ULL | Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Departamento de Ingeniería Informática y de Sistemas

Exploration, Navigation and Localization for Mobile Robots

Under the guidance of:
Leopoldo Acosta Sánchez
Jonay T. Toledo Carrillo

Jose Daniel Perea Ström

2017

Firmado por: JOSE DANIEL PEREA STRÖM
UNIVERSIDAD DE LA LAGUNA
Fecha: 30/06/2017 00:15:39

JONAY TOMAS TOLEDO CARRILLO
UNIVERSIDAD DE LA LAGUNA
30/06/2017 02:34:46

LEOPOLDO ACOSTA SANCHEZ
UNIVERSIDAD DE LA LAGUNA
30/06/2017 08:37:26

ERNESTO PEREDA DE PABLO
UNIVERSIDAD DE LA LAGUNA
06/07/2017 13:51:10

D. Leopoldo Acosta Sánchez, Doctor en Física y Catedrático del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna.

D. Jonay Tomás Toledo Carrillo, Doctor en Informática y Profesor Titular del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna.

CERTIFICAN:

que D. Jose Daniel Perea Ström, Ingeniero en Informática, ha realizado bajo nuestra dirección la presente Tesis, titulada "Exploration, Navigation and Localization for Mobile Robots" (Exploración, Navegación y Localización para Robots Móviles), para optar al grado de Doctor por la Universidad de La Laguna.

Con esta fecha, autorizo la presentación de la misma.

En La Laguna, a 29 de junio de 2017.

Los Directores,

Leopoldo Acosta Sánchez

Jonay Tomás Toledo Carrillo

....

A mis padres, José y Christina,
y a mi hermana, Helena.

Abstract

The main goal of this thesis is the advancement of the state of the art in mobile robot autonomy. In order to achieve this objective, several contributions have been presented that tackle well defined problems in the areas of localization, navigation and exploration.

The very first contribution is focused on the task of robustly finding the localization of a mobile robot in an outdoor environment. Specifically, the presented technique introduces a key methodology to perform sensor fusion of a global localization sensor so ubiquitous as a GPS device, within the context of a particle filter based Monte Carlo localization system. We focus on the management of multiple sensor data sources under noisy and conflicting readings. This strategy allows for a reduced uncertainty in the robot pose estimation, as well as improved robustness of the system.

The second contribution presents a completely integrated navigation system running within a constrained and highly dynamic platform like a quadrotor, applied to full 3D environments. The navigation stack comprises a Simultaneous Localization and Mapping (SLAM) system for RGB-D cameras that provides both the robot pose and an obstacle map of the environment, as well as a 4D path planner capable of finding obstacle free and kinematically feasible trajectories for the quadrotor to navigate this environment.

The third contribution introduces a novel approach for autonomous exploration of unknown environments with robust homing. We present a technique to predict possible environment structures in the unseen parts of the robot's surroundings based on previously explored environments. We exploit this belief to predict possible loop closures that the robot may experience when exploring an unknown part of the scene. This allows the robot to actively reduce the uncertainty in its belief through its exploration actions. Also, we introduce a robust homing system that addresses the problem of returning a robot operating in an unknown environment to its starting position even if the underlying SLAM system fails.

All contributions where designed, implemented and tested on real autonomous robots: a self-driving car, a micro aerial vehicle and an underground exploration platform.

Contents

ix

Contents

Chapter 3. Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry   29

Chapter 4. Robust Exploration and Homing for Autonomous Robots   41

x

III. Conclusions and Future Work   73

Chapter 5. Conclusions   75

IV. Appendix   81

Appendix A. MCL with sensor fusion based on a weighting mechanism vs. particle generation   83

Appendix B. Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry   91

Appendix C. Predictive Exploration Considering Previously Mapped Environments   99

Appendix D. Robust Exploration and Homing for Autonomous Robots 107

Appendix E. Safe and Reliable Path Planning for the Autonomous Vehicle Verdino   121

Appendix F. Stability and performance analysis of unmanned aerial vehicles: Quad vs. Hex   133

Appendix G. VERDINO, prototipo eléctrico de vehículo autoguiado   141

xi

Contents

Bibliography

xii

List of Figures

xiii

List of Figures

xiv

Acronyms

ADAS Advanced Driver Assistance Systems

DGPS Differential Global Positioning System

DOF Degrees of Freedom

EKF Extended Kalman Filter

GPS Global Positioning System

IMU Inertial Measurement Unit

LIDAR Light-Detection And Ranging

MAV Micro Aerial Vehicle

MCL Monte Carlo Localization

AMCL Adaptive Monte Carlo Localization

PF Particle Filter

PDF Probability Density Function

RGB-D Red Green Blue and Depth

ROS Robot Operating System

SIR Sample Importance Resampling

SLAM Simultaneous Localization and Mapping

SURF Speeded Up Robust Features

UAV Unmanned Aerial Vehicle

UKF Unscented Kalman Filter

XV

Part I.

Introduction

1

Firmado por: JOSE DANIEL PEREA STRÖM
UNIVERSIDAD DE LA LAGUNA
Fecha: 30/06/2017 00:15:39

JONAY TOMAS TOLEDO CARRILLO
UNIVERSIDAD DE LA LAGUNA
30/06/2017 02:34:46

LEOPOLDO ACOSTA SANCHEZ
UNIVERSIDAD DE LA LAGUNA
30/06/2017 08:37:26

ERNESTO PEREDA DE PABLO
UNIVERSIDAD DE LA LAGUNA
06/07/2017 13:51:10

Chapter

1

Exploration, Navigation and Localization for Mobile Robots: the road to full autonomy

Achieving the highest level of autonomy is one of the most important problems in the field of robotics. Giving this capability to a robotic system opens up a large spectrum of applications like transportation, rescue, cleaning, logistics, and many other service robotics scenarios. In the context of mobile robotics, full autonomy implies solving the problem of mobility autonomy. For this reason, making a mobile robot capable of perceiving the environment, extracting knowledge about it, and finally navigate it have been major research topics in the robotics community for the last decades.

At a high level of abstraction, autonomy for a mobile robot implies the solution of three major tasks, which are mapping, localization, and navigation (Makarenko et al., 2002). Mapping implies the capability to perceive, accumulate and integrate the information available to the robot's sensors into a model of the environment. This model of the world is what the robot uses as the foundation for the remaining reasoning processes in a mobility task. The mapping task has a strong dependency on the interpretation of sensory data, as well as the representation model chosen to integrate this data in a map. Localization, on the other side, implies the capability to estimate the position and orientation of the robot relative to a model of the environment. How this task is solved is also highly dependent on the interpretation of sensory data, but also on the available information in the environment model that the map contains. Finally, navigation implies

3

Figure 1.1. Visualization of the three major tasks that a robot has to perform in order to achieve autonomous mobility. The overlapping areas represent combinations of the mapping, localization, and navigation tasks (adapted from Makarenko et al. (2002)).

answering the question of how the robot should move through the environment given, the output of the other tasks, so the end goal is reached.

As the reader may guess, these three tasks cannot be solved separate of each other. A robot needs to know where obstacles are before deciding the best trajectory to a specific destination. In order to properly model the obstacles that the robot encounter in the environment, it needs to know its location in relation to these obstacles. Deciding the best trajectory in a map is not very effective without knowing where the robot is in relation to the goal and the obstacles. In resume, each of these tasks are coupled to each other in some degree.

In Figure 1.1 a Venn diagram shows how each of these tasks overlap, and how each overlapping area defines a combined problem effectively becoming a research topic in the mobile robotics field. Solving the combined problem of mapping and localization at the same time is commonly known as the SLAM problem (Simultaneous Localization and Mapping) in the bibliography. These two tasks have a high degree of interdependency, as having a good map enables a robot to properly localize itself, and a good

4

localization is needed to reconstruct a good model of the environment. Active Localization implies guiding the robot through the environment, reaching vantage points that improve the estimation of its own position and orientation on the map. In the case of Exploration, the focus of the trajectory planning is in the acquisition of new knowledge about the environment, assuming the robot pose estimation is accurate. The center region represents the integrated approaches, where all three tasks are executed simultaneously. These approaches allows a mobile robot to acquire sensor data while moving through the environment on its own and being conscious about its localization confidence. Each path planning decision balances the acquisition of new environment knowledge, as well as improving the pose estimate in the map. Finally, an accurate model of the environment is obtained while knowing the relative pose of the robot to this environment.

In the case where the environment changes over time, it is worth mentioning that the previously introduced tasks become more difficult. The majority of techniques developed for these tasks assume a static environment that does not change over time. This assumption is unrealistic in a large set of real-world applications, as mobile robots are expected to be used in environments populated by humans. People moving in the surroundings of the robot, cars passing by, parking lots with changing levels of occupation, etc. introduce complexities in the practical implementation of mapping, localization, navigation and exploration algorithms. Moreover, even in the static scenario, the robot sensors that measure the environment and its own state in the world are subject to noisy readings, bias, false positives, and even faulty modes of operation. Real-world implementations require counter measurements and proper architectural designs to make them reliable, robust and fault tolerant.

In short, the following problems need to be tackled in the implementation of mobile robotic systems in order to achieve full autonomy:

- Interpretation of noisy proprioceptive (the robot itself) and exteroceptive (the robot environment) sensor readings.

- Management of multiple (and possibly conflicting) sensory readings.

- Estimation of the environment obstacles and robot pose under uncertainty.

5

- Selection of the best robot trajectories to safely navigate the environment.

- Selection of the best robot trajectories to effectively acquire new knowledge about the environment.

- Introduction of robustness and fault-tolerance, as well as testing under real-world conditions.

The contributions of this thesis are actual implementations of solutions for the localization, navigation and explorations tasks, specifically addressing these six problems.

We introduce techniques to properly take into account the noisy nature of internal and external sensors in robotic platforms. We focus on the management of multiple sensor data sources under noisy and conflicting readings. This strategy allows for a reduced uncertainty in the robot pose estimation, as well as improved obstacle estimation in the environment.

In order to safely navigate the environment, we present techniques to select the best robot trajectory in a real-world full 3D environment, implemented on computationally-constrained platforms. The implementation is integrated with state-of-the-art depth sensors and volumetric mapping systems on a highly dynamic mobile robot.

Finally, in order to gather new knowledge about the environment, we present a fully integrated approach where the exploration, navigation and localization tasks are combined achieving robust pose estimation while efficiently completing the map information. Our approach provides a novel technique to predict loop closures in the map using state-of-the-art machine learning algorithms, drastically reducing the pose and map uncertainty while informing the best trajectories to follow while exploring the environment.

Additionally, all techniques were implemented in full-scale mobile robots, and robustness and fault-tolerance were introduced to overcome tests in real-world scenarios. The mobile robotic platforms comprise a variety of aerial, ground, and underground vehicles, navigating both indoor and outdoor environments, in both static and dynamic scenarios.

6

1.1. Publications

Parts of this thesis have been published in the following journal articles, conference, symposium, and workshop proceedings:

- D. Perea, I. Bogoslavskyi, and C. Stachniss. Robust exploration and homing for autonomous robots. *Robotics and Autonomous Systems*, 90:125 – 135, 2017. ISSN 0921-8890. doi: 10.1016/j.robot.2016.08.015. Special Issue on New Research Frontiers for Intelligent Autonomous Systems

- D. Perea, F. Nenci, and C. Stachniss. Predictive exploration considering previously mapped environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2761–2766, May 2015a. doi: 10.1109/ICRA.2015. 7139574

- R. G. Valenti, I. Dryanovski, C. Jaramillo, D. Perea, and J. Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5233–5238, May 2014. doi: 10.1109/ICRA.2014.6907628

- D. Perea, J. Hernández-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta. Mcl with sensor fusion based on a weighting mechanism versus a particle generation approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 166–171, Oct 2013. doi: 10.1109/ITSC.2013. 6728228

- R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta. Safe and reliable path planning for the autonomous vehicle verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32, Summer 2016. ISSN 1939-1390. doi: 10.1109/MITS.2015. 2504393

- J. Toledo, L. Acosta, D. Perea, and N. Morales. Stability and performance analysis of unmanned aerial vehicles: Quadrotor against hexrotor. *IET Control Theory*

7

Applications, 9(8):1190–1196, 2015. ISSN 1751-8644. doi: 10.1049/iet-cta.2014. 1032

- D. Perea, J. Toledo, A. Morell, R. Arnay, L. Acosta, N.Morales, A. Hamilton, and J. Hernandez-Aceituno. Demonstration of the autonomous vehicle verdino (invited talk), 2015b. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2015)

- L. Acosta, J. Toledo, A. Hamilton, R. Arnay, J. Espelosin, N. Morales, D. Perea, and L. Moreno. Verdino, prototipo eléctrico de vehículo autoguiado, Sep 2012. XXXIII Jornadas Nacionales de Automática

- T. Cerdena, Y. Callero, D. Perea, P. Betancor, D. Lutzardo, J. Toledo, and L. Acosta. Design and development of a fully autonomous decimeter-scale humanoid robot. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, April 2009. doi: 10.1109/ICMECH.2009.4957211

1.2. Collaborations

Parts of this thesis have been done in collaboration with other people, without whose contributions the obtained results would not have been possible.

The approach to fuse GPS sensor readings within a MCL localization system was developed with the huge collaboration of Antonio Morell, Jonay Toledo, Javier Hernández-Aceituno, Alberto Hamilton and Leopoldo Acosta, as well as the rest of the great team at the Robotics Group at Universidad de La Laguna (Spain).

The integrated navigation system comprised of a 4D path planner on top of a RGB-D SLAM system for aerial vehicles was developed with the huge collaboration of Roberto Valenti, Ivan Dryanovski, Carlos Jaramillo and Jizhong Xiao from the great team at the Robotics Laboratory at The City College of New York (USA), after a 3 month research stay in their lab, as well as ongoing collaborations afterwards.

The autonomous exploration system comprised of a loop closure predictive technique with robust homing was developed with the huge collaboration of Fabrizio Nenci,

8

Igor Bogoslavskyi and Cyrill Stachniss from the great team at the Photogrammetry & Robotics Lab of the University of Bonn (Germany), as well as from the Autonomous Intelligent Systems Lab at the University of Freiburg (Germany), after a 3 month research stay in their lab, as well as ongoing collaborations afterwards.

The team effort of all these people make it possible to reach the stars on the shoulders of giants.

9

Part II.

Results and Discussion

11

MCL with sensor fusion based on a weighting mechanism vs. particle generation

The combined action of several sensing systems, so that they are able to compensate the technical flaws of each other, is common in robotics. Monte Carlo Localization (MCL) is a popular technique used to estimate the pose of a mobile robot, which allows the fusion of heterogeneous sensor data. Several sensor fusion schemes have been proposed which include sensors like GPS to improve the performance of this algorithm. In this chapter, an Adaptive MCL algorithm is presented to combine data from wheel odometry, an inertial measurement unit, a global positioning system and laser scanning. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a particle generation approach. Experiments were conducted on a real robotic car within an urban environment.

2.1. Introduction

Localization is one of the most relevant problems in mobile robotics, specially in outdoor and urban areas. The information obtained from sensor devices might not be as accurate as expected, so it is of great importance to define algorithms that are robust to such problems. Specifically using a known map of static obstacles, dynamic obstacles are also very likely to appear and might add uncertainty to localization algorithms. Using

13

a single source of sensing is not practical, thus many different forms of sensor fusion have been proposed. Multiple devices can mitigate the drawbacks a single sensor might have, especially by combining proprioceptive and exteroceptive measurements, such as odometry and global positioning systems (GPS).

A review of the literature shows that Laser Imaging Detection and Ranging (LIDAR) sensors are very popular because of their data update frequency and precision. They have been combined with wheel odometry, visual cameras (Newman et al., 2006), stereovision, GPS (Wei et al., 2011a), three–dimensional geographic information systems (3D–GIS) (Peng et al., 2009), and combinations thereof (Cappelle et al., 2007; Smaili et al., 2011; Wei et al., 2011b). A common localization algorithm which integrates the information provided by different sensors is the Monte Carlo Localization (MCL) method (Fox et al., 1999). It is based on particle filters (PF), whose samples (or particles) are weighted according to their likelihood computed from each available device (Silver and Stentz, 2011). Fusion of wheel odometry and GPS using MCL has been studied before (Moreira et al., 2007), including omnidirectional vision (Frontoni et al., 2010), LIDAR sensors (Chang et al., 2006a,b), and inertial sensors (Hentschel et al., 2008). Several adaptive variations of MCL with a variable number on PF samples have been proposed, such as Bayesian Bootstrap Filtering (Khalid et al., 2007), Self–Adaptive MCL (SAMCL) (Zhang et al., 2009), or Merge–MCL (Li et al., 2010). Adaptive Monte Carlo Localization (AMCL) (Fox, 2003; Thrun et al., 2005) optimaly adapts the number of samples of the PF by means of Kullback–Leibler divergences (KLD) (Kullback and Leibler, 1951).

This contribution showcases how GPS sensor fusion methods are affected by *multipath interferences*, a phenomenon that leads to misplaced reports of GPS sensors. We propose a fusion of wheel odometry, an Inertial Measurement Unit (IMU), GPS and LIDAR using the AMCL algorithm. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a sample generation approach. On the traditional strategy used in the sample generation method (Hentschel et al., 2008), new particles are added to the PF when new absolute measurements are obtained, whereas our proposal uses this information as a weighting function over the existing particles. Experiments were conducted on both approaches, and the results reflect the robustness

14

and better performance of our implementation.

This chapter is organized as follows. Section 2.2 briefly describes the AMCL algorithm which solves the localization problem. Section 2.3 presents and discusses the proposed GPS integration method. The particle generation approach is described in Sections 2.4, and the robustness of our proposal is discussed and compared with the former. The mobile robot platform and the experiments conducted are described in Section 2.5. Finally, the most relevant conclusions are summarized in Section 2.6.

2.2. Localization with Particle Filters

The state–space of the localization problem on 2–D maps is given by the position of the robot, as a pair of Cartesian coordinates (x, y), and its orientation angle (θ). Thus, let's define this pose as the three–dimensional state vector $x_t = (x_t, y_t, \theta_t)$, where x_t, y_t and θ_t are the position coordinates and the orientation of the pose with respect to a reference frame, respectively. Also, given the capacity of a robot to sense the environment, let's define the data gathered from the sensors as another vector z_t, for all data corresponding to a specific time t. We assume that data from all sensors are available at the same time.

Future states for a mobile robot can be generated stochastically from the prior state and current knowledge, obtained from sensors and the actions performed. The Bayes filter is a recursive technique where the state x_t is calculated from the previously computed state, x_{t-1}. These algorithms are built on top of the Markov assumption: the stochastic evolution of future states from x_t only depends on the conditions and variables at time t. A temporal process that meets this condition is know as a *Markov chain*.

Particle Filter (PF) methods solve state estimation problems as an approximation of the Bayes filter. Efficient implementations have affordable computational complexity and accuracy of the approximations to complex nonlinear problems, like the global localization. The sequential implementation of this problem is called Monte Carlo Localization (MCL), a method that solves the Bayes filtering problem by means of importance sampling, and one of the most popular approaches is the *Sampling Importance Resampling* (SIR) algorithm (Rubin, 1988; Smith and Gelfand, 1992). The SIR algorithm is at the heart of the Adaptive Monte Carlo Localization algorithm, used as a base approach

15

for our contribution.

The Adaptive Monte Carlo Localization (AMCL) algorithm (Fox, 2003) finds an estimation of the posterior $p(x_t \mid z_t)$ for a mobile robot at time t, based on the observations of its sensors (typically LIDAR scans or visual features), merged with wheel odometry as the robot moves. A set of particles or samples represents the posterior about the trajectory of the robot, which are updated following the SIR algorithm, as follows:

1. *Sampling:* A new generation of particles $\{x_t^{(i)}\}$ is obtained from the previous generation $\{x_{t-1}^{(i)}\}$ by sampling from a proposal distribution π. A probabilistic odometry motion model $p(x_t \mid x_{t-1}, u_{t-1})$ is used as such proposal distribution, where u_{t-1} is the odometry measurement at time $t-1$.

2. *Importance weighting:* The importance weight of a pose x_t is a dimensionless value related to how likely is that the robot is located at x_t. LIDARs measurements are used to compute the importance weight of each particle as follows:

$$w_t^{(i)} = p(z_t \mid x_t^{(i)}) \tag{2.1}$$

All weights satisfy:

$$\sum_{i=1}^{N} w_t^{(i)} = 1 \quad , \tag{2.2}$$

on any time t, where N is the total number of samples on the PF.

3. *Resampling:* Particles with a low importance weight are more likely to be replaced by those with a higher weight. This step allows maintaining a discrete set of particles which approximate a continuous distribution with a bounded number of particles, since a high number of particles on the PF is not computationally efficient. The optimal number of particles is typically given by Kullback–Leibler divergences (Kullback and Leibler, 1951).

16

2.3. GPS Integration

Robot localization in outdoor environments usually takes advantage of GPS devices. While providing useful information, GPS measurements might be misleading or even completely erroneous in some circumstances. The most relevant sources of error in urban scenarios are *multipath phenomena*.

Multipath is the propagation phenomenon that results in radio signals, e.g., satellite signals, reaching the receiving antenna by two or more paths. This happens mostly due to reflection and refraction phenomena, i.e., from water bodies and terrestrial objects such as mountains and buildings, resulting in a sudden "jump" in the GPS position estimate. Such source of error should not be ignored, given its negative impact on GPS resulting readings.

We consider GPS measurements given by 2–D Cartesian coordinates (x, y) and their respective covariances, as a *Universal Transverse Mercator* (UTM) projection from the *World Geodetic System* (WGS84) ellipsoid. In addition, an approximation of the orientation (*yaw*) of the robot is given by the Course–Over–Ground (COG) as the orientation (θ) of the vector between consecutive GPS positions, assuming only longitudinal movement of the robot. The covariance of this orientation is also available. When a multipath event is detected, which yields a high covariance on the current reported position, the orientation angle and its covariance are taken from an IMU device. In spite of the fact that IMU orientation reports are not very accurate, (compared with the COG value provided by a differential rover GPS), they are still useful when these reports are not available or are not valid. For example, in situations where the robot has stopped or is moving very slowly, the difference between two consecutive positions does not yield a valid orientation. A multipath event also causes the GPS device to report erroneous COG measurements.

The key idea of the GPS integration approach is to weight the existing particles considering both the pose estimation and the associated covariances reported by a GPS device. This new observation source z_t^{GPS} provides the following position and orientation parameters at a time t:

$$z_t^{\text{GPS}} = \left\langle \begin{bmatrix} \boldsymbol{\mu} \\ \mu_\theta \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \sigma_\theta \end{bmatrix} \right\rangle \quad , \tag{2.3}$$

17

where $\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

Assuming that position (x, y) and orientation (θ) are uncorrelated and follow a Gaussian Probability Density Function (PDF), the posterior given a GPS reading can be obtained as

$$p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}) = f(x, y) \cdot f_{\text{WN}}(\theta) \quad, \tag{2.4}$$

where the PDF for the position is

$$f(x, y) = \frac{1}{2\pi \left| \boldsymbol{\Sigma} \right|^{1/2}} e^{-\frac{1}{2} \left(\begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix} \right)} \quad, \tag{2.5}$$

and the PDF corresponding to the orientation angle, which follows a wrapped normal distribution, is

$$f_{\text{WN}}(\theta) = \frac{1}{\sigma_\theta \sqrt{2\pi}} \sum_{k=-\infty}^{\infty} e^{-(\theta - \mu_\theta + 2\pi k)^2 / 2\sigma_\theta^2} \quad. \tag{2.6}$$

Instead of adding new particles to the PF, the existing set of particles are weighted according to Eq. ((2.4)) and LIDAR measurements z_t^{LIDAR}, which are conditionally independent of past measurements given knowledge of the state $\boldsymbol{x}_t$ (Fox, 2003). Therefore, the new posterior is:

$$p(\boldsymbol{x}_t \mid z_t) = p(\boldsymbol{x}_t \mid z_t^{\text{LIDAR}}) \cdot p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}). \tag{2.7}$$

In the first iteration of the algorithm there is no initial hypothesis available. Although our method does not generate GPS–based particles, an initial particle set is needed. Thus, a particle set is created and distributed following the first GPS measurement. If a *kidnapped robot* event takes place, new particles could be added similarly to overcome this problem.

18

2.4. GPS Particle Generation

When a GPS measurement is reported, it seems natural to add a new particle cluster to the PF (Hentschel et al., 2008). A new set of samples is drawn from the Gaussian PDF centered at the GPS position. The *m* particles with the lowest weight in the filter are replaced with the new sample set. Unfortunately, adding new particles to the PF introduces some flaws. Assuming a correct initial robot localization, if the GPS output greatly differs from the current hypothesis, it may imply that a multipath interference has happened. Misplaced GPS reports should be discarded by the PF in order to not accept misleading hypotheses, incompatible with the current robot location. These misleading hypotheses yield incorrect output in the PF in situations where not enough significant landmarks are available in LIDAR scans. This behavior can be seen in experiment 1 (Section 2.5.1).

Following the proposed GPS weighting scheme, the aforementioned problems will not arise, and there will not be conflicting and ambiguous hypotheses which will eventually lead to a wrong robot localization.

During a multipath event, where a GPS measurement usually drifts from the actual robot position, the GPS sensor covariance values might not be properly delivered. This means that the mean value for the GPS reading is incorrect, usually much further away than one standard deviation, for each of the dimensions reported in the covariance matrix. In this case, the tail of the gaussian distribution is the part that is contributing to the particle weighting, instead of the area within one standard deviation. In our implementation, the way the particles are weighted with the GPS readings produce a much more robust behavior in practical terms. Here are two different examples scenarios that reflect this robustness:

- For multipath events where narrow covariances are incorrectly reported (the GPS sensor is erroneously over-confident about the reported reading confidence), particle weights will not be greatly affected, as the resulting gaussian model will have the gaussian tail very low compared to the area near the mean in the GPS reading.

- On the other hand, when the covariances are wide (the GPS sensor is correctly

19

reporting a reduced confidence about the position reading), it is interesting to note that the resulting GPS–centered gaussian model will contribute more noticeably to the weighting of the correctly localized particles than in the previous scenario. The reason for this is that the gaussian distribution tail is taller than before, as the standard deviation increases. This effect, as counter–intuitive as it may seem, is the proper way to mix two statistical distributions: the instantaneous GPS gaussian and the PF posterior that models the current pose probabilities taking into account previous sensor readings. This proper distribution handling produces direct benefits that will be showcased in the next section with real world experiments, where the contribution of the LIDAR and odometry sensors are included.

It is worth noting that these two scenarios are not handled separately by the PF in our implementation. The GPS weighting is performed with every report received from the device with their respective covariance values. The PF handles them naturally and transparently in either case, in conjunction with the odometry model and the LIDAR weighting scheme.

Finally, if the mean of the reported GPS position is at the current localization hypothesis, there would not be a meaningful difference in the obtained hypothesis between the particle generation method and our implementation, because the current particle distribution would already include that GPS position reading.

2.5. Experiments

The experiments has been conducted on a test platform called VERDINO (Figure 2.1), a fully electric two seat vehicle, based on an *EZ–GO TXT–2* golf cart. It is designed for passenger transportation and surveillance in non–structured environments. The vehicle has been modified by adding several sensors and actuators, which allows performing navigation tasks through urban areas.

Its sensor system includes two differential GPS Javad Triumph–1 devices. The first one is a Rover GPS unit mounted on top of the vehicle, and the second one is a fixed Base station. With its position accurately defined, the Base is used for estimating the error

20

Figure 2.1. VERDINO prototype.

introduced by each satellite, in order to send the corresponding corrections to the moving Rover unit. In addition, an IMU device aids during the estimation of the orientation of the vehicle, together with the Course–Over–Ground (COG) reporting capability of the GPS device. Finally, the robot includes two horizontal Sick LMS111 laser range finders, with a maximum range of 20 meters, and a wheel odometry system. It should be noted that our odometry sensor clearly suffers from a left drift during all experiments, noticeable in experiment 3. However, our method correctly handles these flawed reports.

The experiments were performed at the parking lot of the Computer Science Faculty of our campus, where VERDINO followed the path shown in Figure 2.2, which we consider as our ground truth. This path was recorded under continuous and accurate GPS readings, with a reported position covariance under $0.02\,m$. It was inspected to guarantee that no multipath events occurred. The route was traced on the ground, and the vehicle

21

Figure 2.2. Static localization map, with ground truth (blue) and experiment regions (red) identified. The stating position is marked as a green dot. The map covers an area of approximately $90\,m$ by $90\,m$.

was manually driven along it during the experiments. The error caused by manually driving along the path this way is about $0.3\,m$ in the worst case scenario, thus ensuring that the experimental errors are way below a typical GPS measurement error caused by a multipath event. The map used for Monte Carlo localization is a previously captured model of the static obstacles in the environment, which was georeferenced against the local vector topographic map from the Spatial Data Infrastructure of Canary Islands (IDECanarias).

22

Two different experiments where multipath was present were conducted in order to compare the reliability of the localization. In order to verify the GPS contribution, a final experiment compares our approach with an AMCL implementation without GPS integration. The allowed maximum number of particles was 2000 for the GPS particle generation implementation, and 500 for the GPS weighting approach. The maximum localization error obtained with GPS particle generation was 3.72 meters away from the ground truth, whereas the error yielded by our approach was 0.22 meters. All experiments are described in detail in the following subsections.

2.5.1. Multipath with no static references

In the first experiment, the robot traverses a region of the map where no static references are within the range of the LIDAR system, and therefore the position estimation must rely exclusively on wheel odometry and GPS. The GPS sensor suffers from multipath interference and reports a misplaced absolute position. Figure 2.3 compares the performance of both approaches under these circumstances. Figure 2.3a shows the localization output of the PF with GPS particle generation, while Figure 2.3b shows the localization output with GPS weighting only, as we propose. The dispersion of the particle set in the first case is higher, as the GPS particle generation continuously introduces new particles based on the sensor covariance, while the particle set is densely concentrated in the second case.

The experiment starts with the PF yielding a correct trajectory estimate on both cases. In Figure 2.3a, when the multipath interference appears on the GPS, a new particle cluster is generated some meters away from the correct trajectory. LIDAR scans have no features to match against the map, while more incorrect GPS readings are received. As the PF performance decreases, more particles are generated. The PF evolves and switches its current hypothesis to the new cluster, producing a displaced trajectory. When multipath disappears, the PF switches again to the correct path. This incorrect behaviour is present in other implementations of GPS integration on MCL algorithms where new particles are added (Hentschel et al., 2008). As can be seen on Figure 2.3b, the output is correct in our implementation, in spite of suffering from multipath interference. By not

23

(a) GPS particle generation　　　　　**(b)** GPS weighting

Figure 2.3. Experiment 1 – The dispersion is higher on (a), since the number of particles is also high (2000) and they are generated based on GPS measurements, while the robot traverses an area without enough spatial references, i.e., LIDAR readings do not report obstacles nearby. When applying the weighting approach (b), a maximum of 500 particles manage to keep the robot localized, even while the GPS suffers from a multipath event. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; yellow dotted line: ground truth.

adding new particles, only existing ones are considered and the GPS weighting is very low, given the distance to the displaced readings.

2.5.2. Valid LIDAR measurements with multipath

During the second experiment (see Figure 2.4), the robot traverses an area where obstacles are visible, thus LIDAR scans can be matched with the map. Here, multipath interference also appears, but this time the output is different from Figure 2.3. In Fi-

24

(a) GPS particle generation   **(b)** GPS weighting

Figure 2.4. Experiment 2 – Particles are once again widely spread on the particle generation approach (a) due to the higher number of particles at the GPS position estimate. Unlike experiment 1, the hypothesis does not jump after the multipath event, thanks to LIDAR readings matching a corner successfully. On (b), the particle set is narrower and also does not suffers from an undesired jump. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; green dots: LIDAR scans; yellow dotted line: ground truth.

gure 2.4a the newly created cluster does not affect the output of the PF, as the LIDAR scan matching has more influence than the GPS covariance weighting. Nonetheless, a great number of particles are always present on the wrong track during multipath, and each one is propagated with the odometry model and weighted according to each sensor

25

Figure 2.5. Experiment 3 – Comparison between an implementation of AMCL that does not integrate GPS measurements and our proposal. Light blue arrows: robot trajectory computed by our approach; green arrows: robot trajectory calculated by AMCL without GPS readings; red arrows: particle set distribution generated by our approach; purple arrows: particle set distribution generated without GPS; yellow dotted line: ground truth.

reading. This produces a waste of processing power, and it is a source of potential errors in scan matching that may mislead the PF hypothesis. In Figure 2.4b the output follows the trajectory closer, keeping a narrower particle set. The increased confidence in the hypothesis reduces the number of particles in the current set, freeing up CPU resources.

