculated by multiplying the differences between the current and the target pan-tilt angles by their respective velocities, and computing the maximum of these values.

V. Experimental Evaluation

We evaluated our view planning algorithm in simulation[4]. For *variant (I)*, we compared our algorithm with baseline algorithms in four simulated office environments that vary in size and complexity (Tab. I). The hypothesis was that our algorithm would scale better than the baseline algorithms, especially in larger environments. We used the expected reward as the performance measure. For *variant (II)*, we demonstrate the selection of our algorithm's parameters as a function of problem size ($|S|$) and time limit (T), thus supporting scaling by adapting to spatio-temporal constraints. The hypothesis was that our adaptive approach would outperform fixed strategies in novel environments. We measured the (a) expected rewards from the generated plans; and (b) performance when the plans were executed in a simulated environment—for the latter, performance was measured by the number of objects found.

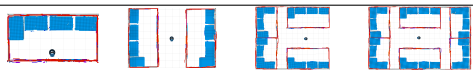
A. Experimental Results: Simulation

Simulation trials were conducted in the robot simulator MORSE [21]. For each environment in Tab. I, we generated 2D and 3D maps, and sampled locations such that 95% of the space was covered. In all environments, the robot had a predefined starting location.

1) Variant (I): For different time limits T, we compared our algorithm (GVP) with a stochastic algorithm for the OP [16] (OP-S), and two sequential approaches that greedily select m best views and sequence them using a TSP solver–one sequences greedily (TSP-G) and the other uses dynamic

[4]An evaluation of the algorithm in the real world is given in [20].

TABLE I: Experimental Environments

Environment	**E1**	**E2**	**E3**	**E4**
Size (m^2)	Small ($32m^2$)	Medium ($96m^2$)	Large ($168m^2$)	Huge ($240m^2$)
Coverage (%)	95%	95%	95%	95%
Locations ($\|S\|$) **(unfiltered)**	91 (322)	139 (402)	319 (1462)	399 (1533)
3D occ. grid size (#voxels)	2362	4519	8775	12950
3D occupancy grid maps				

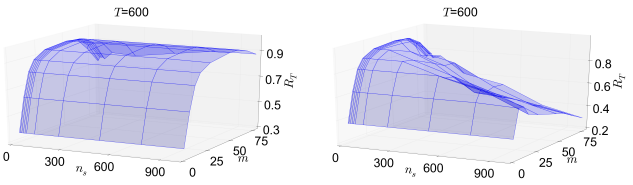
programming (TSP-DP). We expect our approach to scale better in larger environments as it chooses locations freely from the initial distribution—TSP-based methods compute a solution for a fixed set. Tab. II summarizes the results.

Since the sampled locations cover 95% of the space, 0.95 is the maximum reward possible. All approaches degrade in performance as the environments grow larger. Among the two sampling-based approaches, GVP is superior to OP-S in all environments, except in E2 with $T = 120$. Although OP-S achieves a comparable performance in smaller environments, performance is much worse in larger environments. Similarly, TSP-G performs well for the small environments, but the results for the larger environments are poor, as hypothesized. TSP-DP provided good results in multiple trials, but it could not compute a solution in environments E1-E3 for longer intervals, e.g., although planning time is not considered, a solution could not be computed on a standard laptop in ≤ 10 minutes—TSP-DP was able to sequence no more than 21 locations. The performance of the TSP-based approaches depends on the spatial distribution of the best m views. Hence, their performance cannot be predicted easily. Overall, our algorithm provides effective, if sub-optimal, solutions for a range of environments and time limits.

2) Variant (II): The planning time of the algorithms would have had a considerable effect on the results reported above if it were included in the time limits, e.g., T_P (in seconds) of algorithms in Tab. II were: OP-S (5-7); TSP-G (4-6); TSP-DP (4-584); and GVP (102-321). We performed an additional set of over $55k$ trials that evaluated the expected reward by considering both T_P and T_E (for E1 and E3). In Fig. 2, we illustrate the expected reward in environment E3 corresponding to two conditions: (a) $T_E \leq T$, and (b) $T_P + T_E \leq T$. The expected reward in (b) was calculated based on partial trajectories—only locations visited within T contributed to the reward. The reward function in (a) shows a rapid increase with respect to m; most of the configurations in (a) lead to a high reward (mainly dependent on m). However, in (b), configurations with large n_s and m lead to low or even zero rewards as planning with an increasing number of views and trajectories becomes more expensive. The selection of appropriate values for parameters is thus critical.

TABLE II: Comparison with baseline algorithms. Configuration for OP-S and GVP: $n_s = 250$; $m = 80$; $\varrho = 1.0$.

	OP-S		**TSP-G**		**TSP-DP**		**GVP**	
T	R_T	T_E	R_T	T_E	R_T	T_E	R_T	T_E
Environment E1								
120	0.81	119	0.59	67	**0.95**	116	0.89	113
240	0.93	234	0.92	131	**0.95**	136	**0.95**	237
360	0.94	359	**0.95**	216	-	-	**0.95**	358
480	0.94	471	**0.95**	235	-	-	**0.95**	429
600	**0.95**	597	**0.95**	272	-	-	**0.95**	479
Environment E2								
120	**0.51**	119	0.35	116	0.25	45	0.36	110
240	0.54	239	0.70	233	**0.90**	239	0.57	239
360	0.64	341	0.83	328	**0.92**	265	0.68	358
480	0.91	479	0.89	450	-	-	**0.95**	477
600	0.91	597	0.91	587	-	-	**0.95**	530
Environment E3								
120	0.17	118	0.17	135	0.08	64	**0.28**	118
240	0.28	226	0.30	220	0.36	207	**0.39**	237
360	0.35	334	0.46	332	**0.61**	358	0.57	324
480	0.50	477	0.57	458	**0.87**	459	0.60	478
600	0.56	597	0.66	552	-	-	**0.76**	596
Environment E4								
120	0.08	118	0.10	96	0.05	59	**0.17**	117
240	0.12	233	0.20	217	0.15	230	**0.30**	239
360	0.26	356	0.29	337	0.24	317	**0.42**	353
480	0.32	470	0.37	437	0.45	479	**0.54**	479
600	0.36	597	0.52	583	**0.72**	575	0.67	597

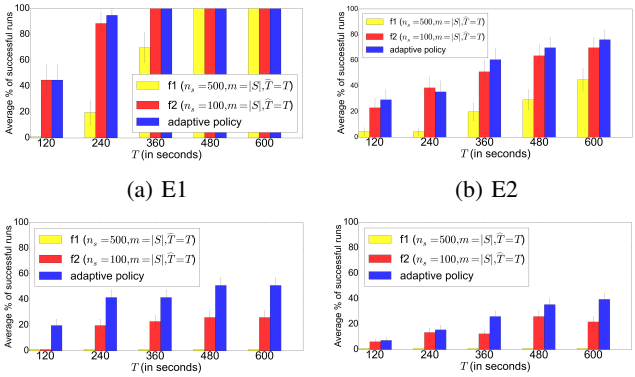


(a) $T_E \leq T$ (b) $T_P + T_E \leq T$

Fig. 2: Reward function for E3 ($T = 600$) considering (a) only execution time, and (b) both planning time and execution time. The range of parameter values explored $n_s \in \{5, ..., 1000\}, m \in \{5, ..., |S|\}, \widehat{T} \in \{30, ..., 600\}$ (only maximal reward with respect to $\widehat{T}$ is shown).

Our adaptive view planning algorithm determines values of parameters n_s and m for a given time limit T and problem size $|S|$ based on planning results of known environments (cf. Sec. IV-B). For instance, consider environments E1 and E3 to have been explored—Fig. 2 shows some of the reward functions for these environments. Based on these reward functions we derive parameters for a novel, medium-sized environment (E2) through interpolation, and a large-sized environment (E4) through extrapolation. We hypothesize that our adaptive approach, can outperform *fixed* strategies in novel environments because it can better adjust to the problem size.

Fig. 3 shows the average performance of our adaptive approach and two fixed strategies ($f1$ and $f2$) in 3900 simulations of object search tasks in all environments (E1-E4). Each trial is successful if a single object (whose location is unknown) is perceived. Performance is measured as the percentage of successful trials. The *adaptive policy* performs at least as well as the fixed strategies in all cases but one. The performance improvement is statistically significant in E2,



(a) E1 (b) E2

(c) E3 (d) E4

Fig. 3: **Average performance per time T.** Average fraction of successful trials for single-object search tasks in E1-E4, considering T_P and T_E. For each time limit T, each strategy was executed multiple times, proportional to the surface area of an environment (E1: 20x, E2: 40x, E3: 80x, E4: 120x; with a total of 3900 searches).

E3, E4 when our adaptive approach is compared with fixed strategy $f1$. Similarly, the results are statistically significant in E3 and E4 when our approach is compared with $f2$. The fixed strategies use the full set of views ($|S|$), whereas the adaptive strategy only uses a subset of views according to the problem size and time limit, thus reducing planning time and leaving more time for execution. The performance improvement is particularly pronounced in the larger environments (E3, E4) where the fixed strategies perform poorly.

Results in Fig. 3 indicate that adaptive parameter selection generalizes to unknown environments (E2 and E4), and that it outperforms the best fixed strategy ($n_{\boldsymbol{s}} = 100, m = |S|, \widehat{T} = T$). This is not entirely surprising, as any fixed strategy will eventually fail as problem size and/or time limit are changed. The adaptive strategy performs worse than one of the fixed strategies in E2 ($T = 240$). Although our adaptive strategy chose the optimal set of plan parameters based on the reward function, the actual performance was sub-optimal. If robot encounters a significant mismatch between the expected reward and its actual performance, this could be an indication that its prior knowledge (here P) is incorrect. In such situations, a robot can optimize its parameters from its actual performance, e.g., through reinforcement learning. These results encourage us to further explore adaptive planning.

VI. Conclusions

This paper has presented a sampling-based algorithm for the challenging open problem of view planning for object search under time constraints. We posed this problem as an OP with history-dependent rewards, and imposed a time limit. Experimental results indicate that our approach outperforms state of the art methods. Furthermore, our adaptive strategy generalizes well in comparison with fixed sampling strategies for object search in different (novel) environments.

Although the proposed approach has been integrated with the components of a mobile robot, the algorithm itself does not depend on any kinematic structure, e.g., it can be used to plan views of a camera on a manipulator arm or a quadcopter. Future work will further explore view planning with time constraints, under partial observability.

Acknowledgments: This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623, STRANDS; and the EPSRC grant EP/M015777/1, ALOOF.

References

[1] J. Tsotsos, "On the relative complexity of active vs. passive visual search," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 127–141, 1992.

[2] S. Frintrop, *VOCUS: A visual attention system for object detection and goal-directed search*. Springer, 2006, vol. 3899.

[3] M. Lorbach, S. Höfer, and O. Brock, "Prior-assisted propagation of spatial information for object search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[4] S. Zhang, M. Sridharan, and C. Washington, "Active visual planning for mobile robot teams using hierarchical POMDPs," *Robotics, IEEE Trans. on*, vol. 29, no. 4, pp. 975–985, 2013.

[5] L. Kunze, K. Doreswamy, and N. Hawes, "Using qualitative spatial relations for indirect object search," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[6] J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G. Kruijff, P. Lison, A. Pronobis *et al.*, "Self-understanding and self-extension: A systems and representational approach," *Autonomous Mental Development, IEEE Trans. on*, vol. 2, no. 4, pp. 282–303, 2010.

[7] M. Hanheide, M. Gobelbecker, G. Horn, A. Pronobis, K. Sjoo, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, H. Zender, G.-J. Kruijff, N. Hawes, and J. Wyatt, "Robot Task Planning and Explanation in Open and Uncertain Worlds," *Artificial Intelligence*, 2015.

[8] S. Zhang, M. Sridharan, and J. Wyatt, "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 699–713, 2015.

[9] B. Nilsson, "Guarding art galleries: Methods for mobile guards," Ph.D. dissertation, Lund University, 1995.

[10] H. González-Baños, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*. ACM, 2001, pp. 232–240.

[11] A. Sarmientoy, R. Murrieta-Cid, and S. Hutchinson, "A sample-based convex cover for rapidly finding an object in a 3-d environment," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3486–3491.

[12] D. Hadfield-Menell and E. Groshev and R. Chitnis and P. Abbeel, "Modular Task and Motion Planning in Belief Space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 28–October 2, 2015.

[13] Roland Geraerts and Mark H. Overras, "A Comparative Study of Probabilistic Roadmap Planners," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Nice, France, December 15-17, 2002.

[14] C. Vasile and C. Belta, "An Automata-Theoretic Approach to the Vehicle Routing Problem," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[15] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 1 – 10, 2011.

[16] T. Tsiligirides, "Heuristic methods applied to orienteering," *The Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.

[17] Q. Wang, X. Sun, B. L. Golden, and J. Jia, "Using artificial neural networks to solve the orienteering problem," *Ann. Oper. Res.*, vol. 61, no. 1, pp. 111–120, 1995.

[18] C. Dimitrakakis, "Complexity of stochastic branch and bound for belief tree search in Bayesian reinforcement learning," in *2nd International Conference on Agents and Artificial Intelligence*, 2009, pp. 259–264.

[19] A. Guez, D. Silver, and P. Dayan, "Efficient Bayes-adaptive reinforcement learning using sample-based search," in *Advances in Neural Information Processing Systems*, 2012, pp. 1025–1033.

[20] L. Kunze, M. Sridharan, C. Dimitrakakis, and J. Wyatt, "View planning with time constraints—an adaptive sampling approach," in *IEEE Int. Conf. on Robotics and Automation (ICRA) Workshop on AI Planning and Robotics: Challenges and Methods*, Singapore, May 29 2017.

[21] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular openrobots simulation engine: Morse," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, 2011.

Predicting Travel Time from Path Characteristics for Wheeled Robot Navigation

Peter Regier Marcell Missura Maren Bennewitz

***Abstract*— Modern approaches to mobile robot navigation typically employ a two-tiered system where first a geometric path is computed in a potentially obstacle-laden environment, and then a reactive motion controller with obstacle-avoidance capabilities is used to follow this path to the goal. However, when multiple path candidates are present, the shortest path is not always the best choice as it may lead through narrow gaps and it may be in general hard to follow due to a lack of smoothness. The assessment of an estimated completion time is a much stronger selection criterion, but due to the lack of a dynamic model in the path computation phase the completion time is typically a priori not known. We introduce a novel approach to estimate the completion time of a path based on simple, readily available features such as the length, the smoothness, and the clearance of the path. To this end, we apply non-linear regression and train an estimator with data gained from the simulation of the actual path execution with a controller that is based on the well-known Dynamic Window Approach. As we show in the experiments, our method is able to realistically estimate the completion time for 2D grid paths using the learned predictor and highly outperforms a prediction that is only based on path length.**

I. Introduction

A fundamental ability of a mobile robot is collision-free navigation in the environment. Many state-of-the-art navigation systems employ a two-stage approach to realize this in an efficient manner. Here, the first stage is dedicated to plan a spatial global path through the environment from the position of the robot to the goal location. Such global path planning is commonly carried out using a grid-based A* planner [1]. To smoothly follow the globally computed 2D path afterwards, one typically employs a local reactive collision avoidance system that efficiently generates velocity commands for the robot [2], [3], [4].

Robots nowadays are facing the challenge of having to solve tasks with ever increasing complexity. In several real world applications, the predictability of the completion time of such tasks plays an important role. In particular in multi-robot scenarios where the actions of multiple robots need to be accurately coordinated, the prediction of the task completion time is pivotal for time-efficient planning. Such scenarios include cooperative floor-cleaning and household tasks, long term autonomous driving systems [5] that need to estimate their travel time for a more efficient power supply management, and museum tour-guide robots that have to schedule their guiding precisely in order to guarantee sufficient entertainment of the visitors [6].

All authors are with the Humanoid Robots Lab, University of Bonn, 53113 Bonn, Germany. This work has been supported by the European Commission under contract number FP7-610532-SQUIRREL.