2.5.3. AMCL without GPS integration

A known, noticeable left drift in our odometry sensor is present in all of the experiments. When comparing our method with an implementation of AMCL without GPS integration, this flaw is corrected in both cases thanks to the observable LIDAR landmarks, as can be seen on the left half of Figure 2.5. However, when these LIDAR references are not available, the output of the AMCL algorithm without GPS weighting drifts away. Due to the lack of information, the dispersion of the particle distribution grows over time. Meanwhile, thanks to GPS integration, the PF hypothesis of our proposal follows the ground truth closely.

2.6. Conclusion

We have compared two different ways of combining GPS/IMU and LIDAR measurements within an AMCL based scheme which fuses data from multiple sensors. Experiments show that adding new particles to a PF on MCL algorithms are likely to create

26

clusters which might trigger a correctly localized hypothesis to change to a wrong location. Our approach uses the provided global location information to weight the existing particles, combined with the corresponding likelihood obtained after the weighting process using LIDARs information.

The maximum localization error yielded by the GPS particle generation method is below the assumed ground truth reproducibility margin. Our experiments show that, although more intuitive, adding new particles ignores the evolution over time of the calculated robot position. Therefore, its reliability is worsen whenever the GPS measurements are not accurate enough, e.g., when multipath interference occurs. Our approach combines all sensor readings successfully, avoiding large hypothesis changes.

Furthermore, by integrating GPS information with our method, the robustness of the hypothesis estimation increases against LIDAR measurements uncertainties and odometry unmodeled behaviors, such as drift. At the same time, the number of particles needed for a correct localization is lower, as the dispersion of the particle set is reduced.

27

Firmado por: JOSE DANIEL PEREA STRÖM
UNIVERSIDAD DE LA LAGUNA
Fecha: 30/06/2017 00:15:39

JONAY TOMAS TOLEDO CARRILLO
UNIVERSIDAD DE LA LAGUNA
30/06/2017 02:34:46

LEOPOLDO ACOSTA SANCHEZ
UNIVERSIDAD DE LA LAGUNA
30/06/2017 08:37:26

ERNESTO PEREDA DE PABLO
UNIVERSIDAD DE LA LAGUNA
06/07/2017 13:51:10

Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry

In this chapter we present a navigation system for Micro Aerial Vehicles (MAV) based on information provided by a visual odometry algorithm processing data from an RGB-D camera. The visual odometry algorithm uses an uncertainty analysis of the depth information to align newly observed features against a global sparse model of previously detected 3D features. The visual odometry provides updates at roughly 30 Hz that is fused at 1 KHz with the inertial sensor data through a Kalman Filter. The high-rate pose estimation is used as feedback for the controller, enabling autonomous flight. We developed a 4DOF path planner and implemented a real-time 3D SLAM where all the system runs on board. The experimental results demonstrate the autonomous flight and 3D SLAM capabilities of integrated navigation system running on the quadrotor.

3.1. Introduction

Micro aerial vehicles such as quadrotors are popular platforms often used by researchers because of their agility, high maneuverability, simple mechanical design and compact size. Their applications range from surveillance, search and rescue, to 3D mapping and photography. In order to perform such tasks, they require a set of sensors suited to the particular use and context, and the capability of ensure stable and autonomous flight.

29

Global Positioning System (GPS) is one of the most common sensors used for outdoor flight. Such solution is not always reliable, particularly when the signal's reception or precision might be unacceptable, as in the case of dense or indoor environments. Furthermore, in tasks like exploration and autonomous navigation in cluttered environments, global position information of the MAV is not sufficient, so a perception of the surroundings is needed. When external motion capture systems can not be deployed, the vehicle needs to rely only on onboard sensors such as laser scanners and cameras. Laser scanners provide range information with high precision and odometry can be derived by scan matching technique. However, laser scanners are not an optimal solution because of their weight and cost, and frequently only single plane is available. Therefore, for more complex environment, the need of lightweight 3D perception sensors is fulfilled by cameras. For this reason, visual odometry algorithm has become very common for flying robots.

An RGB-D camera is a device which provides RGB (red, green, blue) color and depth information for each pixel of the image. Depth is retrieved through the conjunction of an infrared projector and an infrared receiver. Recently new RGB-D cameras such as the Microsoft Kinect and the Asus Xtion, have become popular in the robotics community due to their reduced size, low weight and affordable cost.

In this chapter we present our approach for autonomous quadrotor flight and navigation by means of pose data from an RGB-D visual odometry algorithm, which relies on a frame-to-model registration technique, maintaining a low computation complexity (without any GPU acceleration), while reducing considerably the drift error of a typical frame- to-frame approach. This navigation system implements a 4DOF path planner that provides collision free paths within the short response time requirements of the embedded platform. All systems are implemented and integrated on board the flying robot.

3.2. System Architecture

The platform we use for our experiments is an *Ascending Technologies* Pelican quadrotor, on which we mounted an Asus Xtion Pro Live (see Figure 3.1). The quadrotor is equipped with a 1.86 GHz Core2Duo processor with 4GB of RAM and a Flight Control Unit (FCU) board with 2 ARM7 microcontrollers, an Inertial Measurement Unit (IMU)

30

Figure 3.1. The CityFlyer MAV equipped with an Asus Xtion Pro Live RGB-D camera.

and a pressure sensor. The system architecture is described in detail in Dryanovski et al. (2013a). We use one of the two microcontrollers, the so-called High Level Processor (HLP), to run our custom firmware that handles sensor fusion and control, while the Low Level Processor (LLP) is responsible for attitude control, IMU data fusion and hardware communication. The powerful onboard computer is able to manage visual odometry, 3D SLAM and motion planning. The entire framework is distributed between a ground station, the onboard computer and the FCU. The ground station is only used for visualization and teleoperation. Our framework uses Robot Operating System (ROS) as middleware, allowing communication among all the different components of the software (implemented as nodelets, a ROS mechanism for zero-copy message transport). The HLP and the onboard computer communicate with each other through the serial interface, where the Flyer Interface sends ROS messages and services trasnlated into packets. Communication between the two ARM7 microcontrollers (HLP and LLP) of the FCU board is via an I2C bus.

3.3. Real-Time Visual SLAM

Our navigation system relies on a 3D visual keyframe-based SLAM system for RGB-D cameras from Dryanovski et al. (2013b). The algorithm runs in real time on board the quadrotor computer. The SLAM algorithm takes as input the pose of the quadrotor provided by the visual odometry and generates a sequence of RGB-D keyframes.

31

Each keyframe consists of a RGB and Depth image pair, together with the pose of the camera at that instant and a set of SURF features detected in the RGB image. A new keyframe is generated once the angular or linear distance traveled between the current pose and the pose of the latest keyframe exceeds a given threshold (for example, 0.3 meters or 20 degrees). Incoming keyframes are tested for associations against previous keyframes. An association between two keyframes occurs when they are observing the same scene. This is accomplished in three steps. First, for the incoming keyframe, a set of candidates is built from the set of previous keyframes. Candidates are keyframes whose poses are close enough to be associated with the new keyframe. A liberal pruning threshold is performed (for example, 5 meters and 90 degrees). Next, a descriptor matcher is trained from all the SURF (Speeded Up Robust Features) keypoints in the candidate frames. The descriptor matcher is based on a FLANN (Fast Library for Approximate Nearest Neighbors) search tree (Muja and Lowe, 2009). The tree is used to further limit the candidate keyframes, based on the number of nearest neighbors each feature in the incoming keyframe has in each of the candidate keyframes. Only the k top candidates are kept. For each of the remaining candidates, a robust RANSAC (RANdom SAmple Consensus) (Fischler and Bolles, 1981) random matching is performed on the SURF features. If the RANSAC algorithm finds enough geometric inliers, it is assumed that there is an association between the two keyframes. The association observation is the transformation which best aligns the inliers.

Once the associations are established, a pose graph is built whose nodes are keyframe poses and whose edges are association observations. For consecutive keyframes, the observation comes from the visual odometry. Additional associations are generated through the RANSAC matching described above. Using $g2o$ (Kümmerle et al., 2011), an optimization phase is performed to find the configuration of poses which minimizes the observation error across the whole graph. The procedure runs at a rate between 1 Hz and 2 Hz on board the quadrotor.

The keyframes are used to build a dense Octomap (Hornung et al., 2013), a 3D volumetric occupancy grid-map with explicit representation of free and unknown space (see Figure 3.2). This Octomap, together with the quadrotor pose relative to the map, are passed as inputs for the 4D path planner described in the following section.

32
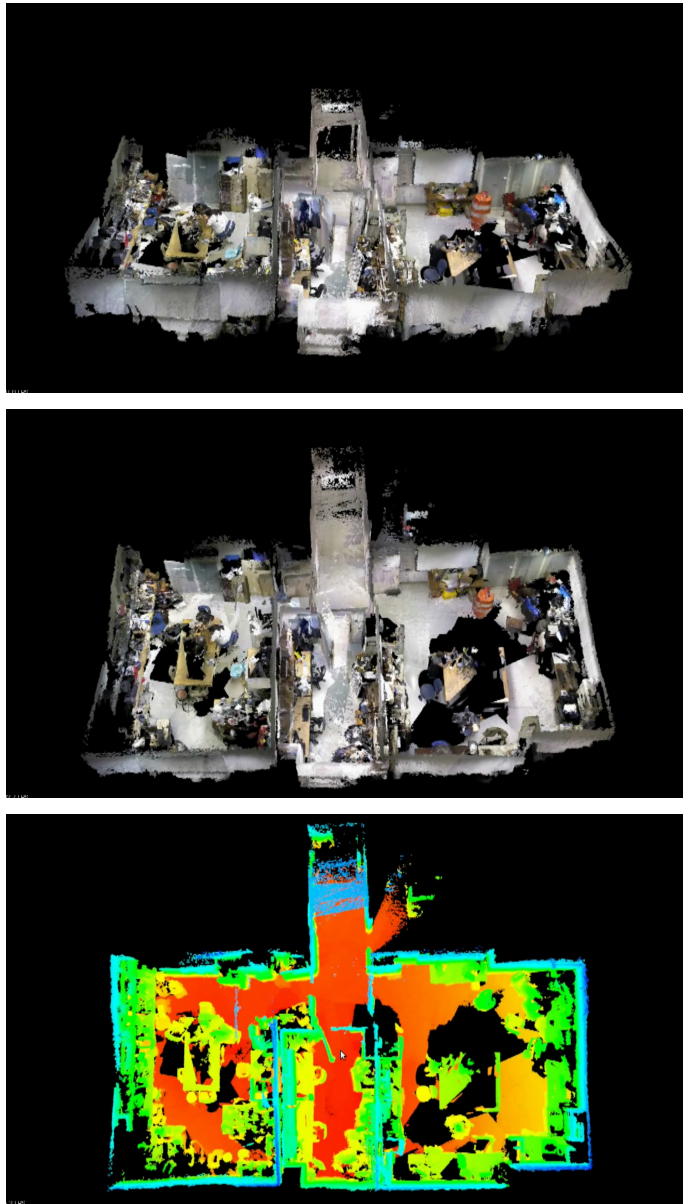
Figure 3.2. Results of the real-time onboard SLAM experiment. Top: Tilted view of the recovered point cloud map. Middle: top view of the point cloud map. Bottom: top view of the point cloud with heatmap colored altitude.

33

3.4. 4D Path Planning

In this section, the quadrotor path planner implementation is presented. This implementation is based on a search approach, where the state space is discretized using a state lattice of motion primitives (Pivtoraiko and Kelly, 2005) and an incremental and anytime version of the A* algorithm with Euclidean distance heuristic, in order to obtain feasible trajectories in practical time. This module consumes the real time ocupancy grid-map obtained by the SLAM system, an Octomap that models a 3D volumetric occupancy grid-map with explicit representation of free and unknown space . This Octomap, together with the quadrotor pose relative to the map, are passed as inputs for the 4D path planner. The system is executed in real-time in combination with the rest of the systems running on the platform, every time a new destination goal is requested to the navigation system.

3.4.1. State space discretization

The quadrotor state space is discretized following a state lattice, a graph search space that integrates motion planning constraints within state exploration. In this case, the state space is four-dimensional, combining the quadrotor position in Euclidean space (x, y, z) with the yaw orientation θ. State space exploration is executed following a set of motion primitives. Motion primitives are short, kinematically feasible path segments, that can be combined together to produce longer and more complex paths. Any combination of motion primitives yield a path that complies to the non-holonomic constraints imposed by the motion planning problem. Motion primitives are computed offline, and their traversal cost is multiplied by a user selected weight to obtain the motion cost. Weights are assigned to each motion primitive, in order to model preferences of one primitive over the others, e.g., penalizing changes in altitude while moving forward, in order to keep next positions centered within sensors field of view. Collision checking is performed online while exploring the search graph.

Planning results are greatly affected by motion primitives selection, in terms of planner times, planner completeness, and resulting path quality. The planner can not obtain

34

a feasible path if it cannot be produced by a combination of available motion primitives. For example, backwards paths cannot be generated if backwards motion primitives are not pre-computed and made available in the set. A richer set of motion primitives improves state space coverage adding flexibility to the planner, but there is a trade-off in computation time, as each new motion primitive increases the branching factor at each state.

3.4.2. Search algorithm

The described state lattice is explored using a graph search algorithm. This algorithm is a variant of the A* search extended with anytime and incremental capabilities called ARA* (Anytime Repairing A*) (Likhachev et al., 2004). ARA* anytime capability is obtained by executing a series of A* searches where the heuristic is inflated by a factor $\epsilon > 1$, and reducing this factor on each execution. With an inflated heuristic, A* search gives more relevance to the heuristic estimation. This results in a faster algorithm by means of losing optimality, but it has been shown that the computed path sub-optimality is bounded to ϵ times the cost of the optimal solution (Likhachev et al., 2004). ARA* starts with a high ϵ value in order to obtain a feasible path very fast. If time is available, ϵ is decreased and a search is executed again reusing computation from previous search. If enough time is available to reduce ϵ to 1, the heuristic is not inflated anymore, and the last search returns the optimal solution.

In this implementation, several restrictions influence the achievable set of motion primitives set available to the path planner. The High Level Controller in the MAV is a position controller, that receives a sequence of waypoints as input. These waypoints are obtained as the intersection of each motion primitve. In order to guarantee the stability and avoid overshooting along the sequence of waypoints, a zero-velocity condition is imposed at every waypoint, and any motion primitive involving a translation is generated as a straight path. This limitation reduces the path execution speed, but simplifies the path planner as every motion primitive is available at any given time regardless of the dynamics of the MAV. Also, the RGB-D sensor has a limited field of view, and the MAV should avoid flying into areas out of sight in order to avoid collisions. Thus, backwards

35

Figure 3.3. Motion primitives set used by the path planner, comprised of 3 forward motions (gaining altitude, keeping altitude, and loosing altitude) and 2 rotations (left turn, right turn)

motions and changes in altitude without moving forward were not allowed.

The motion primitives used in our implementation complies with a state lattice discretization of 0.25m per cell of the 3D Euclidean space, and $\pi/4$rad for yaw orientation θ. The selected motion primitive set comprises 5 motion primitives (3 translations and 2 rotations in place) as shown in Figure 3.3.

3.5. Experimental Results

To evaluate the functionality of our system, we performed several experiments in autonomous flight where destination goals were sent through an off-board workstation. Figure 3.4 demonstrates the 3D SLAM capability of the system. The quadrotor flew autonomously in a large room, with all the computation carried out on board. The 3D

36

Figure 3.4. Results of the autonomous flight experiment. Left: orthogonal projection of the recovered point cloud map. The path generated by the visual odometry and the corrected SLAM path are shown. Middle: side view of the point cloud map. Right: side view of the octomap.

SLAM algorithm receives pose data from the visual odometry and generates a sequence of RGB-D keyframes. The SLAM algorithm tests association between the incoming keyframe and the previous ones to provide correction of the quadrotor trajectory while building a 3D map. Another example of the 3D SLAM output can be seen in Figure 3.2.

The 4D path planner performs a path search in 283ms on average in a single core (maximum time allocated is 500ms), allowing it to settle until it finds the optimal path. These results are obtained in an indoor environment of 30x30x5m in size (see Figure 3.2), at 0.25m Octomap grid resolution. For more computationally expensive scenarios (e.g. larger environments, more motion primitives, finer space resolution) suboptimal obstacle free paths can still be obtained before reaching the optimal path ($\epsilon = 1$) within the 500ms time budget. Examples of the 4D paths obtained through an Octomap of a cluttered indoor environment are shown in Figure 3.5 and Figure 3.6.

37

Figure 3.5. Four-dimensional path (blue) in a cluttered indoor environment. Path starts from actual quadrotor pose (left reference frame) to a user selected goal pose (right reference frame). Intermediate quadrotor poses are shown along the path (colored arrows).

38

Figure 3.6. Another view of the four-dimensional path in a cluttered indoor environment. Path starts from actual quadrotor pose (near reference frame) to a user selected goal pose (far reference frame). Intermediate quadrotor poses are shown along the path (colored arrows).

3.6. Conclusions

In this chapter, we described an autonomous navigation system for a quadrotor based on a RGB-D visual odometry algorithm. We show how the use of the RGB-D camera as the only exteroceptive sensor enables 3D SLAM in autonomous flight in indoor environments. The powerful onboard computer is able to run all the components in real time. We also developed a 4DOF path planner whose functionality has been integrated on board the quadrotor using the 3D SLAM output. All systems have been shown running inside the constrained resources of the embedded platform, maintaining the execution times within the requirements imposed by autonomous flight in cluttered indoor environments.

40

Robust Exploration and Homing for Autonomous Robots

The ability to explore an unknown environment is an important prerequisite for building truly autonomous robots. Central capabilities for autonomous exploration is the selection of the next view point(s) for gathering new observations as well as robust navigation.

In this chapter, we propose a novel exploration strategy that exploits background knowledge by considering previously seen environments to make better exploration decisions. We furthermore combine this approach with robust homing so that the robot can navigate back to its starting location even if the mapping system fails and does not produce a consistent map.

We implemented and tested the proposed approach in ROS. The experiments indicate that our method improves the ability of a robot to explore challenging environments and improves the quality of the resulting maps. Finally, we are able to navigate the robot back home, even if we cannot use the map.

4.1. Introduction

Exploration is the task of selecting view points so that a robot can cover the environment with its sensors to build a map. The ability to robustly operate without user intervention is an important capability for exploration robots, especially if there is no means

41

for communication between the robot and an operator. Most exploration robots always start assuming zero knowledge and do not exploit any background knowledge about the environment or typical environments. They build a map of the environment online and make all navigation decisions based on this map. As long as this map is consistent, the robot can perform autonomous navigation by planning the shortest path—for example using A*—from its current location to its next vantage point using the map. Although recent SLAM systems are fairly robust, there is a chance that they fail, for example, due to wrong data associations generated by the front-end. Even current state-of-the-art SLAM approaches cannot guarantee the consistency of the resulting map. Computing a path based on an inconsistent map, however, is likely to lead to a failure and possibly to loosing the robot if operating in a hazardous environment. Thus, exploring robots should always decide where to go next and at the same time verify if their map is still consistent (see sketch in Figure 4.1). Considering existing approaches, however, it is fair to say that most exploration systems follow the paradigm that they (a) make their navigation and exploration decisions using the current map only and (b) assume that the map is consistent and thus can be used as the basis for path planning and navigation.

In this chapter, we aim at relaxing these assumptions. The key idea is to consider the knowledge gained from previously conducted exploration missions to support the navigation system of the robot. This is motivated by the fact that selecting appropriate target locations during exploration supports the mapping process, and can increase the probability of building a consistent map. Furthermore, we want to be able to safely navigate our robot back to its starting location, even if the mapping process failed.

The first contribution of this chapter is a novel approach to exploiting background knowledge while generating exploration behaviors to support mapping. The key idea is to use previously experienced environments to reason about what to find in the unknown parts of the world. To achieve this, we equip our robot with a database to store all acquired (local) maps and exploit this knowledge when selecting target locations. Our research is motivated by an exploration project for autonomously digitizing the Roman catacombs, which are complex underground environments with repetitive structures. We exploit previously visited areas and consider the similarities with the area around the current next view point, in order to predict possible geometries of the environment that the

42

robot may encounter during exploration. This allows the robot to actively seek for loop-closures and in this way actively reduce its pose uncertainty. Our experiments indicate that this approach is beneficiary for robots when comparing it to a standard frontier-based exploration method.

The second contribution is a robot homing approach with the goal of retrieving our robot even if the SLAM system failed to build a consistent map. To avoid that our robot gets lost, we propose a robust homing system consisting of two distinct parts. Part A performs a statistical analysis of the map and thus provides the information about its consistency. We build upon our previous work (Mazuran et al., 2014) for performing a cascade of pair-wise consistency checks using the observations perceiving the same areas. To avoid performing such checks on the overall map, we reduce the area to analyze by planning the shortest homing route for the robot assuming a consistent map. We then analyze the map consistency only along that path and can estimate on the fly if the map around this path is consistent or not with a given confidence level. If it is consistent, we navigate back on the verified homing path. Part B of our approach is responsible for driving the robot back to its starting location without a map. We achieve this by rewinding the trajectory that the robot took to reach its current pose. If the motions of the robot were perfect, i.e. would lead to the desired robot pose in the world frame, we would be able to simply invert the motion commands performed by the robot and could safely reach the starting location. Motion execution and odometry, however, are often noisy. As a result, simply following inverse motion commands will not bring the robot to the starting location in the real world in most cases. Therefore, we take into account the sensor information to guide the robot back by matching the observations with the past.

43

Figure 4.1. Mobile robot exploration has to answer the question: "Where to go next?". Our approach exploits previously mapped environments to predict potential future loop closures and thus to select better target locations. When the statistical map consistency tester provides the robot with the information that the map is not consistent anymore the robot starts rewinding the trajectory using our robust homing method.

44

4.2. Related Work

The majority of techniques for mobile robot exploration focus on generating motion commands that minimize the time needed to cover the whole terrain. Several techniques also assume that an accurate position estimate is available during exploration (Koenig and Tovey, 2003; Yamauchi, 1998). Whaite and Ferrie (1997) present an approach that uses the entropy to measure the uncertainty in the geometry of objects that are scanned with a laser range sensor. Similar techniques have been applied to mobile robots (Rocha et al., 2005; Stachniss and Burgard, 2003), but such approaches still assume to know the correct pose of the vehicle. Such approaches take the map but not the pose uncertainty into account when selecting the next vantage point. There are, however, exploration approaches that have been shown to be robust against uncertainties in the pose estimates (Duckett et al., 2002; Ko et al., 2003).

Besides the idea of navigating to the next frontier (Yamauchi, 1998), techniques based on stochastic differential equations for goal-directed exploration have been proposed by Shen et al. (2012). Similar to that, constrained partial differential equations that provide a scalar field into unknown areas have been presented by Shade and Newman (2011). An information-theoretic formulation that seeks to minimize the uncertainty in the belief about the map and the trajectory of the robot has been proposed by Stachniss et al. (2005). This approach builds upon the works of Makarenko et al. (2002) and Bourgoult et al. (2002). Both extract landmarks out of laser range scans and use an Extended Kalman Filter to solve the underlying SLAM problem. They furthermore introduce an utility function which trades-off the cost of exploring new terrain with the potential reduction of uncertainty by measuring at selected positions. A similar technique has been presented by Sim et al. (2004), who consider actions to guide the robot back to a known place in order to reduce the pose uncertainty of the vehicle.

In general, the computation of the expected entropy reductions is a complex problem, see Krause and Guestrin (2005), and in all real world systems, approximations are needed. Suitable approximations often depend on the environment model, the sensor data, and the application. In some cases, efficient approximations can be found, for example in the context of monitoring lakes using autonomous boats (Hitz et al., 2014).

45

Other approaches, especially in the context of autonomous micro aerial vehicles (MAVs), seek to estimate the expected feature density in the environment in order to plan a path through areas that support the helicopter localization (Sadat et al., 2014). This can be seen as related to information-theoretic approaches, although Sadat et al. (2014) do not formulate their approach in this framework. A related approach to MAV exploration seeks to select new vantage points during exploration, so that the expected number of visible features is maximized, see Mostegel et al. (2014).

An interesting approach by Fox et al. (2003) aims at incorporating knowledge about *other* environments into a cooperative mapping and exploration system for multiple robots. This allows for predicting simplified laser scans of an unknown environment. This idea was an inspiration for our contribution for predicting possible loop closures given the environment structure explored so far. We use this approach for exploring ancient catacombs, which are repetitive underground environments, with a mobile platform, see Figure 4.1. Chang et al. (2007) propose an approach for predicting the environment using repetitive structures for SLAM. Other background knowledge about the environment, for example semantic information, can support the exploration process as shown by Wurm et al. (2008), Stachniss et al. (2009) as well as Holz et al. (2011).

A central problem in robust exploration, however, is that in case of a SLAM failure, the map becomes inconsistent. This can prevent the robot from continuing its exploration mission and—even worse—from being able to navigate back. It is therefore important to be able to perform reliable navigation without relying on a map.

Sprunk et al. (2013) present a lidar-based teach-and-repeat method to follow a route given by the user. The approach relies on precise localization of the robot based on the lidar measurements with respect to a taught-in trajectory. Similarly, Furgale et al. (2014) perform the ICP-based teach-and-repeat approach on an autonomous robot equipped with a high precision 3D spinning lidar. They extend the standard teach-and-repeat approach by adding a local motion planner to account for dynamic changes in the environment. Our method to rewind the trajectory is similar to the teach-and-repeat setup in this formulation. However, in contrast to the mentioned methods, we use a substantially less accurate robot and thus have to cope with somewhat larger deviation from the reference trajectory.

46

Vision methods are also popular for teach-and-repeat approaches. Furgale and Barfoot (2010) present a vision-based approach to teach-and-repeat for long range rover autonomy. During a learning phase, their system builds a manifold map of overlapping submaps as the rover is piloted along a route. The map is then used for localization as the rover repeats the route autonomously. They present an autonomous planetary rover that is able to navigate even non-planar terrain without relying on an accurate global reconstruction. Nitsche et al. (2014) extend a standard teach-and-repeat approach by adding Monte Carlo localization to localize the robot with respect to the learned path. They present vision-based tests carried out both on a ground robot and an aerial drone. Battesti et al. (2011) present an online localization approach. They use visual loop-closure techniques to create consistent topo-metric maps in real-time while the robot is teleoperated and localizes itself in such maps. This allows the robot to follow the predicted path successfully compensating the odometry drift. These visual methods, however, need substantial adaptation in order to be used in a setup similar to ours: using monocular cameras to localize through feature detection relies on having enough visual information, which is not the case in the typically dark catacombs. The work presented here is based on two conference publications (Bogoslavskyi et al., 2015; Perea et al., 2015a) and a journal publication (Perea et al., 2017). In Perea et al. (2015a), we first described the idea of predictive exploration whereas Bogoslavskyi et al. (2015) addresses homing. Regarding the journal paper in Perea et al. (2017), a unified view to the problem was introduced, and a more detailed description as well as new experiments were added.

47

Figure 4.2. A customized Mesa Element threaded robot designed for exploring and digitizing Roman catacombs is the base mobile robot used in this contribution.

48

4.3. Robot and Sensor Setup

Our robot is a customized Mesa Element platform, see Figure 4.2. It is equipped with a laser range finder scanning in a horizontal 2D plane around 60cm above the ground. The robot is additionally equipped with two ASUS Xtion depth cameras that observe the local area in front of the robot in 3D. Both cameras look forward, one slightly rotated to the left and the other one to the right with a minimal overlap in the middle. Our system relies on the 2D information for solving the exploration task in order to decide which parts of the scene have been explored, and where to move next. For the robust homing presented in Section 4.5, we take into account the 3D depth images from the Xtions as this allows for a more accurate alignment of the scans. Furthermore, a local traversability analysis is done in 3D based on the Xtions (Bogoslavskyi et al., 2013).

4.4. Environment Predictive Exploration

The central question in exploration is "Where to go?". Several different cost functions for making the decision of where to go next can be defined. The most popular one goes back to Yamauchi (1998), who guides the robot to the closest reachable unexplored location. Yamauchi introduces the concept of frontiers, which are the cells of an occupancy grid map at the boundary between the free and the unexplored space. In the standard setting, this approach seeks to minimize the time that is needed to cover the environment with the robot's sensors and is a popular choice in mobile robotics. On the other hand, exploring hazardous environments require trading time for a more robust navigation that supports the mapping system and avoids pose uncertainty.

4.4.1. Information-Driven Exploration

Given the fact that most real robots maintain a probabilistic belief about their pose and the map of the environment, an alternative approach is to select the target location that is expected to minimize the uncertainty in the belief of the robot. In this setting, the exploration problem can be formulated as follows. At each time step t, the robot has

49

to decide which action a to execute (where to move next). During the execution of a, the robot obtains a sequence of observations z (for better readability, we neglect all time indices).Thus, we can define the expected information gain, also called mutual information, of selecting the action a as the expected change in entropy in the belief about the robot's poses X and the map M:

$$I(X, M; Z^a) = H(M, X) - H(M, X \mid Z^a). \tag{4.1}$$

The second term in Eq. ((4.1)) is the conditional entropy and is defined as

$$H(M, X \mid Z^a) = \int p(z \mid a) \, H(M, X \mid Z^a = z) \, dz. \tag{4.2}$$

Unfortunately, reasoning about all potential observation sequences z in Eq. ((4.2)) is intractable in nearly all real world applications since the number of potential measurements grows exponentially with the dimension of the measurement space and with time. It is therefore crucial to approximate the integral of Eq. ((4.2)) so that it can be computed efficiently with sufficient accuracy.

A suitable approximation, however, depends on the environment model, the sensor data, and the application so that no general one-fits-all solution is available. In our previous work (Stachniss et al., 2005), we considered different types of actions: First, *exploration actions* that guide the robot to the closest frontier and this reduces the map uncertainty. As we have no further information about the unseen area, it is difficult to distinguish two frontiers with respect to the expected uncertainty reduction. Second, *loop-closing and re-localization actions*, which are key to the uncertainty reduction about the robot's pose.

In this work, we aim to combine these types of actions into a single one. We seek to predict what the so far unseen environment beyond a frontier *may* look like based on background knowledge of previously seen environments and select the frontier that potentially leads to a loop-closure. In this way, we maximize the expected uncertainty reduction in the belief of the robot about the world.

50

4.4.2. Utility Function for Exploration

Most exploration systems define a utility function to relate the expected gain in information with the cost of obtaining the information. As long as no constraints such as available energy or similar are considered, the distance that the robot has to travel to obtain its measurements is a standard choice. This yields a utility function of the form

$$U(a) = I(M, Z; Z^a) - \text{cost}(a), \tag{4.3}$$

so that the task of selecting the best action can be formulated as

$$a^* = \arg\max_a I(M, Z; Z^a) - \text{cost}(a). \tag{4.4}$$

Throughout this work, we define the cost function $\text{cost}(a)$ as the path length corresponding to action a, i.e. the length of the trajectory from the current location of the robot to the designated target location.

As mentioned in the previous section, estimating the expected information gain is challenging and computationally demanding and thus we use the following approximation. We assume that actions can reduce the robot's uncertainty about the map by exploring unseen areas and/or can reduce its uncertainty about the trajectory by closing a loop:

$$a^* = \arg\max_a I_{map}(a) + I_{traj}(a) - \text{cost}(a). \tag{4.5}$$

As we do not know how large the unknown area and thus the number of unknown grid cells behind a frontier is, we may argue that all frontiers yield the same expected information gain with respect to the map uncertainty. Thus, we can simplify Eq. ((4.5)) as long as we consider only exploration actions to frontiers:

$$a^* = \arg\max_a I_{traj}(a) - \text{cost}(a). \tag{4.6}$$

The expected information gain about the trajectory $I_{traj}(a)$ is mainly influenced by loop closures. The more likely a loop closure can be obtained when executing an exploration

51

query

Figure 4.3. Example of the submap retrieval using FabMAP2. The left image shows the query map, the other ones the best four matches from the database.

action *a*, the higher its expected gain. Thus, the remainder of this section addresses the problem of predicting possible loop closures.

4.4.3. Predictive Exploration

The key contribution here is to model the predictive belief describing what the environment may look like in the unexplored areas. To compute this belief, the robot exploits environment structures it has seen in the past—either in the environment explored so far or even from previous missions. Our exploration system uses this predictive belief to evaluate the frontiers as possible target locations for the exploration. This allows us to select the frontiers that are likely to lead to a loop-closure and thus to an active reduction of the uncertainty in the robot's belief. As we will show during the experimental evaluation, this approach outperforms the traditional frontier-based exploration system.

4.4.4. Querying for Similar Environment Structures

The key idea of this approach is to look for similarities between the known areas around a frontier and portions of previously mapped environments. Under the assumption that environments are not random but expose certain structures and that these structures tend to appear more than once, we can use the already mapped areas in order to predict what the environment beyond the frontier *may* look like.

The first step is to look for portions of the already mapped environments that are similar to the area around the frontier for which the prediction should be performed. To do this, we incrementally build a database storing all local grid maps that the robot experienced. To perform a similarity query, we compare our local maps with the maps stored in

52

the database. To avoid a large number of expensive map-to-map comparisons to search for similar submaps, we rely on a bag-of-words inspired approach, a technique that is frequently used in computer vision to search for image similarities. More concretely, we apply FabMAP2 by Cummins and Newman (2009), an appearance-based approach we can use to efficiently query our database. Although FabMAP2 was originally designed to match camera images, it turns out that we can also use it to effectively search for local grid maps in a large database of maps. As FabMAP2 also provides a likelihood $l(m)$ for each match m, we can obtain a belief about possible environment structures. Figure 4.3 shows an illustration of this procedure. The image on the left is a query image and the other images are the top 4 matches reported by FabMAP2.

4.4.5. Loop Closures Prediction

As we are mainly interested in the possible paths through the unknown environment in order to find loop closures and not necessarily the exact geometry, we reduce the maps reported by FabMAP2 to extended Voronoi graphs (Beeson et al., 2005) and do all further computations on these graphs.

FabMAP2 provides us with candidates of matching maps but no geometric alignment between the query map and the reported ones. Thus, we align each map reported by FabMAP2 with our query map. This can be done in a robust manner through a RANSAC-based alignment of the Voronoi graphs using its junction points. Figure 4.4 shows an example of a Voronoi graph aligned with the map explored so far.

The next step, is to search for possible loop closures, for which we use the extended Voronoi graph. Starting from the frontier point, we traverse the Voronoi graph in a breadth-first manner. During the traversal, we check if the Voronoi graph leads to a position that is close to any other frontier in the map built so far. If this is the case, we regard that as a possible loop closure. Such a situation is illustrated in the left image of Figure 4.5. This process is executed for each frontier.

53
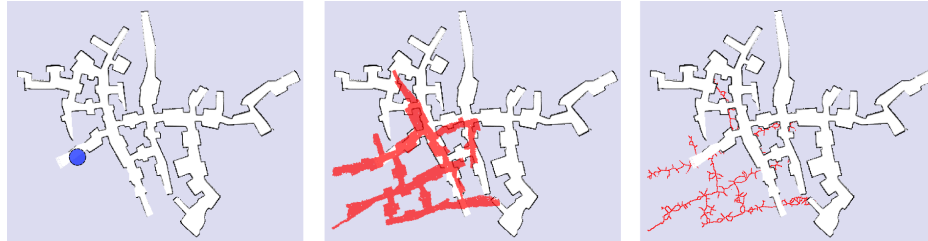
Figure 4.4. Illustration of the loop closures prediction. Left: So far explored map with the frontier under consideration (blue circle). Middle: One map from the predictive belief (in red) superimposed on the map explored so far. Right: Voronoi diagram used for the path search.
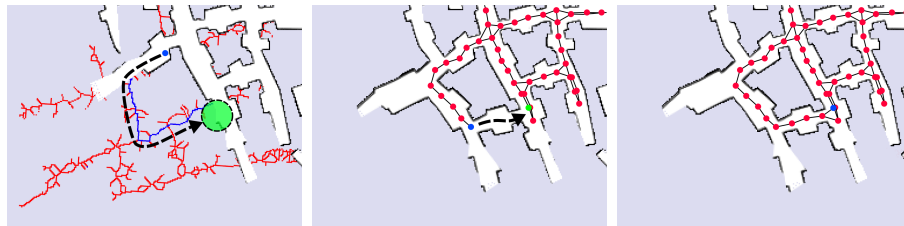
Figure 4.5. Illustration of the active loop closing. Left: prediction of the possible path with the loop closure shown in blue. Middle: the robot explores the path along the predicted loop closure and perceives the actual structure of the scene. The graph in the already explored environment shows the pose graph of the SLAM system. Right: successful loop closure Please note that the predicted environment is actually not identical with the real environment but reveals a similar structure. This similarity resulted in the shown loop closure.

54

4.4.6. Estimating the Probability of Closing a Loop

Each map reported by FabMAP2 comes with a likelihood. Thus, we can approximate the probability of closing a loop when executing an exploration action as

$$\mathbf{S}_f = \sum_{m \in \mathcal{M}(f)} l(m) \sum_{c \in \mathcal{C}(f,m)} l(c \mid m) \tag{4.7}$$

Here, $\mathcal{M}(f)$ is the set of matches returned by FabMAP2 when querying with the frontier f, and $l(m)$ the likelihood of a match m. The term $\mathcal{C}(f,m)$ refers to the set of possible loop closures computed according to the breadth-first traversal explained above and $l(c \mid m)$ is the likelihood that the loop closures can be reached. We assume that $l(c \mid m)$ is proportional to the inverse length of the path of the predicted loop closure. This means that short loop closures are more likely than long ones.

Assuming that every executed loop closure through unknown areas of the map yields the same expected uncertainty reduction, we can approximate the expected information gain I_{traj} of Eq. ((4.6)) with the score $\mathbf{S}_f$ according to Eq. ((4.7)). This is clearly a strong assumption but we argue that a high score indicates a high expected gain from exploring the frontier.

4.5. Robust Homing Using Map Consistency Checks

Under the assumption that we can ensure the consistency of the current map, homing is a comparably easy task. It basically consists of computing a collision-free path from the current location to the starting location and following the planned path with a standard navigation pipeline. Such a navigation system would, for example, localize the robot in the map built so far and plan the shortest path towards home using A* or a similar approach. If the map, however, is not consistent because the underlying SLAM system has failed, this approach is likely to lead to a deadlock situations from which the robot cannot escape easily.

To ensure a robust exploration of the environment, we address the problem of robust homing in a two-stage approach. First, while mapping the environment, a path is com-

55

puted from the current location towards home assuming the map is consistent. Then, we perform the recently proposed map consistency estimation approach by Mazuran et al. (2014) to evaluate if the map is consistent with a given confidence level. If the path towards home is consistent, and we finished exploring the environment, we simply execute this plan. If the path towards home is not consistent, we aim at reversing the trajectory of the robot taken so far. We perform this reversal by aligning the current observation with the observations obtained on the way, starting from the current location back to the starting location. This yields a robust strategy to bring a robot home to its starting location.

4.5.1. Map Consistency Test

Our map consistency estimation approach, proposed previously in Mazuran et al. (2014), builds upon a pose-graph representation that associates each pose of the robot with the individual observations that have been taken from that location. We start with evaluating the consistency of pairs of range readings. The approach of Mazuran *et al.* describes the discrepancy between two range scans by computing how much the two scans occlude each others free space.

To estimate the occlusion of the free space, we compute for each pair of scans the polygon of the robot's pose and all end points of the range scan (see Figure 4.6). Such polygons define the free space covered by the scan taken from the robot's pose. The intuition is that both scans are consistent with each other if none of the end points of the first scan lies inside the polygon of the second one and vice versa. In Mazuran et al. (2014), we define an inconsistency distance $d(p)$ for a point p, which lies inside the polygon of another scan, as the Euclidean distance of a point p to the closest point on the polygon boundary of the other scan. Intuitively speaking, for a consistent map, we assume that the inconsistency distances $d(p)$ are in line with the sensor noise of the proximity sensor. Substantially larger values for $d(p)$ may indicate that the scans are not properly aligned and the map may be inconsistent in local neighborhood of the scans.

More concretely, we can expect that, under the assumption of a perfect alignment of two scans, sensor noise in the range measurements will be the only contributor to the

56
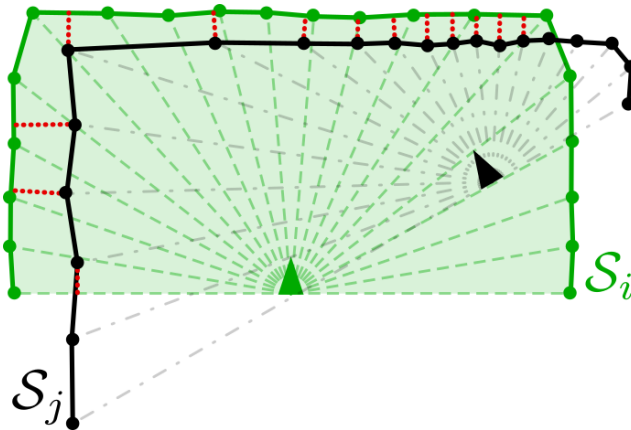
Figure 4.6. Example showing inconsistency distances in scan S_i w.r.t. scan S_j . The set of green and black polygonal chains identify the observable boundary of S_i and S_j, while the shaded green area represents the visibility polygon of S_i. The lengths of the dotted red lines represent the inconsistency distances in S_i w.r.t. S_j, for all inconsistent points w.r.t. S_j falling into the visibility polygon of S_i (adapted from Mazuran et al. (2014))

distance $d(p)$, thus on average 50% of the end points from the first scan have an inconsistency distance $d(p) > 0$ in the second scan and vice versa. According to Mazuran et al. (2014), we can formulate a statistical test for the sum of inconsistency distances $d(p)$. This test evaluates if pairs of scans are consistent given the sensor noise or reveal a larger error and thus an inconsistency.

To assess global map consistency, we could conduct this test for all pairs of scans and consider a map to be consistent if all tests are successful. The problem, however, is that a single statistical test will produce the wrong result with probability α. Thus, if we test a single scan, which overlaps with r other scans, this yields a type I error (i.e. the incorrect rejection of a true null hypothesis, or "false positive") probability of $1 - (1 - \alpha)^r$ and thus renders the direct application of the pairwise approach unsuitable. The key trick is to model the outcome of the pairwise hypothesis test as a Bernoulli-distributed random variable with parameter α. As a result of that, the number of failed tests follows a binomial distribution with parameters α and r. Given that, we can compute

57

the maximum number $\hat{\zeta}$ of tests that are allowed to fail at a confidence level $1 - \alpha'$ as

$$\hat{\zeta} = \min_{0 \leq \zeta \leq r} \left\{ \zeta \; \middle| \; \sum_{i=\zeta+1}^{r} \binom{r}{i} \alpha^i (1 - \alpha)^{r-i} \leq \alpha' \right\}. \tag{4.8}$$

This allows for computing a cascaded hypothesis test for all overlapping scans: We perform all pairwise hypothesis tests. If the number of failed tests is smaller than $\hat{\zeta}$, the overall consistency test is positive otherwise negative. For more details, we refer the reader to Mazuran et al. (2014).

4.5.2. Map Consistency Estimate for Finding the Way Home

Given the consistency test presented above, we can perform a mathematically sound statistical test to evaluate if a map is consistent or not. However, what the robot really needs to know is not the consistency of the full map. Instead, it is sufficient to know if it can safely move along a specific path through the environment to the starting location. Thus, we plan a path with A* assuming that the current map is consistent and extend our previous statistical consistency check to consider only the scans along that path. To achieve this, we select all recording locations that were closer than twice the maximum sensor range away from the trajectory planned with A*. Examples of such partial maps are depicted in Figure 4.7. The top image shows an inconsistent 2D map of the Priscilla catacombs. Directly applying the approach described in Mazuran et al. (2014) would label the whole map as inconsistent. In contrast to that, if the robot only takes into account the shortest route from A to B, he can still safely perform the navigation task, as shown in the middle image of the same figure. This is not the case if the robots wants to go from C to D as it will encounter an inconsistent part of the map on its way.

In terms of the persistent data structure that is used to store all the information, we use a generalization of a pose graph. Each node in the graph corresponds to a pose of the robot at time t. In addition to that, each node stores the original odometry pose X_t and the corresponding 3D point cloud c_t as well as the 2D scan. To efficiently represent this, the pose graph with the nodes X_t itself is kept in memory but the corresponding point clouds c_t are stored on disk and are loaded on demand.

58

4.5.3. Robust Homing by Rewinding the Trajectory

Once the consistency check has identified that the submap including the path is inconsistent, we need to perform the trajectory rewinding to bring the robot home safely. We can view the robot's forward trajectory as a series of 3D poses of the robot $\{X_0, \ldots, X_n\}$. The task of rewinding the trajectory is to drive the robot from X_n to X_0 while correcting for the error in odometry. The correction of the odometry error is done by aligning the point clouds obtained while performing trajectory rewinding with the ones corresponding to poses from X_n to X_0. Note that we subsample the trajectory in such way that each pose X_i is either $1\,m$ away from the previous one or that there is a difference of at least $10°$ in yaw between these two poses.

Without loss of generality, let us consider that the robot has to carry out the action to move from X_i to X_j and to compensate for the error in odometry. To do that, the robot exploits the current point cloud c_{current} obtained after executing the movement from X_i to X_j. In an ideal world, the command should have brought the robot to the pose X_j. In reality, there is an error introduced by slippage, uneven ground etc. Thus, we align c_{current} with c_j. To achieve that, we use a recent robust variant of ICP called NICP (Serafin and Grisetti, 2015) to find the discrepancy between the point cloud that the robot expects to perceive and what it actually perceives. The NICP method extends point-to-plane error metric proposed in Generalized ICP (Segal et al., 2009) by accounting not only for the metric distance between the points but also for the curvature of the underlying surface. The transformation between the point clouds provided by the matching algorithm can be viewed as the difference in the 3D poses at which the two point clouds c_{current} and c_j are obtained. The transformation reported by the NICP algorithm corresponds to T_Δ and thus leads to the relative position of c_{current} expressed in the local coordinate frame defined by X_j. Knowing the pose X_j and the pose of c_{current} relative to it through T_Δ enables us to compute the current position of the robot in the global odometry frame: $X_{\text{current}} = T_j T_\Delta$, where T_j is a transformation matrix that corresponds to the pose X_j in the world coordinate frame.

We use this new 3D pose X_{current} to generate a motion command to reach the next pose chosen from the recorded trajectory. As we have a wheeled platform that moves

59

on the ground, we have no control over the height and attitude. Thus we generate 2D navigation commands for the robot. We continue the above-described process until the robot is within $d_{\max}$ near its starting pose X_0.

Note that our method relies on robust incremental point cloud matching. The ICP-based algorithms tend to converge to a local minimum while performing the optimization of the objective function. This usually happens in either very cluttered environments (objective function has very high variation with multiple local minima) or, on the contrary, in the ones that are very feature-scarce (few distinct very narrow local minima). We found that using the NICP variant of ICP, which takes into account the normals of the surface, our method is able to handle the alignment errors well. In addition to that, we perform a simple consistency check between odometry and the ICP result—in case of a substantial disagreement, we temporarily rely on odometry and after the next motion NICP can again register the point clouds well.

60

Figure 4.7. The top image shows the map built so far with the detected inconsistencies (inconsistent scans are shown in red). The middle one shows a submap that is built using only the scans recorded around the A* path from A to B computed in the full map. In this example, no inconsistencies are present and none are detected. The bottom image is done in the same way as the middle one, but the A* path is computed from C to D and here, the map inconsistencies are correctly detected.

61

Figure 4.8. Partial view of the 3D model of the environment of the Priscilla catacombs built from two ASUS Xtion cameras.

62

Figure 4.9. Two performance comparisons in constant odometry bias scenario. On the left, the original map. In the middle, the closer frontier approach. On the right, our prediction-based approach. Note that the nearest frontier approach produces a map that is non consistent with the original one, so that the robot gets actually lost in it. The map produced by the prediction-based approach is instead consistent with the original one.

63

Robot failed to
continue
the exploration
task

Robot failed to continue
the exploration task

Figure 4.10. Another two performance comparisons in constant odometry bias scenario. On the left, the original map. In the middle, the closer frontier approach. On the right, our prediction-based approach. Note that the nearest frontier approach produces a map that is non consistent with the original one, so that the robot can not continue the exploration task. The map produced by the prediction-based approach is instead consistent with the original one.

64

4.6. Experiments

The experiments are designed to illustrate (i) the advantages of our predictive exploration approach, if it is safe, and (ii) that the robot can rewind trajectories in case of failure of the mapping system.

For evaluating the next view point selection approach, we use a standard frontier-based exploration approach as a baseline and show that our exploration approach selects frontiers that lead to loop closures which in turn result in improved maps of the environment. The underlying mapping framework for all exploration experiments is a state-of-the-art graph-based SLAM system, which uses g2o (Kümmerle et al., 2011) and FLIRT features to speed up the search for possible data associations (Tipaldi and Arras, 2010), uses scan matching for incremental alignments, and applies single cluster graph partitioning to resolve ambiguities as proposed by Olson (2009). The exploration and homing systems have been implemented as ROS modules.

4.6.1. Map Comparisons

First, we compare the quality of the maps obtained with frontier-based exploration vs. our predictive exploration. The environments considered here are parts of the Roman catacomb Priscilla, a difficult to traverse and large-scale underground environment in Rome. The robot is equipped with tracks and thus its odometry is in general worse than the one of a wheeled robot and it sometimes reveals a (temporarily) bias to one side.

Figure 4.9 and Figure 4.10 illustrate the obtained results for four environments using exactly the same mapping system and identical parameters for the comparison. The images on the left are the "ground truth" maps obtained from manual surveys. The images in the second column correspond to the results of the frontier-based exploration, while the images on the right show our approach. As can be seen already visually, our approach yielded a consistent model of the environment, while the frontier-based approach failed. Using the frontier-based approach the robot was unable to continue its exploration task due to an inconsistent map that prevented the computation of further exploration actions. We performed similar experiments in different nested tunnel environment and obtained

65
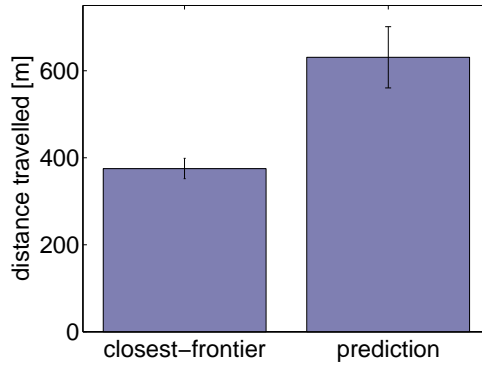
Figure 4.11. Mean and standard deviation of the distances traveled in the frontier-based approach and in the proposed approach.

complete results.

4.6.2. Exploration Path Length

The advantages of the prediction-based approach come at a cost—the cost of traversing exploration paths that are longer than the ones generated by the frontier-based approach. This experiment is designed to evaluate the increase in path length. As we are not able to obtain consistent maps for the frontier-based approach under a realistic noise model for the task under consideration, we executed this evaluation under zero noise in the simulator. Using a zero noise odometry, also the frontier-based system is able to build consistent maps. In this setting there is no advantage in using our predictive approach as the pose uncertainty is zero and no uncertainty reduction is gained from closing loops. We compared the distances traveled for the frontier-based and our approach. The distances traveled are summarized in Figure 4.11. In the worst case scenario, the path generated by our approach was 1.85 times longer than the one of the frontier-based approach. The minimum increase was a factor of 1.5. Generating on average a 1.7 times longer trajectory is clearly an overhead—for actively closing loops and in this way reducing uncertainty, however, this price must be paid.

66

4.6.3. Statistical Map Consistency Check and Robust Homing

After the robot finishes exploring the environment, it needs to find its way home. The evaluation of our framework is designed to illustrate the performance of the statistical map consistency check in conjunction with an approach to safely and robustly rewind the trajectory to return the robot to the starting position should the consistency check report the map as inconsistent.

First, Figure 4.7 illustrates an example of the statistical map consistency check performed on range data from the Priscilla catacombs in Rome. The partial maps computed around the shortest path are usually substantially smaller than the map of the whole environment, especially if the environment has multiple alternative branches and forms a complicated network of corridors or rooms as we experience it often in catacombs or underground mines. Testing smaller maps results in speed-up of the statistical consistency evaluation procedure. The timings for the maps presented in Figure 4.7 are as follows: full map shown on top—2,930 ms; middle—140 ms; bottom—170 ms. The computational time depends on the number of scans to analyze and the gain in speed grows with the difference between the sizes of the full and the reduced maps and the overlapping scans. We performed the map consistency test on five different datasets recorded in the Priscilla catacomb and the consistency check always generated correct results. In sum, testing a map along the planned path for consistency takes less than 200 ms and thus can be executed on the fly on the robot. Additionally, most of the computations could be cached when dealing with huge maps (although this was not done here). In this case, the test would only require a re-computation if the SLAM back-end changes the configuration of the pose graph substantially.

Second, if the proposed statistical consistency check evaluates the map as inconsistent we need a robust way to return the robot home to the starting location. We evaluate the ability of our approach to rewind the trajectory by carrying out 20 experiments in our lab environment. One of these experiments is illustrated in Figure 4.12. We steered the robot on a rather complicated trajectory through an obstacle parcour containing narrow passages as well as areas with lots of flat wall, which represents a challenge for the matcher. The robot activated the "rewind the trajectory" behavior after we (manually)

67

broke the SLAM system so that it followed the way in reverse order using the NICP-based pose correction.

In Figure 4.12, the original odometry measurements from the *forward path* are drawn in black (hardly visible as the red trajectory perfectly overlays it). The red line illustrates the subsampled trajectory that the robot has selected as its sequence $\{X_0, \ldots, X_n\}$ for rewinding the trajectory. Both trajectories overlay because the robot does not use any global map and relies solely on the poses he recorded in the odometry frame (to navigate back).

The gray line depicts the pure odometry measurements recorded while performing rewinding while the blue line shows the poses of the robot after the alignment by NICP. As can be seen, the robot accurately follows the previous trajectory with our approach as the blue and the red trajectories are similar. In this experiment, the average deviation of the rewinding trajectory is approximately 5 cm. From the gray trajectory, we can furthermore see that the odometry error must be taken into account—otherwise, the robot would deviate substantially from the reference path (and would collide with walls and obstacles).

We executed similar experiments in 20 different settings and were always able to robustly drive the robot back to the start location. Three examples are illustrated in Figure 4.13. Overall, this evaluation suggest that our robot is able to rewind different trajectories through the environment, robustly handling corridor-like environments with multiple narrow passages such as the doorways. Note that the robot cannot observe the doorways before it fully passed through it. Only by following the reference trajectory precisely, the robot can return.

68

Figure 4.12. Illustration of rewinding the trajectory through the office environment. The robot is steered from the bottom "tail" of the depicted trajectory to the upper-right one. Black line denotes the odometry poses saved while the robot is steered, gray denotes the odometry on the way back, red shows the temporary destination poses picked from the odometry and blue shows the same poses after the ICP correction. The pictures show several example locations visited by the robot. These feature tight doors to rooms as well as feature-scarce corridors.

69

Figure 4.13. Three experiments performed in different settings. The meaning of the lines is the same as in Figure 4.12 with the difference that the top and the bottom graphs do not show the pure odometry measurement on the return path. The average deviation from the original trajectory is between 4 cm (top dataset) and 6 cm (bottom dataset).

70

4.7. Conclusion

The ability to robustly operate without user intervention is an important capability for exploration robots in real-world settings. In this chapter, we proposed a novel approach for autonomous exploration of unknown environments with robust homing. The key contributions of this work are two-fold. First, we presented a technique to predict possible environment structures in the unseen parts of the robot's surroundings based on previously explored environments. We exploit this belief to predict possible loop closures that the robot may experience when exploring an unknown part of the scene. This allows the robot to actively reduce the uncertainty in its belief through its exploration actions. Secondly, we presented a homing system that addresses the problem of returning a robot operating in an unknown environment to its starting position even if the underlying SLAM system fails. We combined a statistical map consistency test with an NICP-based approach to precisely rewind a previously taken trajectory.

We implemented our approach and executed it both, in simulation and on a real autonomous robot. Our experiments illustrate that our technique allows for an effective exploration of difficult to map environments. By actively closing loops, we are able to obtain consistent maps of the environment. In contrast to that, a traditional frontier-based exploration approach is not able to successfully explore the scene if the SLAM system fails. In the case of a mapping failure leading to an inconsistent map, the proposed robust homing system can accurately rewind trajectories guiding the robot through narrow passages such as doorways, even when the robot could not see these narrow spaces while navigating through them.

71

Part III.

Conclusions and Future Work

73

Conclusions

The main goal of this thesis is the advancement of the state of the art in mobile robot autonomy. In order to achieve this objective, several contributions have been presented that tackle well defined problems in the areas of localization, navigation and exploration.

The very first contribution is focused on the task of robustly finding the localization of a mobile robot in an outdoor environment. Specifically, the presented technique introduces a key methodology to perform sensor fusion of a global localization sensor so ubiquitous as a GPS device. This contribution shows how this sensor measurements can be integrated with state of the art localization systems, taking the most of its advantages while properly handling its drawbacks. The technique includes a formal definition of how the sensor fusion should be performed in the context of a localization system based on Particle Filters, and approach that is becoming mainstream in the field of mobile robotics, and beyond.

The proposed particle weighting technique was compared against the more traditional approach that involves generating particles in the GPS measurement location. These two ways of combining GPS/IMU and LIDAR measurements within an AMCL based scheme were compared on real self-driving vehicle. Experiments show that the traditional method of adding new particles to the filter is likely to create clusters which might trigger wrong localization hypothesis to appear. Our approach uses the provided global location information to weight the existing particles, combined with the corresponding likelihood obtained after the weighting process using LIDARs information.

The maximum localization error yielded by the GPS particle generation method is below the assumed ground truth reproducibility margin. Our experiments show that,

75

although more intuitive, adding new particles ignores the evolution over time of the calculated robot position. Therefore, its reliability is worsen whenever the GPS measurements are not accurate enough, e.g., when multipath interference occurs. Our approach combines all sensor readings successfully, avoiding large hypothesis changes.

Furthermore, by integrating GPS information with our method, the robustness of the hypothesis estimation increases against LIDAR measurements uncertainties and odometry unmodeled behaviours, such as drift. At the same time, the number of particles needed for a correct localization is lower, as the dispersion of the particle set is reduced. Overall, the experiments have shown the benefits of our contribution in terms of hypothesis robustness, computational costs and tolerance to external disturbances.

The second contribution pushes the level of autonomy requirements for a mobile robot into a new dimension. We present a completely integrated navigation system running within a constrained and highly dynamic platform like a quadrotor, applied to full 3D environments. The navigation stack comprises a Simultaneous Localization and Mapping (SLAM) system for RGB-D cameras that provides both the robot pose and an obstacle map of the environment, as well as a 4D path planner capable of finding obstacle free and kinematically feasible trajectories for the quadrotor to navigate this environment.

We show how the use of the RGB-D camera as the only exteroceptive sensor enables 3D SLAM in autonomous flight in indoor environments. The onboard computer is able to run all the components in real time maintaining an updated model of the environment in 3D with explicit representations for free and unknown space. This information is passed into a 4DOF path planner which performs searches on the known free space within the remaining computational resources available. All systems have been shown running inside the capabilities of the embedded platform, maintaining the execution times within the requirements imposed by autonomous flight in cluttered indoor environments.

This contribution enables an autonomous robot to fully navigate a 3D environment from any location of the map to a user selected destination. Still, the capability of gathering knowledge about this environment without user intervention is key to achieve a fully autonomous mobile robot.

For this reason, the third contribution focuses on performing exploration of the environment with robust techniques. The ability to robustly operate without user intervention

76

is an important capability for exploration robots in real-world settings. We proposed a novel approach for autonomous exploration of unknown environments with robust homing. The key contributions of this work are two-fold. First, we presented a technique to predict possible environment structures in the unseen parts of the robot's surroundings based on previously explored environments. We exploit this belief to predict possible loop closures that the robot may experience when exploring an unknown part of the scene. This allows the robot to actively reduce the uncertainty in its belief through its exploration actions. Secondly, we presented a homing system that addresses the problem of returning a robot operating in an unknown environment to its starting position even if the underlying SLAM system fails.

We implemented our approach and executed it both in simulation and on a real autonomous robot. Our experiments illustrate that our technique allows for an effective exploration of difficult to map environments. By actively closing loops, we are able to obtain consistent maps of the environment. In contrast to that, a traditional frontier-based exploration approach is not able to successfully explore the scene if the SLAM system fails. In the case of a mapping failure leading to an inconsistent map, the proposed robust homing system can accurately rewind trajectories guiding the robot through narrow passages such as doorways, even when the robot could not see these narrow spaces while navigating through them.

Finally, all the contributions presented in this thesis where implemented following a modular architecture design, on top of the industry standard framework ROS (Robot Operating System). This modular design allowed for heavy re-usability of the code base between highly heterogeneous robot platforms (a self-driving car, an aerial micro-vehicle and an underground exploration platform), as well as enabling strong and continuous collaboration between geographically spread research teams (Spain, Italy, United States of America, and Germany).

All contributions where designed, implemented and tested on real autonomous robots, contributing to the state of the art in the field of mobile robotics, and enabling ongoing initiatives of practical applications for improving the quality of life of our society.

77

Future Work

The work presented in this thesis is a very small step in the field, relatively compared to the great potential of future developments, as well as the great number of challenges remaining to be solved. Here are some open lines of research worth exploring:

1. The sensor fusion techniques presented in this work open up a very interesting oportunity to be generalized, in order to support a broad set of sensors and robot applications. In terms of sensor properties, several common traits can be found between GPS, IMU, LIDAR, RGB-D cameras, 2D cameras, etc. Unified sensor fusion frameworks have been proposed and open implementations are available, but most of the time they are implemented as variations of Extended or Unscented Kalman Filters. We consider that an unified and modular sensor fusion framework tightly coupled with a global localization system based on Particle Filters, or even with a complete SLAM front-end, could greatly contribute to the mobile robotics field.

2. As shown in the GPS experiments where multipath effects are present, conflicting and/or erroneous localization hypothesis from complementary (or redundant) sources represent a challenging problem to be solved. Integrating failure mode detection and independent quality assessment of sensory data within the proposed sensor fusion techniques seems a great opportunity to improve both the performance and the robustness of localization and mapping systems.

3. The corrections that Particle Filters provide in the global robot pose hypothesis vs. each other sensory source of location data represent an invaluable source of bias and noise information of each independent source. Exploitation of this information has been proposed before to model and calibrate each sensor separately, but few times this information is correlated with the platform as a whole or with the environment (by means of the mapping system). We consider that establishing the correlation of the sensor errors with the robot location in the environment, as well as the internal status, would open up very interesting applications.

78

4. Consumer RGB-D cameras like Microsoft Kinect and Asus Xtion definitively opened up a whole spectrum of applications for aerial vehicles that need to navigate in full 3D environments. Nonetheless, the field of view and depth detection distance is still quite limited for long distance or high speed trajectories. Outdoor environments are also not well supported by these cameras because of the high infrared noise levels caused by sunlight. Recently, new sensors like solid-state LIDARs and event cameras, as well as advances in the processing of large field-of-view cameras represent a great opportunity to fully automate the navigation of aerial vehicles (or any other mobile platform) in arbitrary 3D environments.

5. The task of gathering environment knowledge through exploration is inherently at a higher level of abstraction than tasks performed by traditional navigation systems, that just focus on obtaining safe or efficient trajectories through a map. An exploration task is initially loosely defined, and its practical definition is at the same time influenced by the kind of information the sensors onboard are able to provide, as well as the navigation task needed to be solved by the robot to perform its final application. It is quite different the exploration task of a 3D environment that a flying platform has to execute to fulfill a site survey inside a factory, compared to a wheeled robot that has to update a 2D map of a changing environment while performing other tasks like people or goods transportation. Finding the best sources of information to guide exploration processes that comply with multi-objective goals (e.g. localization, mapping, and final end user application) is an open and very interesting field of research.

6. Finally, and potentially the most interesting of all, is the line of research of machine learning techniques applied to mobile robotics. In this thesis we just scratched the surface using bag-of-words techniques for location geometry prediction, but many of the current problems at hand in mobile robotics are ripe to be solved, in part or completely, with machine learning. The flexibility and power of these systems to find the best solutions by example or by discovery are changing the way robotics research is done, and it is an unprecedented opportunity to push the frontier of what is possible.

79

Part IV.