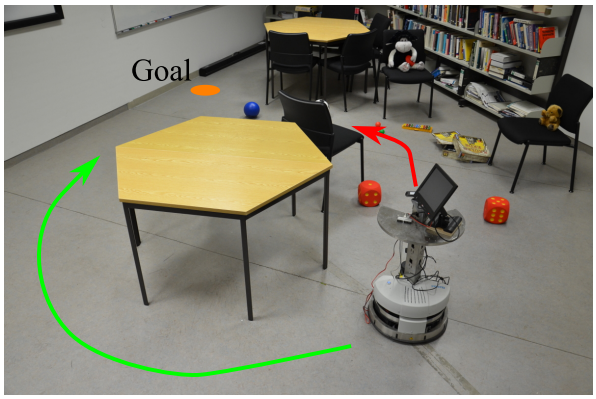


Fig. 1. Motivation of our approach. The robot can choose between two options to reach the goal location. The shorter path (red) leads through dense clutter where the robot needs to drive carefully and needs accurate sensing and pose estimation to avoid collisions. The second path (green) is longer but leads through wide free space where the robot can drive with a faster velocity profile. The work presented in this paper learns to predict the travel time along 2D paths from training data to decide which path leads to the fastest completion time.

When it comes to navigation, and most applications of mobile robots involve phases of locomotion in an environment, the travel time can make up a significant amount of the total task completion time. The fastest option, however, often differs from the shortest or cost-minimal solution found using 2D path planning on a cost grid map. Consider for example Fig. 1, where the robot has two options to reach the goal location. The first option (red path) contains narrow passages with several cluttered objects. The second option (green path) traverses solely free space. Taking the red path, the robot needs to drive slowly since it needs accurate sensing and precise motion execution to avoid collisions. At very tight spots the robot even has to stop frequently and rotate in order to adjust its heading. In contrast, the green path leads through free space where the robot can drive with higher velocities. Thus, the robot is faced with the question which route to choose to reach the goal location as quickly as possible.

With two-stage navigation systems where a global path planner is combined with a local motion controller, the precise outcome of the motion execution is typically difficult or impossible to predict in advance, especially when traversing narrow or cluttered passages. A reactive robot control system such as the popular Dynamic Window Approach (DWA) [4] can neither ensure a time-optimal trajectory, nor control stability, nor convergence of the system [7]. Additionally, many sources of noise randomly influence the navigation performance, i.e., the slippage of the wheels, noise in the

sensor measurements, and inaccuracies in the localization. The resolution of the grid-based environment representation or the choice of the navigation cost function can also influence the performance significantly.

In this paper, we present a novel method to predict the travel time for a mobile robot based on path features that are available ahead of the execution time. This will allow the robot to evaluate different options and choose the path that is predicted to be the most time-efficient. Given a 2D path in a grid representation of the environment, our method predicts the completion time by means of regression analysis based on general path characteristics such as its length, clearance, and curvature. We extensively evaluated our method in various environments of different complexity. As the experiments show, our method is able to realistically estimate the completion time of 2D grid paths and outperforms a prediction that is solely based on the path length. To the best of our knowledge, this is the first approach that can efficiently predict the completion time of navigation tasks without applying computationally expensive calculations using an exact kino-dynamic model of the robot.

II. Related Work

Grid-based planners are usually very fast in computing a spatial global path to a goal location and are widely used in navigation systems for wheeled robots [8], [9], [10]. Typically, these systems employ a local reactive collision avoidance method to generate actual velocity commands for the robot to follow a globally computed 2D path such as the DWA [3], [4] or the Nearness Diagram Method [2]. Note that the global paths typically contain sharp corners in the vicinity of obstacles so that these low-level reactive systems explicitly take into account deviations from the global 2D grid path to achieve smooth trajectories with a faster progress towards the goal location compared to stopping and turning on the spot. A different technique was developed by Stachniss and Burgard who suggested to plan directly in a 5D space in a local channel around the global 2D path with A* [11]. This space additionally contains the orientation as well as discretized translational and rotational velocities. In this way, smooth trajectories that directly consider kino-dynamic constraints are obtained.

All the approaches mentioned above assume that the time-optimal trajectory lies close to the computed 2D path, which is often the case but might not be true in the presence of many obstacles. In such cases it might actually be better to also take into account different paths with fewer obstacles that need to be passed. Therefore, we present a method that learns the time the robot needs to navigate along a given 2D path based on path characteristics. Based on general features of a 2D path, the robot can then estimate the completion time it would need to follow the path towards the goal location and choose the best option among different possibilities.

Murphy and Newman considered robots operating in large outdoor environments and developed an approach to trade off the risk of planning a path with suboptimal length for planning time and plan over probabilistic costmaps [8]. To create such a probabilistic costmap, one typically needs a priori knowledge about the terrain such as an overhead image of the environment. The work of Murphy and Newman focuses on traversing special types of terrain, whereas our approach is optimized for dealing with challenging indoor environments with mainly flat floor where the terrain properties play a minor role for the performance. Zhu and Qingbao proposed path planning based on a genetic algorithm [12]. The authors introduced functions to describe path characteristics that allow to choose an optimized path from a given set. This approach does not consider the motion control system of the robot. Philippsen [13] used probabilistic navigation functions to trade off the risk of colliding with dynamic obstacles against the length of a detour to avoid those. However, the approach requires tuning and user-defined heuristics and does not involve a trained model.

Lau *et al.* [7] developed an approach to time-optimal control from sparse way points to the goal based on quintic Bézier splines. Starting from a given straight-line path, the trajectory is optimized for smoothness and time taking into account the constraints of the system. In this paper, we consider general navigation in environments of different complexity also containing highly cluttered and narrow passages. Our goal is to estimate the travel time based on simple, readily available features describing the path characteristics and in this way enable the robot to choose the best option, i.e., the path assumed to lead fastest to the goal using a standard DWA-based controller that generates velocity commands in an efficient manner.

Recently, Regier *et al.* proposed to estimate obstacle densities beyond observed areas based on already detected objects and predict corresponding traversal costs [14]. The authors hereby assume that the robot possesses only partial knowledge about the obstacles in its surrounding. The work presented in this paper can be combined with such a prediction step in order to recompute the best path whenever new information about obstacles becomes available.

III. Navigation Framework

In this work, we assume that a 2D grid map representation of the environment is given and consider a mobile robot control system that applies a classical two-stage navigation approach, where the global path is computed by a grid planner on a costmap.

Afterwards, the task of the reactive local controller is to find velocity commands that allow the robot to follow this global path with collision-free motions. A well-known approach is to use roll out or look-ahead methods as in the DWA [3], [4]. A DWA-controller considers at each time step only a local costmap, which is a small fraction of the environment model allowing the system to operate in real time. The basic principles of the DWA approach are as follows:

At each time step, a local goal on the global 2D path is determined right outside the local costmap. In the second step, a set of feasible steering commands from the robot control space is computed to reach the local goal. For each sampled velocity command a simulated trajectory is determined and

evaluated through a predefined cost function. Based on the evaluation, the velocities of the best trajectory are taken for the control. The terms of the cost function used to evaluate the trajectories are based on the distance to the global path, the distance to the local goal, and the traversal cost given by the costmap.

Such a two-layered approach can generate robust, collision-free motion even in obstacle-laden environments. However, the highly unpredictable nature of the DWA controller as well as the influence of noisy perception and localization make the estimation of the completion time of a motion task a difficult endeavor. The cheapest path in a costmap is not always the best choice as it may lead through narrow or cluttered passages and it may be in general hard to follow without slowing down and rotating on the spot.

To illustrate this, we performed two navigation experiments in a large cluttered environment where the robot had the choice to drive through or around the cluttered region. Fig. 2 shows this scenario with the two paths and their corresponding velocity profiles. The red profile in Fig. 2 shows that driving around a cluttered region allows the robot to navigate at full speed and reach the goal in a shorter time even though the total path length is longer. Driving through the dense clutter, however, leads to higher localization errors due to more frequent rotations, repeated velocity drops in order to avoid collisions and on-spot rotations, which are necessary in regions with very little space to navigate.

Thus, the estimated completion time is in many situations a much stronger selection criterion than the path costs, but due to the lack of a dynamic model in the path computation phase the completion time is typically not known a priori. In the following, we introduce a novel approach to estimate the completion time from path features to enable the robot to choose the most promising path among different possible routes through the environment.

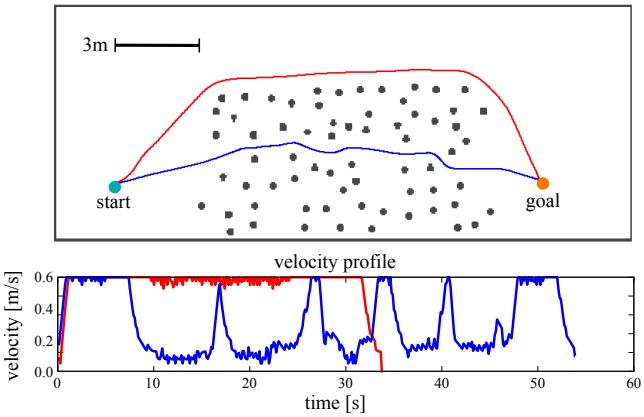


Fig. 2. Example velocity profiles of a robot driving through and around the cluttered region. Obstacles are displayed in grey. In order to reach the goal, the robot has the choice to either navigate through (blue path) or around (red path) the clutter. The corresponding velocity profiles are displayed to indicate common navigation issues that arise from navigating close to multiple obstacles. Considering the red profile, it is easy to see that the robot can constantly drive close to the maximum velocity and, thus, reaches the goal after only 34s with a traveled distance of 18.54m. The blue profile shows that constant speed drops occur, which are necessary in order to avoid collisions. Additionally, on-spot rotations are performed if too tight directional changes are necessary. This leads to a lower traveled distance of 14.46m while the execution time increased to 55.36s.

IV. Predicting Travel Time from Path Characteristics

In principle, the only way to predict the completion time is to simulate the path execution and measure the time the robot takes to navigate to the goal. Our idea is to apply a machine learning approach and to train a predictor function for the execution time based on a small number of generic features that can be efficiently computed from a given global 2D grid path.

A. Features for Describing Path Characteristics

We define a path $\mathbf{P} = \{\mathbf{p}_0, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ between the current position of the robot and the goal location as a sequence of two-dimensional coordinates (nodes) $\mathbf{p}_i = (x_i, y_i)$, $i \in \{0 \ldots n\}$, as illustrated for an example path in Fig. 3. A segment $\mathbf{s}_i$ of the path is then given by the vector $\mathbf{s}_{i+1} = \mathbf{p}_{i+1} - \mathbf{p}_i$. We found out that the length of the path, its clearance, and smoothness are expressive features that can be used to effectively estimate the time the robot needs to travel along the path towards the goal location. These features are described in detail in the following:

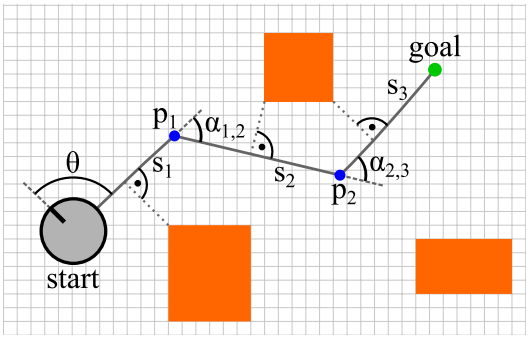


Fig. 3. Visualization of the features we use for path characterization. The figure shows an example path from the robot's current positions (grey circle) to the goal location through an environment with three obstacles (yellow rectangles). The path consists of three segments and two nodes. The angles $\alpha_{1,2}$, $\alpha_{2,3}$, and θ used in Eq. (2) are also shown. The shortest distances between the segments and the obstacle cells are illustrated as black dashed lines and are used in Eq. (3).

1) The *total length* of the path is given by the sum of the lengths of each path segment:

$$L_p = \sum_{i=1}^{n} |\mathbf{s}_i| \tag{1}$$

2) The *average smoothness* of a path expresses its deviation from being a straight line:

$$S_p = \frac{\theta + \sum_{j=1}^{n-1} \alpha_{j,j+1}}{n}, \tag{2}$$

where θ is the angle between the initial heading of the robot and the first path segment, and $\alpha_{j,j+1}$ denotes the angle between two path segments $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$. For example, $\alpha_{1,2}$ in Fig. 3 denotes the angle between $\mathbf{s}_1$ and $\mathbf{s}_2$.

3) Finally, the *average path clearance* is computed as follows:

$$C_p = \frac{\sum_{i=1}^{n} \max\{D_{max} - D_{min}(\mathbf{s}_i, c_{occ}), 0\}}{n} \quad (3)$$

using the shortest distances $D_{min}(\mathbf{s}_i, c_{occ})$ between each path segment $\mathbf{s}_i$ and the occupied cells c_{occ} closer than a threshold $D_{max} > 0$. We assume that obstacles with a distance greater than D_{max} have no effect on the task execution. The clearance is illustrated in Fig. 3 as a dashed line from a path segment $\mathbf{s}_i$ to its closest obstacle.

B. Prediction of Travel Time

Using the three path features defined above, we train a predictor function

$$T_p = \mathscr{F}(L_p, S_p, C_p) \quad (4)$$

that estimates the expected path execution time T_p based on the total length, average smoothness, and average clearance of the path. These features are readily available before the actual path execution starts by a local controller.

C. Regression Models

Regression is a common tool in statistical analysis to find relationships among variables. The goal of the regression analysis is to find a model that fits well the given data points use it for prediction afterwards. Different models can map different types of relationships between the variables. Linear regression, for example, is a very fast algorithm, but can only model linear coherences. A special case of linear regression is the simple linear regression that fits the data with a simple regression line. Linear regression, in contrast, models the relationship among several independent variable to predict the requested dependent quantity. For systems with non-linear behavior, linear predictors are often not sufficient. Better results can be achieved with more advanced methods. Support vector machines, for example, are kernel methods that map the data input into a high-dimensional feature space using kernel functions. This kernel trick allows to detect non-linear coherence in data sets.

To find the right regression approach for our problem, we evaluate the simple linear regression, linear regression, and support vector method for the task of completion time prediction based on the path features length, smoothness, and clearance that are described above.

V. Experiments

In this section, we discuss the data collection process, the regression analysis, as well as the prediction results in different environments.

A. Data Collection

Our goal is to obtain a single regression model that covers as many scenarios as possible. In order to gather data that is well distributed over the feature space, we performed experiments on a variety of maps such as the Willow office environment and artificially created maps (see Fig. 4). One

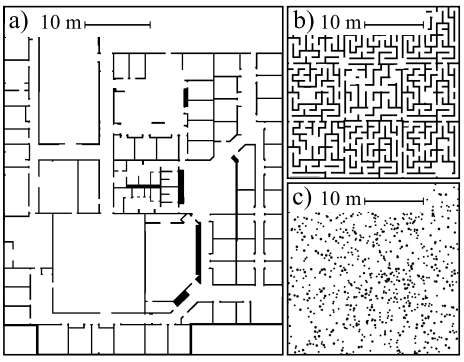


Fig. 4. Maps used in the experiments. (a) Office environment created by Willow Garage, (b) narrow maze-like environment, and (c) cluttered environment with many randomly distributed obstacles.

type of artificial maps we used are highly cluttered maps consisting of uniformly distributed or Gaussian distributed pillars, where pillars are randomly generated in varying quantity of 75 pillars per hundred square meters, 50 pillars per hundred square meters, and 25 pillars per hundred square meters with varying radii from 20-60 cm according to the distribution used. Another type of artificially created maps consist of narrow maze-like structures with a corridor width between 0.6 m and 0.9 m. We generated three different artificial maps of both types.

To collect training data, we used the Gazebo simulation environment [15] to simulate a model of the omnidirectional Robotino robot by *Festo Didactics*. We first compute a global path from the current position of the robot to a goal position and then let the robot follow this path with a DWA-controller. To obtain ground truth data, we measure the task completion time when the position of the robot is close to the x-y-coordinates of the goal position. The final heading is not considered. In each experiment, the start position, the initial heading of the robot, and the goal position were chosen randomly. We used an A*-planner for computing the global path. The lengths of the grid-based paths varied between 4 m and 50 m. The A*-planner and the DWA-controller are implemented in the ROS navigation stack [16].