Appendix

81

Firmado por: JOSE DANIEL PEREA STRÖM
            UNIVERSIDAD DE LA LAGUNA
            Fecha: 30/06/2017 00:15:39

            JONAY TOMAS TOLEDO CARRILLO
            UNIVERSIDAD DE LA LAGUNA
            30/06/2017 02:34:46

            LEOPOLDO ACOSTA SANCHEZ
            UNIVERSIDAD DE LA LAGUNA
            30/06/2017 08:37:26

            ERNESTO PEREDA DE PABLO
            UNIVERSIDAD DE LA LAGUNA
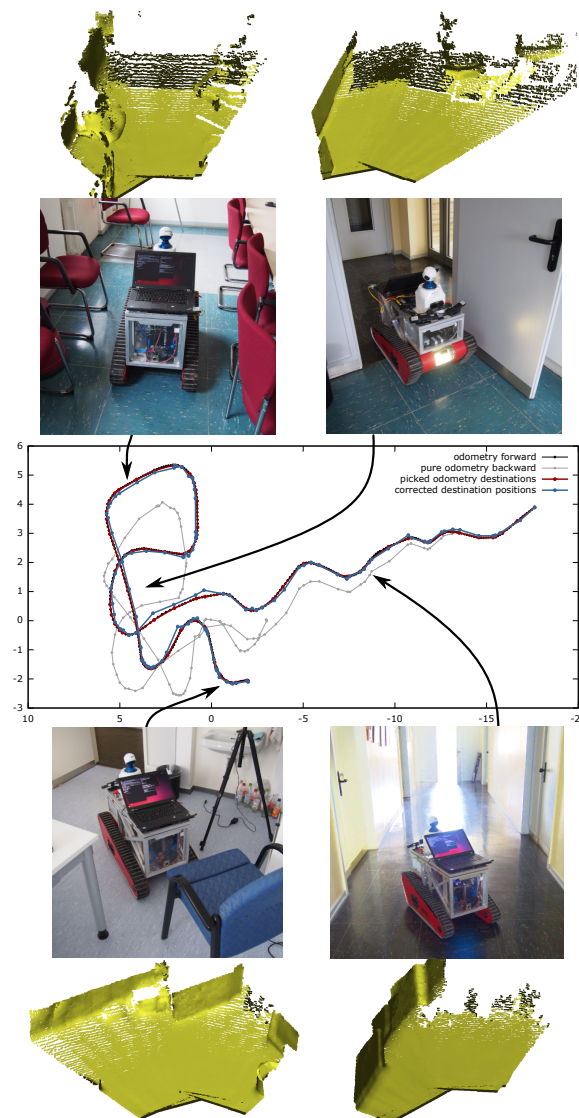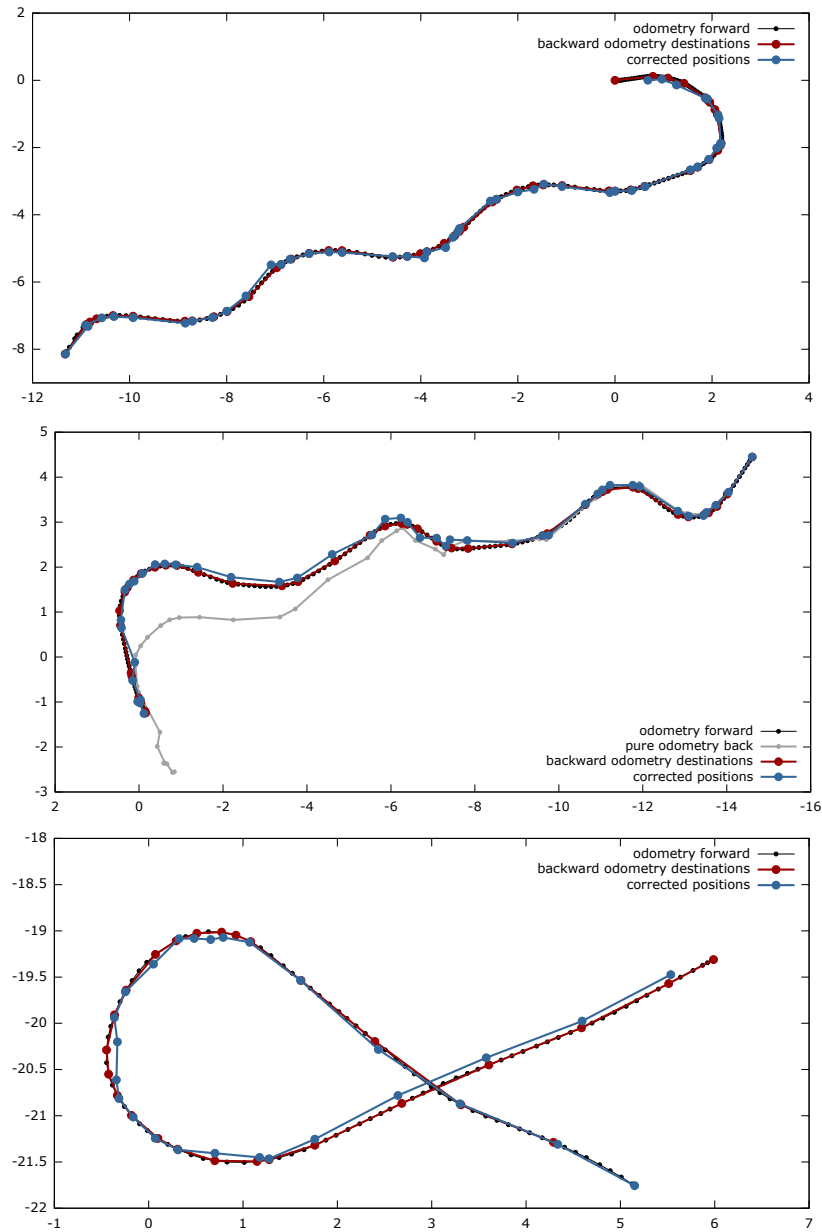   06/07/2017 13:51:10

MCL with sensor fusion based on a weighting mechanism vs. particle generation

This appendix includes the full text for the following article. The article is part of the PhD by publication:

83

MCL with sensor fusion based on a weighting mechanism
versus a particle generation approach

Daniel Perea, Javier Hernández–Aceituno, Antonio Morell,
Jonay Toledo, Alberto Hamilton and Leopoldo Acosta (*Member, IEEE*)

Abstract— **The combined action of several sensing systems, so that they are able to compensate the technical flaws of each other, is common in robotics. Monte Carlo Localization (MCL) is a popular technique used to estimate the pose of a mobile robot, which allows the fusion of heterogeneous sensor data. Several sensor fusion schemes have been proposed which include sensors like GPS to improve the performance of this algorithm. In this paper, an Adaptive MCL algorithm is used to combine data from wheel odometry, an inertial measurement unit, a global positioning system and laser scanning. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a particle generation approach. Experiments were conducted on a real robotic car within an urban environment.**

I. INTRODUCTION

Localization is one of the most relevant problems in mobile robotics, specially in outdoor and urban areas. The information obtained from sensor devices might not be as accurate as expected, so it is of great importance to define algorithms that are robust to such problems. Specifically using a known map of static obstacles, dynamic obstacles are also very likely to appear and might add uncertainty to localization algorithms. Using a single source of sensing is not practical, thus many different forms of sensor fusion have been proposed. Multiple devices can mitigate the drawbacks a single sensor might have, especially by combining proprioceptive and exteroceptive measurements, such as odometry and global positioning systems (GPS).

A review of the literature shows that Laser Imaging Detection and Ranging (LIDAR) sensors are very popular because of their data update frequency and precision. They have been combined with wheel odometry, visual cameras [1], stereovision, GPS [2], three–dimensional geographic information systems (3D–GIS) [3], and combinations thereof [4]–[6]. A common localization algorithm which integrates the information provided by different sensors is the Monte Carlo Localization (MCL) [7] method. It is based on particle filters (PF), whose samples (or particles) are weighted according to their likelihood computed from each available device [8]. Fusion of wheel odometry and GPS using MCL has been studied before [9], including omnidirectional vision [10], LIDAR

Daniel Perea (corresponding author), Javier Hernández–Aceituno, Antonio Morell, Jonay Toledo, Alberto Hamilton and Leopoldo Acosta are with Departamento Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores Universidad de La Laguna, 38203 La Laguna, Tenerife, Spain. E–mail address: `dani at isaatc.ull.es`, `javier at isaatc.ull.es, amorell at isaatc.ull.es, jonay at isaatc.ull.es, alberto at isaatc.ull.es, lacosta at ull.es`

sensors [11], [12], and inertial sensors [13]. Several adaptive variations of MCL with a variable number on PF samples have been proposed, such as Bayesian Bootstrap Filtering [14], Self–Adaptive MCL (SAMCL) [15], or Merge–MCL [16]. Adaptive Monte Carlo Localization (AMCL) [17], [18] optimaly adapts the number of samples of the PF by means of Kullback–Leibler divergences (KLD) [19].

This work showcases how GPS sensor fusion methods are affected by *multipath interferences*, a phenomenon that leads to misplaced reports of GPS sensors. We propose a fusion of wheel odometry, an Inertial Measurement Unit (IMU), GPS and LIDAR using the AMCL algorithm. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a sample generation approach. On the sample generation method, new particles are added to the PF when new absolute measurements are obtained, whereas our proposal uses this information as a weighting function over the existing particles. Experiments were conducted on both approaches, and the results reflect the robustness and better performance of our implementation.

This paper is organized as follows. Section II briefly describes the AMCL algorithm which solves the localization problem. Section III presents and discusses the proposed GPS integration method. The particle generation approach is described in Sections IV, and the robustness of our proposal is discussed and compared with the former. The mobile robot platform and the experiments conducted are described in Section V. Finally, the most relevant conclusions are summarized in Section VI.

II. LOCALIZATION WITH PARTICLE FILTERS

The state–space of the localization problem on 2–D maps x_t is given by the position of the robot, as a pair of Cartesian coordinates (x, y), and its orientation angle (θ). The AMCL algorithm finds an estimation of the posterior $p(x_t \mid z_t)$ for a mobile robot at time t, based on the observations (typically LIDAR scans) z_{t-1}.

A classical Monte Carlo method for solving the Bayesian filtering problem is the Bayesian importance sampling. One of the most popular particle filtering schemes is the *Sampling Importance Resampling* (SIR) algorithm [20], [21]. Typically, an AMCL algorithm merges wheel odometry and LIDAR information as they are available. A set of particles or samples represents the posterior about the trajectory of the robot, which are updated following the SIR algorithm, as follows:

84

1) *Sampling:* A new generation of particles $\{\boldsymbol{x}_t^{(i)}\}$ is obtained from the previous generation $\{\boldsymbol{x}_{t-1}^{(i)}\}$ by sampling from a proposal distribution $\boldsymbol{\pi}$. A probabilistic odometry motion model $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, u_{t-1})$ is used as such proposal distribution, where u_{t-1} is the odometry measurement at time $t-1$.

2) *Importance weighting:* The importance weight of a pose is a dimensionless value related to how likely is that the robot is located at $\boldsymbol{x}_t$. LIDARs measurements are used to compute the importance weight of each particle as follows:

$$w_t^{(i)} = p(z_t \mid \boldsymbol{x}_t^{(i)}) \tag{1}$$

All weights satisfy:

$$\sum_{i=1}^{N} w_t^{(i)} = 1 \quad , \tag{2}$$

on any time t, where N is the total number of samples on the PF.

3) *Resampling:* Particles with a low importance weight are more likely to be replaced by those with a higher weight. This step allows maintaining a discrete set of particles which approximate a continuous distribution, since a high number of particles on the PF is not computationally efficient. The optimal number of particles is typically given by Kullback–Leibler divergences.

III. GPS Integration

Robot localization in outdoor environments usually takes advantage of GPS devices. While providing useful information, GPS measurements might be misleading or even completely erroneous in some circumstances. The most relevant sources of error in urban scenarios are *multipath phenomena*.

Multipath is the propagation phenomenon that results in radio signals, e.g., satellite signals, reaching the receiving antenna by two or more paths. This happens mostly due to reflection and refraction phenomena, i.e., from water bodies and terrestrial objects such as mountains and buildings, resulting in a sudden "jump" in the GPS position estimate. Such source of error should not be ignored, given its negative impact on GPS resulting readings.

We consider GPS measurements given by 2–D Cartesian coordinates (x, y) and their respective covariances, as a *Universal Transverse Mercator* (UTM) projection from the *World Geodetic System* (WGS84) ellipsoid. In addition, an approximation of the orientation (*yaw*) of the robot is given by the Course–Over–Ground (COG) as the orientation (θ) of the vector between consecutive GPS positions, assuming only longitudinal movement of the robot. The covariance of this orientation is also available. When a multipath event is detected, which yields a high covariance on the current reported position, the orientation angle and its covariance are taken from an IMU device. In spite of the fact that IMU orientation reports are not very accurate, (compared with the COG value provided by a differential rover GPS), they are still useful when these reports are not available or

are not valid. For example, in situations where the robot has stopped or is moving very slowly, the difference between two consecutive positions does not yield a valid orientation. A multipath event also causes the GPS device to report erroneous COG measurements.

The key idea of the GPS integration approach is to weight the existing particles considering both the pose estimation and the associated covariances reported by a GPS device. This new observation source z_t^{GPS} provides the following position and orientation parameters at a time t:

$$z_t^{\text{GPS}} = \left\langle \begin{bmatrix} \boldsymbol{\mu} \\ \mu_\theta \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \sigma_\theta \end{bmatrix} \right\rangle \quad , \tag{3}$$

where $\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

Assuming that position (x, y) and orientation (θ) are uncorrelated and follow a Gaussian Probability Density Function (PDF), the posterior given a GPS reading can be obtained as

$$p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}) = f(x, y) \cdot f_{\text{WN}}(\theta) \quad , \tag{4}$$

where the PDF for the position is

$$f(x, y) = \frac{1}{2\pi \, |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}\left(\begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix} \right)} \quad , \tag{5}$$

and the PDF corresponding to the orientation angle, which follows a wrapped normal distribution, is

$$f_{\text{WN}}(\theta) = \frac{1}{\sigma_\theta \sqrt{2\pi}} \sum_{k=-\infty}^{\infty} e^{-(\theta - \mu_\theta + 2\pi k)^2 / 2\sigma_\theta^2} \quad . \tag{6}$$

Instead of adding new particles to the PF, the existing set of particles are weighted according to (4) and LIDAR measurements z_t^{LIDAR}, which are conditionally independent of past measurements given knowledge of the state $\boldsymbol{x}_t$ [17]. Therefore, the new posterior is:

$$p(\boldsymbol{x}_t \mid z_t) = p(\boldsymbol{x}_t \mid z_t^{\text{LIDAR}}) \cdot p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}). \tag{7}$$

In the first iteration of the algorithm there is no initial hypothesis available. Although our method does not generate GPS–based particles, an initial particle set is needed. Thus, a particle set is created and distributed following the first GPS measurement. If a *kidnapped robot* event takes place, new particles could be added similarly to overcome this problem.

IV. GPS Particle Generation

When a GPS measurement is reported, it seems natural to add a new particle cluster to the PF [13]. A new set of samples is drawn from the Gaussian PDF centered at the GPS position. The m particles with the lowest weight in the filter are replaced with the new sample set. Unfortunately, adding new particles to the PF introduces some flaws. Assuming a correct initial robot localization, if the GPS output greatly differs from the current hypothesis, it may imply that a multipath interference has happened. Misplaced

85

Appendix A. MCL with sensor fusion based on a weighting mechanism vs. particle generation

GPS reports should be discarded by the PF in order to not accept misleading hypotheses, incompatible with the current robot location. These misleading hypotheses yield incorrect output in the PF in situations where not enough significant landmarks are available in LIDAR scans. This behaviour can be seen in experiment 1 (Section V-A).

Following the proposed GPS weighting scheme, the aforementioned problems will not arise, and there will not be conflicting and ambiguous hypotheses which will eventually lead to a wrong robot localization.

During a multipath event, where a GPS measurement usually drifts from the actual robot position, the covariance values might not be properly delivered. In our implementation, the GPS weighting scheme behaves as follows:

- For multipath events where narrow covariances are incorrectly reported, particle weights will not be greatly affected, as the resulting gaussian model will have its mean far away from the position of the particles.
- When covariances are wide, the resulting GPS–centered gaussian model will affect more noticeably to the weighting of correctly localized particles, but we cannot just reject these reports. A high covariance is not only due to multipath interference phenomena, but also due to low number of visible satellites or bad constellation geometry. It is safer to treat them as any other GPS pose report.

It is worth noting that these two cases are not handled separately by the PF in our implementation. The GPS weighting is performed with every report received from the device with their respective covariance values. The PF handles them naturally in either case, in conjunction with the odometry model and the LIDAR weighting scheme.

Finally, if the mean of the reported GPS position is at the current localization hypothesis, there would not be a meaningful difference between the particle generation method and our implementation, because the current particle distribution would already include that GPS position.

V. Experiments

The experiments has been conducted on a test platform called VERDINO (Fig. 2), a fully electric two seat vehicle, based on an *EZ–GO TXT–2* golf cart. It is designed for passenger transportation and surveillance in non–structured environments. The vehicle has been modified by adding several sensors and actuators, which allows performing navigation tasks through urban areas.

Its sensorial system includes two differential GPS Javad Triumph–1 devices. The first one is a Rover GPS unit mounted on top of the vehicle, and the second one is a fixed Base station. With its position accurately defined, the Base is used for estimating the error introduced by each satellite, in order to send the corresponding corrections to the moving Rover unit. In addition, an IMU device aids during the estimation of the orientation of the vehicle, together with the Course–Over–Ground (COG) reporting capability of the GPS device. Finally, the robot includes two horizontal Sick LMS111 laser range finders, with a maximum range of 20

Fig. 1. Static localization map, with ground truth (blue) and experiment regions (red) identified. The stating position is marked as a green dot. The map covers an area of approximately $90\,m$ by $90\,m$.

Fig. 2. VERDINO prototype.

meters, and a wheel odometry system. It should be noted that our odometry sensor clearly suffers from a left drift during all experiments, noticeable in experiment 3. However, our method correctly handles these flawed reports.

The experiments were performed at the parking lot of the Computer Science Faculty of our campus, where VERDINO followed the path shown in Fig. 1, which we consider as our ground truth. This path was recorded under continuous and accurate GPS readings, with a reported position covariance under $0.02\,m$. It was inspected to guarantee that no multipath events occurred. The route was traced on the ground, and the vehicle was manually driven along it during the experiments. We assume that the error caused by reproducing the path this way is about $0.3\,m$ in the worst case scenario, way below a typical GPS measurement "jump" caused by a multipath event. The map used for Monte Carlo localization is a previously captured model of the static obstacles in the environment, which was georeferenced against the local vector topographic map from the Spatial Data Infrastructure of Canary Islands (IDECanarias).

86

Two different experiments where multipath was present were conducted in order to compare the reliability of the localization. In order to verify the GPS contribution, a final experiment compares our approach with an AMCL implementation without GPS integration. The allowed maximum number of particles was 2000 for the GPS particle generation implementation, and 500 for the GPS weighting approach. The maximum localization error obtained with GPS particle generation was 3.72 meters away from the ground truth, whereas the error yielded by our approach was 0.22 meters. All experiments are described in detail in the following subsections.

A. Multipath with no static references

In the first experiment, the robot traverses a region of the map where no static references are within the range of the LIDAR system, and therefore the position estimation must rely exclusively on wheel odometry and GPS. The GPS sensor suffers from multipath interference and reports a misplaced absolute position. Fig. 3 compares the performance of both approaches under these circumstances. Fig. 3a shows the localization output of the PF with GPS particle generation, while Fig. 3b shows the localization output with GPS weighting only, as we propose. The dispersion of the particle set in the first case is higher, as the GPS particle generation continuously introduces new particles based on the sensor covariance, while the particle set is densely concentrated in the second case.

The experiment starts with the PF yielding a correct trajectory estimate on both cases. In Fig. 3a, when the multipath interference appears on the GPS, a new particle cluster is generated some meters away from the correct trajectory. LIDAR scans have no features to match against the map, while more incorrect GPS readings are received. As the PF performance decreases, more particles are generated. The PF evolves and switches its current hypothesis to the new cluster, producing a displaced trajectory. When multipath disappears, the PF switches again to the correct path. This incorrect behaviour is present in other implementations of GPS integration on MCL algorithms where new particles are added [13]. As can be seen on Fig. 3b, the output is correct in our implementation, in spite of suffering from multipath interference. By not adding new particles, only existing ones are considered and the GPS weighting is very low, given the distance to the displaced readings.

B. Valid LIDAR measurements with multipath

During the second experiment (see Fig. 4), the robot traverses an area where obstacles are visible, thus LIDAR scans can be matched with the map. Here, multipath interference also appears, but this time the output is different from Fig. 3. In Fig. 4a the newly created cluster does not affect the output of the PF, as the LIDAR scan matching has more influence than the GPS covariance weighting. Nonetheless, a great number of particles are always present on the wrong track during multipath, and each one is propagated with the odometry model and weighted according to each sensor

reading. This produces a waste of processing power, and it is a source of potential errors in scan matching that may mislead the PF hypothesis. In Fig. 4b the output follows the trajectory closer, keeping a narrower particle set. The increased confidence in the hypothesis reduces the number of particles in the current set, freeing up CPU resources.

C. AMCL without GPS integration

A known, noticeable left drift in our odometry sensor is present in all of the experiments. When comparing our method with an implementation of AMCL without GPS integration, this flaw is corrected in both cases thanks to the observable LIDAR landmarks, as can be seen on the left half of Fig. 5. However, when these LIDAR references are not available, the output of the AMCL algorithm without GPS weighting drifts away. Due to the lack of information, the dispersion of the particle distribution grows over time. Meanwhile, thanks to GPS integration, the PF hypothesis of our proposal follows the ground truth closely.

VI. Conclusion

We have compared two different ways of combining GPS/IMU and LIDAR measurements within an AMCL based scheme which fuses data from multiple sensors. Experiments show that adding new particles to a PF on MCL algorithms are likely to create clusters which might trigger a correctly localized hypothesis to change to a wrong location. Our approach uses the provided global location information to weight the existing particles, combined with the corresponding likelihood obtained after the weighting process using LIDARs information.

The maximum localization error yielded by the GPS particle generation method is below the assumed ground truth reproducibility margin. Our experiments show that, although more intuitive, adding new particles ignores the evolution over time of the calculated robot position. Therefore, its reliability is worsen whenever the GPS measurements are not accurate enough, e.g., when multipath interference occurs. Our approach combines all sensor readings successfully, avoiding large hypothesis changes.

Furthermore, by integrating GPS information with our method, the robustness of the hypothesis estimation increases against LIDAR measurements uncertainties and odometry unmodeled behaviours, such as drift. At the same time, the number of particles needed for a correct localization is lower, as the dispersion of the particle set is reduced.

87

Appendix A. MCL with sensor fusion based on a weighting mechanism vs. particle generation

(a) GPS particle generation (b) GPS weighting

Fig. 3. Experiment 1 – The dispersion is higher on (a), since the number of particles is also high (2000) and they are generated based on GPS measurements, while the robot traverses an area without enough spatial references, i.e., LIDAR readings do not report obstacles nearby. When applying the weighting approach (b), a maximum of 500 particles manage to keep the robot localized, even while the GPS suffers from a multipath event. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; yellow dotted line: ground truth.

REFERENCES

[1] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using Visual Appearance and Laser Ranging," in *IEEE Int. Conf. Robotics and Automation*, May 2006, pp. 1180–1187.

[2] L. Wei, C. Cappelle, and Y. Ruichek, "Unscented Information Filter Based Multi–Sensor Data Fusion using Stereo Camera, Laser Range Finder and GPS Receiver for Vehicle Localization," in *XIV Int. IEEE Conf. Intell. Transportation Systems*, Oct. 2011, pp. 1923–1928.

[3] J. Peng, M. El Najjar, C. Cappelle, D. Pomorski, F. Charpillet, and A. Deeb, "A Novel Geo–Localisation Method using GPS, 3D–GIS and Laser Scanner for Intelligent Vehicle Navigation in Urban Areas," in *Int. Conf. Advanced Robotics*, June 2009, pp. 1–6.

[4] C. Cappelle, M. El Badaoui El Najjar, F. Charpillet, and D. Pomorski, "Outdoor Obstacle Detection and Localisation with Monovision and 3D Geographical Database," in *IEEE Intell. Transportation Systems Conference*, Oct. 2007, pp. 1102–1107.

[5] L. Wei, C. Cappelle, and Y. Ruichek, "Localization of Intelligent Ground Vehicles in Outdoor Urban Environments using Stereovision and GPS Integration," in *XV Int. Conf. Advanced Robotics*, June 2011, pp. 192–197.

[6] C. Smaili, M. El Najjar, and C. Francois, "A Chained Form State Representation for Outdoor Vehicle Localisation," in *XIV Int. IEEE Conf. Intell. Transportation Systems*, Oct. 2011, pp. 1386–1391.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *XVI Nat. Conf. Artificial Intell., XI Innovative Applications of Artificial Intell. Conf.* American Association for Artificial Intelligence, 1999, pp. 343–349.

[8] D. Silver and A. Stentz, "Monte Carlo Localization and Registration to Prior Data for Outdoor Navigation," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Sept. 2011, pp. 510–517.

[9] M. Moreira, H. Machado, C. Mendonca, and G. Pereira, "Mobile Robot Outdoor Localization using Planar Beacons and Visual Improved Odometry," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Nov. 2007, pp. 2468–2473.

[10] E. Frontoni, A. Ascani, A. Mancini, and P. Zingaretti, "Robot Localization in Urban Environments using Omnidirectional Vision Sensors and Partial Heterogeneous apriori Knowledge," in *IEEE/ASME Int. Conf. Mechatronics and Embedded Systems and Applications*, July 2010, pp. 428–433.

[11] H. Chang, J. Choi, and M. Kim, "Experimental Research of Probabilistic Localization of Service Robots using Range Image Data and Indoor GPS System," in *IEEE Conf. Emerging Technologies and Factory Automation*, Sept. 2006, pp. 1021–1027.

[12] ——, "Probabilistic Localization of Service Robot by Sensor Fusion," in *Int. Joint Conf.*, Oct. 2006, pp. 3626–3631.

[13] M. Hentschel, O. Wulf, and B. Wagner, "A GPS and Laser–Based Localization for Urban and Non–Urban Outdoor Environments," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Sept. 2008, pp. 149–154.

[14] T. Khalid, Z. Mourad, C. Jean-Bernard, and B. Mohammed, "Bayesian Bootstrap Filter for Integrated GPS and Dead Reckoning Positioning," in *IEEE Int. Symposium on Industrial Electronics*, June 2007, pp. 1520–1524.

[15] L. Zhang, R. Zapata, and P. Lepinay, "Self–Adaptive Monte Carlo for Single–Robot and Multi–Robot Localization," in *IEEE Int. Conf. Automation and Logistics*, Aug. 2009, pp. 1927–1933.

[16] T. Li, S. Sun, and J. Duan, "Monte Carlo Localization for Mobile Robot using Adaptive Particle Merging and Splitting Technique," in *IEEE Int. Conf. Information and Automation*, June 2010, pp. 1913–1918.

[17] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1004, 2003.

88

(a) GPS particle generation   (b) GPS weighting
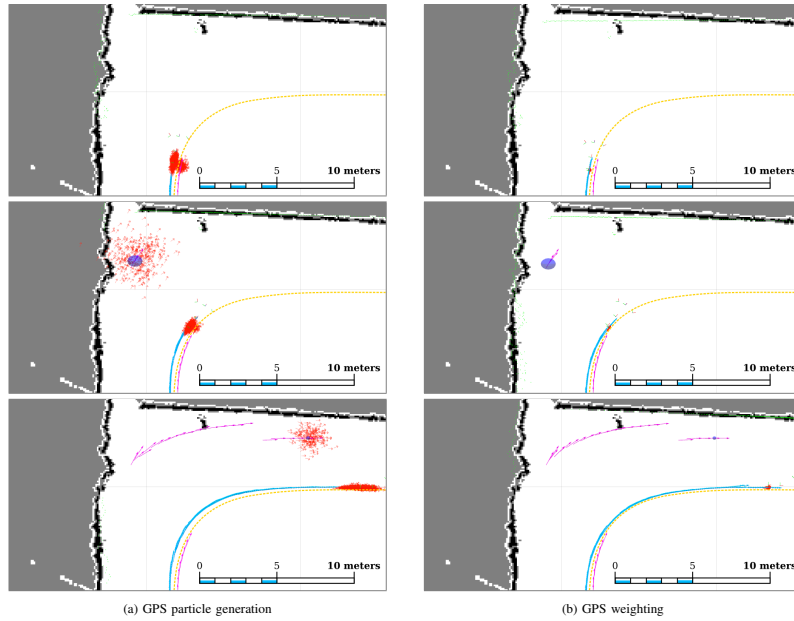
Fig. 4.   Experiment 2 – Particles are once again widely spread on the particle generation approach (a) due to the higher number of particles at the GPS position estimate. Unlike experiment 1, the hypothesis does not jump after the multipath event, thanks to LIDAR readings matching a corner successfully. On (b), the particle set is narrower and also does not suffers from an undesired jump. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; green dots: LIDAR scans; yellow dotted line: ground truth.
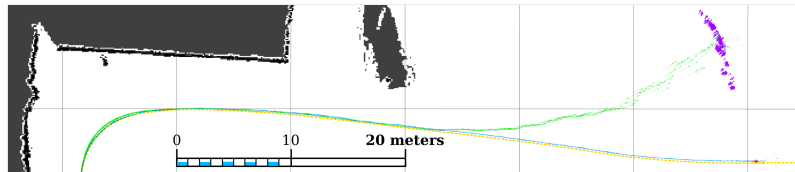
Fig. 5.   Experiment 3 – Comparison between an implementation of AMCL that does not integrate GPS measurements and our proposal. Light blue arrows: robot trajectory computed by our approach; green arrows: robot trajectory calculated by AMCL without GPS readings; red arrows: particle set distribution generated by our approach; purple arrows: particle set distribution generated without GPS; yellow dotted line: ground truth.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents.   MIT Press, 2005.

[19] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[20] D. B. Rubin, "Using the SIR algorithm to simulate posterior distributions," in *Bayesian Statistics 3*, M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith, Eds.   Oxford University Press, 1988, pp. 395–402.

[21] A. F. M. Smith and A. E. Gelfand, "Bayesian Statistics without Tears: A Sampling–Resampling Perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, may 1992.

89

Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry

This appendix includes the full text for the following article. The article is part of the PhD by publication:

91

Autonomous Quadrotor Flight Using Onboard RGB-D Visual Odometry

Roberto G. Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Ström, Jizhong Xiao*

* *Senior Member, IEEE*

Abstract—In this paper we present a navigation system for Micro Aerial Vehicles (MAV) based on information provided by a visual odometry algorithm processing data from an RGB-D camera. The visual odometry algorithm uses an uncertainty analysis of the depth information to align newly observed features against a global sparse model of previously detected 3D features. The visual odometry provides updates at roughly 30 Hz that is fused at 1 KHz with the inertial sensor data through a Kalman Filter. The high-rate pose estimation is used as feedback for the controller, enabling autonomous flight. We developed a 4DOF path planner and implemented a real-time 3D SLAM where all the system runs on-board. The experimental results and live video demonstrates the autonomous flight and 3D SLAM capabilities of the quadrotor with our system.

Fig. 1. The CityFlyer MAV equipped with an Asus Xtion Pro Live RGB-D camera.

I. INTRODUCTION

Micro aerial vehicles such as quadrotors are popular platforms often used by researchers because of their agility, high maneuverability, simple mechanicale design and compact size. Their applications range from surveillance, search and rescue, to 3D mapping and photography. In order to perform such tasks, they require a set of sensors suited to the particular use and context, and the capability of ensure stable and autonomous flight. Global Positioning System (GPS) is one of the most common sensors used for outdoor flight. Such solution is not always reliable, particularly when the signal's reception or precision might be unacceptable, as in the case of dense or indoor environments. Furthermore, in tasks like exploration and autonomous navigation in cluttered environments, global position information of the MAV is not sufficient, so a perception of the surroundings is needed. When external motion capture systems can not be deployed, the vehicle needs to rely only on onboard sensors such as laser scanners and cameras. Laser scanners provide range

Roberto. G. Valenti and Jizhong Xiao are with the Electrical Engineering Department, City College of New York, Convent Ave & 140th Street, New York, NY 10031 rvalent00@citymail.cuny.edu, jxiao@ccny.cuny.edu, corresponding author

Ivan Dryanovski and Carlos Jaramillo are with the Dept. of Computer Science, The Graduate Center, The City University of New York, 365 Fifth Avenue, New York, NY 10016 idryanovski@gc.cuny.edu, cjaramillo@gc.cuny.edu

Daniel Perea Ström is with Dept. of Ingeniera de Sistemas y Automtica y Arquitectura y Tecnologa de Computadores, Universidad de La Laguna, Calle Astrofisico Fco. Sanchez s/n Edificio Fisica/Matematicas, San Cristbal de La Laguna, Santa Cruz de Tenerife, Spain 38271 dani@isaatc.ull.es

information with high precision and odometry can be derived by scan matching technique. However, laser scanners are not an optimal solution because of their weight and high power consumption. Therefore, for more complex environment, The need of lightweight 3D perception sensors is fulfilled by cameras. For this reason, visual odometry algorithm has become very common for flying robots.