The choice of the parameters of the navigation systems has a pivotal influence on the performance of the robot during the experiments. We found the following parameters to work best in practice. We used a resolution of 5 cm for the global costmaps of the environments and 1 cm for the local map. The frequency of the control loop was set to 8 Hz and the size of the local costmap was chosen to be 1.5 m $\times$ 1.5 m. The maximum linear velocity was set to $0.6\frac{m}{s}$ and the maximum rotational velocity was set $0.6\frac{rad}{s}$. The acceleration limits for linear and rotational movement were set to $0.7\frac{m}{s^2}$ and $0.7\frac{rad}{s^2}$, respectively. Naturally, the capabilities of the underlying physical system are instilled into a trained regressor. A deviation from the configuration parameters at a later time may work to some extent, we have not evaluated this in our work so far, but in general it must be assumed that the model is not transferable to a new system with significantly different navigational capabilities. The training

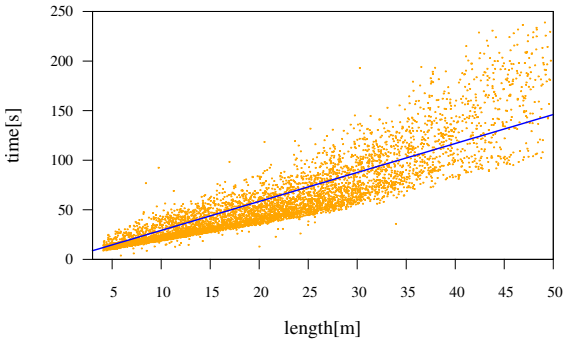


Fig. 5. Completion time of the simulated execution of the generated paths in the three environments (yellow) over path length. With increasing length, the data spreads broader around the regression line (blue). These results were obtained from the experiments with noisy localization.

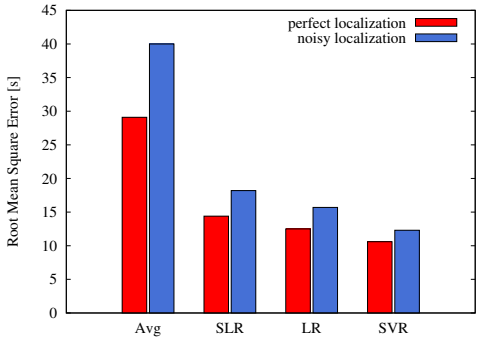


Fig. 6. Comparison of four different regression methods for perfect (red) and noisy (blue) localization. The regression methods are the constant average estimate (Avg), a simple linear estimate (SLR) based on path length only, linear regression (LR) using all features, and support vector machine regression (SVR) also using all features. As can be seen, the SVR has the smallest RMSE of all approaches. This non-linear model seems to be the best approximation for our robot and controller setup. Furthermore, we see a clear improvement of the LR model in comparison to SLR with the additional independent features.

must be performed for each individual combination of robot and navigation software.

We created two data sets with each containing 5500 navigation tasks. The first dataset was gathered without any sources of noise, i.e., no noise in the sensors and without slippage of the wheels. In particular this also includes a perfect localization. Naturally, this model is not entirely realistic, but it helps to analyze the data with respect to the correlation of the features with the estimate. The second data set was collected from experiments with a localization system that adds noise to the simulation due to faulty pose estimates. Note that in the second data set, the sensors and motion itself are still noise-free. Using these two data sets, we can evaluate our model with noise in comparison to noise-free results and also see how much the noise in the system affects the navigation performance.

B. Regression Results

In this section, we present the results of our regression analysis. For every data set, we learned an estimator for each of the different approaches. We used a simple linear regression (SLR) method based on the path length alone as computed in Eq. (1), a linear regression (LR) model which considers all the features mentioned above (see Sec. IV-A), and we trained a support vector machine for regression (SVR), also using all features. For training and testing we used WEKA, a well-established data mining software [17]. To evaluate the different regression models, we performed a 10-fold cross validation on the data set, i.e., during one validation run, 90% of the data set is used for training and the other 10% for testing this specific model. In the next validation round, another subset of 10% is used for testing and we repeated this process 10 times until every subset of the data set has been tested. We computed the average of the root mean square errors (RMSE) of every ten testing runs. As a reference, we additionally computed the RMSE of the constant average estimator over the entire data set.

It is not sufficient to assume a linear distribution, as in particular in the presence of clutter and narrow gaps our navigation system exhibits a highly non-linear behavior. Nevertheless, linear models are easy to fit and fast to compute and thus serve as a good reference. Fig. 5 illustrates the completion time of every run in the dataset over the path length (computed according to Eq. (1)). Note that the spread of the data points (yellow) around the linear regression line (blue) increases with path length.

The regression results depicted in Fig. 6 show that the features introduced in Sec. IV have a substantial influence on the time estimation, as we can see a 14% improvement for both data sets of the LR compared to SLR. A further reduction of the prediction error can be achieved using the non-linear model. Using the SVR results in a RMSE that is further reduced by 15% compared to the LR for the data set with perfect localization and 22% for noisy localization. These results support that the system behaves highly non-linear and a linear regression method is not sufficient if a more accurate estimate is desired.

Comparing both data sets against each other, we can clearly see the influence of the noisy localization which is close to real-world runs. The results also show that some of the noise can be estimated by our approach, since the error reduction from LR to SVR is larger for noisy compared to perfect localization. This improvement stems from the fact that a much higher localization error correlates with certain non-linear behaviors, e.g., rotating fast on the spot or traversing monotone environments with only few features. Thus, the non-linear SVR method is best suited for real-world scenarios.

The evaluation shown in Fig. 6 is well suited for a comparison of the different regression approaches. Additionally, we are interested in the relative root mean square error of the estimate, which is defined as follows:

$$\sigma_{est} = \sqrt{\frac{\sum_{i=1}^{N} \frac{(Y_i - Y_i')^2}{Y_i^2}}{N}}\ , \tag{5}$$

where Y_i is the completion time of experiment i, Y_i' is the corresponding estimated value, and N is the number of

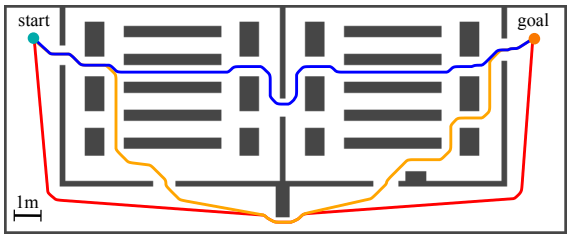


Fig. 7. A environment with two rooms and a corridor, that was not used for the training data, with three different path options to get from the start to the goal. The longest path (red) leads the robot through wide free-space area. The shortest path (blue) guides the robot through narrow space between obstacles. An alternative path (yellow) leads partly through the narrow passages and through wide-space. The the evaluation of the path is shown in Tab. I.

experiments in a set. As we evaluated a wide variety of scenarios which contained both very short and very long paths, σ_{est} is a better measurement of the relative deviation per experiment, as we first scale every separate squared error by the corresponding completion time. We computed the values of Eq. (5) for both the LR and SVR due to the superior performance compared to SLR. For the LR and SVR, σ_{est} evaluates to 0.32 and 0.13, respectively. These results show that the use of SVR not only highly decreases the average deviation, but also shows an improved estimate for the whole spectrum of path lengths. As these results show, our approach is able to predict the path completion time with an error of only 13% in average.

C. Temporal Gain

To demonstrate the temporal gain when applying our prediction, we performed an experiment on a completely new map (see Fig. 7). In this experiment, three different path choices to navigate from start to the goal location exist. The first option is the shortest path (blue), which leads through narrow areas. The red path is the longest, but it is smooth and has a high clearnace to obstacles. The third alternative consists of segments of the other two paths. Based on the completion time predicted by our approach, the longest path is chosen as the fastest option followed by the shortest path. The third path is the slowest according to our prediction. By executing all three options in simulation, the actual completion time in Tab. I confirms the prediction and the path choice. The actual temporal gain when executing the red path in comparison to the execution of the shortest path amounts to 6 s, which is 9.8% of the travel time.

VI. Conclusions

In this paper, we presented a technique to estimate the completion time for 2D grid paths. The completion time is in general not known in advance as it strongly depends on the capabilities of the underlying motion controller. Through a low-dimensional categorization of the paths using three generic features—their length, smoothness, and clearance—and the simulation of a large variety of motion tasks on different types of maps, we were able to regress an estimator that predicts the path completion time with a low error of around 10% before motion execution starts. Naturally, as the completion time depends strongly on the navigation performance of the robot, it needs to be trained individually for a specific hardware and motion controller combination.

TABLE I

Evaluation of the paths shown in Fig. 7

	red	yellow	blue
length	28.5145 m	26.0843 m	20.9335 m
clearance	0.2685	0.4401	0.5131
smoothness	0.0137	0.0457	0.0483
prediction time	47.373 s	58.756 s	50.392 s
completion time	55.512 s	63.751 s	61.511 s

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics*, vol. 20, no. 1, 2004.

[3] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, pp. 23–33, 1997.

[5] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous vehicles in city traffic*. Springer, 2009.

[6] S. Thrun, M. Bennewitz, W. Burgard, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[7] C. Lau, B.and Sprunk and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2009.

[8] L. Murphy and P. Newman, "Risky planning on probabilistic costmaps for path planning in outdoor environments," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 445–457, 2013.

[9] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 5568–5575.

[10] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[11] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2002.

[12] H. Jun and Z. Qingbao, "Multi-objective mobile robot path planning based on improved genetic algorithm," in *Int. Conf. on Intelligent Computation Technology and Automation (ICICTA)*, vol. 2, 2010, pp. 752–756.

[13] R. Philippsen, S. Kolski, K. Macek, and B. Jensen, "Mobile robot planning in dynamic environments and on growable costmaps," in *Workshop on Planning with Cost Maps at the IEEE Intl. Conf. on Robotics and Automation*, 2008.

[14] P. Regier, S. Oßwald, P. Karkowski, and M. Bennewitz, "Foresighted navigation through cluttered environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2016.

[15] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2004.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

Deep Detection of People and their Mobility Aids for a Hospital Robot

Andres Vasquez Marina Kollmitz Andreas Eitel Wolfram Burgard

***Abstract*— Robots operating in populated environments encounter many different types of people, some of whom might have an advanced need for cautious interaction, because of physical impairments or their advanced age. Robots therefore need to recognize such advanced demands to provide appropriate assistance, guidance or other forms of support. In this paper, we propose a depth-based perception pipeline that estimates the position and velocity of people in the environment and categorizes them according to the mobility aids they use: pedestrian, person in wheelchair, person in a wheelchair with a person pushing them, person with crutches and person using a walker. We present a fast region proposal method that feeds a Region-based Convolutional Network (Fast R-CNN [1]). With this, we speed up the object detection process by a factor of seven compared to a dense sliding window approach. We furthermore propose a probabilistic position, velocity and class estimator to smooth the CNN's detections and account for occlusions and misclassifications. In addition, we introduce a new hospital dataset with over 17,000 annotated RGB-D images. Extensive experiments confirm that our pipeline successfully keeps track of people and their mobility aids, even in challenging situations with multiple people from different categories and frequent occlusions.**

I. INTRODUCTION

Mobile robots operating in populated environments need to perceive and react to the people they encounter. Our research is part of the project NaRKo, which aims at employing autonomous robots for delivery tasks as well as for guiding people to treatment rooms in hospitals. Hospital environments pose special challenges to autonomous robot operation, because the people interacting with the robot might have very different needs and capabilities. It is therefore desirable for the robot to adapt its behavior accordingly, e.g. by adjusting its velocity and path when guiding a person with a walking frame compared to a healthy person without motion impairments.

Our work targets the detection and categorization of people according to the mobility aids they use. Privacy concerns play an important role in the hospital, which is why our approach is based only on depth data. However, depth data conveys a lot less information than RGB images, which makes the problem more challenging. We propose a perception pipeline which uses depth images from a Kinect v2 camera at 15 frames per second and outputs the perceived class, position, velocity and the tracked motion path of people. Our object detection pipeline uses the Fast R-CNN method proposed by Girshick [1], which takes an image together with a set of regions of interest (ROIs) as its input and outputs classification scores for each ROI. We propose a fast depth-based ROI proposal method that uses ground plane removal and clustering to generate a set of regions and applies a set of local sliding templates over each region. We compare our method against a dense sliding window baseline and show that our approach is significantly faster and yields improved performance. The perceived class of each ROI as well as its position in the world frame are further processed by our probabilistic position, velocity and class estimator. In addition to tracking the position and velocity of each person in the environment with a Kalman filter, we use a hidden Markov model (HMM) to estimate the class of each track. As depicted in Fig. 1, the probabilistic position, velocity and class estimator resolves occlusions and outputs a probability distribution over the five classes, taking the previous observations into account. This paper further presents our

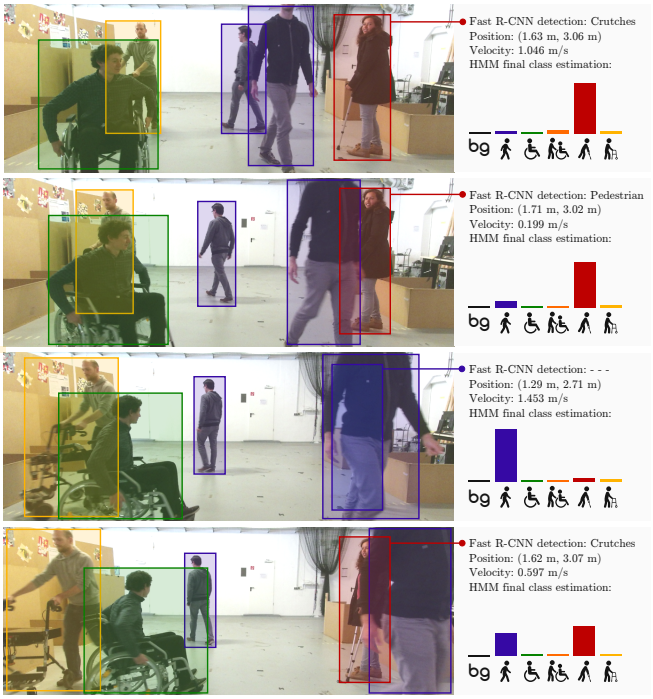


Fig. 1. We present a depth-based perception pipeline that estimates the position, velocity and the class of people, recorded in different populated environments, including a hospital. Top: a person with crutches is detected at the right corner of the image and the class estimation predicts high probability for the respective class. Second and third: the person is occluded by another person and the estimated class switches gradually to pedestrian, while the tracker keeps the track alive. Bottom: the class estimator reflects the ambiguity of the belief about the class of the person. We show RGB images for visualization purposes only.

*This work has been partially supported by the German Federal Ministry of Education and Research (BMBF), contract number 01IS15044B-NaRKo and by the DFG under grant number EXC 1086.

The authors are with the Faculty of Computer Science, University of Freiburg, Freiburg, Germany. Corresponding author: kollmitz@informatik.uni-freiburg.de

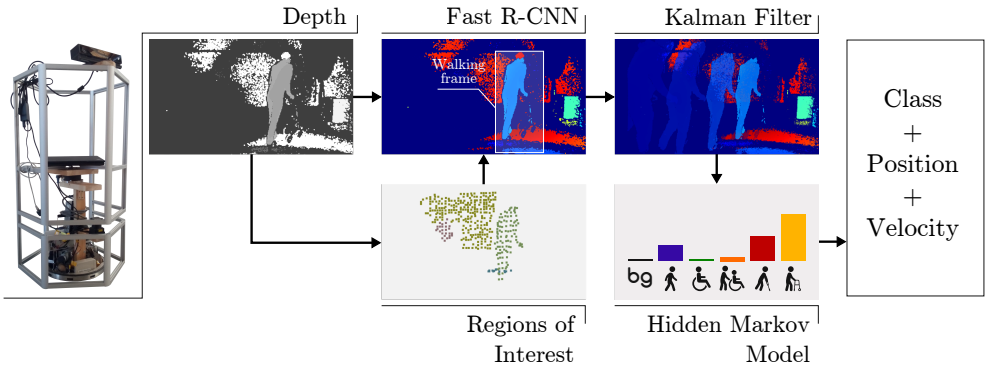


Fig. 2. Our pipeline operates on depth images collected from a Kinect v2 sensor. To achieve a fast runtime we present a depth-based region proposal method that generates regions of interest based on Euclidean clustering, which we feed into a Fast R-CNN detection network. We filter the resulting bounding box detections using a Kalman filter for position and velocity estimation of people. Further, we employ a hidden Markov model for filtering the class predictions over time.

hospital dataset that contains over 17,000 annotated RGB-D images with 960x540 pixel resolution. It is publicly available at `http://www2.informatik.uni-freiburg.de/~kollmitz/MobilityAids`. We collected the dataset in the facilities of the Faculty of Engineering of the University of Freiburg and in a hospital in Frankfurt. The webpage also shows a video of the final results of our approach.

II. RELATED WORK

People detection and tracking is a widely studied field in both computer vision and robotics. Given the large amount of previous work in this area, we will focus only on approaches that integrate both people detection and tracking in a combined system. Further special emphasis is laid on approaches for mobile platforms that are equipped with vision-based sensors such as cameras or RGB-D sensors.

Ess et al. [2] address the problem of multi person tracking and detection using a stereo vision system mounted on a mobile platform, integrating visual odometry, depth estimation and pedestrian detection for improved perception. Choi et al. [3] propose a method to track multiple people from a moving platform based on a particle filter approach. Several different detectors are used such as upper body, face, depth-based shape, skin color and a motion detector. Recently, extensive frameworks that include several people detection and tracking methods for mobile robots operating in indoor environments have been presented [4, 5]. In comparison to the mentioned frameworks, we focus on a multi-class detection problem and do not only track position and velocity but also the class throughout time. Further, previous approaches rely on manually designed detectors for different body parts while we use a single neural network detector that learns those body features automatically.

Our work is further related to the research area of object detection, which recently is dominated by deep neural network approaches, most prominently by region-based convolutional neural networks [1, 6], which achieve very good results but are not applicable for real-time yet. An interesting extension to the region-based CNN detection approaches is the very recently introduced region-based fully convolutional neural network presented by Li et al. [7], which increases the test-time speed. Recently Redmon and Farhadi [8] proposed an approach that formulates object detection as a regression problem. It can operate in real-time and achieves very impressive performance on several object detection benchmarks. We also employ a region-based convolutional neural network classifier and, to achieve a fast runtime, we combine it with our depth-based region proposal method. Recent work on multi-class object recognition and detection applied to mobile robot scenarios include a Lidar-based wheelchair/walker detector [9] and a human gender recognition approach [10]. To the best of our knowledge there exists no prior work that presents multi-class people detection applied to service robot scenarios.

Region of interest (ROI) extraction is often used to speed up the detection process and to reduce the number of false positives. Often employed is the assumption that most objects occur on a dominant ground surface and several methods exist that generate ROIs based on this ground plane assumption [11, 12, 13]. Munaro and Menegatti [11] detect and remove the ground plane, then they apply Euclidean clustering. To overcome the problem of having two or more people in the same cluster they run an additional head detector. Our method is similar, although we use a local sliding window approach with templates that are applied to each cluster instead of a head detector. The people detector of Jafari et al. [14] extracts ROIs by fusing point clouds over a time window to compute a 2D occupancy map. Spinello and Arras [15] also exploit depth information to reduce the number of candidates in a sliding window approach for people detection in RGB-D data. However, their approach is sensitive to false depth readings that can result in very large proposal windows. Chen et al. [16] present a more evolved method to generate ROIs formulating an energy function that encodes informative features such as object size priors, ground plane and depth information. Despite good performance, the algorithm has a computation time of approximately one second. Our ROI extraction approach instead is fast and simple.

Another contribution of this paper is a novel, annotated

large-scale dataset for multi-class people detection. In the literature there are several other datasets that include multi-class labels for people, mostly from the area of human attribute recognition [17, 18] or more specifically gender recognition [10]. Our dataset can be valuable for the robotics community, because on one hand it provides a large number of labeled images and on the other hand it is recorded from a mobile platform. Very recently and most similar to our dataset Sudowe et al. [19] recorded video sequences of people from a moving camera for the task of human attribute recognition.

III. PEOPLE PERCEPTION FRAMEWORK

Fig. 2 gives an overview of our overall system, which takes as input a stream of depth images, computes a set of ROIs, classifies those proposals and filters the network output over time. Specifically we filter the position and velocity of a person using a Kalman filter and their category using a hidden Markov model. The resulting output of our framework, which contains the class, position and velocity of people, is visualized in Fig. 1.

A. Fast Depth-Based Region Proposals Generation

To obtain the regions of interests we follow a sequence of steps. After converting the depth image into a point cloud representation we remove all points belonging to the ground plane, apply Euclidean clustering and slide a set of local sliding templates over the obtained segments, see Fig. 3.

1) Ground plane removal: For a fast ground plane estimation we apply Random Sample Consensus to estimate the parameters of the ground plane. After the removal of the ground plane we segment the point cloud by means of the Euclidean clustering method.

2) Local sliding templates: People of the five classes have different outlines in the images, because of the different typical shapes. A pedestrian, for example, takes up a rather narrow and tall part while a person in a wheelchair will take up a wider and lower part. To account for the different contours we use five different local sliding templates as shown in Fig. 3. All five templates are projected around the centroid of each computed point cloud segment to compute the ROIs. The first bounding box considers the size of an average pedestrian with height $h_p = 1.75m$ and width $w_p = 0.4m$, this rectangle is our template box T_1 for detecting pedestrians and the front views of the other categories except people using wheelchairs. We stretch T_1 horizontally with a factor of $1/3$ to each side to obtain the template box T_2 for side views of people using crutches or walking frames. In the same way, T_1 is stretched again with a factor of $2/3$ to obtain T_3 for the side view cases of people pushing people in wheelchairs. Finally, we obtain the template boxes T_4 and T_5 for people using wheelchairs by reducing the height of T_1 and T_2 by a factor of $1/4$.

Note that a candidate in the point cloud might contain only a part of a human body and since we do not have this prior information, we take into account that the centroid of a segment is not always the center of a body. Therefore, we

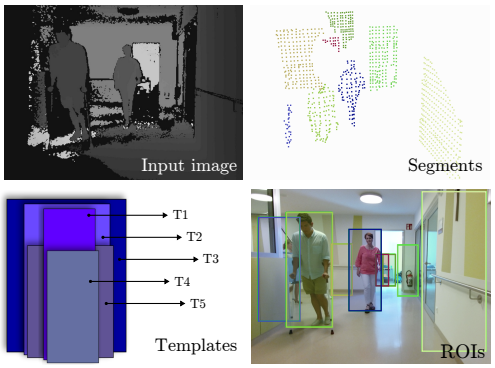


Fig. 3. We obtain regions of interest (ROIs) from depth images, based on point cloud segmentation and the use of templates. For demonstration purposes, we only show the template $T1$ applied to each segment in the lower right image.

horizontally slide these five proposals to l different positions around the center of the candidate, using a stride of n_s pixels. Accordingly, we end up with $5l$ proposals for every segment that will be fed into the Fast R-CNN detector.

B. Detection using Fast R-CNN

The input of the Fast R-CNN network is a color-encoded depth image together with a set of ROIs and its output are both softmax scores for each ROI and bounding box coordinates from a regression layer. We use the deep network architecture proposed by Mees et al. [20], which contains 21 convolutional and six max pooling layers (GoogLeNet-xxs).

1) Training: Fast R-CNN jointly optimizes a multi-task loss that contains a softmax classifier and a bounding box regressor. We build the set of ROIs for the training stage by applying a dense multi-scale sliding window approach using our five template bounding boxes previously described $(T1, \ldots, T5)$. We slide them at different scales all over the image, which produces a set of dense ROIs containing more than 29'000 bounding boxes for a single image, from which we sample during training. For every sampled ROI, we compute the overlap with a ground-truth bounding box. This overlap we measure in terms of Intersection over Union (IoU). A sample with IoU greater than 0.6 is considered a positive training example of the class contained in the ground-truth bounding box. Samples with IoU below 0.4 are instances of the background class. We run stochastic gradient descent SGD for 140,000 iterations, using a learning rate of 0.001 and momentum of 0.9.

2) Test-time detection: Fast R-CNN outputs scores s^r_m for each class $m \in \{1, \ldots, M\}$ and each proposal r. The class c^r corresponding to each proposal is the one with the highest score:

$$c^r = \underset{m=1}{\overset{M}{\operatorname{argmax}}}\ s^r_m \tag{1}$$

Since we manage $5l$ different proposals (local sliding templates) for the same candidate segment, we will have several proposal classes c^r for one segment. We assign the final class of the segment as the class with the highest number of appearances for all associated proposals. Note that this

procedure assumes that each segment contains not more than one person, and in practice this assumption works reasonably well. Two segments corresponding to the same person or very close segments in the point cloud might result in overlapping detections. We overcome this problem by applying non-maximum suppression (NMS) as a final step. NMS chooses a subset of the remaining c^r, which is the final output c of the Fast R-CNN detector. The corresponding position (x_c, y_c, z_c) of the person in camera coordinates is obtained from the bounding box coordinates, which are also provided by the Fast R-CNN network, and the distance of the segment from the camera.

C. Probabilistic Position, Velocity and Class Estimation

The detection stage provides us with a set of coordinates for each bounding box containing a person of the form (x, y, z, c), where (x, y, z) is the detected pose of the person transformed from camera coordinates into the world frame and c is the perceived class. The world frame transformation requires knowledge of the robot's position in the environment, which we obtain from its odometry. Our position, velocity and class estimator computes the belief $\text{Bel}(\mathbf{x})$ of the person state $\mathbf{x} = (x_x, x_y, x_{\dot{x}}, x_{\dot{y}}, x_c)$, where (x_x, x_y) represent the person's true position and $(x_{\dot{x}}, x_{\dot{y}})$ the true velocity on the ground plane and x_c represents the true class.

Each person has one Kalman filter and one HMM associated to it. The Kalman filter uses a constant velocity motion model, where the motion and the observation noise are obtained experimentally by analyzing the labeled ground truth trajectories and the corresponding Fast R-CNN detections of people in the training data set. We solve the data association problem between frames using the Mahalanobis distance

$$d_{ij}^2 = v_{ij}^T(t)\hat{S}^{-1}(t)v_{ij}(t) \quad (2)$$

$$\text{with} \quad v_{ij} = z_i(t) - H(t)\hat{x}_j(t) \quad (3)$$

$$\text{and} \quad \hat{S}(t) = H(t)\hat{P}(t)H^T(t) + R(t), \quad (4)$$

where $\hat{x}(t)$ and $\hat{P}(t)$ are the predicted state mean and covariance at time t, $z(t)$ is the observation, $H(t)$ the observation model and $R(t)$ the observation noise of the filter. The observation and filter indices at time t are i and j. We use the Hungarian algorithm to assign tracks to observations, according to the pairwise Mahalanobis distances. If the distance is larger than a threshold, the observation is not paired to a track. Instead, a new Kalman filter is initialized. If there is no observation for a track, we perform a prediction without observation update.

Each Kalman filter has one HMM associated to it for estimating the class x_c of the tracked person, according to

$$p(x_{c,t} \mid c_{1:t}) = \eta p(c_t \mid x_{c,t}) \sum_{x_{c,t-1}} p(x_{c,t} \mid x_{c,t-1}) p(x_{c,t-1} \mid c_{1:t-1}). \quad (5)$$

Eq. 5 models the probability of the tracked person to belong to a given class $x_{c,t}$, given the past observations $c_{1:t}$. Here, $x_{c,t}$ is hidden, since we only get measurements c_t for it. The measurement model $p(c_t \mid x_{c,t})$ connects the hidden with the observed variable for time step t. The HMM further assumes that the class $x_{c,t}$ can randomly change from one time step to the next, represented by the transition model $p(x_{c,t} \mid x_{c,t-1})$. In a hospital, a possible transition could be person with crutches $\rightarrow$ pedestrian $\rightarrow$ person in wheelchair for a patient who has just finished physiotherapy and hands over his crutches to return to his wheelchair. We need to further specify the class prior $p(x_{c,1})$ for the initialization of the HMM.

The output of deep neural network classifiers like Fast R-CNN can be interpreted as $p(x_{c,t} \mid c_t)$, since it represents a probability distribution. However, training with one-hot encoded labels results in very peaky distributions and over-confident estimates. Therefore, we will not employ the network scores s_m^r directly for our HMM. Instead, we analyze our training data to determine the underlying probability distributions statistically. To this end, we first generate and label all proposals for each frame in the validation set of our data, given by our ROI generator and obtain a classification output for each ROI. The percentage of labels for each class determines the class prior $p(x_{c,1})$. The measurement model $p(c_t \mid x_{c,t})$ is determined by the amount of detections for each class, given a certain label. As a side node, the measurement model corresponds to the classifier's confusion matrix. The transition model $p(x_{c,t} \mid x_{c,t-1})$ is given by the amount of transitions from one class to the other with respect to the total number of transitions.

Due to the limited amount of examples in the validation set, we might not observe all class transitions, even if they are possible. Therefore, we assign small probabilities to all unobserved but possible transitions and misdetections, using a Dirichlet prior. The data association for the HMM is given by the Kalman filter. If a tracked person is in the field of view of the camera and there is no observation c_t for time step t, we treat it as a background detection in the HMM. If the track is outside the field of view, we apply the transition model to the previous state estimate without considering an observation. The position, velocity and class estimator removes tracks with a standard deviation in position above a threshold. Furthermore, tracks which are estimated as background with a probability above a threshold are deleted.

IV. EXPERIMENTS

In this section we evaluate the performance of the two submodules Fast R-CNN object detection and Kalman filter tracking. Additionally, we present quantitative results for the combined perception pipeline. For these experiments we used our hospital dataset. Finally, we present a real-robot scenario where the robot uses our perception pipeline in order to give special assistance in a person guidance task.

A. Dataset

In order to train and evaluate our pipeline we annotated our hospital dataset. Images were collected in the facilities of the Faculty of Engineering of the University of Freiburg and in a hospital in Frankfurt using a mobile Festo Robotino

TABLE I

OBJECT DETECTION PERFORMANCE IN TERMS OF MEAN AVERAGE PRECISION (MAP) USING SOLELY FAST R-CNN.

						MAP
Ours	73.0	89.6	39.5	66.9	54.9	**64.98**
DSW	70.6	86.9	42.0	69.4	49.5	**63.67**

TABLE II

PERFORMANCE ON INOUTDOORPEOPLE [20] TEST SEQUENCE 4.

	AP
Ours	70.0
DSW [20]	**71.6**
Upper-body detector [14]	69.1

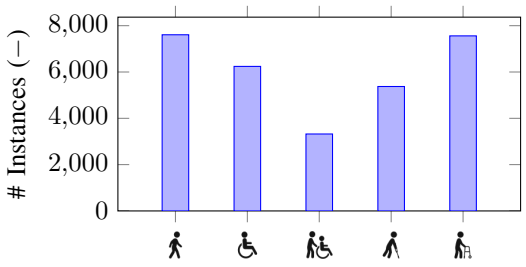


Fig. 4. Number of instances per class in our dataset.

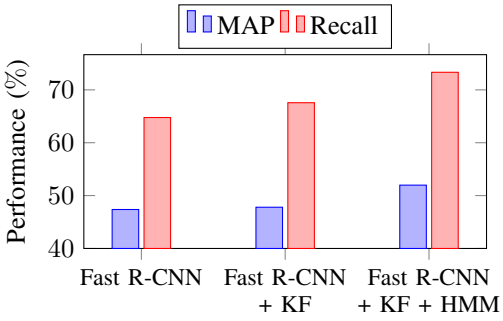


Fig. 5. Object detection performance evolution with respect to stand-alone Fast R-CNN. Addition of the two modules KF and HMM to the Fast R-CNN baseline improves overall MAP and recall.

robot, equipped with a Kinect v2 camera mounted 1 m above the ground and capturing images at 15 frames per second. The robot was controlled by a notebook computer running ROS (Robot Operating System).