An RGB-D camera is a device which provides RGB (red, green, blue) color and depth information for each pixel of the image. Depth is retrieved through the conjuction of an infrared projector and an infrared receiver. Recently new RGB-D cameras such as the Microsoft Kinect and the Asus Xtion, have become popular in the robotics community due to their reduced size, low weight and affordable cost.

In this paper we present our approach for autonomous quadrotor flight and navigation by means of pose data from an RGB-D visual odometry algorithm, which relies on a frame-to-model registration technique, mantaining a low computation complexity (without any GPU acceleration), while reducing considerably the drift error of a typical frame-to-frame approach.

II. PREVIOUS WORK

Autonomous MAV flight using data from visual odometry has been achieved in the past by several researcher groups with different approaches.

Algorithms based on stereo vision have been used by Johnson et al. [13] and Yu et al. [26] to control the altitude of a UAV. Park et al. [19] and Zamudio et al. [27] used a stereo camera to control a quadrotor. In [19] a stereo vision system is used to perform collision avoidance. Other

92

strategies using monocular vision, which better meet the needs of a limited payload of a MAV, have been adopted. Achtelick et al [1] controlled a quadrotor for both indoor and outdoor flight by using visual information coming from a single camera pointing down, where depth information was recoverred by fusing a pressure sensor and an accelerometer through an Extended Kalman Filter. A similar system is presented by Blosch et al. [3] to perform indoor exploration with a MAV (with the difference that it does not rely on any other exteroceptive sensor). Conroy et al. [5] and Zingg et al. [28], present a biologically inspired vision system for safe and stable corridor navigation of a MAV based on optical flow from an omnidirectional camera. Rondon et al. [22] present a monocular vision system to estimate and control altitute, lateral position and forward velocity of a MAV via optical flow.

In order to perform several tasks with the same platform, both monocular and stereo vision techniques have been adopted. For instance Hrabar et al. [11] combine stereo vision from a forward-facing camera pair and optical flow from two sideways-looking monocular cameras in order to avoid obstacles and navigate inside a canyon with maximum clearance. Meier et al. [17] and Fraundorfer et al. [9] integrate a down-facing monocular camera with a forward-facing stereo camera for flight control and obstacle avoidance. For a more robust navigation in different environments, visual odometry is sometimes fused with other exteroceptive position sensors. In the work by Tomic et al. [14] and Bachrach et al. [2], a quadrotor is equipped with a laser range-finder and a stereo camera to enable stable flight and SLAM in a number of large-scale, GPS-denied environments, such as an urban canyon.

Unlike visual odometry from standard cameras, RGB-D visual odometry has not been widely used by researchers to control a flying robot. Stowers et al. [24] presented one of the first results of using RGB-D camera for real-time robot-control application. They use a Kinect pointed towards the ground to estimate and control the height of a quadrotor. The work presented by Huang et al. [12] is the most closely related to ours. They use a RGB-D visual-odometry (based on standard stereo visual odometry) to control a MAV in unknown indoor environments and do mapping. The pose estimation runs on the onboard computer while mapping and loop closure runs on an off-board computer.

In our proposed system, we use a recently developed visual odometry algorithm, able to reduce the drift error thanks to a fast frame-to-model approach. Further, all the components of the system (including mapping and loop closure) run onboard on the same kind of computer as the system proposed by Huang et al. [12].

III. SYSTEM ARCHITECTURE

The platform we use for our experiments is an AscTec Pelican quadrotor [10], on which we mounted an Asus Xtion Pro Live. The quadrotor is equipped with a 1.86 GHz Core2Duo processor with 4GB of RAM and a Flight Control Unit (FCU) board with 2 ARM7 microcontrollers,
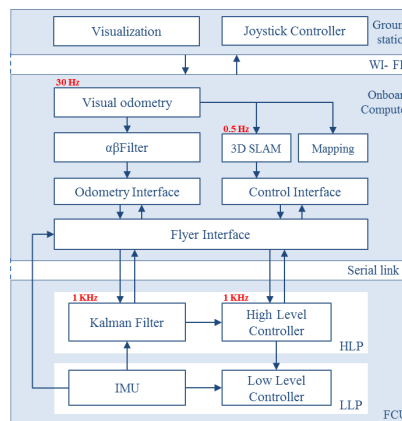
Fig. 2. System diagram. All the software running on the on-board computer and on the FCU are our contribution except mapping and the Low Level Controller provided by Asctec Technology.

an Inertial Measurement Unit (IMU) and a pressure sensor. The system architecture is shown in Fig. 2 which is described in detail in our previous work [6]. We use one of the two microcontrollers, the so-called High Level Processor (HLP), to run our custom firmware that handles sensor fusion and control, while the Low Level Processor (LLP) is responsible for attitude control, IMU data fusion and hardware communication. The powerful on-board computer is able to manage visual odometry, 3D SLAM and motion planning. The entire framework is distributed between a ground station, the on-board computer and the FCU. The ground station is only used for visualization and teleoperation. Our framework uses ROS [21] as middleware, allowing communication among all the different components of the software (implemented as *nodelets*, a ROS mechanism for zero-copy message transport). The HLP and the onboard computer communicate with each other through the serial interface, where the `Flyer Interface` sends ROS messages and services traslated into packets. Communication between the two ARM7 microcontrollers (HLP and LLP) of the FCU board is via an I^2C bus.

IV. STATE ESTIMATION

A. Visual Odometry

The visual odometry adopted in this paper uses a frame-to-model registration approach to compute the transformation between two consecutive camera poses. This approach allows us to considerably decrease the drift in the pose as demonstrated in [7]. We first detect the features in the

93

captured scene by using Shi-Tomasi [23] algorithm and their 3D coordinates in the camera frame, then we align these features against a global model of 3D features (previously detected). We perform data association and filtering using a probabilistic method, which employs a novel uncertainty estimation based on a Gaussian mixture model (described in our previous work [7]). We use this 3D normal distribution model for each feature detected in the incoming RGB-D image. This set of 3D features (with mean and covariance matrix $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$), is expressed with respect to the camera reference frame. We refer to this set as *Data*. We have a similar set that we call *Model* and is expressed in the fixed frame. We use ICP [4] to align *Data* against *Model* and then it is transformed into the fixed frame. The alignment produces the transformation T, composed by a Rotation matrix R and a translation vector $\mathbf{t}$. We also need to express the mean and the covariance matrix of each feature in *Data* with respect to the fixed frame. We can do so according to:

$$\boldsymbol{\mu}' = \mathbf{R}\mu + \mathbf{t} \tag{1a}$$

$$\boldsymbol{\Sigma}' = \mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^T. \tag{1b}$$

Once we have *Data* expressed in the fixed frame we generate correspondences adopting the following steps. First we build a Kd-tree [18] of the *Model*, and then for each feature of *Data* we find k nearest Euclidean neighbors from the *Model*. Next, for each point $\mathbf{d}'_i$ of the transformed *Data*, we compute the Mahalanobis distance between the point and its nearest neighbor in the *Model*, $\mathbf{m}_j$.

$$dist(\mathbf{d}',\mathbf{m}) = \sqrt{\Delta_{\mathbf{d}'\mathbf{m}}(\Sigma^{[M]}+\Sigma^{[D']})^{-1}\Delta^T_{\mathbf{d}'\mathbf{m}}} \tag{2}$$

If this distance is lower than a certain threshold, the two points are associated establishing the correspondence. All the points which cannot be associated are inserted in the *Model*. The update is performed by a Kalman Filter, which takes the *Model* and its covariance matrix $\boldsymbol{\Sigma}$ as prediction and updates it with the new features and their covariances (for more details refer to [7]). In order to guarantee costant-time performance, we constrain the model's maximum size. If the model grows beyond a certain upper bound, the oldest features are discarded and overwritten with the new ones.

The images are streamed at QVGA resolution and processed in the on-board computer. The visual odometry runtime is shown in Fig. 3, consists of two parts: feature extraction and motion estimation. The average processing time is 12.3 ms with a maximum of 43 ms and a standard deviation of 2.5 ms.

B. Sensor Fusion

The output pose of the visual odometry is sent through the serial interface to the HLP, where is fused with IMU data at a rate of 1 Khz. The high frequency KF's output are fed into the controller, enabling stable flight. As in [6], we cascade an Alpha Beta Filter ($\alpha\beta F$) and a Kalman Filter (KF). The ($\alpha\beta F$) runs on the on-board computer and provides a smoother evaluation of its input data (without an actual probabilistic analysis). We use it to reduce the
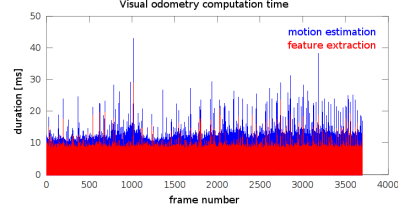
Fig. 3. Onboard computer Processing duration for each incoming QVGA image.

noise in the first estimation of the velocity, which is a simple derivative of the visual odometry position data. Hence, the output of the ($\alpha\beta F$) is sent over serial interface and serves as correction in the KF. In this indoor application we assume that the quadrotor moves with low velocity and following quasi-rectilinear path and in-place rotations. This assumption allows us to decouple the axis in the KF design and to compute the linear acceleration relative to the fixed frame, from the IMU reading, as explained in [1]. We use three smaller KF's for each position axis as well as a KF for yaw. The discrete state space linear model of the KF is:

$$\boldsymbol{x_k} = \boldsymbol{A}_k \cdot \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \cdot \boldsymbol{u}_k \tag{3a}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k \cdot \boldsymbol{x}_k \tag{3b}$$

where the state, input and measurement for x (and similary y and z) are:

$$\boldsymbol{x} = [x\ v_x]^T \quad \boldsymbol{u} = [a_x] \quad \boldsymbol{z} = [x_{vo}\ v_{x_{vo}}]^T \tag{4}$$

while for yaw we have:

$$\boldsymbol{x} = [\psi] \quad \boldsymbol{u} = [\omega_z] \quad \boldsymbol{z} = [\psi_{vo}] \tag{5}$$

where a is the linear acceleration detected by the IMU, expressed with respect to the fixed frame and ω, the angular velocity. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ of the system in (3) for x, y and z are:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{6}$$

and for yaw:

$$\boldsymbol{A} = [1] \quad \boldsymbol{B} = [\Delta T] \quad \boldsymbol{H} = [1] \tag{7}$$

Fig. 4 shows the ouput result of the KF for the $x-$component of the linear velocity.

V. CONTROL

The control system provides position and velocity control separately for each axis. It is based on a cascade structure of two loops, where the inner loop is provided by the Low level Controller (LLC) implemented in the LLP. It controls roll, pitch, yaw-rate and thrust (RPYT). The outer controller generetes RPYT commands to the inner loop controller. Roll

94
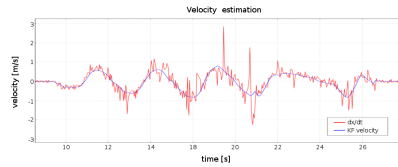
Fig. 4. Linear x velocity estimation output of the Kalman Filter compared to the rough estimation by position derivative. We obtained similar results for y and z.

and pitch commands are generated from the outer loop $x-y-$ controllers as reference to the inner attitude controller. Similarly, thrust and yaw rate commands are generated from the height and yaw controller, respectively. The position controller is based on a modified PID, while velocity and yaw controller are PI and P controllers, respectively, as explained in [6].

VI. REAL-TIME VISUAL SLAM

We developed a visual keyframe-based SLAM system. The algorithm runs in real time on-board the quadrotor in a separate thread. The SLAM algorithm takes as input the pose of the quadrotor provided by the visual odometry and generates a sequence of RGB-D *keyframes*. Each keyframe consists of a RGB and Depth image pair, together with the pose of the camera at that instant and a set of SURF features detected in the RGB image. A new keyframe is generated once the angular or linear distance traveled between the current pose and the pose of the latest keyframe exceeds a given threshold (for example, 0.3 meters or 20 degrees). Incoming keyframes are tested for associations against previous keyframes. An association between two keyframes occurs when they are observing the same scene. This is accomplished in three steps. First, for the incoming keyframe, we build a set of *candidates* from the set of previous keyframes. Candidates are keyframes whose poses are close enough to be associated with the new keyframe. We use a liberal pruning threshold (for example, 5 meters and 90 degrees). Next, we train a descriptor matcher from all the SURF (Speeded Up Robust Features) keypoints in the candidate frames. The descriptor matcher is based on a FLANN (Fast Library for Approximate Nearest Neighbors) search tree [18]. We use the tree to further limit the candidate keyframes, based on the number of nearest neighbors each feature in the incoming keyframe has in each of the candidate keyframes. We keep only the k top candidates. For each of the remaining candidates, we perform robust RANSAC (RANdom SAmple Consensu) [8] matching of the SURF features. If the RANSAC algorithm finds enough geometric inliers, we assume there is an association between the two keyframes. The association observation is the transformation which best aligns the inliers.

Once the associations are established, we build a graph whose nodes are keyframe poses and whose edges are association observations. For consecutive keyframes, the observation comes from the visual odometry. Additional associations are generated through the RANSAC matching described above. Using g^2o [15], we find the configuration of poses which minimizes the observation accross the whole graph.

The procedure runs at a rate between 1 Hz and 2 Hz onboard the quadrotor.

The keyframes are used to build a dense Octomap [25] which can be used for path-planning.

VII. 4DOF PATH PLANNING

This section introduces a quadrotor path planner in x, y, z and yaw directions. This implementation is based on a search approach where the state space is discretized using a state lattice of motion primitives [20] and an incremental and anytime version of the A* algorithm with Euclidean distance heuristic. This module has been tested in simulation in real-time in combination with the rest of the systems presented in this paper, on an identical computer as the CPU onboard the MAV.

A. State space discretization

The quadrotor state space is discretized following a state lattice, a graph search space that integrates motion planning constraints within state exploration. In this case, the state space is four-dimensional, combining the quadrotor position in Euclidean space (x, y, z) with the yaw orientation ψ. State space exploration is executed following a set of motion primitives. Motion primitives are short, kinematically feasible path segments, that can be combined together to produce longer and more complex paths. Any combination of motion primitives yield a path that complies to the non-holonomic constraints imposed by the motion planning problem. Motion primitives are pre-computed, and their traversal cost is multiplied by a user selected weight to obtain the motion cost. Weights are assigned to each motion primitive, in order to model preferences of one primitive over the others, e.g., penalizing changes in altitude while moving forward, in order to keep next positions centered within sensors field of view. Collision checking is performed online while exploring the search graph.

Planning results are greatly affected by motion primitives selection, in terms of planner times, planner completeness, and resulting path quality. The planner can not obtain a feasible path if it cannot be produced by a combination of available motion primitives. For example, backwards paths cannot be generated if backwards motion primitives are not pre-computed and made available in the set. A richer set of motion primitives improves state space coverage adding flexibility to the planner, but there is a trade-off in computation time, as each new motion primitive increases the branching factor at each state.

B. Search algorithm

The described state lattice is explored using a graph search algorithm. This algorithm is a variant of the A*

95

Fig. 5. Four-dimensional path (blue) in a cluttered indoor environment. Path starts from actual quadrotor pose (left reference frame) to a user selected goal pose (right reference frame). Intermediate quadrotor poses are shown along the path (colored arrows).

Fig. 6. Control performance in a hovering experiment over 100s. 3D view (left) and top view (right)

Fig. 7. Position control with a sequence of waypoints, varying x and y.
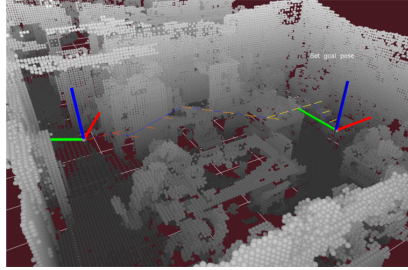
search extended with anytime and incremental capabilities called ARA* (Anytime Repairing A*) [16]. ARA* anytime capability is obtained by executing a series of A* searches where the heuristic is inflated by a factor $\epsilon > 1$, and reducing this factor on each execution. With an inflated heuristic, A* search gives more relevance to the heuristic estimation. This results in a faster algorithm by means of losing optimality, but it has been shown that the computed path sub-optimality is bounded to ϵ times the cost of the optimal solution [16]. ARA* starts with a high ϵ value in order to obtain a feasible path very fast. If time is available, ϵ is decreased and a search is executed again reusing computation from previous search. If enough time is available to reduce ϵ to 1, the heuristic is not inflated anymore, and the last search returns the optimal solution.

The motion primitives used in our implementation complies with a state lattice discretization of 0.25m per cell of the 3D Euclidean space, and $\pi/4$rad for yaw orientation θ. A typical path query takes 283ms average in a single core (maximum time allowed is 500ms) until the optimal path is obtained, in an indoor environment 30x30x5m in size at 0.25m resolution. For larger environments, more motion primitives, or finer space resolution, obstacle free paths can still be obtained before reaching the optimal path ($\epsilon = 1$) within the 500ms time budget. An example of the 4D path obtained in a cluttered indoor environment is shown in Fig. 5.

VIII. EXPERIMENTAL RESULTS

To evaluate the functionality of our system, we performed several experiments in automous flight where waypoints were sent through an off-board workstation. In the first experiment shown in Fig. 6, we commanded the quadrotor to hover in place for a time of 100 seconds. In the experiment of Fig. 7, the quadrotor changed its x and y position after sending a sequence of waypoints. Both experiments prove the effectiveness of state estimation and control with a maximum error of 20 cm or less. Fig. 8 demonstrates the 3D SLAM
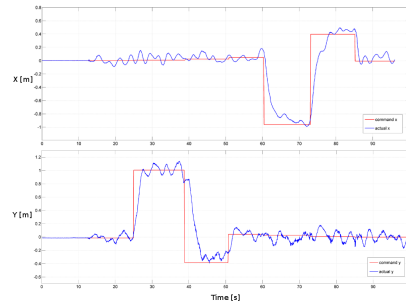
capability of the system. The quadrotor flew autonomously in a large room, with all the computation carried out on-board. The 3D SLAM algorithm receives pose data from the visual odometry and generates a sequence of RGB-D keyframes as explained in Sec. VI. The SLAM algorithm tests association between the incoming keyframe and the previous ones to provide correction of the quadrotor trajectory while building a 3D map. The video is provided as a real-time demonstration of the autonomous flight.

IX. CONCLUSIONS

In this paper, we describe an autonomous navigation system for a quadrotor based on our recently developed RGB-D visual odometry algorithm. We show how the use of the RGB-D camera as the only exteroceptive sensor enables 3D SLAM in autonomous flight in indoor environments. The powerful on-board computer is able to run all the components in real time. We also developed a 4DOF path planner whose functionality has been verified by simulation and it will be implemented on-board the quadrotor as future work. In addition we will tackle some challenging problems related to RGB-D sensors which have limited range and improve the visual odometry algorithm in featureless environment.

96

Fig. 8. Results of the real-time onboard SLAM experiment. Left: orthogonal projection of the recovered point cloud map. The path generated by the visual odometry and the corrected SLAM path are shown. Middle: side view of the point cloud map. Right: side view of the octomap.

REFERENCES

[1] Markus Achtelik, Michael Achtelik, Stephan Weiss, and Roland Siegwart. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 3056–3063, May 2011.

[2] Abraham Bachrach, Samuel Prentice, and Nicholas Roy. RANGE - Robust Autonomous Navigation in GPS-denied Environments. *Journal of Field Robotics*, 28:644–666, 2011.

[3] M Blosh, S Weiss, D Scaramuzza, and R Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 21–28, May 2010.

[4] Andrea Censi. An ICP variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25, Pasadena, CA, May 2008. Ieee.

[5] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J. Sean Humbert. Implementation of wide-field integration of optic flow forautonomous quadrotor navigation. *Autonomous Robots*, 27(3):189–198, August 2009.

[6] Ivan Dryanovski, Roberto G Valenti, and Jizhong Xiao. An Open-Source Navigation System for Micro Aerial Vehicles. *Autonomous Robots*, pages 1–12, 2013.

[7] Ivan Dryanovski, Roberto G Valenti, Jizhong Xiao, and Senior Member. Fast Visual Odometry and Mapping from RGB-D Data. In *IEEE Int. Conf. on Robotics and Automation*, volume 10031, 2013.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564. Ieee, October 2012.

[10] Ascending Technologies GmbH. http://www.asctec.de.

[11] S. Hrabar, G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3309–3316. Ieee, 2005.

[12] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[13] Andrew Johnson, James Montgomery, and Larry Matthies. Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain *. In *International Conference on Robotics and Automation*, number April, 2005.

[14] Michael Kassecker, By Teodor Tomi, Korbinian Schmid, Philipp Lutz, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Research Platform for Indoor and Outdoor Urban Search and Rescue. *Robotics & Automation Magazine, IEEE*, 19(SEPTEMBER):46–56, 2012.

[15] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, 2011.

[16] Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *in advances in neural information processing systems 16: proceeding of the 2003 conference (NIPS-03*. MIT Press, 2004.

[17] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, pages 1–19, 2012.

[18] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP09),*, pages 331–340, 2009.

[19] Jongho Park and Youdan Kim. Stereo Vision Based Collision Avoidance of Quadrotor UAV. In *International Conference on Control, Automation and Systems*, pages 173–178, 2012.

[20] Mihail Pivtoraiko and Alonzo Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3231–3237. IEEE, 2005.

[21] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, number Figure 1, 2009.

[22] Eduardo Rondon, Luis-Rodolfo Garcia-Carrillo, and Isabelle Fantoni. Vision-based altitude, position and speed regulation of a quadrotor rotorcraft. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 628–633. Ieee, October 2010.

[23] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[24] John Stowers, Michael Hayes, and Andrew Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. In *IEEE International Conference on Mechatronics*, pages 358–362. Ieee, April 2011.

[25] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, May 2010.

[26] Zhenyu Yu, Demian Celestino, and Kenzo Nonami. Development of 3D Vision Enabled Small-scale Autonomous Helicopter. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2912–2917, October 2006.

[27] Z Zamudio, R Lozano, J Torres, and V Rosas. Stereo vision for the stabilization of a quadrotor. In *System Theory, Control and Computing (ICSTCC), 2012 16th International Conference on*, pages 1–6, 2012.

[28] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. MAV navigation through indoor corridors using optical flow. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3361–3368. IEEE, 2010.

97

Predictive Exploration Considering Previously Mapped Environments

This appendix includes the full text for the following article. The article is part of the PhD by publication:

99

Predictive Exploration Considering Previously Mapped Environments

Daniel Perea Ström   Fabrizio Nenci   Cyrill Stachniss

Abstract—The ability to explore an unknown environment is an important prerequisite for building truly autonomous robots. The central decision that a robot needs to make when exploring an unknown environment is to select the next view point(s) for gathering observations. In this paper, we consider the problem of how to select view points that support the underlying mapping process. We propose a novel approach that makes predictions about the structure of the environments in the unexplored areas by relying on maps acquired previously. Our approach seeks to find similarities between the current surroundings of the robot and previously acquired maps stored in a database in order to predict how the environment may expand in the unknown areas. This allows us to predict potential future loop closures early. This knowledge is used in the view point selection to actively close loops and in this way reduce the uncertainty in the robot's belief. We implemented and tested the proposed approach. The experiments indicate that our method improves the ability of a robot to explore challenging environments and improves the quality of the resulting maps.
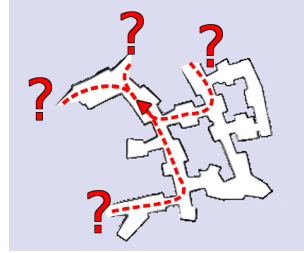
I. Introduction

Exploration is the task of selecting view points so that a robot can cover the environment with its sensors to build a map. Most exploring robots always start from scratch and do not use any background knowledge about the environment or typical environments. This may be seen as suboptimal as we humans also reason about typical structures even exploring an unknown environment.

While exploring the environment, a robot has to make decisions about where to go and which area to inspect in order to build a model of the environment, see Figure 1 for a small example. The decision of which place to visit can impact the underlying mapping system and can thus be critical for the quality of the resulting map. A typical approach to exploration is the frontier-based approach proposed by Yamauchi [28]. It identifies the frontiers between the free space and unknown areas and guides the robot to the closest one. This strategy typically yields short exploration paths but is generally unaware about the uncertainty in the robot belief, for example, about its current position in the world. Information-theoretic approaches such as [13], [2], [23], [22] consider the expected information gain to evaluate possible target locations. A key challenge, for example in [23], is to reason about the possible environment that the robot may experience when navigating through unknown environments.

Fig. 1. Mobile robot exploration has to answer the question: "Where to go next?". Our approach exploits previously mapped environments to predict potential future loop closures and thus to select better target locations.

In this paper, we take first steps towards exploiting background knowledge during autonomous exploration. The key idea is to use previously experienced environments to reason about what to find in the unknown parts of the world. Thus, we equip the robot with a database to store all acquired local maps and exploit this knowledge when selecting target locations. Our research is motivated by an autonomous exploration project for autonomously digitizing the Roman catacombs, which are complex underground environments with repetitive structures. To predict possible structures the robot may experience during exploration, we exploit previously visited areas and consider the similarities with the area around the current frontiers. This allows the robot to actively seek for loop-closures and in this way actively reduce its pose uncertainty. Our experiments indicate that this approach is beneficiary for robots when comparing it to a standard frontier-based method.

II. Related work

The majority of techniques for mobile robot exploration focus on generating motion commands that minimize the time needed to cover the whole terrain. Several techniques also assume that an accurate position estimate is available during exploration [10], [28]. Whaite and Ferrie [26] present an approach that uses the entropy to measure the uncertainty in the geometry of objects that are scanned with a laser range sensor. Similar techniques have been applied to mobile robots [21], [16], but such approaches still assume to know the correct pose of the vehicle. None of the approaches mentioned above takes the pose uncertainty into account when selecting the next vantage point. There are, however,

100

exploration approaches that have been shown to be robust against uncertainties in the pose estimates [5], [9].

Besides the idea of navigating to the next frontier [28], techniques based on stochastic differential equations for goal-directed exploration have been proposed by Shen *et al.* [19]. Similar to that, constrained partial differential equations that provide a scalar field into unknown areas have been presented by Shade et al. [18]. An information-theoretic formulation that seeks to minimize the uncertainty in the belief about the map and the trajectory of the robot has been proposed by Stachniss *et al.* [23]. This approach builds upon the works of Makarenko *et al.* [13] and Bourgault *et al.* [2]. Both extract landmarks out of laser range scans and use an Extended Kalman Filter to solve the underlying SLAM problem. They furthermore introduce an utility function which trades-off the cost of exploring new terrain with the potential reduction of uncertainty by measuring at selected positions. A similar technique has been presented by Sim *et al.* [20], who consider actions to guide the robot back to a known place in order to reduce the pose uncertainty of the vehicle.

In general, the computation of the expected entropy reductions is a complex problem, see Krause and Guestrin [11], and in all real world systems, approximations are needed. Suitable approximations often depend on the environment model, the sensor data, and the application. In some cases, efficient approximations can be found, for example in the context of monitoring lakes using autonomous boats [7].

Other approaches, especially in the context of autonomous micro aerial vehicles (MAVs), seek to estimate the expected feature density in the environment in order to plan a path through areas that support the helicopter localization [17]. This can be seen as related to information-theoretic approaches, although Sadat et al. [17] do not formulate their approach in this framework. A related approach to MAV exploration seeks to select new vantage points during exploration, so that the expected number of visible features is maximized, see Mostegel et al. [14].

An interesting approach by Fox *et al.* [6] aims at incorporating knowledge about *other* environments into a cooperative mapping and exploration system for multiple robots. This allows for predicting simplified laser scans of an unknown environment. This idea was an inspiration for our paper for predicting possible loop closures given the environment structure explored so far. We use this approach for exploring ancient catacombs, which are repetitive underground environments, with a mobile platform, see Figure 2. Chang *et al.* [3] propose an approach for predicting the environment using repetitive structures for SLAM. Other background knowledge about the environment, for example semantic information, can support the exploration process as shown by Wurm et al. [27], Stachniss *et al.* [24] as well as Holz *et al.* [8].

III. Environment Predictive Exploration

The central question in exploration is "Where to go?". Several different cost functions for making the decision of

Fig. 2. A robot for exploring and digitizing Roman catacombs was the motivation for our research.

where to go next can be defined. The most popular one goes back to Yamauchi [28], who guides the robot to the closest unexplored location. Yamauchi introduces the concept of frontiers, which are the cells of an occupancy grid map at the boundary between the free and the unexplored space. In the standard setting, this approach seeks to minimize the time that is needed to cover the environment with the robot's sensors and is a popular choice in mobile robotics.

A. Information-Driven Exploration

Given the fact that most real robots maintain a probabilistic belief about their pose and the map of the environment, an alternative approach is to select the target location that is expected to minimize the uncertainty in the belief of the robot. In this setting, the exploration problem can be formulated as follows. At each time step t, the robot has to decide which action a to execute, i.e., where to move next. During the execution of a, it is assumed that the robot obtains a sequence of observations z (for better readability, we neglect all time indices).

Thus, we can define the expected information gain, also called mutual information, of selecting the action a as the expected change in entropy in the belief about the robot's poses X and the map M:

$$I(X, M; Z^a) = H(M, X) - H(M, X \mid Z^a). \quad (1)$$

The second term in Eq. (1) is the conditional entropy and is defined as

$$H(M, X \mid Z^a) = \int p(z \mid a) H(M, X \mid Z^a = z) \, dz. \quad (2)$$

Unfortunately, reasoning about all potential observation sequences z in Eq. (2) is intractable in nearly all real world applications since the number of potential measurements grows exponentially with the dimension of the measurement space and with time. It is therefore crucial to approximate the integral of Eq. (2) so that it can be computed efficiently with sufficient accuracy.

A suitable approximation, however, depends on the environment model, the sensor data, and the application so that no general one-fits-all solution is available. Given our previous work [23], we considered different types of actions: First, *exploration actions* that guide the robot to the closest frontier and this reduces the map uncertainty. As we have no further information about the unseen area, it is difficult

101

query

Fig. 3.   Example of the submap retrieval using FabMAP2. The left image shows the query map, the other ones the best four matches from the database.

to distinguish two frontiers with respect to the expected uncertainty reduction. Second, *loop-closing and re-localization actions*, which are key to the uncertainty reduction about the robot's pose.

In this work, we aim at combining these types of actions into a single one. We seek to predict what the so far unseen environment beyond a frontier *may* look like, based on background knowledge of previously seen environments, and select the frontier that potentially leads to a loop-closure. In this way, we maximize the expected uncertainty reduction in the belief of the robot about the state of the world.

B. Utility Function for Exploration

Most exploration systems define a utility function to relate the expected gain in information with the cost of obtaining the information. As long as no constraints such as available energy or similar are considered, the distance that the robot has to travel to obtain its measurements is a standard choice. This yields a utility function of the form

$$U(a) = I(M, Z; Z^a) - \text{cost}(a) \qquad (3)$$

so that the task of selecting the best action can be formulated as

$$a^* = \underset{a}{\arg\max}\, I(M, Z; Z^a) - \text{cost}(a). \qquad (4)$$

Throughout this work, we define the cost function $\text{cost}(a)$ as the path length corresponding to action a, i.e. the length of the trajectory from the current location of the robot to the designated target location.

As mentioned in the previous section, estimating the expected information gain is challenging and computationally demanding and thus we use the following approximation. We assume that actions can reduce the robot's uncertainty about the map by exploring unseen areas and/or can reduce its uncertainty about the trajectory by closing a loop.

$$a^* = \underset{a}{\arg\max}\, I_{map}(a) + I_{traj}(a) - \text{cost}(a). \qquad (5)$$

As we do not know how large the unknown area and thus the number of unknown grid cells behind a frontier is, we may argue that all frontiers yield the same expected information gain with respect to the map uncertainty. Thus, we can simplify Eq. (5) as long as we consider only exploration actions to frontiers:

$$a^* = \underset{a}{\arg\max}\, I_{traj}(a) - \text{cost}(a). \qquad (6)$$

The expected information gain about the trajectory $I_{traj}(a)$ is mainly influenced by loop closures. The more likely a

loop closure can be obtained when executing an exploration action a, the higher its expected gain. Thus, the remainder of this section addresses the problem of predicting possible loop closures.