The dataset is subdivided into subsets for training, validation, and evaluation of the pipeline. We use two test sets to evaluate the performance of the pipeline. Test set 1 is used to evaluate the Fast R-CNN detector, and it has two sets of annotations used for different experiments, whereas Test set 2 is used for the overall evaluation of the pipeline and its ground truth labels consider also occluded objects. Test set 1 is used also as validation set to design the hidden Markov model (HMM) and the tracking algorithm. Fig. 4 shows the number of instances for each class contained in the set.

B. Detection using Fast R-CNN

In order to evaluate the object detection performance, we use the standard mean average precision (MAP) metric. We compare the final detection performance using proposals from a dense sliding window method against our fast depth-based region proposal method. In order to create the set of dense sliding window proposals, we applied our templates on the implementation of Mees et al. [20]. Tab. I compares both approaches at an IoU of 0.5. Our depth-based proposal method performs better than the sliding window baseline method.

We also compare the runtime of both methods on a computer with 12-Core CPU and a GeForce GTX TITAN X with 12GB of memory. Following the dense sliding window algorithm, the set of ROIs for a single image contains approximately 29,000 bounding boxes, while our approach generates an average of 450 proposals. By using dense sliding window (DSW), a single frame is classified in 297 ms, whereas our algorithm requires 43 ms. Our approach is therefore a better choice for employment on a real-world system. We also evaluated the classification performance by means of a confusion matrix where the three most noticeable confusions correspond to 26.8% of people using crutches that were confused with people pushing other people in wheelchairs, 31.7% of people using walking frames that were classified as pedestrians and 23.8% of people using crutches classified as pedestrians. Qualitative detection results are shown in Fig. 6.

Further, we compare our detector on the InOutDoorPeople dataset [20] against a dense sliding window approach and an implementation [5] of the depth-based upper-body detector by Jafari et al. [14]. This dataset only contains labels for pedestrians. Therefore, all class predictions of our network are counted as detections of pedestrians. Table II shows that our approach outperforms the upper-body detector but achieves slightly worse results than the DSW baseline. We hypothesize that our detector would improve if trained solely on pedestrians. For all experiments we use the same network that we trained on five classes, without retraining.

C. Multi-Tracking using Kalman Filter

To evaluate the performance of the tracking algorithm we use the CLEAR MOT metrics proposed by Bernardin et al. [21] which considers the Multiple Object Tracking Precision (MOTP) and the Multiple Object Tracking Accuracy (MOTA). We obtained a MOTP of 70.60% and 62.19% of MOTA.

D. Framework Evaluation

We further aim to evaluate our complete pipeline in order to assess the contribution of different stages such as classification, tracking and class estimation in the overall detection performance. In this experiment, we challenge the overall system to estimate the position and class of temporarily occluded people. We compare the performance of the pipeline in terms of MAP and recall. Results were obtained



Fig. 6. Qualitative object detection results obtained using our trained Fast R-CNN network. Left: positive examples. Right: cases of failure with missed detections and wrong classifications. We show the RGB image only for demonstration purposes.

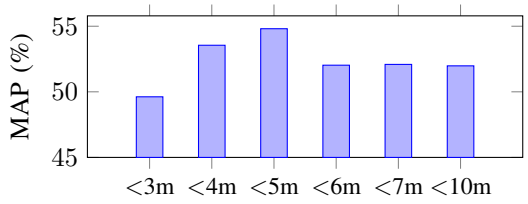


Fig. 7. Variation of the performance with respect to the distance.

using our Test set 2 that contains four sequences. Each sequence was evaluated independently and at each frame we use Intersection over Union and the Hungarian algorithm to find ground truth detection pairs. Hypotheses outside of the field of view of the camera were not considered. Test set 2 is especially challenging because of occlusions, which explains the drop in MAP compared to Test set 1. As can be seen in Fig. 5, the addition of the Kalman filter and the HMM improves the performance of the overall pipeline. The MAP improves by 4.6% from 47.37% to 51.98% compared to the raw Fast R-CNN output. Recall increases from 64.77% to 73.33%.

We also assess the variation of the performance with respect to the distance from the sensor. For a given distance d, this experiment considers detections and ground truth bounding boxes up to d meters only. Fig. 7 shows that our method performs best when detecting people up to five meters, achieving an MAP of 54.81%. The performance decreases to 51.98% MAP for larger distances and also notably for very short distances.

E. Person Guidance Scenario

To show the applicability of our framework to a real-world service robot task, we test our system in a person guidance scenario. The robot's task is to guide visitors to the professor's office in our lab, building 80 at the Faculty of Engineering of the University of Freiburg. The professor's office is located at the first floor, opposite the staircase at the main entrance. An elevator is available at the other side of the corridor for people with walking impairments.

Our robot uses a laptop computer with an 8-Core CPU and a GeForce GTX 1080 with 8GB of memory in order to process data at approximately 15 frames per second. We use the ROS navigation stack[1] for the navigation parts of the experiment. We select the initial waiting pose of the robot close to the main entrance as well as two goal poses, see Fig. 8. The robot observes an area of interest $3\,\mathrm{m}$ in front of it and within $\pm 20°$ from its center. Once it detects a person in this area, it waits for $4\,\mathrm{s}$ and until the confidence for one category exceeds $90\,\%$ before navigating to one of the two goals. For pedestrians, it navigates to the goal by the stairs; people with mobility aids are guided to the elevator. In addition, the robot adjusts its velocity according to the perceived category of the person. Once it reached its goal, it returns to the waiting position and waits for the next visitor. The robot further uses predefined speech commands to ask the visitors to follow it and inform them how to proceed to the professor's office once its navigation goal is reached.

We tested thirteen guidance runs with different people from our lab. Each category was tested twice, except pedestrian for which we tested five runs. In all of the runs, the robot perceived the correct category and successfully navigated the person to the correct location. However, in some runs, the people had to adjust their positions to trigger the navigation, because they were too close to the robot or outside of the observed area. The experiment confirms that our approach

[1] `http://wiki.ros.org/navigation`



Fig. 8. We use our framework to provide assistance in a person guidance experiment. The robot's task is to guide all pedestrians to the nearest staircase (left image) and all people with mobility aids to the elevator (right image). We told the test subjects to approach the robot at the initial position and then follow the robot's predefined speech commands.

can be applied on a real robot. Further, it shows how our framework can be used to give appropriate assistance to people, according to their needs.

V. CONCLUSIONS

We proposed a perception system to detect and distinguish people according to the mobility aids they use, based on a deep neural network and supported by tracking and class estimation modules. Our experiments show an increase in performance by the addition of these two modules to the pipeline. Moreover, we demonstrated that our approach for the proposal generation can speed up the classification process by a factor of up to seven compared to a dense sliding window baseline while achieving better performance. Additionally, we introduce an RGB-D dataset containing over 17,000 annotated images. In our person guidance experiment we showed that our detection pipeline enables robots to provide individual assistance to people with advanced needs. In the future, we plan to use sensor data from multiple sources to improve our pipeline such as using laser range finder readings to increase the accuracy of the proposal generation at larger distances. We will further examine how the additional information provided by our framework can improve the behavior of robots in populated environments, for example during autonomous navigation.

ACKNOWLEGMENTS

We would like to thank Oier Mees for his help with the detection experiments.

REFERENCES

[1] R. Girshick, "Fast R-CNN," *Int. Conf. on Computer Vision (ICCV)*, 2015.
[2] A. Ess, B. Leibe, K. Schindler, and L. van Gool, "Robust multiperson tracking from a mobile platform," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2009.
[3] W. Choi, C. Pantofaru, and S. Savarese, "A general framework for tracking multiple people from a moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.
[4] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.
[5] C. Dondrup, N. Bellotto, F. Jovan, and M. Hanheide, "Real-time multisensor people tracking for human-robot spatial interaction," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015.
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.
[7] Y. Li, K. He, J. Sun *et al.*, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.
[8] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.
[9] L. Beyer, A. Hermans, and B. Leibe, "DROW: Real-time deep learning-based wheelchair detection in 2D range data," *IEEE Robotics and Automation Letters*, 2017.
[10] T. Linder, S. Wehner, and K. O. Arras, "Real-time full-body human gender recognition in (RGB)-D data," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.
[11] M. Munaro and E. Menegatti, "Fast RGB-D people tracking for service robots," *Autonomous Robots*, 2014.
[12] L. Spinello, R. Triebel, and R. Siegwart, "Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.
[13] P. Sudowe and B. Leibe, "Efficient use of geometric constraints for sliding-window object detection in video," *Int. Conf. on Computer Vision Systems (ICVS)*, 2011.
[14] O. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," *Int. Conf. on Robotics and Automation (ICRA)*, 2014.
[15] L. Spinello and K. O. Arras, "People detection in RGB-D data," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.
[16] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals for accurate object class detection," *Advances in Neural Information Processing Systems (NIPS)*, 2015.
[17] G. Sharma and F. Jurie, "Learning discriminative spatial representation for image classification," in *British Machine Vision Conf. (BMVC)*, 2011.
[18] L. Bourdev, S. Maji, and J. Malik, "Describing people: A poselet-based approach to attribute classification," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2011.
[19] P. Sudowe, H. Spitzer, and B. Leibe, "Person attribute recognition with a jointly-trained holistic CNN model," in *IEEE Int. Conf. on Computer Vision Workshops (ICCVW)*, 2015.
[20] O. Mees, A. Eitel, and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.
[21] K. Bernardin, E. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," *IEEE Int. Workshop on Visual Surveillance*, 2006.

Mobile Robot for Retail Surveying and Inventory Using Visual and Textual Analysis of monocular pictures based on Deep Learning

Marina Paolanti[1], Mirco Sturari[1], Adriano Mancini[1], Primo Zingaretti[1] and Emanuele Frontoni[1]

***Abstract*— This paper describes a novel system for automating data collection and surveying in a retail store using mobile robots. The manpower cost for surveying and monitoring the shelves in retail stores are high, because of which these activities are not repeated frequently causing reduced customer satisfaction and loss of revenue. Further, the accuracy of data collected may be improved by avoiding human related factors. We use a mobile robot platform with on-board cameras to monitor the shelves autonomously (based on indoor UWB Localization and planning). The robot is designed to facilitate automatic detection of Shelf Out of Stock (SOOS) situations. The paper contribution is an approach to estimate the overall stock assortment based of pictures from both visual and textual clues. Based on visual and textual features extracted from two trained Convolutional Neural Networks (CNNs), the type of the product is identified by a machine learning classifier. The approach was applied and tested on a newly collected dataset and several machine learning algorithms are compared. The experiments yield high accuracy, demonstrating the effectiveness and suitability of the proposed approach, also in comparison of existing state of the art SOOS solutions.**

I. INTRODUCTION

Temporary Shelf Out of Stock (SOOS) situation is a considerable problem in the retail field. SOOS events are often strongly related to planogram design, where a planogram is the way how the stock keeping units (SKUs) are organized among the shelves [1]. The global average out-of-stock rate is estimated to be about 8%. This means for retailers about 4% losses in sales [2]. The out-of-stock situations arise from several reasons. The 70-90% of stockouts are caused by defective shelf replenishment practices. Instead, the 10-30% resulting from the supply chain problems. The first one leads to SOOS while the second one leads to store-OOS [3]. We are mainly focused in case of the items are available in the store warehouse or backroom but they are out-of-stock on the shelves. This situation frequently occurs with fast moving consumer goods (FMCG) which are depleted faster than their replenishment [4]. Higher checking frequency could reduce the SOOS related problems to a greater extent.

At present, several surveys are human-based at defined intervals, because if these are carried out assiduously, they would be expensive and time consuming, leading to lesser profit margins [5]. Furthermore, the data collected by humans is erroneous and unreliable. RFID based technologies are useful in dealing with SOOS situation apart from automating the supply chain management [6], [7], but these solutions require altering the store environment to accommodate antennae, sensors etc. Thus, more time and money for deployment are required. RFID-based solutions which require item-level tagging are also expensive for low cost grocery items.



Fig. 1: A usual example of combined SOOS and Promo in a supermarket shelf.

In this context, we propose ROCKy (Retail Out of stoCK) a mobile robot for detecting SOOS in real-time as well as on-demand, with a novigation approach based on carts and basket tracking system based on the assumption that where shoppers are passing and purchasing are the place to survey for SOOS. In addition to the identification of SOOS and misplaced items, ROCKy can provide several other value added services, like surveying and informing customers for promotions or discounts. It is also able to model changes in planogram and store layouts. It could also be used to monitor the warehouse and runs surveillance during night time. Finally, our approach enables retailers to compare the impact of different shelf layouts (planograms) on issues such as ease of selection, trading up and the overall shopping experience. Advantages of using robots for localization and main contributions of this paper are the proposed vision based surveying approach for SOOS and marketing materials and promo, the novel navigation system based on data coming from a monitor system and using carts and baskets tracking data, the first real scenario testing and dataset collection on this particular application. ROCKy consists of a TurtleBot [8], equipped on-board Kinect camera for navigation, a low-power netbook for running fundamental algorithms and a top camera for shelf image collection . It actually uses two cameras to capture images and videos of the shelves on either side of the robot (shelves on left and right side when passing in an aisle). The TurtleBot with cameras is shown in Figure 2.

[1]Marina Paolanti is with Dipartimento di Ingegneria dell'Informazione (DII), Universitá Politecnica delle Marche, 60131, Ancona, Italy m.paolanti@pm.univpm.it



Fig. 2: TurtleBot robotic platform with cameras and UWB tags for localization

The ROCKy tracking system is based on Ultra-WideBand (UWB) technology for robots collaboration. The system provides the use of several UWB antennas properly positioned inside a predetermined area and powered battery tags free to move inside the area. This techonology helps it to build up a map of the store and avoid obstacles, including customers pushing trolleys. Using the Kinect camera ROCKy captures images of the shelves in the store every minute during the business hours. The images obtained by ROCKy can be divided into three categories:

- *positive*: images of shelf with products at a special offer;
- *neutral*: images of planogram in a standard layout;
- *negative*: images of SOOS situations.

This paper also introduces an approach to estimate the overall content of the images acquired by ROCKy, based on both visual and textual information [9]. In fact, to classify these pictures as positive, neutral and negative, it is essential not only to judge the visual elements but also the included text at once. While a picture showing cookies with the phrase "Special Offer 50% off" is positive, the same picture containing the words "Gluten Free" might be neutral. The picture category is identified by a machine learning classifier based on visual and textual features extracted from two specially trained Deep Convolutional Neural Networks (DCNNs). The visual feature extractor, applied to the whole image, is based on the VGG16 network architecture and it is trained by fine-tuning a model pretrained on the ImageNet dataset. The textual feature extractor has to detect and recognize texts before extracting features. The textual feature extractor is based on the DCNN architecture proposed by [10] and is created by fine-tuning a model that has been previously trained on synthesized planogram images. Based on these features, six state-of-the-art classifiers, such as kNearest Neighbors (kNN) [11], [12], Support Vector Machine (SVM) [13], Decision Tree (DT) [14], Random Forest (RF) [15], Naïve Bayes (NB) [16] and Artificial Neural Network (ANN) [17], are compared to recognize the overall planogram image content. The approach has been applied to a newly collected dataset "ROCKy Dataset" of pictures acquired by ROCKy in a real retail environment during business hours. This dataset comprises 4200 images containing visual and textual elements concerning shelves in the store target. In contrast to many existing datasets, the true content is not automatically judged by the accompanying texts but it has been manually estimated by human annotators, thus providing a more precise dataset. The application of our approach to this dataset yields good results in terms of precision, recall and F1-score and demonstrates the effectiveness of the proposed approach.

The paper is organized as follows: Section II is an overview of the research status of robots and techonologies employed in retail field; Section III describes the UWB techology used to monitor the trajectories in store; Section IV introduces our approach consisting of a visual model (Subsection IV-A), a textual model (Subsection IV-B), a fusion model (Subsection IV-C) and gives details on the "ROCKy Dataset" (Subsection IV-D); final sections present results (Section V) and conclusions (Section VI) with future work.

II. RELATED WORK

This section is an overview of the papers in retail fields in which the robots as well as computer vision technologies are applied. In [18], the authors use a network robot system to incorporate recommendation methods used in e-commerce system into a real-world retail shop. While sensors like LRF and cameras are employed in order to analyze pre-purchase behaviour of customers [19], the physical robots are used to present recommendations and show directions. In another paper [20], Matsuhira et. al. has developed a robotic transport system to assist people during shopping. The system consists of two robots: one for following the customer and the other acts as shopping cart carrying purchased items. The on-board robot sensors and environmental cameras facilitate human-robot interaction. In [21], it is presented RoboCart that assists visually impaired customers to navigate a typical grocery store and carry purchased items. It is based on RFID tags for localization and laser for navigation. A mobile robot system for assisting and interacting with customers is presented in [22]. The mobile robot was capable of moving from one location to another and could accept inputs from a customer and give output verbally or in written messages. A shopping cart that can detect the items lying on the bottom of shopping cart is developed in [23], [24]. It used a visual sensor that identifies products using visual features like SIFT. The aim was to speed up the check out process. Purohit et al. [25] have developed SugarTrail, a system for indoor navigation assistance in retail environments that does not require existing maps. By using the measurement sequence as combined signatures (paths), the proposed system learns traversable path-clusters to build a navigable virtual roadmap of the environment. A similar work proposed for retail field is [26].

The authors discuss the results and implications of front-end and formative evaluation studies that they conducted during the development of MyGrocer, a second-generation pervasive retail system. They developed this system, that improves shopping experience, to investigate grocery supply-chain optimization using pervasive computing technologies. Currently, in intelligent retail environment two strategy are adopted for monitoring the customers behaviour and trajectories inside the store during the shopping experience: UWB tecnologies and RGB-D cameras. The UWB technology for indoor tracking is one of the most promising solution in terms of accuracy, reliability, battery life and system cost, and it is able to monitor the movement of consumers in stores and to send tracking data to a cloud server. It allows to derive several information on the shoppers behaviour inside the store (flows of walking, most visited areas inside the space dedicated to the shopping and average travel times). Other useful information are provided by computer vision technologies. In fact, several papers have proposed the use of RGB-D cameras for shoppers monitoring in front of a shelf [27], [28]. The design of an embedded wireless sensor network able to detect in real time a shelf out-of-stock event is presented in [29]. In particular, the authors present a shelf detector sensor based on a low cost and low power wireless sensor network design that can automatically discover SOOS for all the stores of a retail chain. In this paper, we improve our previous experiences in solutions for intelligent retail envronment, by proposing ROCKy to monitor the shelves autonomously (based on indoor UWB Localization and planning). The robot is designed to facilitate automatic detection of SOOS situations. The important contribution is also an approach to estimate the classification of stock assortment by shelves pictures acquired by ROCKy. The details of the proposed scheme is discussed in the next sections.

III. SYSTEM OVERVIEW AND UWB NAVIGATION TECHNOLOGY

ROCKy robotic system detects SOOS in real-time as well as on-demand. The system leverages the structured movement trajectories in store to enhance accuracy and minimize time for surveying. This aspect is important for robots collaboration. The authors have experience in mobile robot localization using computer vision techniques [30], [31]. On the contrary, ROCKy navigation is based on UWB localization and tracking technology. The same system is used in the real store used for results and test, to localize and track carts and baskets for marketing research purposed and consumer behaviours. The system is able to monitor the trajectories in stores and send the collected tracking data to a cloud server. These data, properly processed and stored in a database, are useful to obtain information about ROCKy and customers behaviour during the day. UWB signals are transmitted with a shorter duration, consuming less power and can operate in a wide area of the radio spectrum. UWB and RFID can operate in the same area without interference thanks to the differences in signal types and radio spectrum. Moreover, UWB signal is able to pass through walls, devices and clothes with no interferences. The technology has been used to deploy a RTLS (Real Time Locating System), in order to collect real time localization data. Tests conducted in a real store achived an accuracy in the position measurement of 20 cm in terms of indoor localization and a smart power management provides a high autonomy for the battery-powered tags. Conventional wireless applications (such as RFID, WLAN and others) do not reach this value, and so it is very useful when a high levels of precision in real time for 2-D and 3-D localization is required. The UWB radio module DWM1000 advent (IEEE802.15.4-2011 UWB compliant) by decaWAVE, at a low price (10$ per 1000 unit), has given a great number of tracking solutions, developed by European companies. The parameters, that can be measured, are: time of arrival, the direction of arrival and the signal strength.

The components that compose the tracking system are listed below.

- **Anchors**: static devices (antennas) that form a grid covering the whole store. They are placed in the dropped ceiling of the shop and gather signals from the tags. This data are forworded (timestamps of received signals and tags related information such as ID and battery level) to the RTLS server. The anchors are connected and powered via Ethernet through a PoE switch to the RTLS server. Since all the anchors must be at the same time, one of them is chosen as the master anchor for synchronizing them.
- **Tags**: mobile devices to be tracked. They send data to the anchors at a specified transmission rate as well as a broadcast message, received by the anchors in a communication system (this makes their number fully scalable). The tag presents also an accelerometer for movement detection that prolongs battery lifetime. In this way, new data are sent only if the tags exceed a certain and adjustable threshold.
- **RTLS Server**: collects data from the anchors, estimates the 3D position of tags and sends these to the cloud server. A TDOA (Time Difference of Arrival) algorithm is used for estimating a tag position through multilateration. It takes into account only timestamps coming from at least three anchors with the same time.

The RTLS Server sends localization data on a TCP/IP socket. A software is developed to collect from the stream the following information: master ID, tag ID, position coordinates (x, y and z), battery level of the tag and timing information (such as tracking system and RTLS server timestamp). ROCKy is using previous collected daily information about the carts and basket store heat-map to plan its navigation based on maxima in the heatmap (Fig 3). Navigation system uses waypoints in the red areas assuming that most visited store categories are also the most affected by SOOS.

IV. VISUAL SURVEYING METHODS

In this section, we introduce the combined use of visual and textual features as already done in [9]. The framework for joint visual and textual analysis as well as the dataset used for evaluation comprises three main components: the visual



Fig. 3: The Store heatmap based carts and basket tracking using UWB tags is the base for motion planning of the ROCKy platform. The heatmap is a daily localization data aggregation of about 300 carts and baskets moving in the store in 10 hours, tracked at 1Hz.

feature extractor, the textual feature extractor and the fusion classifier. For visual and textual feature extraction we use two especially trained DCNNs. Then, the visual and textual features are combined and fed into the fusion classifier. For estimating the overall content of image, we compare common machine learning algorithms. Further details about the main components are given in the following subsections.

The framework is comprehensively evaluated on the "ROCKy Dataset", a proprietary dataset collected for this work. The details of the data collection and ground truth labelling are discussed in Subsection IV-D.

A. VISUAL FEATURE EXTRACTOR

The visual feature extractor provides information about the visual part of a picture and is therefore trained with image labels indicating the visual category of the images. The training is performed by fine-tuning a VGG16 net [32]. We fine-tune by cutting off the final classification layer (fc8) and replacing it by a fully connected layer with 3 outputs (one for each category class). In addition, the learning rate multipliers are increased for that layer so that it would learn more aggressively than all the other layers. Finally, loss and accuracy layers are adapted to take input from the new fc8 layer. Since the image classifier serves as feature extractor, the output of the next to last fc7 layer is passed to the overall polarity classifier. The image feature extractor is implemented using standard Caffe tools [33].

B. TEXTUAL FEATURE EXTRACTOR

The textual feature extractor aims at providing information about the textual category of a picture. It is therefore trained with image labels indicating the textual category of the images. The textual feature extractor consists of multiple components. The central component is a character-level DCNN with an architecture, as described in [10], which has been extended by one additional convolution layer. The extra convolution layer, inserted before the last pooling layer, has a kernel size of 3 and produces 256 features. The textual feature extractor was trained in two phases: first training a base model on synthesized planogram images and then fine-tuning that base model on our dataset. Since the text is embedded in the picture as pixels, the text has to be transformed to characters before it can be processed by the character-level DCNN. For this purpose, the following steps have been performed:

1) *Text Detection*: individual text boxes are detected in an image with the TextBoxes Caffe model [34].
2) *Text Arrangement*: detected text boxes are put in order based on a left-to-right, top-to-bottom policy, thus forming logical lines.
3) *Text Recognition*: each text box is processed by the OCR model [35] to transcribe the text of the box.
4) *Text Encoding*: the recognized text is encoded into one-hot vectors based on the alphabet of the character-level DCNN.

The textual features of the next to last layer of the character-level DCNN are passed to final fusion classifier.

C. FUSION CLASSIFIER

On the basis of the visual and textual features, the fusion classifier estimates the overall content of an image. For this purpose, it is trained with labels indicating the overall category of the images.

Based on all features, six state-of-the art classifiers, kNN, SVM, DT, RF, NB and ANN are compared to recognize the overall content of the images and we evaluate performance in terms of precision, recall and F1-score.

D. ROCKY DATASET

In this work, we build a visual and textual retail dataset from the pictures acquired by ROCKy. The "ROCKy Dataset" is composed of shelves images. As stated previously, these images are divided into three categories. In particular, it includes:

- 1400 positive images;
- 1400 neutral images;
- 1400 negative images.

To obtain the ground truth of the collected pictures, the true content has been manually estimated by human annotators, thus providing a more precise and less noisy dataset. All pictures are annotated with respect to their visual, textual and overall content.

Figure 4 shows three examples of pictures of "ROCKy Dataset". Figure 4a is a negative picture; Figure 4b is a neutral image; and Figure 4c shows a positive picture. As can be seen, the overall content does not only depend on the visual content of a picture, but also on its textual content.

V. RESULTS AND DISCUSSION

In this section, the results of the experiments conducted on "ROCKy Dataset" are reported. In addition to the performance of the fusion classifier, we also present the performance of the visual and textual category classifiers which



(a) (b) (c)

Fig. 4: Pictures of "ROCKy Dataset". Figure 4a is an example of negative picture, Figure 4b represents a neutral image and Figure 4c shows a positive picture.

TABLE I: Performance evaluation of the visual model

Category	Precision	Recall	F1-Score
positive	0.83	0.82	0.82
neutral	0.86	0.89	0.88
negative	0.72	0.67	0.69
MEAN()	***0.81***	***0.79***	***0.80***

TABLE II: Performance evaluation of the text model

Category	Precision	Recall	F1-Score
positive	0.84	0.89	0.71
neutral	0.71	0.68	0.70
negative	0.67	0.61	0.76
MEAN()	***0.74***	***0.73***	***0.74***

TABLE III: Performance evaluation of the Fusion Classifier.

Classifier	Precision	Recall	F1-Score
NB	0.72	0.72	0.72
DT	0.72	0.72	0.72
RF	0.74	0.74	0.74
SVM	0.77	0.77	0.77
kNN	0.78	0.78	0.78
ANN	**0.79**	**0.79**	**0.79**

form the basis of the visual and textual feature extractors and are key to the overall classification.

The experiments are performed by splitting the labelled dataset into a training set and a test set. Each classifier will only be trained based on the training set. Likewise, the test set is also fixed in the beginning and used for all test purposes. The dataset is split into 80% training and 20% test images, taking into account all permutations of overall, visual, and textual annotations.

In order to create the visual feature extractor we trained a DCNN to classify the visual part of a picture. The performance of the visual classification is reported in Table I. As can be seen, high values of precision and recall can be achieved, especially for pictures with positive and neutral visual content. The recognition of visually negative pictures is more difficult due to the smaller amount of available training data. Generally, retailers pay more attention to the SOOS situation and they tried to solve this problem immediatly when it occours.

For creating a textual feature extractor, we trained a DCNN to estimate the text in the pictures. Table II depicts precision and recall of the textual classification. The performance of the textual classification is good, but lower than the performance of the visual classification. While the judgement of visual and textual image content is equally difficult for humans, the classification of text in pictures much more challenging for machines as the text has to be detected and recognized first before it can be classified thus beeing more error-prone. Comparing the different classes reveals that positive and neutral texts can be recognized better than positive texts. This fact is also reflected by the characteristics of the dataset.

Based on the visual and textual features, a machine learning classifier is trained to identify the overall content of a pictures. We train several classifiers, namely SVM, DT, NB, RF and ANN and compare their performance. As Table III shows, the performance of all classifiers is good, with a F1-Scores ranging from 0.72 for NB to 0.79 for ANN, thus demonstrating the effectiveness and the suitability of the proposed approach. The performance of the overall content classification is much higher than the performance of the textual classification but slightly lower than the performance of the visual classification. This comparision shows that recognizing the overall content is more challenging than only the visual part.

VI. CONCLUSIONS

SOOS situation is an important problem in the retail field. In this paper, we propose ROCKy a mobile robot to detect SOOS in real-time as well as on-demand. In addition to the identification of SOOS and misplaced items, ROCKy can provide several other value added services like informing customers in the form of promotions or discounts. It is also able to model changes in planogram and store layouts. It could also be used to monitor the warehouse and runs surveillance during night time. ROCKy is not trying to replace workers, but instead free them up to spend more time helping customers.

ROCKy consists of a TurtleBot equipped on-board Kinect camera for navigation and a low-power netbook for running

fundamental algorithms. It uses two USB cameras to capture images and videos of the shelves on either side of the robot. It is based on indoor UWB Localization and planning. In this paper, we also introduce a deep learning approach for recognizing the content of shelves pictures acquired by ROCKy by taking visual as well as textual information into account. The content of a picture is identified by a machine learning classifier based on visual and textual features extracted from two trained DCNNs. By combining DCNNs with machine learning algorithms such as kNN, SVM, DT, RF, NB and ANN the approach is able to learn a high level representation of both visual and textual content and achieve high precision and recall for classification. The experiments on "ROCKy Dataset" yield high accuracy and demonstrate the effectiveness and suitability of our approach. Further investigation will be devoted to improve our approach by employing a larger dataset and extracting additional informative features.

REFERENCES

[1] E. Frontoni, F. Marinelli, R. Rosetti, and P. Zingaretti, "Shelf space re-allocation for out of stock reduction," *Computers & Industrial Engineering*, vol. 106, pp. 32–40, 2017.

[2] H. Che, J. Chen, and Y. Chen, "Investigating effects of out-of-stock on consumer sku choice," 2011.

[3] T. W. Gruen and D. S. Corsten, "A comprehensive guide to retail out-of-stock reduction in the fast-moving consumer goods industry," 2007.

[4] S. Kumar, G. Sharma, N. Kejriwal, S. Jain, M. Kamra, B. Singh, and V. K. Chauhan, "Remote retail monitoring and stock assessment using mobile robots," in *Technologies for Practical Robot Applications (TePRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1–6.

[5] D. Calvaresi, A. Vincentini, A. Di Guardo, D. Cesarini, P. Sernani, and A. F. Dragoni, "Exploiting a touchless interaction to drive a wireless mobile robot powered by a real-time operating system." in *RTA-CSIT*, 2016, pp. 1–10.

[6] M. Kärkkäinen, "Increasing efficiency in the supply chain for short shelf life goods using rfid tagging," *International Journal of Retail & Distribution Management*, vol. 31, no. 10, pp. 529–536, 2003.

[7] D. Corsten and T. Gruen, "Desperately seeking shelf availability: an examination of the extent, the causes, and the efforts to address retail out-of-stocks," *International Journal of Retail & Distribution Management*, vol. 31, no. 12, pp. 605–617, 2003.

[8] "Turtlebot, open-source robot development kit for apps on wheels." [Online]. Available: http://www.turtlebot.com

[9] M. Paolanti, C. Kaiser, R. Schallner, E. Frontoni, and P. Zingaretti, "Visual and textual sentiment analysis of brand-related social media pictures using deep convolutional neural networks," in *Image Analysis and Processing, 2017. ICIAP 2017. 19th International Conference on*.

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[11] T. H. Bø, B. Dysvik, and I. Jonassen, "Lsimpute: accurate estimation of missing values in microarray data with least squares methods," *Nucleic acids research*, vol. 32, no. 3, pp. e34–e34, 2004.