C. Predictive Exploration

The key contribution of this work is to model the predictive belief describing what the environment may look like in the unexplored areas. To compute this belief, the robot exploits environment structures it has seen in the past—either in the environment explored so far or even from previous mapping runs. Our exploration system uses this predictive belief to evaluate the frontiers as possible target locations for the exploration. This allows us to select the frontiers that are likely to lead to a loop-closure and thus to an active reduction of the uncertainty in the robot's belief. As we will show later during the experimental evaluation, our approach outperforms the traditional frontier-based exploration system.

D. Querying for Similar Environment Structures

The key idea of this belief is to look for similarities between the known areas around a frontier and portions of previously mapped environments. Under the assumption that environments are not random but expose certain structures and that these structures tend to appear more than once, we can use the already mapped areas in order to predict what the environment beyond the frontier may look like.

The first step is to look for portions of the already mapped environment(s) that are similar to the area around the frontier for which the prediction should be performed. To do this, we incrementally build a database storing all local grid maps. To perform a similarity query, we compare our local maps with the maps stored in the database. To avoid a large number of expensive map-to-map comparisons to search for similar submaps, we rely on a bag-of-words inspired approach, a technique that is frequently used in computer vision to search for image similarities. More concretely, we apply FabMAP2 by Cummins and Newman [4], an appearance based approach to efficiently query the database. Although FabMAP2 was originally designed to match camera images, it turns out that we can also use it to effectively search for local grid maps in a large database of maps. As FabMAP2 also provides a likelihood $l(m)$ for each match m, we can obtain a belief about possible environment structures. Figure 3 shows an illustration of this procedure. The image on the left is a query image and the other images are the top 4 matches reported by FabMAP2.

102
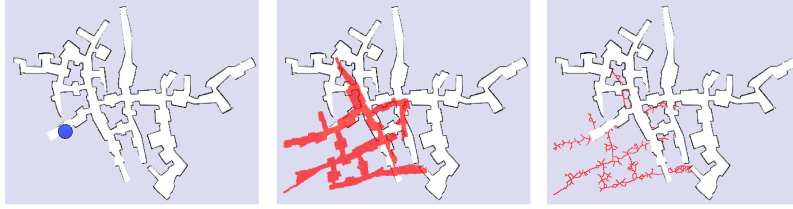
Fig. 4. Illustration of the loop closures prediction. Left: So far explored map with the frontier under consideration (blue circle). Middle: One map from the predictive belief (in red) superimposed on the map explored so far. Right: Voronoi diagram used for the path search.

E. Loop Closures Prediction

As we are mainly interested in the possible paths through the unknown environment in order to find loop closures and not necessarily the exact geometry, we reduce the maps reported by FabMAP2 to extended Voronoi graphs [1] and do all further computations on these graphs.

FabMAP2 provides us with candidates for matching maps but no geometric alignment between the query map and the reported ones. Thus, we align each map reported by FabMAP2 with our query map. This can be done in a robust manner through a RANSAC-based alignment of the Voronoi graphs using its junction points. Figure 4 shows an example of a Voronoi graph aligned with the map explored so far.

The next step, is to search for possible loop closures, for which we use the generalized Voronoi graph. Starting from the frontier point, we traverse the Voronoi graph in a breadth-first manner. During the traversal, we check if the Voronoi graph leads to a position that is close to any other frontier in the map built so far. If this is the case, we regard that as a possible loop closure. Such a situation is illustrated in the left image of Figure 5. This process is executed for each frontier.

F. Estimating the Probability of Closing a Loop

Each map reported by FabMAP2 comes with a likelihood. Thus, we can approximate the probability of closing a loop when executing an exploration action as

$$\mathbf{S}_f = \sum_{m \in \mathcal{M}(f)} l(m) \sum_{c \in \mathcal{C}(f,m)} l(c \mid m) \qquad (7)$$

Here, $\mathcal{M}(f)$ is the set of matches returned by FabMAP2 when querying with the frontier f, and $l(m)$ the likelihood of a match m. The term $\mathcal{C}(f,m)$ refers to the set of possible loop closures computed according to the breadth-first traversal explained above and $l(c \mid m)$ is the likelihood that the loop closures can be reached. We assume that $l(c \mid m)$ is proportional to the inverse length of the path of the predicted loop closure. This means that short loop closures are more likely than long ones.

Assuming that every executed loop closure through unknown areas of the map yields the same expected uncertainty reduction, we can approximate the expected information gain

I_{traj} of Eq. (6) with the score $\mathbf{S}_f$ according to Eq. (7). This is clearly a strong assumption but we argue that a high score indicates a high expected gain from exploring the frontier.

IV. EXPERIMENTS

The experiments are designed to illustrate the advantages of our predictive exploration approach. We show that our approach selects frontiers that lead to loop closures which in turn result in improved maps of the environment. As a baseline, we use a standard frontier-based exploration approach.

The underlying mapping framework that is used for all experiments is a state-of-the-art graph-based SLAM system that relies on laser range data. The backend is g2o [12] and the frontend uses FLIRT features to search for possible data associations [25], uses correlative scan matching to align scans, and applies single cluster graph partitioning to resolve ambiguities as proposed by Olson [15]. The exploration system is integrated into the mapping framework and has been implemented using ROS.

A. Map Consistency

First, we compare the quality of the maps obtained with frontier-based exploration vs. our predictive exploration. The environments considered here are parts of the Roman catacomb St. Priscilla, a difficult to traverse and large-scale underground environment in Rome. As the robot is equipped with tracks, see Figure 2, its odometry is in general worse than the one of a wheeled robot and it often reveals a bias to one side. This experiment has been conducted in simulation but the environment actually represents the catacomb, with each experiment covering an exploration area of 2,500 m². Odometry noise is simulated following a probabilistic motion model, sampling over normal distributions for each motion parameter, with bias for the rotation distribution.

Figure 6 illustrates the obtained results for two environments using exactly the same mapping system and identical parameters for the comparison. The images on the left are the ground truth 2D map used for the simulation. The images in the second column correspond to the results of the frontier-based exploration, while the images on the right show our approach. As can be seen already visually, our approach yielded a consistent model of the environment, while the

103

Fig. 5. Illustration of the active loop closing. Left: prediction of the possible path with the loop closure shown in blue. Middle: the robot explores the path along the predicted loop closure and perceives the actual structure of the scene. The graph in the already explored environment shows the pose graph of the SLAM system. Right: successful loop closure Please note that the predicted environment is actually not identical with the real environment but reveals a similar structure. This similarity resulted in the shown loop closure.

Fig. 6. Two performance comparisons in constant odometry bias scenario. On the left, the original map. In the middle, the closer frontier approach. On the right, our prediction-based approach. Note that the nearest frontier approach produces a map that is non consistent with the original one, so that the robot gets actually lost in it. The map produced by the prediction-based approach is instead consistent with the original one.

frontier-based approach failed. Using the frontier-based approach the robot was unable to continue its exploration task due to an inconsistent map that prevented the computation of further exploration actions. This was the case in all six exploration experiments that we conducted in St. Priscilla environment.

B. Path Length

The advantages of the prediction-based approach come at a cost—the cost of traversing exploration paths that are longer than the ones generated by the frontier-based approach. This experiment is designed to evaluate the increase in path length.

As we are not able to obtain consistent maps for the frontier-based approach under a realistic noise model for the task under consideration, we set the noise to zero in the simulator and repeated the previous experiments. Using a zero noise odometry, also the frontier-based approach is able to build consistent maps. In this settings there is no advantage in using our predictive approach as the pose uncertainty is zero

104
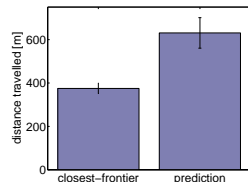
Fig. 7. Mean and standard deviation of the distances travelled in the frontier-based approach and in the proposed approach.

and no uncertainty reduction is gained from closing loops. We compared the distances travelled for the frontier-based and our approach. The distances travelled are summarized in Figure 7. In the worst case scenario, the path generated by our approach was 1.85 times longer than the one of the frontier-based approach. The minimum increase was a factor of 1.5. Generating on average approximatively a 1.7 times longer trajectory is clearly an overhead—for actively closing loops and in this way reducing uncertainty, however, this price must be paid.

V. Conclusions

In this paper, we proposed a novel approach for autonomous exploration of unknown environments. The key contribution of this work is a technique to predict possible environment structures in the unseen parts of the robot's surroundings based on previously explored environments. In our approach, we exploit this belief to predict possible loop closures that the robot may experience when exploring an unknown part of the scene. This allows the robot to actively reduce the uncertainty in its belief through its exploration actions. We implemented and tested our approach. Our experiments illustrate that our technique allows for an effective exploration of difficult to map environments. By actively closing loops, we are able to obtain consistent maps of the environment. In contrast to that, a traditional frontier-based exploration approach is not able to successfully explore the scene.

References

[1] P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.
[2] F. Bourgoult, A.A. Makarenko, S.B. Williams, B. Grocholsky, and F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.
[3] H.J. Chang, C.S.G. Lee, Y. Lu, and Y.C. Hu. P-slam: Simultaneous localization and mapping with environmental-structure prediction. *IEEE Transactions on Robotics*, 23(2):281–293, 2007.
[4] M. Cummins and P. Newman. Highly scalable appearance-only slam fab-map 2.0. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.
[5] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Journal of Autonomous Robots*, 12(3):287 – 300, 2002.
[6] D. Fox, J. Ko, K. Konolige, and B. Stewart. A hierarchical bayesian approach to the revisiting problem in mobile robot map building. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2003.
[7] G. Hitz, A. Gotovos, F. Pomerleau, M.-E. Garneau, C. Pradalier, A. Krause, and R.Y. Siegwart. Fully autonomous focused exploration for robotic environmental monitoring. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.
[8] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. A comparative evaluation of exploration strategies and heuristics to improve them. In *Proc. of the European Conference on Mobile Robots (ECMR)*, Oerebro, Sweden, 2011.
[9] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.
[10] S. Koenig and C. Tovey. Improved analysis of greedy mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.
[11] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, 2005.
[12] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613, 2011.
[13] A.A. Makarenko, S.B. Williams, F. Bourgoult, and F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.
[14] C. Mostegel, A. Wendel, and H. Bischof. Active monocular localization: Towards autonomous monocular exploration for multirotor mavs. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.
[15] E. Olson. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172, 2009.
[16] R. Rocha, J. Dias, and A. Carvalho. Exploring information theory for vision-based volumetric mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2409–2414, Edmonton, Canada, 2005.
[17] S.A. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, and R. Vaughan. Feature-rich path planning for robust navigation of mavs with mono-slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Hong Kong, China, 2014.
[18] R. Shade and P. Newman. Choosing where to go: Complete 3d exploration with stereo. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.
[19] S. Shen, N. Michael, and V. Kumar. 3d indoor exploration with a computationally constrained mav. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.
[20] R. Sim, G. Dudek, and N. Roy. Online control policy optimization for minimizing map uncertainty during exploration. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.
[21] C. Stachniss and W. Burgard. Mapping and exploration with mobile robots using coverage maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 476–481, 2003.
[22] C. Stachniss and W. Burgard. Particle filters for robot navigation. 3(4):211–282, 2012. Published 2014.
[23] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.
[24] C. Stachniss, O. Martinez Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52:205ff, 2009.
[25] G.D. Tipaldi and K.O. Arras. Flirt–interest regions for 2d range data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3616–3622, 2010.
[26] P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.
[27] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.
[28] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Second International Conference on Autonomous Agents*, pages 47–53, Minneapolis, MN, USA, 1998.

105

Robust Exploration and Homing for Autonomous Robots

This appendix includes the full text for the following article. The article is part of the PhD by publication:

107

Robust Exploration and Homing for Autonomous Robots

Daniel Perea Ström[a], Igor Bogoslavskyi[b], Cyrill Stachniss[b,*]

[a]*Universidad de La Laguna, Departamento de Ingeniería Informática, Av. Francisco Sánchez, 38206 Tenerife, Spain.*
[b]*University of Bonn, Institute for Geodesy and Geoinformation, Nussallee 15, 53115 Bonn, Germany*

Abstract

The ability to explore an unknown environment is an important prerequisite for building truly autonomous robots. Two central capabilities for autonomous exploration are the selection of the next view point(s) for gathering new observations and robust navigation. In this paper, we propose a novel exploration strategy that exploits background knowledge by considering previously seen environments to make better exploration decisions. We furthermore combine this approach with robust homing so that the robot can navigate back to its starting location even if the mapping system fails and does not produce a consistent map. We implemented the proposed approach in ROS and thoroughly evaluated it. The experiments indicate that our method improves the ability of a robot to explore challenging environments as well as the quality of the resulting maps. Furthermore, the robot is able to navigate back home, even if it cannot rely on its map.

Keywords: Exploration, Background knowledge, Homing, Navigation, SLAM

1. Introduction

Exploration is the task of selecting view points so that a robot can cover the environment with its sensors to build a map. The ability to robustly operate without user intervention is an important capability for exploration robots, especially if there is no means for communication between the robot and an operator. Most exploration robots always start assuming zero knowledge and do not exploit any background knowledge about the environment or typical environments. They build a map of the environment online and make all navigation decisions based on this map. As long as this map is consistent, the robot can perform autonomous navigation by planning the shortest path—for example using A*—from its current location to its next vantage point using the map. Although recent SLAM systems are fairly robust, there is a chance that they fail, for example, due to wrong data associations generated by the front-end. Even current state-of-the-art SLAM approaches cannot guarantee the consistency of the resulting map. Computing a path based on an inconsistent map, however, is likely to lead to a failure and possibly to loosing the robot if operating in a hazardous environment. Thus, exploring robots should always decide where to go next and at the same time verify if their map is still consistent (see sketch in Fig. 1). Considering existing approaches, however, it is fair to say that most exploration systems follow the paradigm that they (a) make their navigation and exploration decisions using the current map only and (b) assume that the map is consistent and thus can be used as the basis for path planning and navigation.
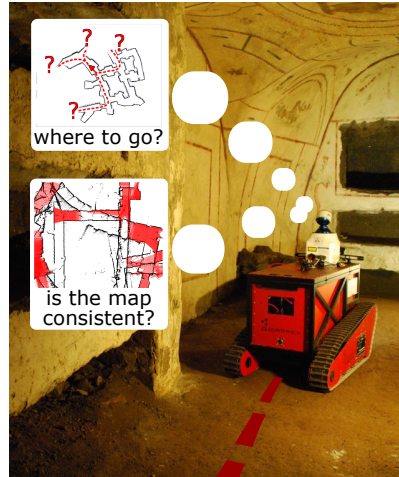
Figure 1: Mobile robot exploration has to answer the question: "Where to go next?". Our approach exploits previously mapped environments to predict potential future loop closures and thus to select better target locations. When the statistical map consistency tester provides the robot with the information that the map is not consistent anymore the robot starts rewinding the trajectory using our robust homing method.

---

*Corresponding author
Email addresses:* `dani@isaatc.ull.es` (Daniel Perea Ström), `igor.bogoslavskyi@uni-bonn.de` (Igor Bogoslavskyi), `cyrill.stachniss@igg.uni-bonn.de` (Cyrill Stachniss)

108

In this paper, we aim at relaxing these assumptions. The key idea is to consider the knowledge gained from previously conducted exploration missions to support the navigation system of the robot. This is motivated by the fact that selecting appropriate target locations during exploration supports the mapping process, and can increase the probability of building a consistent map. Furthermore, we want to be able to safely navigate our robot back to its starting location, even if the mapping process failed.

The first contribution of this paper is a novel approach to exploiting background knowledge while generating exploration behaviors to support mapping. The key idea is to use previously experienced environments to reason about what to find in the unknown parts of the world. To achieve this, we equip our robot with a database to store all acquired (local) maps and exploit this knowledge when selecting target locations. Our research is motivated by an exploration project for autonomously digitizing the Roman catacombs, which are complex underground environments with repetitive structures. To predict possible geometries of the environment the robot may experience during exploration, we exploit previously visited areas and consider the similarities with the area around the currently planned next view point. This allows the robot to actively seek for loop-closures and in this way actively reduce its pose uncertainty. Our experiments indicate that this approach is beneficiary for robots when comparing it to a standard frontier-based exploration method.

The second contribution is a robot homing approach with the goal of retrieving our robot even if the SLAM system failed to build a consistent map. To avoid that our robot gets lost, we propose a robust homing system consisting of two distinct parts. Part A performs a statistical analysis of the map and thus provides the information about its consistency. We build upon our previous work [1] for performing a cascade of pair-wise consistency checks using the observations perceiving the same areas. To avoid performing such checks on the overall map, we reduce the area to analyze by planning the shortest homing route for the robot assuming a consistent map. We then analyze the map consistency only along that path and can estimate on the fly if the map around this path is consistent or not with a given confidence level. If it is consistent, we navigate back on the verified homing path. Part B of our approach is responsible for driving the robot back to its starting location without a map. We achieve this by rewinding the trajectory that the robot took to reach its current pose. If the motions of the robot were perfect, i.e. would lead to the desired robot pose in the world frame, we would be able to simply invert the motion commands performed by the robot and could safely reach the starting location. Motion execution and odometry, however, are often noisy. As a result, simply following inverse motion commands will not bring the robot to the starting location in the real world in most cases. Therefore, we take into account the sensor information to guide the robot back by matching the observations with the past.

2. Related Work

The majority of techniques for mobile robot exploration focus on generating motion commands that minimize the time needed to cover the whole terrain. Several techniques also assume that an accurate position estimate is available during exploration [2, 3]. Whaite and Ferrie [4] present an approach that uses the entropy to measure the uncertainty in the geometry of objects that are scanned with a laser range sensor. Similar techniques have been applied to mobile robots [5, 6], but such approaches still assume to know the correct pose of the vehicle. Such approaches take the map but not the pose uncertainty into account when selecting the next vantage point. There are, however, exploration approaches that have been shown to be robust against uncertainties in the pose estimates [7, 8].

Besides the idea of navigating to the next frontier [3], techniques based on stochastic differential equations for goal-directed exploration have been proposed by Shen *et al.* [9]. Similar to that, constrained partial differential equations that provide a scalar field into unknown areas have been presented by Shade *et al.* [10]. An information-theoretic formulation that seeks to minimize the uncertainty in the belief about the map and the trajectory of the robot has been proposed by Stachniss *et al.* [11]. This approach builds upon the works of Makarenko *et al.* [12] and Bourgault *et al.* [13]. Both extract landmarks out of laser range scans and use an Extended Kalman Filter to solve the underlying SLAM problem. They furthermore introduce a utility function which trades-off the cost of exploring new terrain with the potential reduction of uncertainty by measuring at selected positions. A similar technique has been presented by Sim *et al.* [14], who consider actions to guide the robot back to a known place in order to reduce the pose uncertainty of the vehicle. Such information-driven techniques have also been used for perception selection to limit the complexity of the underlying optimization problems in SLAM [15].

In general, the computation of the expected entropy reductions is a complex problem, see Krause and Guestrin [16], and in all real world systems, approximations are needed. Suitable approximations often depend on the environment model, the sensor data, and the application. In some cases, efficient approximations can be found, for example in the context of monitoring lakes using autonomous boats [17].

Other approaches, especially in the context of autonomous micro aerial vehicles (MAVs), seek to estimate the expected feature density in the environment in order to plan a path through areas that support the helicopter localization [18]. This can be seen as related to information-theoretic approaches, although Sadat *et al.* [18] do not formulate their approach in this framework. A related approach to MAV exploration seeks to select new vantage points during exploration, so that the expected number of visible features is maximized, see Mostegel *et al.* [19].

An interesting approach by Fox *et al.* [20] aims at incorporating knowledge about *other* environments into a cooperative mapping and exploration system for multiple robots. This allows for predicting simplified laser scans of an unknown environment. This idea was an inspiration for our paper for predicting possible loop closures given the environment structure

2

explored so far. We use this approach for exploring ancient catacombs, which are repetitive underground environments, with a mobile platform, see Fig. 1. Chang *et al.* [21] propose an approach for predicting the environment using repetitive structures for SLAM. Other background knowledge about the environment, for example semantic information [22], can support the exploration process as shown by Wurm *et al.* [23], Stachniss *et al.* [24] as well as Holz *et al.* [25].

A central problem in robust exploration, however, is that in case of a SLAM failure, the map becomes inconsistent. This can prevent the robot from continuing its exploration mission and—even worse—from being able to navigate back. It is therefore important to be able to perform reliable navigation without relying on a map.

Sprunk *et al.* [26] present a lidar-based teach-and-repeat method to follow a route given by the user. The approach relies on precise localization of the robot based on the lidar measurements with respect to a taught-in trajectory. Similarly, Furgale *et al.* [27] perform the ICP-based teach-and-repeat approach on an autonomous robot equipped with a high precision 3D spinning lidar. They extend the standard teach-and-repeat approach by adding a local motion planner to account for dynamic changes in the environment. Our method to rewind the trajectory is similar to the teach-and-repeat setup in this formulation. However, in contrast to the mentioned methods, we use a substantially less accurate robot and thus have to cope with somewhat larger deviation from the reference trajectory.

Vision methods are also popular for teach-and-repeat approaches. Furgale *et al.* [28] present a vision-based approach to teach-and-repeat for long range rover autonomy. During a learning phase, their system builds a manifold map of overlapping submaps as the rover is piloted along a route. The map is then used for localization as the rover repeats the route autonomously. They present an autonomous planetary rover that is able to navigate even non-planar terrain without relying on an accurate global reconstruction. Nitsche *et al.* [29] extend a standard teach-and-repeat approach by adding Monte Carlo localization to localize the robot with respect to the learned path. They present vision-based tests carried out both on a ground robot and an aerial drone. Battesti *et al.* [30] present an online localization approach. They use visual loop-closure techniques to create consistent topo-metric maps in real-time while the robot is teleoperated and localizes itself in such maps. This allows the robot to follow the predicted path successfully compensating the odometry drift. These visual methods, however, need substantial adaptation in order to be used in a setup similar to ours: using monocular cameras to localize through feature detection relies on having enough visual information, which is not the case in the typically dark catacombs. The work presented here is based on a conference publication [31], which described the idea of predictive exploration.

3. Robot and Sensor Setup

Our robot is a customized Mesa Element platform, see Fig. 1. It is equipped with a laser range finder scanning in a horizontal

2D plane around 60cm above the ground. The robot is additionally equipped with two ASUS Xtion depth cameras that observe the local area in front of the robot in 3D. Both cameras look forward, one slightly rotated to the left and the other one to the right with a minimal overlap in the middle. Our system relies on the 2D information for solving the exploration task in order to decide which parts of the scene have been explored, and where to move next. For the robust homing presented in Sec. 5, we take into account the 3D depth images from the Xtions as this allows for a more accurate alignment of the scans. Furthermore, a local traversability analysis is done in 3D based on the Xtions [32].

4. Environment Predictive Exploration

The central question in exploration is "Where to go?". Several different cost functions for making the decision of where to go next can be defined. The most popular one goes back to Yamauchi [3], who guides the robot to the closest reachable unexplored location. Yamauchi introduces the concept of frontiers, which are the cells of an occupancy grid map at the boundary between the free and the unexplored space. In the standard setting, this approach seeks to minimize the time that is needed to cover the environment with the robot's sensors and is a popular choice in mobile robotics. On the other hand, exploring hazardous environments requires trading time for a more robust navigation that supports the mapping system and avoids pose uncertainty.

4.1. Information-Driven Exploration

Given the fact that most real robots maintain a probabilistic belief about their pose and the map of the environment, an alternative approach is to select the target location that is expected to minimize the uncertainty in the belief of the robot. In this setting, the exploration problem can be formulated as follows. At each time step t, the robot has to decide which action a to execute (where to move next). During the execution of a, the robot obtains a sequence of observations z (for better readability, we neglect all time indices).Thus, we can define the expected information gain, also called mutual information, of selecting the action a as the expected change in entropy in the belief about the robot's poses X and the map M:

$$I(X, M; Z^a) = H(M, X) - H(M, X \mid Z^a). \quad (1)$$

The second term in Eq. (1) is the conditional entropy and is defined as

$$H(M, X \mid Z^a) = \int p(z \mid a) H(M, X \mid Z^a = z) \, dz. \quad (2)$$

Unfortunately, reasoning about all potential observation sequences z in Eq. (2) is intractable in nearly all real world applications since the number of potential measurements grows exponentially with the dimension of the measurement space and with time. It is therefore crucial to approximate the integral

3

110

Figure 2: Example of the submap retrieval using FabMAP2. The left image shows the query map, the other ones the best four matches from the database.

of Eq. (2) so that it can be computed efficiently with sufficient accuracy.

A suitable approximation, however, depends on the environment model, the sensor data, and the application so that no general one-fits-all solution is available. In our previous work [11], we considered different types of actions: First, *exploration actions* that guide the robot to the closest frontier and reduce the map uncertainty. As we have no further information about the unseen area, it is difficult to distinguish two frontiers with respect to the expected uncertainty reduction. Second, *loop-closing and re-localization actions*, which are key to the uncertainty reduction about the robot's pose.

In this work, we aim at combining these types of actions into a single one. We seek to predict what the so far unseen environment beyond a frontier *may* look like based on background knowledge of previously seen environments and select the frontier that potentially leads to a loop-closure. In this way, we maximize the expected uncertainty reduction in the belief of the robot about the world.

4.2. Utility Function for Exploration

Most exploration systems define a utility function to relate the expected gain in information with the cost of obtaining the information. As long as no constraints such as available energy or similar are considered, the distance that the robot has to travel to obtain its measurements is a standard choice. This yields a utility function of the form

$$U(a) = I(M, Z; Z^a) - \text{cost}(a), \tag{3}$$

so that the task of selecting the best action can be formulated as

$$a^* = \operatorname*{argmax}_a I(M, Z; Z^a) - \text{cost}(a). \tag{4}$$

Throughout this work, we define the cost function $\text{cost}(a)$ as the path length corresponding to action a, i.e. the length of the trajectory from the current location of the robot to the designated target location.

As mentioned in the previous section, estimating the expected information gain is challenging and computationally demanding and thus we use the following approximation. We assume that actions can reduce the robot's uncertainty about the map by exploring unseen areas and/or can reduce its uncertainty about the trajectory by closing a loop:

$$a^* = \operatorname*{argmax}_a I_{map}(a) + I_{traj}(a) - \text{cost}(a). \tag{5}$$

As we do not know how large the unknown area and thus the number of unknown grid cells behind a frontier is, we may argue that all frontiers yield the same expected information gain with respect to the map uncertainty. Thus, we can simplify Eq. (5) as long as we consider only exploration actions to frontiers:

$$a^* = \operatorname*{argmax}_a I_{traj}(a) - \text{cost}(a). \tag{6}$$

The expected information gain about the trajectory $I_{traj}(a)$ is mainly influenced by loop closures. The more likely a loop closure can be obtained when executing an exploration action a, the higher its expected gain. Thus, the remainder of this section addresses the problem of predicting possible loop closures.

4.3. Predictive Exploration

The key contribution here is to model the predictive belief describing what the environment may look like in the unexplored areas. To compute this belief, the robot exploits environment structures it has seen in the past—either in the environment explored so far or even from previous missions. Our exploration system uses this predictive belief to evaluate the frontiers as possible target locations for the exploration. This allows us to select the frontiers that are likely to lead to a loop-closure and thus to an active reduction of the uncertainty in the robot's belief. As we show during the experimental evaluation, this approach outperforms the traditional frontier-based exploration system.

4.4. Querying for Similar Environment Structures

The key idea of this approach is to look for similarities between the known areas around a frontier and portions of previously mapped environments. Under the assumption that environments are not random but expose certain structures and that these structures tend to appear more than once, we can use the already mapped areas in order to predict what the environment beyond the frontier *may* look like.

The first step is to look for portions of the already mapped environments that are similar to the area around the frontier for which the prediction should be performed. To do this, we incrementally build a database storing all local grid maps that the robot experienced. To perform a similarity query, we compare our local maps with the maps stored in the database. To avoid a large number of expensive map-to-map comparisons to search for similar submaps, we rely on a bag-of-words inspired approach, a technique that is frequently used in computer vision to search for image similarities. More concretely, we apply FabMAP2 by Cummins and Newman [33], an appearance-based approach we can use to efficiently query our database.

4

111
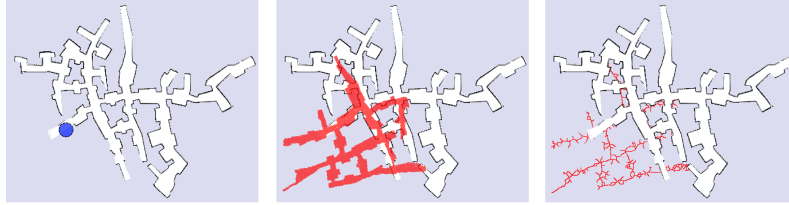
Figure 3: Illustration of the loop closures prediction. Left: So far explored map with the frontier under consideration (blue circle). Middle: One map from the predictive belief (in red) superimposed on the map explored so far. Right: Voronoi diagram used for the path search.
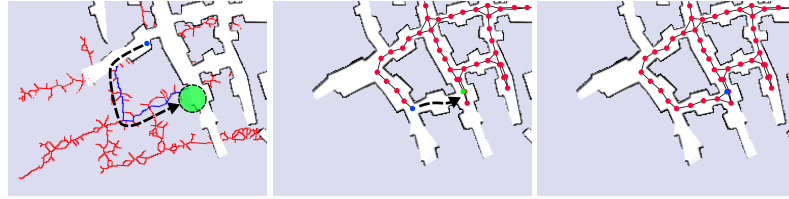
Figure 4: Illustration of the active loop closing. Left: prediction of the possible path with the loop closure shown in blue. Middle: the robot explores the path along the predicted loop closure and perceives the actual structure of the scene. The graph in the already explored environment shows the pose graph of the SLAM system. Right: successful loop closure Please note that the predicted environment is actually not identical with the real environment but reveals a similar structure. This similarity resulted in the shown loop closure.

Although FabMAP2 was originally designed to match camera images, it turns out that we can also use it to effectively search for local grid maps in a large database of maps. As FabMAP2 also provides a likelihood $l(m)$ for each match m, we can obtain a belief about possible environment structures. Fig. 2 shows an illustration of this procedure. The image on the left is a query image and the other images are the top 4 matches reported by FabMAP2.

4.5. Loop Closures Prediction

As we are mainly interested in the possible paths through the unknown environment in order to find loop closures and not necessarily the exact geometry, we reduce the maps reported by FabMAP2 to extended Voronoi graphs [34] and do all further computations on these graphs.

FabMAP2 provides us with candidates of matching maps but no geometric alignment between the query map and the reported ones. Thus, we align each map reported by FabMAP2 with our query map. This can be done in a robust manner through a RANSAC-based alignment of the Voronoi graphs using its junction points. Fig. 3 shows an example of a Voronoi graph aligned with the map explored so far.

The next step, is to search for possible loop closures, for which we use the extended Voronoi graph. Starting from the frontier point, we traverse the Voronoi graph in a breadth-first manner. During the traversal, we check if the Voronoi graph leads to a position that is close to any other frontier in the map built so far. If this is the case, we regard that as a possible loop closure. Such a situation is illustrated in the left image of Fig. 4. This process is executed for each frontier.

4.6. Estimating the Probability of Closing a Loop

Each map reported by FabMAP2 comes with a likelihood. Thus, we can approximate the probability of closing a loop when executing an exploration action as

$$\mathbf{S}_f = \sum_{m \in \mathcal{M}(f)} l(m) \sum_{c \in \mathcal{C}(f,m)} l(c \mid m) \qquad (7)$$

Here, $\mathcal{M}(f)$ is the set of matches returned by FabMAP2 when querying with the frontier f, and $l(m)$ the likelihood of a match m. The term $\mathcal{C}(f,m)$ refers to the set of possible loop closures computed according to the breadth-first traversal explained above and $l(c \mid m)$ is the likelihood that the loop closures can be reached. We assume that $l(c \mid m)$ is proportional to the inverse length of the path of the predicted loop closure. This means that short loop closures are more likely than long ones.

Assuming that every executed loop closure through unknown areas of the map yields the same expected uncertainty reduction, we can approximate the expected information gain I_{traj} of Eq. (6) with the score $\mathbf{S}_f$ according to Eq. (7). This is clearly a strong assumption but we argue that a high score indicates a high expected gain from exploring the frontier.

5

112

5. Robust Homing Using Map Consistency Checks

Under the assumption that we can ensure the consistency of the current map, homing is a comparably easy task. It basically consists of computing a collision-free path from the current location to the starting location and following the planned path with a standard navigation pipeline. Such a navigation system would, for example, localize the robot in the map built so far and plan the shortest path towards home using A* or a similar approach. If the map, however, is not consistent because the underlying SLAM system has failed, this approach is likely to lead to a deadlock situations from which the robot cannot escape easily.

To ensure a robust exploration of the environment, we address the problem of robust homing in a two-stage approach. First, while mapping the environment, a path is computed from the current location towards home assuming the map is consistent. Then, we perform the recently proposed map consistency estimation approach by Mazuran *et al.* [1] to evaluate if the map is consistent with a given confidence level. If the path towards home is consistent, and we finished exploring the environment, we simply execute this plan. If the path towards home is not consistent, we aim at reversing the trajectory of the robot taken so far by aligning the current observation with the observations obtained on the way *from the starting location* to the current one. This yields a robust strategy to bring a robot home to its starting location.

5.1. Map Consistency Test

Our map consistency estimation approach proposed previously in [1] builds upon a pose-graph representation, i.e., the location of the robot from which individual observations have been taken. We start with evaluating the consistency of pairs of range readings. The approach of Mazuran *et al.* describes the discrepancy between two range scans by computing how much the two scans occlude each others free space.

To estimate the occlusion of the free space, we compute for each scan the polygon of the robot's pose and all end points of the range scan. Such polygons define the free space covered by the scan taken from the robot's pose. The intuition is that both scans are consistent with each other if none of the end points of the first scan lies inside the polygon of the second one and vice versa. In [1], we define an inconsistency distance $d(p)$ for a point p, which lies inside the polygon of another scan, as the Euclidean distance of a point p to the closest point on the polygon boundary of the other scan. Intuitively speaking, for a consistent map, we assume that the inconsistency distances $d(p)$ are in line with the sensor noise of the proximity sensor. Substantially larger values for $d(p)$ may indicate that the scans are not properly aligned and the map may be inconsistent in local neighborhood of the scans.

More concretely, we can expect that, under the assumption of a correct alignment of two scans, on average 50% of the end points from the first scan have an inconsistency distance $d(p) > 0$ in the second scan and vice versa. This is due to the sensor noise in the range measurements. According to [1], we can formulate a statistical test for the sum of inconsistency
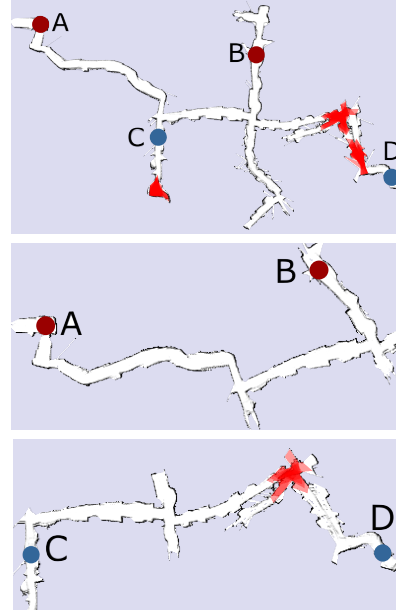
Figure 5: The top image shows the map built so far with the detected inconsistencies (inconsistent scans are shown in red). The middle one shows a submap that is built using only the scans recorded around the A* path from A to B computed in the full map. In this example, no inconsistencies are present and none are detected. The bottom image is done in the same way as the middle one, but the A* path is computed from C to D and here, the map inconsistencies are correctly detected.

distances $d(p)$. This test evaluates if pairs of scans are consistent given the sensor noise or reveal a larger error and thus an inconsistency.

To assess global map consistency, we could conduct this test for all pairs of scans and consider a map to be consistent if all tests are successful. The problem, however, is that a single statistical test will produce the wrong result with probability α. Thus, if we test a single scan, which overlaps with r other scans, this yields a type I error probability of $1 - (1-\alpha)^r$ and thus renders the direct application of the pairwise approach unsuitable. The key trick is to model the outcome of the pairwise hypothesis test as a Bernoulli-distributed random variable with parameter α. As a result of that, the number of failed tests follows a binomial distribution with parameters α and r. Given that, we can compute the maximum number $\hat{\xi}$ of tests that are

6

113

allowed to fail at a confidence level $1 - \alpha'$ as

$$\hat{\xi} = \min_{0 \leq \xi \leq r} \left\{ \xi \; \middle| \; \sum_{i=\xi+1}^{r} \binom{r}{i} \alpha^i (1-\alpha)^{r-i} \leq \alpha' \right\}. \quad (8)$$

This allows for computing a cascaded hypothesis test for all overlapping scans: We perform all pairwise hypothesis tests. If the number of failed tests is smaller than $\hat{\xi}$, the overall consistency test is positive otherwise negative. For more details, we refer the reader to [1].

5.2. Map Consistency Estimate for Finding the Way Home

Given the consistency test presented above, we can perform a mathematically sound statistical test to evaluate if a map is consistent or not. However, what the robot really needs to know is not the consistency of the full map. Instead, it is sufficient to know if it can safely move along a specific path through the environment to the starting location. Thus, we plan a path with A* assuming that the current map is consistent and extend our previous statistical consistency check to consider only the scans along that path. To achieve this, we select all recording locations that were closer than twice the maximum sensor range away from the trajectory planned with A*. Examples of such partial maps are depicted in Fig. 5. The top image shows an inconsistent 2D map of the Priscilla catacombs. Directly applying the approach described in [1] would label the whole map as inconsistent. In contrast to that, if the robot only takes into account the shortest route from A to B, he can still safely perform the navigation task, as shown in the middle image of the same figure. This is not the case if the robots wants to go from C to D as he will encounter an inconsistent part of the map on its way.

In terms of the persistent data structure that is used to store all the information, we use a generalization of a pose graph. Each node in the graph corresponds to a pose of the robot at time t. In addition to that, each node stores the original odometry pose X_t and the corresponding 3D point cloud c_t as well as the 2D scan. To efficiently represent this, the pose graph with the nodes X_t itself is kept in memory but the corresponding point clouds c_t are stored on disk and are loaded on demand.

5.3. Robust Homing by Rewinding the Trajectory

Once the consistency check has identified that the submap including the path is inconsistent, we need to perform the trajectory rewinding to bring the robot home safely. At this moment the robot does not have a consistent map, so it needs to rewind the trajectory without relying on a map. We assume that the environment remains static while the robot performs homing which is typically the case for the underground environments we explore. We can view the robot's forward trajectory as a series of 3D poses of the robot $\{X_0, \ldots, X_n\}$. The task of rewinding the trajectory is to drive the robot from X_n to X_0 while correcting for the error in odometry. The correction of the odometry error is done by aligning the point clouds obtained while performing trajectory rewinding with the ones corresponding to poses from X_n to X_0. Note that we subsample the trajectory in such way that each pose X_i is either $1\,m$
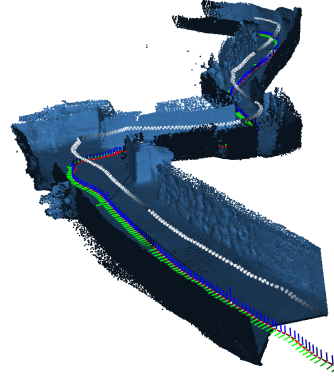
Figure 6: Partial view of the 3D model of the environment of the Priscilla catacombs built from two ASUS Xtion cameras.

away from the previous one or that there is a difference of at least $10°$ in yaw between these two poses.

Without loss of generality, let us consider that the robot has to carry out the action to move from X_i to X_j and to compensate for the error in odometry. To do that, the robot exploits the current point cloud c_{current} obtained after executing the movement from X_i to X_j. In an ideal world, the command should have brought the robot to the pose X_j. In reality, there is an error introduced by slippage, uneven ground etc. Thus, we align c_{current} with c_j. To achieve that, we use a recent robust variant of ICP called NICP [35] to find the discrepancy between the point cloud that the robot expects to perceive and what it actually perceives. The NICP method extends point-to-plane error metric proposed in Generalized ICP [36] by accounting not only for the metric distance between the points but also for the curvature of the underlying surface. The transformation between the point clouds provided by the matching algorithm can be viewed as the difference in the 3D poses at which the two point clouds c_{current} and c_j are obtained. The transformation reported by the NICP algorithm corresponds to T_Δ and thus leads to the relative position of c_{current} expressed in the local coordinate frame defined by X_j. Knowing the pose X_j and the pose of c_{current} relative to it through T_Δ enables us to compute the current position of the robot in the global odometry frame: $X_{\text{current}} = T_j T_\Delta$, where T_j is a transformation matrix that corresponds to the pose X_j in the world coordinate frame.

We use this new 3D pose X_{current} to generate a motion command to reach the next pose chosen from the recorded trajectory. As we have a wheeled platform that moves on the ground, we have no control over the height and attitude. Thus,

7

114

we generate 2D navigation commands for the robot. We continue the above-described process until the robot is within d_{max} near its starting pose X_0.

Note that our method relies on matching point clouds, typically seen from similar view points, i.e., no global search is needed. The vanilla ICP algorithm may converge to a local minimum while performing the optimization of the objective function. This can happen in very cluttered environments. Here, the objective function shows high variations with multiple local minima. On the contrary, this can also happen in places that are very feature-scarce as this may yield few distinct, narrow local minima. We found that using the NICP variant of ICP avoids such shortcomings in most practical situations, particularly when dealing with small view point changes, as is the case for our homing strategy. Refer to the work of Serafín and Grisetti [35] for a robustness analysis.

6. Experiments

The experiments are designed to illustrate (i) the advantages of our predictive exploration approach, if it is safe, and (ii) that the robot can rewind trajectories in case of failure of the mapping system.

For evaluating the next view point selection approach, we use a standard frontier-based exploration approach as a baseline and show that our exploration approach selects frontiers that lead to loop closures which in turn result in improved maps of the environment. The underlying mapping framework for all exploration experiments is a state-of-the-art graph-based SLAM system, which uses g2o [37] and FLIRT features to speed up the search for possible data associations [38], uses scan matching for incremental alignments, and applies single cluster graph partitioning to resolve ambiguities as proposed by Olson [39]. The exploration and homing systems have been implemented in C++ as ROS modules.

6.1. Map Comparisons

First, we compare the quality of the maps obtained with frontier-based exploration vs. our predictive exploration. The environments considered here are parts of the Roman catacomb Priscilla, a difficult to traverse and large-scale underground environment in Rome. The robot is equipped with tracks and thus its odometry is in general worse than the one of a wheeled robot and it sometimes reveals a (temporarily) bias to one side.

Fig. 7 illustrates the obtained results for two environments using exactly the same mapping system and identical parameters for the comparison. The map database consists of maps constructed from other catacomb sites representing a similar type of environment but not the same one. The images on the left are the "ground truth" maps obtained from manual surveys. The images in the second column correspond to the results of the frontier-based exploration, while the images on the right show our approach. As can be seen already visually, our approach yielded a consistent model of the environment, while the frontier-based approach failed. Using the frontier-based approach the robot was unable to continue its exploration task due

to an inconsistent map that prevented the computation of further exploration actions. We performed similar experiments in different nested tunnel environments and obtained comparable results.

The exploration task strongly benefits from achieving loop closures as early as possible, avoiding a high uncertainty in the pose-graph. Our approach improves the amount of loop closures whenever the current environment resembles previously seen maps, either in previous or current explorations runs. Thus, a new environment with recurrent structures also benefits from this approach. In case there is no similarity between the current environment and the maps stored in the database, no map should be retrieved and thus the system falls back to frontier-based exploration.

The execution time of our approach depends on the number of unexplored frontiers, as well as on the map size and resolution. On a standard computer and a map size of 150 m by 100 m with a grid resolution of 5 cm, next view point selection time ranged from 131 ms up to 4.8 s in the most complex situation. In practice, time consumed by the robot reaching the next view point usually dominates the time consumed by the selection task.

6.2. Exploration Path Length

The advantages of the prediction-based approach come at a cost—the cost of traversing exploration paths that are longer than the ones generated by the frontier-based approach. This experiment is designed to evaluate the increase in path length. As we are not able to obtain consistent maps for the frontier-based approach under a realistic noise model for the task under consideration, we executed this evaluation under zero noise in the simulator. Using a zero noise odometry, also the frontier-based system is able to build consistent maps. In this setting there is no advantage in using our predictive approach as the pose uncertainty is zero and no uncertainty reduction is gained from closing loops. We compared the distances traveled for the frontier-based and our approach. The distances traveled are summarized in Fig. 8. In the worst case scenario, the path generated by our approach was 1.85 times longer than the one of the frontier-based approach. The minimum increase was a factor of 1.5. Generating on average a 1.7 times longer trajectory is clearly an overhead—for actively closing loops and in this way reducing uncertainty, however, this price must be paid.

6.3. Statistical Map Consistency Check and Robust Homing

After the robot finishes exploring the environment, it needs to find its way home. The evaluation of our framework is designed to illustrate the performance of the statistical map consistency check in conjunction with an approach to safely and robustly rewind the trajectory to return the robot to the starting position should the consistency check report the map as inconsistent.

First, Fig. 5 illustrates an example of the statistical map consistency check performed on range data from the Priscilla catacombs in Rome. The partial maps computed around the shortest path are usually substantially smaller than the map of

8

115

Figure 7: Two performance comparisons in constant odometry bias scenario. On the left, the original map. In the middle, the closer frontier approach. On the right, our prediction-based approach. Note that the nearest frontier approach produces a map that is non consistent with the original one, so that the robot can not continue the exploration task. The map produced by the prediction-based approach is instead consistent with the original one.
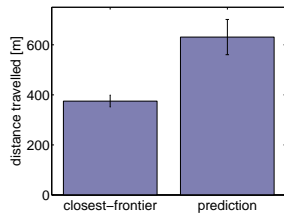
Figure 8: Mean and standard deviation of the distances traveled in the frontier-based approach and in the proposed approach.

the whole environment, especially if the environment has multiple alternative branches and forms a complicated network of corridors or rooms as we experience it often in catacombs or underground mines. Testing smaller maps results in speed-up of the statistical consistency evaluation procedure. The timings for the maps presented in Fig. 5 are as follows: full map shown on top—2,930 ms; middle—140 ms; bottom—170 ms. The computational time depends on the number of scans to analyze and the gain in speed grows with the difference between the sizes of the full and the reduced maps and the overlapping scans. We performed the map consistency test on five different datasets recorded in the Priscilla catacomb and the consistency check always generated correct results. In sum, testing a map along the planned path for consistency takes less than 200 ms and thus can be executed on the fly on the robot. Additionally, most of the computations could be cached when dealing with huge maps (although this was not done here). In this case, the test would only require a recomputation if the SLAM back-end

9

116

Figure 9: Illustration of rewinding the trajectory through the office environment. The robot is steered from the bottom "tail" of the depicted trajectory to the upper-right one. Black line denotes the odometry poses saved while the robot is steered, gray denotes the odometry on the way back, red shows the temporary destination poses picked from the odometry and blue shows the same poses after the ICP correction. The pictures show several example locations visited by the robot. These feature tight doors to rooms as well as feature-scarce corridors.

changes the configuration of the pose graph substantially.

Second, if the proposed statistical consistency check evaluates the map as inconsistent we need a robust way to return the robot home to the starting location. We evaluate the ability of our approach to rewind the trajectory by carrying out 20 experiments in our lab environment. One of these experiments is illustrated in Fig. 9. We steered the robot on a rather complicated trajectory through an obstacle parcour containing narrow

Figure 10: Three experiments performed in different settings. The meaning of the lines is the same as in Fig. 9 with the difference that the top and the bottom graphs do not show the pure odometry measurement on the return path. The average deviation from the original trajectory is between 4 cm (top dataset) and 6 cm (bottom dataset).

passages as well as areas with lots of flat wall, which represents a challenge for the matcher. The robot activated the "rewind the trajectory" behavior after we (manually) broke the SLAM system so that it followed the way in reverse order using the NICP-based pose correction.

In Fig. 9, the original odometry measurements from the *forward path* are drawn in black (hardly visible as the red trajectory perfectly overlays it). The red line illustrates the sub-sampled trajectory that the robot has selected as its sequence $\{X_0, \ldots, X_n\}$ for rewinding the trajectory. Both trajectories overlay because the robot does not use any global map and relies solely on the poses he recorded in the odometry frame (to navigate back).

The gray line depicts the pure odometry measurements recorded while performing rewinding while the blue line shows the poses of the robot after the alignment by NICP. As can be seen, the robot accurately follows the previous trajectory with our approach as the blue and the red trajectories are similar. We compute the deviation from the original odometry path by

10

117

searching the closest pose of the robot from the forward traversal (black) for each pose from the backwards traversal (blue) and averaging over all such distances. In this experiment, the average deviation of the rewinding trajectory is approximately 5 cm. From the gray trajectory, we can furthermore see that the odometry error must be taken into account—otherwise, the robot would deviate substantially from the reference path (and would collide with walls and obstacles).

We executed similar experiments in 20 different settings with trajectory lengths ranging from 10 m to nearly 100 m and the robot was always able to robustly drive back to the start location. The trajectory in Fig. 9 is approximately 52 m long while three smaller examples are illustrated in Fig. 10. Overall, this evaluation suggest that our robot is able to rewind different trajectories through the environment, robustly handling corridor-like environments with multiple narrow passages such as the doorways. Note that the robot cannot observe doorways before it fully passed through them. Only by following the reference trajectory precisely, the robot can return.

7. Conclusion

The ability to robustly operate without user intervention is an important capability for exploration robots in real-world settings. In this paper, we proposed a novel approach for autonomous exploration of unknown environments with robust homing. The key contributions of this work are two-fold. First, we presented a technique to predict possible environment structures in the unseen parts of the robot's surroundings based on previously explored environments. We exploit this belief to predict possible loop closures that the robot may experience when exploring an unknown part of the scene. This allows the robot to actively reduce the uncertainty in its belief through its exploration actions. Secondly, we presented a homing system that addresses the problem of returning a robot operating in an unknown environment to its starting position even if the underlying SLAM system fails. We combined a statistical map consistency test with an NICP-based approach to precisely rewind a previously taken trajectory.

We implemented our approach and executed it both, in simulation and on a real autonomous robot. Our experiments illustrate that our technique allows for an effective exploration of difficult to map environments. By actively closing loops, we are able to obtain consistent maps of the environment. In contrast to that, a traditional frontier-based exploration approach is not able to successfully explore the scene if the SLAM system fails. In the case of a mapping failure leading to an inconsistent map, the proposed robust homing system can accurately rewind trajectories guiding the robot through narrow passages such as doorways, even when the robot could not see these narrow spaces while navigating through them.

8. Acknowledgments

References

[1] M. Mazuran, G. Tipaldi, L. Spinello, W. Burgard, C. Stachniss, A statistical measure for map consistency in slam, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Hong Kong, China, 2014.

[2] S. Koenig, C. Tovey, Improved analysis of greedy mapping, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2003.

[3] B. Yamauchi, Frontier-based exploration using multiple robots, in: Proc. of the Int. Conf. on Autonomous Agents, 1998, pp. 47–53.

[4] P. Whaite, F. P. Ferrie, Autonomous exploration: Driven by uncertainty, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (3) (1997) 193–205.

[5] C. Stachniss, W. Burgard, Mapping and exploration with mobile robots using coverage maps, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2003, pp. 476–481.

[6] R. Rocha, J. Dias, A. Carvalho, Exploring information theory for vision-based volumetric mapping, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Canada, 2005, pp. 2409–2414.

[7] T. Duckett, S. Marsland, J. Shapiro, Fast, on-line learning of globally consistent maps, Journal of Autonomous Robots 12 (3) (2002) 287 – 300.

[8] J. Ko, B. Stewart, D. Fox, K. Konolige, A practical, decision-theoretic approach to multi-robot mapping and exploration, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2003, pp. 3232–3238.

[9] S. Shen, N. Michael, V. Kumar, 3d indoor exploration with a computationally constrained mav, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2012.

[10] R. Shade, P. Newman, Choosing where to go: Complete 3d exploration with stereo, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2011.

[11] C. Stachniss, G. Grisetti, W. Burgard, Information gain-based exploration using rao-blackwellized particle filters, in: Proc. of Robotics: Science and Systems (RSS), Cambridge, MA, USA, 2005, pp. 65–72.

[12] A. Makarenko, S. Williams, F. Bourgoult, F. Durrant-Whyte, An experiment in integrated exploration, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2002.

[13] F. Bourgoult, A. Makarenko, S. Williams, B. Grocholsky, F. Durrant-Whyte, Information based adaptive robotic exploration, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 2002.

[14] R. Sim, G. Dudek, N. Roy, Online control policy optimization for minimizing map uncertainty during exploration, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2004.

[15] H. Kretzschmar, C. Stachniss, Information-theoretic pose graph compression for laser-based SLAM, Int. Journal of Robotics Research 31 (2012) 1219–1230.

[16] A. Krause, C. Guestrin, Near-optimal nonmyopic value of information in graphical models, in: Proc. of Uncertainty in Artificial Intelligence (UAI), 2005.

[17] G. Hitz, A. Gotovos, F. Pomerleau, M.-E. Garneau, C. Pradalier, A. Krause, R. Siegwart, Fully autonomous focused exploration for robotic environmental monitoring, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2014.

[18] S. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, R. Vaughan, Feature-rich path planning for robust navigation of mavs with mono-slam, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Hong Kong, China, 2014.

[19] C. Mostegel, A. Wendel, H. Bischof, Active monocular localization: Towards autonomous monocular exploration for multirotor mavs, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Hong Kong, China, 2014.

[20] D. Fox, J. Ko, K. Konolige, B. Stewart, A hierarchical bayesian approach to the revisiting problem in mobile robot map building, in: Proc. of the Int. Symposium of Robotics Research (ISRR), 2003.

11

118

[21] H. Chang, C. Lee, Y. Lu, Y. Hu, P-slam: Simultaneous localization and mapping with environmental-structure prediction, IEEE Transactions on Robotics 23 (2) (2007) 281–293.

[22] C. Stachniss, O. Martínez-Mozos, A. Rottmann, W. Burgard, Semantic labeling of places, in: Proc. of the Int. Symposium of Robotics Research (ISRR), San Francisco, CA, USA, 2005.

[23] K. Wurm, C. Stachniss, W. Burgard, Coordinated multi-robot exploration using a segmentation of the environment, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2008.

[24] C. Stachniss, O. Martinez Mozos, W. Burgard, Efficient exploration of unknown indoor environments using a team of mobile robots, Annals of Mathematics and Artificial Intelligence 52 (2009) 205ff.

[25] D. Holz, N. Basilico, F. Amigoni, S. Behnke, A comparative evaluation of exploration strategies and heuristics to improve them, in: Proc. of the European Conference on Mobile Robots (ECMR), Örebro, Sweden, 2011, pp. 25–30.

[26] C. Sprunk, G. Tipaldi, A. Cherubini, W. Burgard, Lidar-based teach-and-repeat of mobile robot trajectories, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Tokyo, Japan, 2013.

[27] P. Furgale, P. Krüsi, F. Pomerleau, U. Schwesinger, F. Colas, R. Siegwart, There and back again-dealing with highly-dynamic scenes and long-term change during topological/metric route following, in: ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots, 2014.

[28] P. Furgale, T. Barfoot, Visual teach and repeat for long-range rover autonomy, Journal on Field Robotics 27 (5) (2010) 534–560.

[29] M. Nitsche, T. Pire, T. Krajnik, M. Kulich, M. Mejail, Monte carlo localization for teach-and-repeat feature-based navigation, in: M. Mistry, A. Leonardis, M. Witkowski, C. Melhuish (Eds.), Advances in Autonomous Robotics Systems, Vol. 8717 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 13–24.

[30] E. Battesti, S. Bazeille, D. Filliat, Qualitative localization using vision and odometry for path following in topo-metric maps, in: Proc. of the European Conference on Mobile Robots (ECMR), 2011, pp. 303–308.

[31] D. P. Ström, F. Nenci, C. Stachniss, Predictive exploration considering previously mapped environments, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2015.

[32] I. Bogoslavskyi, O. Vysotska, J. Serafin, G. Grisetti, C. Stachniss, Efficient traversability analysis for mobile robots using the kinect sensor, in: Proc. of the European Conference on Mobile Robots (ECMR), 2013.

[33] M. Cummins, P. Newman, Highly scalable appearance-only slam fab-map 2.0, in: Proc. of Robotics: Science and Systems (RSS), 2009.

[34] P. Beeson, N. Jong, B. Kuipers, Towards autonomous topological place detection using the extended voronoi graph, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2005.

[35] J. Serafin, G. Grisetti, NICP: Dense normal based point cloud registration and mapping, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2015.

[36] A. Segal, D. Haehnel, S. Thrun, Generalized-icp., in: Proc. of Robotics: Science and Systems (RSS), Vol. 2, 2009.

[37] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g2o: A general framework for graph optimization, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2011, pp. 3607–3613.

[38] G. Tipaldi, K. Arras, Flirt–interest regions for 2d range data, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2010, pp. 3616–3622.

[39] E. Olson, Recognizing places using spectrally clustered local matches, Robotics and Autonomous Systems.

12

Safe and Reliable Path Planning for the Autonomous Vehicle Verdino

This appendix includes the full text for the following article:

121

Safe and Reliable Path Planning for the Autonomous Vehicle Verdino

Rafael Arnay, Néstor Morales, Antonio Morell, Javier Hernández-Aceituno, Daniel Perea Ström, Jonay Toledo, Alberto Hamilton, Javier Sánchez-Medina, and Leopoldo Acosta
Universidad de La Laguna, Canary Islands, Spain Email: rafa@isaatc.ull.es

Abstract—This paper introduces a local planner which computes a set of commands, allowing an autonomous vehicle to follow a given trajectory. To do so, the platform relies on a localization system, a map and a cost map which represents the obstacles in the environment. The presented method computes a set of tentative trajectories, using a schema based on a Frenét frame obtained from the global planner. These trajectories are then scored using a linear combination of weighted cost functions. In the presented approach, new weights are introduced in order to satisfy the specificities of our autonomous platform, Verdino. A study on the influence of the defined weights in the final behavior of the vehicle is introduced. From these tests, several configurations have been chosen and ranked according to two different proposed behaviors. The method has been tested both in simulation and in real conditions.

122

I. Introduction

Generally speaking, the navigation system of autonomous vehicles is composed of two main levels: the global planner and the local planner. The first level consists of the generation of a feasible route from the current position of the vehicle to a desired goal. The second level computes the necessary commands to control the vehicle in order to follow the global plan, while dynamically adapting to the changing environment conditions.

The problem we want to solve is safely following a predefined route while avoiding dynamic obstacles. This problem is not trivial, since several factors have to be taken into account. For starters, the safety of pedestrians is crucial: the vehicle has to navigate close to the desired route while keeping a safe distance to the obstacles. Secondly, the navigation has to be comfortable from the passengers point of view: when following the global path and avoiding obstacles, the performed maneuvers have to prevent both abrupt changes of linear and angular speed and high curvature trajectories.

The method presented in this paper is the local planner of an autonomous robotic prototype called Verdino[1] [1], shown in Fig. 1. This electric vehicle was designed for people transportation in pedestrian environments.

When a new destination is selected, the global planner builds a feasible path from the current vehicle position to the desired goal. The vehicle then follows this path by using the method presented in this paper. In this method, the euclidean space surrounding the vehicle is transformed to the Frenét space, using the computed global trajectory as basis. Then, a set of tentative paths is computed, considering the following restrictions: all paths should start at the position and orientation of the vehicle, and should end parallel to the global trajectory, at a parameterized lateral distance from it. This way paths are computed using just geometrical information, making the problem simpler by not needing to define a kinematic model of the vehicle to generate the trajectories in the euclidean space. Once trajectories are computed, they are transformed back to the euclidean space, in which they are scored based on different variables like, for example, their curvature or their distance to obstacles. Impossible paths (those which can not be followed by the vehicle) are removed. Using these scores, a winner path is selected and used for the computation of the next speed and steering commands.

This vehicle used as a testing platform is a standard golf car, which has been electronically and mechanically modified so it can be controlled by an on-board computer. It is equipped by default with six $6\,V$ batteries, a speed controller, a $36\,Vcc$ electrical motor, mechanical brakes and steering, and has a maximum speed between 19 and 23 Km/h.

In order to localize itself, the vehicle is equipped with an odometry system attached to each wheel, which allows making relative position estimations. This information is combined with the information provided by an Inertial Measurement Unit (IMU) and a centimetric DGPS. Several Light-Detection And Ranging (LIDAR) sensors are used both for SLAM (Simultaneous Localization and Mapping) and obstacle detection. All this information is combined using the method in [2], so the vehicle is properly localized. The vehicle is controlled by an on board computer.

II. Previous Work

In the literature, it is possible to find several planning methods that have been applied in the generation and selection of local paths. Most of these methods are based on a discrete optimization scheme [3], [4], [5] and [6]. From all the approaches of this kind, Rapidly-exploring Random Trees (RRT) and its variants are widely used in non-holonomic motion planning applications. However, real time implementations require efficient heuristics for the sampling configuration. Some examples of this kind of methods are [7], [8] and [9].

---

[1]http://verdino.webs.ull.es

FIG 1 Verdino prototype.

Some other methods, like the method introduced in this paper, are based in the transformation of the configuration space through the Frenét space. Some examples of this technique are the methods in [5], [3] and [10]. In [5], long term objectives are pursued, like speed keeping, merging, following, stopping. This is done through optimal control strategies within the Frenét frame of the street. In [3], lateral offset is defined as the perpendicular direction to an established base trajectory. This allows the vehicle to drive along the road, parallel to this trajectory. In [10], a set of candidate paths are also generated, with endpoints in fixed positions at different offsets with respect to the base frame, but they do not set this base frame in the center of the road, using a security cost for each candidate path instead. The safety of the path is computed by blurring the binary data of the obstacles.

III. Method

In order for the vehicle to navigate, a global plan has to be defined. This global plan is a rough estimate of the path that the vehicle has to follow to go from its current position to a desired goal.

The global path is generated using the NavFn global planner[2]. This planner implements Dijkstra's algorithm to find the best path through a cost map, which represents the goodness of the navigable areas taking into account static obstacles.

The vehicle can follow the global plan using the local planner. Obstacles are represented in a *costmap*, which is an occupancy grid which the local planner needs in order to select the best trajectories and avoid obstacles.

A. Generation of the Costmap

The costmap maintains information about occupied/free areas in the map, as an occupancy grid. It uses sensor data and information from the static map to store and update information about obstacles in the world, which are marked in the map (or cleared, if they are no longer there).

[2]http://wiki.ros.org/navfn

Each cell in the map can have 255 different cost values:
- A value of 255 means that there is no information about this specific cell in the map.
- 254 means that a sensor has marked this specific cell as occupied. This is considered as a lethal cell, so the vehicle should never enter there.
- The rest of cells are considered as free, but with different cost levels depending on an inflation method relative to the size of the vehicle and its distance to the obstacle.

The cost value of free cells decreases with the distance to the nearest occupied one, following the expression:

$$C(i, j) = \exp\left(-1.0 \cdot \alpha \cdot (\|c_{ij} - \bar{o}\| - \rho_{\text{inscribed}})\right) \cdot 253 \quad (1)$$

In this expression, α is a scaling factor which increases or decreases the decay rate of the cost of the obstacle. $\|c_{ij} - \bar{o}\|$ is the distance between cell $c_{ij} \in C$ (where C is the set of cells in the costmap) and the obstacle. Finally, $\rho_{\text{inscribed}}$ is the inscribed radius, which is the inner circle of the limits of the car. For a better explanation of the way in which the costmap is computed, please refer to [11]. An implementation of this method is available at http://wiki.ros.org/costmap_2d, as part of the Robotic Operating System (ROS) framework used for the development and testing of our approach. In the tests described in section IV a value of $\alpha = 3.0$ has been used.

B. Local Planner

Once the global path is defined, a method to compute the steering and speed commands is needed, in order to control the vehicle along this path. This method should also be able to avoid the obstacles present in the road. This has to be done in a safe and efficient way. The method developed to solve this problem is based on the solution described in [10], in combination with some ideas proposed in [3], taking into account the characteristics of the *Verdino* prototype.

The basic idea of the local path generation is to define a set of *feasible paths* and choose the best option in terms of their cost. The winner path defines the steering and speed commands that the vehicle will use. Having options among local paths is useful to overcome unforeseen obstacles in the road.