[12] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[13] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[14] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[15] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[17] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

[18] K. Kamei, K. Shinozawa, T. Ikeda, A. Utsumi, T. Miyashita, and N. Hagita, "Recommendation from robots in a real-world retail shop," in *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*. ACM, 2010, p. 19.

[19] E. Frontoni, P. Raspa, A. Mancini, P. Zingaretti, and V. Placidi, "Customers activity recognition in intelligent retail environments," in *International Conference on Image Analysis and Processing*. Springer, 2013, pp. 509–516.

[20] N. Matsuhira, F. Ozaki, S. Tokura, T. Sonoura, T. Tasaki, H. Ogawa, M. Sano, A. Numata, N. Hashimoto, and K. Komoriya, "Development of robotic transportation system-shopping support system collaborating with environmental cameras and mobile robots," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*. VDE, 2010, pp. 1–6.

[21] V. Kulyukin, C. Gharpure, and J. Nicholson, "Robocart: Toward robot-assisted navigation of grocery stores by the visually impaired," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 2845–2850.

[22] A. Bancroft and C. W. Ward, "Methods for facilitating a retail environment," Apr. 17 2007, uS Patent 7,206,753.

[23] J. Ostrowski, L. Goncalves, M. Cremean, A. Simonini, and A. Hudnut, "System and methods for merchandise checkout," Sep. 5 2006, uS Patent 7,100,824.

[24] A. Hudnut, A. Simonini, M. Cremean, and H. Morgan, "Method of merchandising for checkout lanes," Jul. 24 2007, uS Patent 7,246,745.

[25] A. Purohit, Z. Sun, S. Pan, and P. Zhang, "Sugartrail: Indoor navigation in retail environments without surveys and maps," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*. IEEE, 2013, pp. 300–308.

[26] P. Kourouthanassis and G. Roussos, "Developing consumer-friendly pervasive retail systems," *IEEE Pervasive Computing*, vol. 2, no. 2, pp. 32–39, 2003.

[27] A. Mancini, E. Frontoni, P. Zingaretti, and V. Placidi, "Smart vision system for shelf analysis in intelligent retail environments," in *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, Portland, OR, USA*, vol. 47, 2013.

[28] D. Liciotti, M. Contigiani, E. Frontoni, A. Mancini, P. Zingaretti, and V. Placidi, "Shopper analytics: A customer activity recognition system using a distributed rgb-d camera network," in *International Workshop on Video Analytics for Audience Measurement in Retail and Digital Signage*. Springer, 2014, pp. 146–157.

[29] E. Frontoni, A. Mancini, P. Zingaretti, M. Contigiani, L. Di Bello, and V. Placidi, "Design and test of a real-time shelf out-of-stock detector system," *Microsystem Technologies*, pp. 1–9, 2016.

[30] P. Zingaretti and E. Frontoni, "Vision and sonar sensor fusion for mobile robot localization in aliased environments," in *Mechatronic and Embedded Systems and Applications, Proceedings of the 2nd IEEE/ASME International Conference on*. IEEE, 2006, pp. 1–6.

[31] E. Frontoni and P. Zingaretti, "A vision based algorithm for active robot localization," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*. IEEE, 2005, pp. 347–352.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[33] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[34] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," *arXiv preprint arXiv:1611.06779*, 2016.

[35] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

Robotic Platform for Deep Change Detection for Rail Safety and Security

Mirco Sturari[1], Marina Paolanti[1], Emanuele Frontoni[1], Adriano Mancini[1] and Primo Zingaretti[1]

***Abstract*— Felix is the first robot measuring switch dimension parameters and geometrical railway ones together with potentially dangerous situations. It increases the quality and the consistency of the measures and offers the chance to increase the safety of operators and ultimately of final users. This paper presents a novel approach mixing visual and point cloud information for effective change detection for railways safety and security application. Main contributions are on the proposed data platform and on the mix of point-based measurements (switch dimensions) and on surrounding change detection (dangerous trees or abnormal railway trawlers) based multi-view camera and linear laser, processed by a classification process. Results, coming from a real rail scenario using the Felix platform, show the feasibility of the approach and the fast surveying capabilities with strong implications on safety and security.**

I. INTRODUCTION

Railways are currently the most common transportation means for both passengers and freights in Europe. Their capacity, speed, and reliability have increased their popularity, even if high risks sometimes occur in term of human life and cost of assets. The poor maintenance of the railways is a critical point for security. In last years, several technologies for better safety measures are introduced but this problem is not completely solved. Thus, a proper strategy is required for maintenance and inspection of tracks. The defects principally include weld problems, internal defects worn out rails, head checks, squats, spalling and shelling, corrugations and rolling contact fatigue (RCF) as well as surface cracks [1]. The risk of derailments increases if mud, snow, ice and debris collect on and around tracks, as well as the trees that overhang railway lines can be hazardous, especially within the switches and crossings (S&C) that ensure all possible connections. A switch consists of moving point blades that ensure a change in the train route. S&C are key elements in a railway network because they provide flexibility to traffic operation [2].

There are many challenges to rail community and the infrastructure maintenance operators such as to perform effective inspection and cost effective maintenance decisions. In this field, several papers are proposed with the main goal to improve the railway safety and security [3].

For this reasons, Loccioni Research for innovation and RFI (Rete Ferroviaria Italiana, the main Italian Railways Group) have developed Felix, the first mobile robot for the automatic railway switch inspection, with the objective to increase railway switches reliability, guarantee railways safety and solve maintenance planning problems. Felix robot is able to monitor S&C status. It is equipped with modules to scan the railway as profilometers that create a 3D reconstruction of the inspected segment. According to this characteristics, Felix is the first robot measuring switch dimension parameters and geometrical railway ones, increasing the quality and the consistency of the measures and offering the chance to increase safety of operators and ultimately of final users. The system stores, processes and displays in real time all measurements, creating specific reports. It improves the quality of measurement, makes surveys objective and avoids errors in data acquisition and data transfer phases. During the measurement, the system is remotely managed with a PDA (Personal Digital Assistant) and a handheld remote control unit, ensuring greater safety to the operators. The motorized device makes it possible to analyse a sequence of several switches. Speed and minimum distance between two consecutive measurements can be set on customer requirements. Felix brings innovation to railway maintenance processes, facilitating and enhancing them. The automation of measurement makes it faster and easier to perform diagnosis and quality control, with a significant impact on the safety of operators and ultimately of final users. Figure 1 is a real picture of Felix during inspection.



Fig. 1: Felix robot for the automatic railway switch inspection.

This paper describes a Felix evolution architecture with a novel approach that combines multi-view visual information and point cloud information for effective change detection on

[1]Mirco Sturari is with Dipartimento di Ingegneria dell'Informazione (DII), Universitá Politecnica delle Marche, 60131, Ancona, Italy m.sturari@pm.univpm.it

railways safety and security application. The fusion of laser and visual data is performed to increase the overall accuracy of the process [4]. It can be done both on line and in real time during the robot motion, but data can be also collected for off line processing. Both data sources can be accessed for human visualization. These can be finalized to human augmented operators inspections directly at raw data level or, after the processing for automatic change detection, with a semantic change detection layer over imposed on the point clouds or on the panoramic images for change detection. In this way, the augmented operator inspection interface. is better highlighted.

Main contributions are on the novel mobile robotics application, on the proposed data platform and on the mix of point based measurements (switch dimensions) and on surrounding change detection (dangerous trees or abnormal railway trawlers) based multi view camera and linear laser, processed by a deep learning approach. Results, coming from a real rail scenario using the Felix platform, show the feasibility of the approach and the fast surveying with strong implications on safety and security.

The paper is organized as follows: Section II is an overview of the research status of security railways technologies; Section III introduces the proposed architecture consisting of Point Cloud data collection and visualization (Subsection III-A), Multi-camera data collection and visualization (Subsection III-B), Data visualization for augmented operators (Subsection III-C) and Change Detection automatic processing on both data sources (Subsection III-D); Section IV gives details on the classification process and presents results gathered in a real scenario; final section provide conclusions and future works.

II. RELATED WORKS

In this context, several works are proposed. In [5], the authors have focused on the inspections methods limitations to detect defects in the rail foot, especially in the side edges away from the region directly below the web. They proposed the Long Range Ultrasonic Testing (LRUT), a technique to examine the foot of rails, mainly in track regions where corrosion and associated fatigue cracking is likely, such as at level crossings.

A technique based on Ground-penetrating radar (GPR) is proposed in [6] for obtaining quantitative information about the depth and degree of deterioration of the track. This work aims at automating the processing and interpretation of data to the extent whereby on-site interpretations may be achieved with minimal intervention of the expert. This is done through the development of new image and signal processing tools specifically for GPR data and the range of anomalies found on the trackbed.

A model for placing sensors on the railway track is described in the system presented in [7]. The data are collected from sensor nodes distributed on the railway tracks.

In [8], the authors propose a rail track inspection technique using automated video analysis. The system replaces manual visual checks performed by the railway engineers for track inspection. In the paper, it is used a combination of image processing and analysis methods in order to achieve high performance automated rail track inspection.

For evaluating if a structure is safe or damaged and for determining whether or not movement of the object has taken place, change detection of linear infrastructure features such as railways, highways, and bridges involves the systematic measurement and tracking of the alteration in the shape of the monitored object [9], [10]. Change detection is for a binary answer, a situation changed or not, whereas deformation analysis looks for a quantified change [11]. Several change detection techniques have been developed and reviews of their applications and recommendations for selection of suitable change detection methods have been reported in [12].

Agouris et al. [13] developed a differential snake model for change detection in road segments, by the use of medium-resolution imagery with a ground pixel resolution of 5 m. In [14], it has been proposed a Modified Iterated Hough transform (MIHT) algorithm for change detection in roads, employing aerial photographs and digital maps. An approach for detecting the channel change in the Chapping River in three periods (i.e., pre-disaster, during the disaster, and post-disaster), based on an edge-based segmentation algorithm and a support vector machine (SVM) classier is proposed by Gong in [15].

Different point cloud comparison techniques have been applied by many researchers to detect change between sequentially gathered point clouds [16].

The cloud-to-mesh method has been used to investigate structural and surface deformation. Gridded data sets can be compared to produce a DEM of difference, which highlight areas of loss or accumulation [17]. Currently, many researchers have implemented methods for deforming object surface modelling that involve a simple griddding [18]. Anyway, isolationg regions of interest and creating DEMs or surface models are time-consuming an interpolation is required. Moreover, a topographic data reduction occurs when point cloud data are reduced from 3D to 2.5D. In [19], the authors have stated that deformation studies are performed by directly comparing DEMs from different TLS campaigns. This approach is quite easy to implement using commercial software, but general-purpose algorithms are usually not sufficient for specific purposes such as change detection [20].

Our paper aims at describing a feasibility approach mixing visual and point cloud informations for effective change detection on railways safety and security application. Results are related to inspections performed in a real scenario using the Felix platform.

III. SYSTEM OVERVIEW

As stated previously, this system is developed in an actual railway context that imposes constrains on the system design. The solution proposed is described by the work flow scheme shown in Figure 2.

This architecture can be described in 4 important steps:

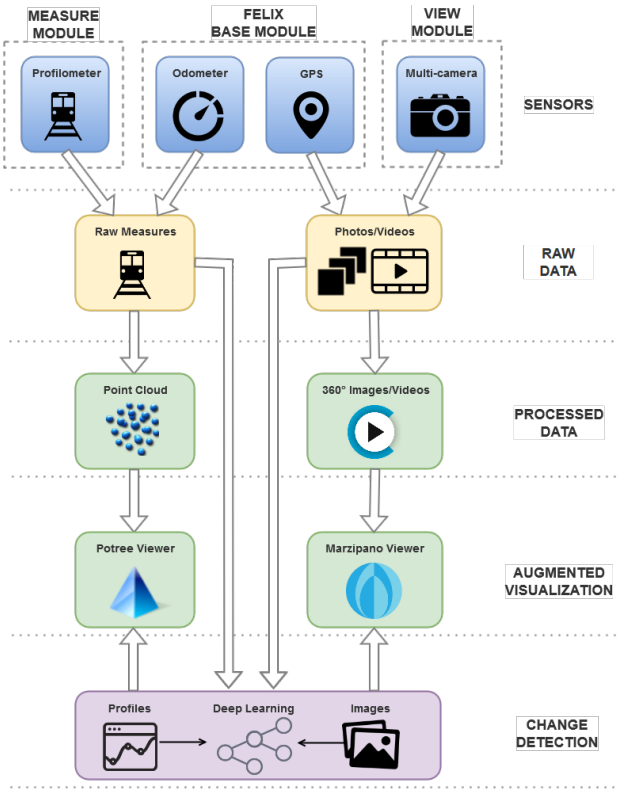


Fig. 2: System Architecture.

- Point Cloud data collection and visualization (Section III-A);
- Multi-camera data collection and visualization (Section III-B);
- Augmented operators with data visualization features on both raw data layer and processed data layer with over imposed semantic annotations based on change detection (Section III-C);
- Change detection automatic processing on both data sources (Section III-D).

This section gives an overview of our approach.

A. Point Cloud data collection and visualization

Felix is equipped by a laser profilometer for the reconstruction of surfaces under inspection. To obtain the high-meaningful data required for 3D reconstruction of the inspected parts, Felix moves at a speed that ranges from $0.36\,km/h$ to $3.6\,km/h$ while the profilometer describes a profile every $2-20\,mm$. In the most critical areas of railway switch, Felix proceeds at a slower rate to get additional profiles. The on board computer adjusts the speed identifying different sections of the railway from laser profiles. Laser profilometers follow the same principles of structured light, for which a laser line fitting on the target results deformed. Assuming that the relative position of laser and camera are known, triangulation laws can derive the position of the line in an absolute reference system [21]. The laser profilometer is a non-contact device that allows obtaining x, y, and z coordinates, which are processed and then converted in .pts

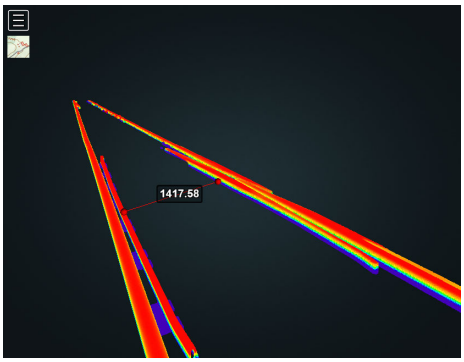


Fig. 3: Augmented operator: web based and VR compliance Point Cloud visualization based on Potree Viewer.

files. The collected dataset of a railway surface consists of both the coordinates and corresponding heights of the object surface. Finally, for each object under investigation, a 3D point cloud is generated using the PotreeConverter tool [22]. This tool is usually employed to convert an input coordinate files to the required point cloud data structure. The data structure in output can be easily visualised on a web platform based on Potree, a viewer for a large point cloud/LIDAR datasets (Figure 3).

B. Multi-camera data collection and visualization

Commercial GoPro Inc. HD personal cameras are used in extreme action video photography. They are compact, light, rugged and wearable or mountable on robots. These fullframe fish eye cameras capture still photos and/or video in HD through wide angle lens and can be configured to work automatically with minimum intervention or remotely controlled. The GoPro Hero 4 Black cameras, which are used in this paper, are set to record video with a resolution of 1920×1440 at frame rates of $24\,frames/s$. At the same time, these cameras take 12 MP photos ($4:3$ format with a resolution of 4000×3000 and a horizontal FOV of 122.6 degrees) every $5\,seconds$. Six cameras are mounted in a 360H6 rig and are fixed in the top of Felix, as in Figure 1. The acquisition was controlled remotely and the six videos are processed simultaneously to obtain a $360°$ video using Video Stitch software and $360°$ panoramic photos using Hugin software. Both softwares use the same preset to calibrate the six cameras for obtaining the stitch videos/photos. The output of this two processes can be visualised on a web platform based on Marzipano (Figure 4). Marzipano is a viewer of panoramic photos and $360°$ videos with the capability to organize them in a virtual tour.

C. Data visualization for augmented operators

The data visualization layer is designed to be accessed by web browsers, with mobile responsive interfaces, to allow on field data visualization both for raw data and for processed and classified data for change detection. The visualization layer, using also geolocation data, can also be accessed as an Augmented Reality (AR) layer using the AR browser Layar. Main visualization features are:

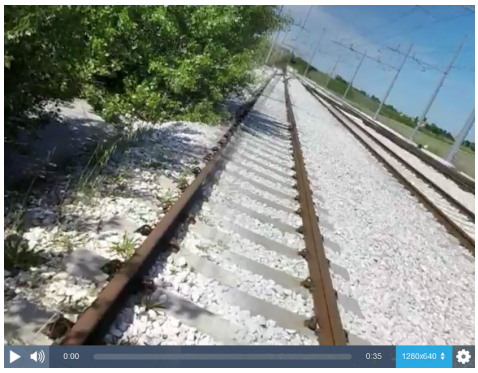


Fig. 4: Augmented operator: web based and VR compliance multi-view visualization based on Marzipano Viewer.

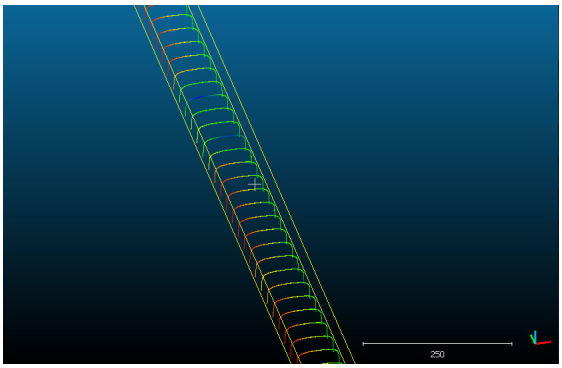


Fig. 5: Change detection visualization based on point cloud colorimetric features. Point Cloud visualization based on Potree Viewer.

- Data navigation in the space with pan, tilt and zooming functions;
- Layer annotation and selection on point cloud data;
- Metric dimensional measurement in distances, surfaces and volumes;
- Areas definition and annotation for dataset annotation and change detection classification visualization;
- Mobile accelerometers integration for immersive visualization;

D. Change Detection Approach on both data sources

Railways face a lot of changes all the time due to the defects that principally include weld problems, internal defects worn out rails, head checks, squats, spalling and shelling, corrugations and rolling contact fatigue, surface cracks and the trees overhanging. 360° images are used for change detection by applying image differencing and, thus, point clouds can be used to detect changes in a similar way. A change detection method should automatically find the changes in the studied area and update the corresponding map, model or database accordingly. Change detection can be done in several ways: by the comparison of raw point clouds, colored point clouds, voxels, range images, surface models or modeled objects. The common way to compare two point clouds is to measure distances between points in different point clouds and evaluate if a point in one point cloud has neighbors nearby in the other point cloud, otherwise the point is marked as a potential change. Adding the average color of images into the point clouds provides additional information about the possible changes. Point clouds can be also converted into 3D surfaces. Changes can also be detected using object models. If the measured point cloud can be modeled into separate classified objects, it can compared if the objects can be found in both point clouds.

Every generated point cloud is associated with a specific segment that is uniquely identified in the infrastructure network. This is necessary to correctly manage S&C and trace variation over time.

IV. CLASSIFICATION PROCESSES AND RESULTS

The proposed method uses two different classifiers working on the two described data sources: i) a feature extractors based on point cloud segments is used with classical machine learning methods to obtain a segment based classification with a linear map of changes and anomalies ; ii) a deep learning method is used to classify different classes of anomalies with respect to the rail road. Here below a detailed description of proposed features, methods and related results are presented.

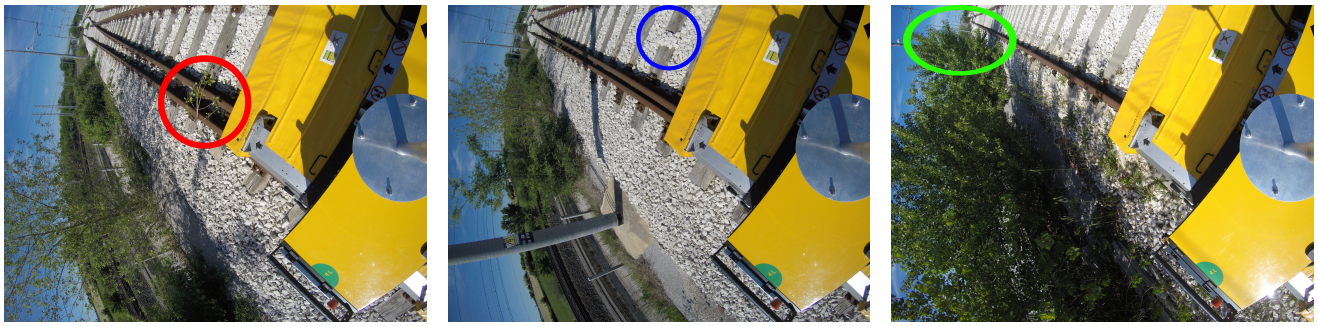
Every scanned profile is represented by 4400 points, each one described by $P_i(x_i, y_i, z_i)$, for each profile (left and right side) for a total of 8800 points. Every profile is acquired at a fixed distance of 2 mm from the previous one along the y dimension. The full set of profilometer data is divided in 64 different slices in the x dimension and every slice is described by 12 geometric features: 4 different statistical features (avg, trend, min-max, standard deviation) for 3 different geometric measures (hight, gradient, 3-point curvature). Every slice is labelled in the data set used for the learning in two classes (normal, abnormal) and the combination of the classes give back to the user the location of the area affected by usury and considered abnormal for safety reasons. The actual dataset is composed by 11228 different real profiles and is annotated by experts according to the EU law on railway safety. The purpose of the classification process is the safety detection map estimation for every new profile acquired by Felix to be used together with image based deep learning approach for real time dangerous area detection.

The six cameras have obtained 10100 frames with a resolution of 960×720 and on the bases of these frames we built a dataset. These images are divided into four classes and they have been manually estimated by human annotators, thus providing a more precise dataset. In particular, it includes:

- normal images;
- tree-anomaly images;
- trawlers anomaly images;
- railway lines anomaly images.

Figure 6 shows three anomaly examples. Figure 6a is a example of track anomaly Figure 6b shows an example of trawlers anomaly; and Figure 6c is a tree anomaly photo.

The training is performed by fine-tuning a VGG16 net [23] to classify images into four categories similarly to what done



(a) (b) (c)

Fig. 6: Examples of anomaly. Figure 6a represents an example of track anomaly, Figure 6b shows an example of trawlers anomaly and Figure 6c is a tree anomaly.

in [24]. A fine-tuning by cutting off the final classification layer (fc8) is done and replacing it by a fully connected layer. In addition, the learning rate multipliers are increased for that layer so that it would learn more aggressively than all the other layers. Finally, loss and accuracy layers are adapted to take input from the new fc8 layer. Since the classifier serves as feature extractor, the output of the last fc7 layer is passed to the new fc8. The features extractor is implemented using standard Caffe tools[1].

Results of the experiments conducted on both profiles and the images acquired are reported. We present the performance of the classifiers on the basis of the features extractors that are key to the rail road profile classification and of the Deep Convolutional Neural Network (DCNN) for vision based inspections.

The experiments are performed by splitting the labelled dataset into a training set and a test set. Each classifier will only be trained based on the training set. Likewise, the test set is also fixed in the beginning and used for all test purposes. The dataset is split into 80% training and 20% test.

On the basis of the features, the classifier estimates the anomaly on active railroad. Based on these features, four state-of-the art classifiers, k-Nearest Neighbors (kNN) [25], Support Vector Machine (SVM) [26], Decision Tree (DT) [27] and Random Forest (RF) [28], are compared and we evaluate performance in terms of precision, recall and F1-score. The task is solved using a SVM with a quadratic degree of the polynomial kernel function. For kNN classifier has been chosen minkowski as metric distance and n neighbors = 5. As Table I shows, the performance of all classifiers is good, with a F1-Scores ranging that exceed the 70% in all the cases analysed, thus demonstrating the effectiveness and the suitability of the proposed approach.

As Table II shows, the performance of the DCNN is good with values of F1-score that reached the 70%, thus demonstrating the effectiveness and the suitability of the proposed approach, even if the smaller amount of available training data and the higher variation in motives. The small number of anomaly photos is due to the fact that railroad crossing is a critical railway infrastructure anomaly/attacks and it had to be carefully controlled in order to preserve safety on active railroad crossings.

[1]http://caffe.berkeleyvision.org/

TABLE I: Performance evaluation of the Profiles.

Classifier	Precision	Recall	F1-Score
DT	0.72	0.72	0.72
RF	0.74	0.74	0.74
SVM	0.77	0.77	0.77
kNN	0.78	0.78	0.78

TABLE II: Performance evaluation of the DCNN.

Category	Precision	Recall	F1-Score
normal	0.78	0.72	0.72
tree-anomaly	0.66	0.69	0.68
trawlers anomaly	0.72	0.67	0.69
railway lines anomaly	0.71	0.67	0.69
MEAN()	***0.72***	***0.69***	***0.70***

V. CONCLUSIONS

Felix is the first mobile robot that can automatically inspect railway points, crossings and tracks. It has been developed with the objective to increase railway switches reliability, guarantee railways safety and solve maintenance planning problems. Felix robot is also able to monitor S&C status and it is equipped with profilometers to create a 3D reconstruction of the inspected part. In this paper, we introduce an approach for a proper change detection, in order to improve railway safety and security. The data structure in output can be visualised on a web platform making faster and easier to perform diagnosis and quality control on track degradation, with a significant impact on the safety of operators and ultimately of final users. The main goal is to increase points reliability, guarantee rail safety and solve maintenance scheduling problems. This approach is important to reduce train accidents, as well as human and material losses suffered in these accidents, and to ensure security in train transportation. A classification process is carried out with the aim to automatically detect anomaly in railway.

Future works are the the collection of more rail wear data and the development of quality control testing systems

to customers specifications. The collected change detection information aim to lead the way for augmented reality applications as well as tools for the security risk assessment and management specifically tailored to the context of railway transportation systems.

Further investigation will be devoted to improve our approach by employing a larger dataset for anomaly detection and extracting additional informative features. Furthermore, as we plan to put Felix robot on board of the train tractor, it will become a sort of mobile mapping system to extract and recognise surrounding objects [29]. In this context, a vision based approach similar to [30] could be used to solve localization problems in the aliased environment (i.e., an environment where several place look the same) constituted by the sequence of sleepers and trawlers.

ACKNOWLEDGMENT

This work was supported by Ministry of Economic Development, FCS – Fondo per la crescita sostenibile Decree 20 June 2013. Project n.129

REFERENCES

[1] C. Esveld, *Modern railway track*. MRT-productions Zaltbommel, The Netherlands, 2001, vol. 385.

[2] A. Johansson, B. Pålsson, M. Ekh, J. C. Nielsen, M. K. Ander, J. Brouzoulis, and E. Kassa, "Simulation of wheel–rail contact and damage in switches & crossings," *Wear*, vol. 271, no. 1, pp. 472–481, 2011.

[3] X. Tan and B. Ai, "The issues of cloud computing security in high-speed railway," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, vol. 8. IEEE, 2011, pp. 4358–4363.

[4] P. Zingaretti and E. Frontoni, "Vision and sonar sensor fusion for mobile robot localization in aliased environments," in *Mechatronic and Embedded Systems and Applications, Proceedings of the 2nd IEEE/ASME International Conference on*. IEEE, 2006, pp. 1–6.

[5] C. Campos-Castellanos, Y. Gharaibeh, P. Mudge, and V. Kappatos, "The application of long range ultrasonic testing (lrut) for examination of hard to access areas on railway tracks," in *Railway Condition Monitoring and Non-Destructive Testing (RCM 2011), 5th IET Conference on*. IET, 2011, pp. 1–7.

[6] W. Al-Nuaimy, A. Eriksen, and J. Gasgoyne, "Train-mounted gpr for high-speed rail trackbed inspection," in *Ground Penetrating Radar, 2004. GPR 2004. Proceedings of the Tenth International Conference on*. IEEE, 2004, pp. 631–634.

[7] Z. S. Daliri, S. Shamshirb, and M. A. Besheli, "Railway security through the use of wireless sensor networks based on fuzzy logic," *International Journal of Physical Sciences*, vol. 6, no. 3, pp. 448–458, 2011.

[8] M. Singh, S. Singh, J. Jaiswal, and J. Hempshall, "Autonomous rail track inspection using vision based system," in *Computational Intelligence for Homeland Security and Personal Safety, Proceedings of the 2006 IEEE International Conference on*. IEEE, 2006, pp. 56–59.

[9] H. Setan and R. Singh, "Deformation analysis of a geodetic monitoring network," *GEOMATICA-OTTAWA-*, vol. 55, no. 3, pp. 333–346, 2001.

[10] D. Calvaresi, P. Sernani, M. Marinoni, A. Claudi, A. Balsini, A. F. Dragoni, and G. Buttazzo, "A framework based on real-time os and multi-agents for intelligent autonomous robot competitions," in *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*. IEEE, 2016, pp. 1–10.

[11] R. Lindenbergh and P. Pietrzyk, "Change detection and deformation analysis using static and mobile laser scanning," *Applied Geomatics*, vol. 7, no. 2, pp. 65–74, 2015.

[12] D. Lu, P. Mausel, E. Brondizio, and E. Moran, "Change detection techniques," *International journal of remote sensing*, vol. 25, no. 12, pp. 2365–2401, 2004.

[13] P. Agouris, A. Stefanidis, and S. Gyftakis, "Differential snakes for change detection in road segments," *Photogrammetric Engineering and Remote Sensing*, vol. 67, no. 12, pp. 1391–1400, 2001.

[14] X. Tong, Z. Hong, S. Liu, P. Chen, Y. Jin, H. Xie, S. Liu, S. Zhang, and T. Feng, "Detection of geometric change in railway curves caused by earthquakes, using high-resolution stereo satellite images," *Natural Hazards*, vol. 79, no. 1, pp. 409–436, 2015.

[15] J. Gong, Y. Yue, J. Zhu, Y. Wen, Y. Li, J. Zhou, D. Wang, and C. Yu, "Impacts of the wenchuan earthquake on the chaping river upstream channel change," *International journal of remote sensing*, vol. 33, no. 12, pp. 3907–3929, 2012.

[16] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. part 3, p. W19, 2005.

[17] T. B. Barnhart and B. T. Crosby, "Comparing two methods of surface change detection on an evolving thermokarst using high-temporal-frequency terrestrial laser scanning, selawik river, alaska," *Remote Sensing*, vol. 5, no. 6, pp. 2813–2837, 2013.

[18] M. Tsakiri, D. Lichti, and N. Pfeifer, "Terrestrial laser scanning for deformation monitoring," 2006.

[19] O. Monserrat and M. Crosetto, "Deformation measurement using terrestrial laser scanning data and least squares 3d surface matching," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 63, no. 1, pp. 142–154, 2008.

[20] W. Mukupa, G. W. Roberts, C. M. Hancock, and K. Al-Manasir, "A review of the use of terrestrial laser scanning application for change detection and deformation monitoring of structures," *Survey Review*, vol. 49, no. 353, pp. 99–116, 2017.

[21] J.-H. Wu, C.-C. Pen, and J.-A. Jiang, "Applications of the integrated high-performance cmos image sensor to range findersfrom optical triangulation to the automotive field," *Sensors*, vol. 8, no. 3, pp. 1719–1739, 2008.

[22] M. Schuetz, "Potree: Rendering large point clouds in web browsers," Ph.D. dissertation, Technische Universität Wien, 2016.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] M. Paolanti, C. Kaiser, R. Schallner, E. Frontoni, and P. Zingaretti, "Visual and textual sentiment analysis of brand-related social media pictures using deep convolutional neural networks," in *Image Analysis and Processing, 2017. ICIAP 2017. 19th International Conference on*.

[25] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.

[26] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[27] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[28] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[29] A. Mancini, E. Frontoni, and P. Zingaretti, "Road change detection from multi-spectral aerial data," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 448–451.

[30] E. Frontoni and P. Zingaretti, "A vision based algorithm for active robot localization," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*. IEEE, 2005, pp. 347–352.