The current euclidean coordinate system is transformed into a new system based on the Frenét space. This space is computed as follows: the global path is considered as the base frame of a curvilinear coordinate system. The feasible local paths are defined in terms of this base frame in the following way:
- The nearest point of the main trajectory to the vehicle (where the distance is computed perpendicular to the global path), will be the origin of the curvilinear coordinate system.

124

- The horizontal axis will be represented by the distance over the global path, along its direction.
- The vertical axis is represented by the vector which is perpendicular to the origin point and points left of the path direction.

In this schema, trajectories can be easily computed in the curvilinear space (that is, maneuvering information is generated). These are then transformed to the original euclidean space, in which the obstacles information is added by assigning costs to each path.

As seen, the local path generation can be divided into two stages: the *candidate paths generation* and *the winner path selection*.

1) Candidate Paths Generation

In this stage, the base frame of the curvilinear coordinate system is defined, so that the algorithm will be able to compute the trajectories in this space as if the global plan were a rectilinear trajectory. The geometric relationship of the path in euclidean and curvilinear coordinates is shown at Fig. 2.

The coordinate origin of the base frame is defined as the closest point to the vehicle in the global path. The arc length of the base frame (s on the right image) is obtained as the distance of each point along the global plan (represented as a green line) to the coordinate origin. This distance is represented on the x-axis of the curvilinear system. On the y-axis, q represents the perpendicular lateral distance with respect to the path. The left side is represented by positive values and the right side by negative values.

For the computation of the transformation between the euclidean and the curvilinear coordinate system, path curvature κ is needed. This value is computed as follows [10], [5], [12]:

$$\kappa = \frac{S}{Q} \cdot \left(\kappa_b \cdot \frac{(1 - q \cdot \kappa_b) \cdot (\partial^2 q / \partial s^2) + \kappa_b \cdot (\partial q / \partial s)^2}{Q^2} \right), \quad (2)$$

$$\text{where} \begin{cases} S = \text{sign}(1 - q \cdot \kappa_b) \\ Q = \sqrt{\left(\frac{\partial q}{\partial s}\right)^2 + (1 - q \cdot \kappa_b)^2} \end{cases} \quad (3)$$

There, κ_b is the curvature of the segment of the base trajectory used for the computation of the Frenét space. The candidate paths generation is performed in the curvilinear space, without considering the obstacles in the environment. These will be taken into account later, once the tentative trajectories are transformed to the euclidean space.

Maneuvering Paths Generation: The curvature of the generated paths is defined by the lateral offset q with respect to the base frame. First and second order derivatives of q are needed to compute κ (see equations 2 and 3), so a function dependent on the lateral offset is needed to compute a smooth lateral change.

q can be defined by a sequence of a cubic polynomial and a set of constants [10]:
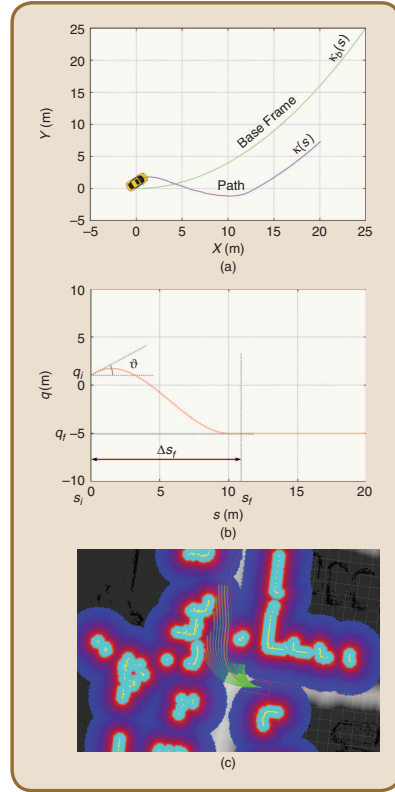
FIG 2 Conversion of a trajectory between the Cartesian and Frenét spaces, and paths truncation example.

$$q(s) = \begin{cases} a \cdot \Delta s^3 + b \cdot \Delta s^2 + c \cdot \Delta s + q_i & \text{if } s_i \le s < s_f \\ q_f & \text{if } s_f \le s \end{cases}$$
$$\frac{\partial q}{\partial s}(s) = \begin{cases} 3 \cdot a \cdot \Delta s^2 + 2 \cdot b \cdot \Delta s + c & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}, \quad (4)$$
$$\frac{\partial^2 q}{\partial s^2}(s) = \begin{cases} 6 \cdot a \cdot \Delta s + 2 \cdot b & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}$$

where $\Delta s = s - s_i$.

In Fig. 2b, the components involved in this process are depicted.

125

- The initial length s_i is zero, since the global planner is being pruned as the vehicle advances. Lateral offset q_i with respect to the global path's origin is also known.
- Angle θ defines the difference between the vehicle heading angle and the tangent angle of the base frame at the current position. If $\theta > 40°$, the vehicle enters in a recovery state, described in section III-B3. Once the vehicle is headed properly towards the path, this recovery state ends.
- s_f is a parameter that controls the longitudinal distance needed to reach offset q_f. This distance should be dependent on the speed. However, as the top speed of the prototype is not too high, Δs_f can be considered as the distance needed to go from q_i to the biggest q_f at top speed.
- The different q_f are computed separately for each path attending to the parameters defined by the user. In our implementation, the method receives as input the maximal width covered between the outer left and the outer right generated path w_{max}, and the total number of paths to be generated (n_{paths}). So the value of q_f is computed as $w_{max}/(n_{paths}-1)$. s_f is also a free parameter provided directly by the user as the maximal longitudinal length that is desired to be covered by the paths.

  Once the paths are generated in the curvilinear coordinate system, they are transformed to the euclidean space. In this new space, it is possible to evaluate their corresponding costs.

  To define the maximum length of each candidate path, the cost of the cells corresponding to the points in the trajectory are inspected. If this cost is over threshold $\tau_{circumscribed}$, (which is the cost of cells at a distance equal to the radius of the minimum circumference that contains the outer limits of the vehicle footprint), the path is truncated at this point, as shown in Fig. 2c, where the generated paths are shown together with a colored costmap representation. In this figure, blue represents a low cost value, while the red color is used for the higher costs. Yellow and cyan correspond to lethal cells and inscribed cells, respectively.

2) Winner Path Selection

At each iteration, the winner path is selected through the use of a linear combination $J[i]$ of weighted cost functions, related to the following parameters: occlusion, length, distance to the global path, curvature and consistency of the path. $J[i]$ is evaluated as follows:

$$J[i] = \omega_o C_o[i] + \omega_l C_l[i] + \omega_d C_d[i] + \omega_\kappa C_\kappa[i] + \omega_c C_c[i] \quad (5)$$

Here, i is the path index, and C_o, C_l, C_d, C_κ and C_c are the costs of occlusion, length, distance to the global path, curvature and consistency, respectively. Their relative factors ω_k, $k \in \{o, l, d, \kappa, c\}$ are the associated weights that allow to adjust the influence of each of the costs to the final cost value.

The following costs will be computed for each candidate path independently in the euclidean space.

a) Occlusion

The occlusion cost is related to the safety of the path. This cost estimates the goodness of a path, such that the best paths are those which pass far enough from the obstacles. To do so, the footprint of the vehicle is simulated in each point of the path. The occlusion cost corresponds to the normalized maximum cost along the path:

$$C_o = \frac{\max\{c_i\}}{255}, \quad i = 1...L \quad (6)$$

In this expression, L is the length of the current path being evaluated. $\max\{c_i\}$ is the maximum value of all the costs, associated to a point in the path.

b) Length

This cost is related to the length of the current path. The longer the path is, the lower its associated cost is. In general, a long path implies the existence of an area which is free of obstacles and can be safely traversed.

$$C_l = 1 - \frac{\sum_{i=1}^{L} \| p_i - p_{i-1} \|}{q_{f_{max}} + s_f} \quad (7)$$

Here, p_i is a certain point inside the evaluated path. $q_{f_{max}}$ is the maximum value that q_f can have for a certain path. Lengths are normalized to a value that a path will never reach.

c) Distance to the Global Path

This cost represents the lateral offset of the vehicle with respect to the global path. Tuning the associated weight of this cost will change the behavior of the vehicle when returning to the global path, after an occasional obstacle is avoided. It is computed as follows:

$$C_d = \frac{\sum_{i=1}^{L} \| p_i - \text{nearest}(p_i, g) \|}{L \cdot q_{f_{max}}}, \quad (8)$$

where $\text{nearest}(p, g)$ is the nearest point in the global path g to point p. This cost is normalized with respect to the maximum expected offset, $q_{f_{max}}$.

d) Curvature

This cost allows to give priority to smoother paths. Let $p(x_i, y_i), i = 1 \cdots L$, be a point in the path. Then,

$$C_\kappa = \max\left\{ \frac{x_i' \cdot y_i'' - x_i'' \cdot y_i'}{(x_i'^2 + y_i'^2)^{3/2}} \right\}, \quad i = 1...L \quad (9)$$

e) Consistency

This cost avoids continuous changes in winner paths between iterations. Once the vehicle starts a maneuver, it is

126

preferable to keep the same behavior during the following iterations. This is done through the following expression:

$$C_c = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} l_i \, ds \qquad (10)$$

Lateral cost $l_i(s)$ is the distance between the current path and the previous winner path, at the same longitudinal position s; s_1 and s_2 are the first and last positions over s for which there are common points in both trajectories. At the beginning of the trip, the oscillation cost is 0, so it does not affect to the final choice of the path.

Once all costs are computed, the expression described in equation 5 is applied. In those paths for which it is impossible to advance, due to the presence of a nearby obstacle, the cost is forced to a negative value in order to indicate the rest of the system that this path is invalid.

The path with the lowest cost is selected (winner path W). If for some reason there is no valid path, the vehicle stops until the road is free of obstacles. If this situation does not change for a while, the recovery behavior process starts.

3) Recovery behavior

There are two scenarios in which the vehicle executes a recovery maneuver. The first one occurs when the vehicle is not correctly aligned with respect to the global plan, and the initial angle is too large to produce feasible local paths that comply with the curvature restrictions of the vehicle. The second case happens when the vehicle is correctly aligned but there are no feasible local paths to follow, due to the presence of a nearby obstacle.

In the first case, the recovery maneuver is intended to align the vehicle to the global path again. Same as for the local planner, the recovery behavior of the vehicle is composed of
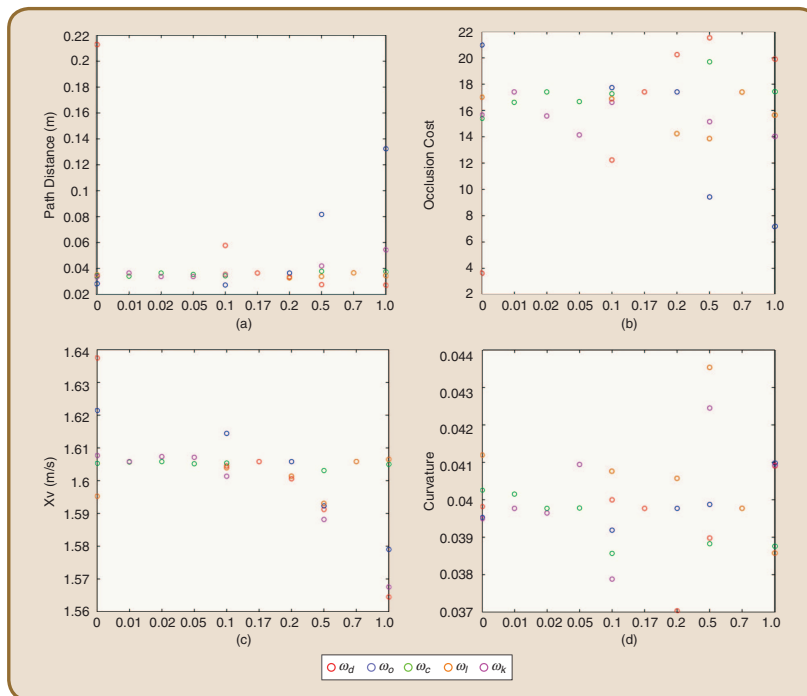
FIG 3 Results obtained from parameters.

127

two main phases. In the first phase, a set of feasible paths is generated. In the second phase, these paths are weighed in order to choose the best option. The vehicle will then try to follow the winner path. The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides. The recovery maneuver is composed of a sequence of one or more of these paths.

In the second case, if possible, the vehicle just moves backwards for a short distance, in order to obtain enough space for the generation of feasible local paths.

IV. Results
In this section, the behavior of the local planner is described, using different cost weights as input.

A. Experimental Setup
In order to study the behavior of our method, several trajectories were followed, while recording a set of measured

variables. These variables were *distance to path*, which measures the distance from the center of the vehicle to the closest point in the global path; *occlusion cost*, which measures the maximum cost of the cells below the vehicle footprint at each iteration; the *speed*, assuming that faster trajectories are preferred; and the *curvature* of the followed trajectory.

Since keeping the exact same conditions for all the tests is desirable, a simulator was used. In each test, the vehicle started at the exact same position and traveled towards the exact same goal. Obstacles were always in the same locations, and the only changing values were the input parameters under evaluation. The obtained results were then validated with some tests under real conditions, using the Verdino platform.

As seen in section III-B2, there are five different parameters that influence the overall cost, which will determine the chosen winner path. Each parameter has an associated weight. A base configuration of $\omega_d = 0.17, \omega_o = 0.2,\ \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$, which

■ $\omega_d = 0.17, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.2, \omega_k = 0.01$	■ $\omega_d = 0.17, \omega_o = 0.1, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$
■ $\omega_d = 1.0, \omega_o = 1.0, \omega_c = 0.5, \omega_l = 5, \omega_k = 1.0$	■ $\omega_d = 0.17, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.0$
■ $\omega_d = 1.0, \omega_o = 0.1, \omega_c = 0.05, \omega_l = 1.0, \omega_k = 0.0$	■ $\omega_d = 1.0, \omega_o = 0.1, \omega_c = 0.05, \omega_l = 1.0, \omega_k = 0.0$
■ $\omega_d = 0.17, \omega_o = 0.5, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$	■ $\omega_d = 0.17, \omega_o = 0.5, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$
■ $\omega_d = 0.1, \omega_o = 0.5, \omega_c = 0.0, \omega_l = 0.5, \omega_k = 0.1$	■ $\omega_d = 0.1, \omega_o = 0.5, \omega_c = 0.0, \omega_l = 0.5, \omega_k = 0.1$
■ $\omega_d = 0.17, \omega_o = 1.0, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$	■ $\omega_d = 0.17, \omega_o = 1.0, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$
■ $\omega_d = 0.0, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$	■ $\omega_d = 0.0, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$
■ $\omega_d = 0.0, \omega_o = 0.5, \omega_c = 0.0, \omega_l = 0.2, \omega_k = 0.0$	■ $\omega_d = 0.0, \omega_o = 0.5, \omega_c = 0.0, \omega_l = 0.2, \omega_k = 0.0$
■ $\omega_d = 0.5, \omega_o = 0.0, \omega_c = 0.02, \omega_l = 0.0, \omega_k = 0.5$	■ $\omega_d = 0.5, \omega_o = 0.0, \omega_c = 0.02, \omega_l = 0.0, \omega_k = 0.5$
■ $\omega_d = 0.0, \omega_o = 0.0, \omega_c = 0.5, \omega_l = 0.7, \omega_k = 0.01$	■ $\omega_d = 0.0, \omega_o = 0.0, \omega_c = 0.5, \omega_l = 0.7, \omega_k = 0.01$
■ Chu et al. [10] $\omega_o = 0.1, \omega_c = 0.02, \omega_k = 0.01$	■ Chu et al. [10] $\omega_o = 0.1, \omega_c = 0.02, \omega_k = 0.01$
(a)	(b)

FIG 4 Results obtained from rankings.

128

provided good empirical results, has been used. Using the base configuration as a starting point, different weight configurations have been obtained by incrementally varying each weight.

Fig. 3 shows how the variation of each individual weight influences the measured variables. In order to establish the relative importance of each weight with respect to the base configuration, for each performed test one of them varies in the [0, 1] range, while the rest keep their default values. Fig. 4 shows a ranking with different cost weights configurations. In Fig. 4a, the evaluation has been done by taking into account the path distance and the occlusion cost. In Fig. 4b, the evaluation takes into account the occlusion cost and path distance again, but also the velocity and the curvature of the followed path. Verdino is intended to be used in pedestrian

> The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides.

areas. If the speed and curvature costs are taken into account when computing the overall cost of the trajectories, paths with high speed and predominantly straight are more likely to be selected. This means that the vehicle will be more aggressive and less capable of maneuvering. In crowded environments, it is usually better to take a longer, slower path that skirts obstacles (people) by a large margin, than a fast, straight path that traverses near obstacles. For this reason, in order to reach a compromise between speed and maneuverability, the
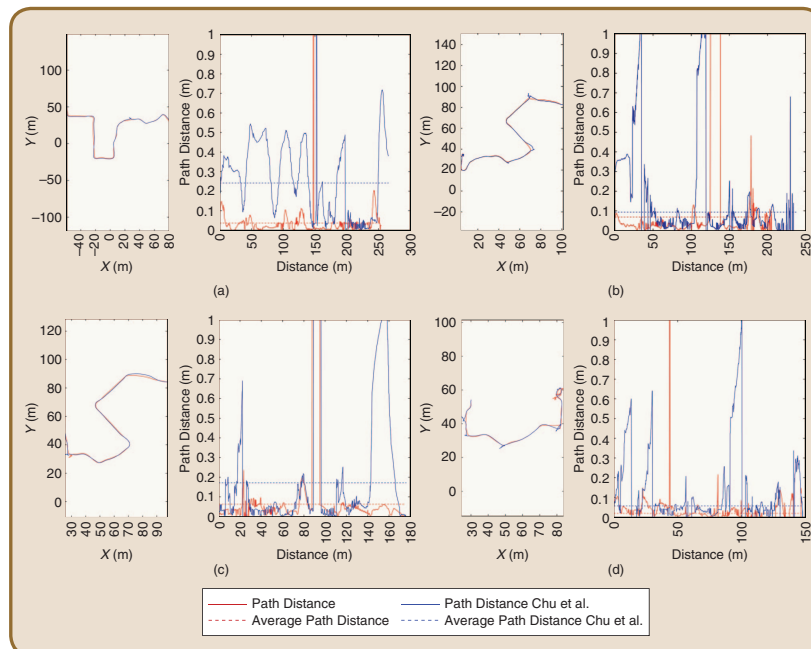
FIG 5 Traversed routes comparison of the proposed method (red) and the method proposed in Chu et al. [10] (blue).

129

> The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly.

measured speed and the curvature are weighted by 0.5 to lessen their influence.

In order to aggregate variables with different units, all measured variables are normalized between 0 and 1 using the maximum and minimum values for each range. Also, prior to aggregating results, some variables are inverted. For example, the occlusion cost and the path distance are to be minimized, whereas speed and path length are to be maximized.

B. Cost weights determination

As expected, the measured variables are observed to be more greatly affected when varying their corresponding cost weight, whereas the influence of the other weights is not as evident. For instance, as Fig. 3a shows, varying the path distance cost causes the path distance variable to change proportionally. As Fig. 3b shows, the relationship between this cost weight and the occlusion cost is inversely proportional. This behavior may be produced due to the global plan traversing relatively high cost areas, mainly in closed curves. This occurs because the global planner does not take the vehicle dynamics into account when computing the global plan. When the cost weight of the path distance is high, the vehicle is forced to stick to the global path as much as possible and thus the cost can raise. If the occlusion cost weight is high, the vehicle's priority is to avoid obstacles as much as possible, so the distance to the global path grows.

Additionally, if the path length cost weight increases, the selection of long paths is favored. As the velocity command is computed as inversely proportional to the cost, in general, higher speeds are obtained when paths are longer (Fig. 3c). The navigation speed also depends on the distance to the obstacles, which is why the occlusion cost weight also influences this measurement. Moreover, if the curvature cost weight is high, the cost of the winner path in closed curves also increases. As speed is inversely proportional to the cost of the winner path, this causes the vehicle to slow down.

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly (see Fig. 3d).

The rankings of Fig. 4a and Fig. 4b show which weight configurations produce the best results, based on different criteria. In Fig. 4a the ranking takes into account the occlusion cost and the path distance. In this sense, the configurations in the first places of the ranking favor mainly following the global path and avoiding obstacles. In the ranking shown in Fig. 4b the speed achieved by the vehicle and the curvature of the trajectory are also taken into account.

For the ranking in Fig. 4a, the best configuration was $\omega_d = 0.17, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.2,$ and $\omega_k = 0.01$ (shown in blue color); and for the ranking in Fig. 4b, this configuration was $\omega_d = 0.17, \omega_o = 0.1, \omega_c = 0.02, \omega_l = 0.7,$ and $\omega_k = 0.01$ (shown in light blue color).

The method presented in [10] uses three weights: occlusion, curvature and consistency. For the sake of comparison, the path distance and path length costs weights have been set to zero in order to replicate the configuration used in [10]. This configuration is shown in violet color in Figs. 4a and 4b. As can be seen, there are several configurations with five costs that outperform the configuration with three costs proposed in [10]. As depicted in the charts, the use of the additional cost weights proposed in this work (ω_d and ω_l) produces better results in terms of vehicle behavior.

The charts of Fig. 5 compare the proposed method using the winner five-costs configuration of Fig. 4b (light blue color), to the method proposed in [10] that uses a three costs configuration, shown in violet color in Fig. 4b. On the left side of each figure, the traversed trajectory of the proposed method (red) and the method proposed in [10] (blue) are shown. On the right side of each figure, it is depicted the distance to the global path along the traversed trajectory of the presented method (red) and the method proposed on [10] (blue). Dashed lines show averaged path distance results.

As can be seen, the addition of the path distance costs makes the vehicle navigate closer to the global plan. Controlling the distance to the global plan is of capital importance in complex scenarios, like the ones Verdino is intended for. In these scenarios, mainly pedestrian areas, there are sharp turns and narrow navigable zones. Without a path distance cost, the vehicle may not follow the global plan properly. For example, when approaching a curve in which the vehicle passes from a wider to a narrower area, if the car predominantly selects paths with low occlusion costs, it will turn late. In sharp turns this can force the vehicle to initiate a recovery behavior to reorient itself, before it can continue following the global plan again. In the left side of the examples shown in Fig. 5, there are several situations where the method proposed in [10] has to

130

initiate a recovery behavior. In these situations, there is an abrupt change in the distance to the path, as shown in the right side of the examples. This occurs noticeably less frequently with the proposed method, which sticks properly to the global path.

Taking into account the obtained results, the winner configuration of Fig. 4b was chosen for the Verdino prototype. This method works at 10 *Hz*, on a *i7-3770K* processor, 16 Gb of *RAM DDR-3* memory, *SSD* storage and a NVIDIA GeForce GT 640. These times have been obtained under real conditions.

V. Conclusions

This paper presents a system which follows a global plan, by generating different tentative paths and choosing the most suitable one.

Different configurations have been tested and ranked, in order to select the parameters that will ultimately influence the vehicle behavior. These parameters are the length of the generated path, its distance to the global path, its proximity to obstacles, its curvature and its consistency. Using the obtained results, a winner configuration has been selected to be used in the real prototype Verdino.

The results are compared to a similar method that uses three cost weights to select the local paths. As shown in the results section, the inclusion of two additional weights improve the navigation behavior of the prototype. This is the main contribution of the presented method with respect to the method presented in [10]. The inclusion of the path distance cost makes the vehicle follow the global plan more accurately than in the previous work presented in [10]. Additionally, the path length cost allows us to control the influence that the length of the winner path has in the speed of the vehicle. Another advantage is the use of a recovery behavior to overcome unforeseen situations, complementing the local planner and ensuring that the vehicle never gets stuck.

However, there is still room for improvement in the presented approach. One of the main drawbacks of the method is that if the angle between the vehicle and the path is too big, it is not possible to generate the paths, or they could not be followed by the vehicle due to physical restrictions. However, the use of the recovery maneuvers minimizes the impact of this limitation. Also, the obstacles are being considered as static, and no information about their previous motion is in use, which could lead to a more intelligent behavior of the vehicle. Although the iteration frequency of the method is high enough to reduce the effects of this lack of information, further research must be done in order to detect obstacles trajectories and to be able to include them in the costmap.

About the Authors

Dr. Rafael Arnay was born in 1982. He received the M.Sc. and Ph.D. degrees (with honors) in Computer Science from the University of La Laguna (Spain), in 2007 and 2014, respectively. He joined the Department of Computer Science and Systems, University of La Laguna in 2007, where he is currently a Postdoctoral Researcher. His current research interests include unsupervised learning, object segmentation/recognition, feature extraction and deep learning.

Dr. Néstor Morales is a researcher at Universidad de La Laguna. He received his Ph.D. in Computer Sciences on 2014, and his Computer Sciences Engineer degree on 2007. His PhD was related to different techniques for computer vision based obstacle detection for autonomous vehicles both using monocular and stereo vision, and the ways used for their avoidance. He has published several papers in several national and international conferences and journals. His research interests are computer vision, obstacle detection and tracking, and mobile robots planning.

Antonio Morell was born in 1981 in Santa Cruz de Tenerife, Spain. He received his B.Sc. degree in Computer Science (Hardware Engineering) in 2006 and his B.Sc. degree in Automation and Industrial Electronics Engineering in 2010 from the University of La Laguna (ULL), Spain. After receiving his M.Sc. degree in Research, Development and Innovation in Science and Engineering in 2011 also from the ULL, he worked as a researcher in Project SAGENIA, funded by the Spanish Ministry of Economy and Competitiveness until 2013. Since then, he is a Research Assistant supported with a grant from the Canarian Agency for Research, Innovation and Information Society (ACIISI), pursuing a Ph.D. in Engineering Physics. His research interests include sensor fusion, mobile robots, autonomous navigation, biped locomotion and robot modeling and control.

Javier Hernández-Aceituno was born in 1986 in Santa Cruz de Tenerife, Spain. He graduated in Computer

Engineering in 2010 at the Universidad de La Laguna (ULL), Spain. He is currently conducting his research as a Ph.D. student in ULL. His current research includes mapping, navigation, and obstacle detection algorithms for autonomous vehicles.

Daniel Perea Ström was born in Stockholm, Sweden. He received the B.Eng. degree in computer science and systems engineering in 2007, and the M. Eng. degree in computer science engineering from the University of La Laguna, in 2008. Since 2007 he is with the Robotics Group of the University of La Laguna, conducting research in the field of mobile robotics. Since 2009 he is developer of the self driving car Verdino in charge of the localization and mapping systems. His research interests are in the areas of robot navigation, simultaneous localization and mapping, exploration, and machine learning approaches.

Dr. Jonay Toledo is an Assistant Professor at University of La Laguna (ULL). He received his Master in Computer Science in 2001, Master in Electronics in 2002 and Ph.D. in Automatic Control in 2008. His current research interests include mobile robots, autonomous vehicles, automatic control and embedded systems. He is part of the development team of Verdino, an autonomous electric car designed to drive autonomously in pedestrian areas. He has published several conference and journal papers in robotics, automatic control, artificial intelligent and autonomous cars.

Dr. Alberto Hamilton was born in 1968. He received the M. Sc. in 1991 in Physics from University Complutense of Madrid and Ph.D. in Computer Science in 1997 from University of La Laguna (Spain). He joined the Department of Computer Science and Systems, University of La Laguna in 1991, where he is currently senior lecturer. His current research interests include mobile robotics, intelligent agents and pervasive computing.

Dr. Javier Sánchez-Medina earned his Engineering Master Degree at the Telecommunications Faculty on 2002, and his PhD at the Computer Science Department on 2008. He is interested in the application of Evolutionary and

Parallel Computing techniques to Intelligent Transportation Systems. He has more than 20 international conference and 10 international journal papers. He is also very active as a volunteer of the IEEE ITS Society, where he has been serving in a number of different positions. Currently he is EiC of the ITS Podcast, the ITS Newsletter Vice-president of the IEEE ITSS's Spanish chapter and General Chair for IEEE ITSC2015.

Dr. Leopoldo Acosta is a Full Professor at the Universidad de La Laguna, Spain. He has been involved in several competitive nationally-funded research projects related to Artificial Intelligence and Robotics, four of them as project leader. He has directed six Ph.D.

References

[1] L. Acosta, J. Toledo, R. Arnay, J. Espelosín, N. Morales, D. Perea, and L. Moreno, "Verdino, prototipo eléctrico de vehículo autoguiado," in *XXXIII Jornadas de Automática*, Sept. 2012, pp. 729–756.

[2] D. Perea, J. Hernandez-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta, "MCL with sensor fusion based on a weighting mechanism versus a particle generation approach," in *Proc. 16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 166–171.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[5] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, May 2010, pp. 987–993.

[6] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *J Field Robot.*, vol. 25, no. 11-12, pp. 939–960, Nov. 2008.

[7] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Contr.*, vol. 8, no. 11, pp. 995–1020, Sep. 1998.

[8] S. M. LaValle, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[9] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Contr. Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[10] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transport. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.

[11] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proc. Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ Int. Conf.*, 2014, pp. 709–715.

[12] T. Barfoot and C. Clark, "Motion planning for formations of mobile robots," *Robot. Autonomous Syst.*, vol. 46, no. 2, pp. 65–78, Feb. 2004.

ITS

132

Stability and performance analysis of unmanned aerial vehicles: Quad vs. Hex

This appendix includes the full text for the following article:

133

Stability and performance analysis of unmanned aerial vehicles: Quadrotor against Hexrotor

Jonay Toledo ✉ *, Leopoldo Acosta, Daniel Perea, Nestor Morales*
Computer Science Departament, University of La Laguna, La Laguna, Tenerife 38200, Spain
✉ *E-mail: Jonay@isaatc.ull.es*

Abstract: Multirotor helicopters are very powerful flying robots used in many applications, but their lack is robustness. A fail in any of their rotors can drive the helicopter to fall. In this study two different micro-helicopter structures are analysed, a standard Quadrotor and a Hexrotor. An uncertainty model is generated and a robust controller is designed. The structured singular value μ_Δ is used to test the robust stability and performance of these two plants, obtaining better results for Hex than Quadrotor. At the end of the study, a set of flight tests, where some of the motors are partially damaged, is presented confirming the analysis of μ_Δ. Results of this study show Hexrotor as a very interesting structure for unmanned aerial vehicles because of its stability and performance characteristics compared with Quadrotor.

1 Introduction

Small scale unmanned aerial vehicles (UAVs) have been extensively studied in the literature [1–3] showing their utility in a broad range of applications working alone or in formation [4, 5]. Some examples are surveillance, power line inspection, rescue robotics and so on.

One of the most popular UAV is Quadrotor, presenting the following advantages against comparably-scaled helicopters. First, Quadrotors do not require mechanical linkages to vary the rotor blade pitch angle as they spin. This simplifies vehicle design, and reduces maintenance time and cost. Second, the use of four rotors allows each individual rotor to have a smaller diameter than equivalent helicopter rotor for a given vehicle size, allowing them to store less kinetic energy during flight. Even still dangerous, damages caused if the rotors hit any object are smaller. Finally, by enclosing rotors within a frame, rotors can be protected during collisions, allowing for indoor flights and in obstacle-dense environments, with low risk of damaging the vehicle, its operators or its surroundings.

The main inconvenient of this kind of helicopters is that they are intrinsically unstable. One of the keys for a successful control is the sensorial subsystem, requiring measurements of the helicopter position and orientation (pose) in real time and with accuracy. They can fly for a long time and the probability of motor failures during flights is high. A short fail in any of their rotors can drive the helicopter to fall. This failure could be generated by an overheating of the motor control electronic, a temporal blocking of the propeller, using cheap motors controllers and so on. Especially at hover, motors are running heavily, which causes overheating. Therefore failure occurrence rate is higher. One of the objectives of this paper is to study the motor fail tolerance of this kind of helicopters.

Quadrotors are not the only multirotor design; other designs can be used as an alternative looking for better stability, fault tolerance or stronger lift force. One alternative design is Hexrotor, a multirotor helicopter that uses six motors in a star structure (Fig. 1). The objective of this paper is to study the robust stability and performance of Quad and Hexrotor. The structured singular value (SSV, μ_Δ) is used to test the robust stability and performance of these prototypes. SSV is a classical stability tool but with the advance of computers it can be applied to new problems [6–8] in our case, multirotor helicopters.

Many control techniques have been designed, from classical PID-based control to non-linear ones, a comparison of control techniques including nested saturations, backstepping and sliding modes can be found in [9, 10]. In [2] a switching model predictive controller is designed to achieve precise trajectory control, under the presence of forcible wind gusts. In [3] a dynamic inversion to linearise the system and control the selected output variables is done.

Robust framework hasn't been applied extensively to Quadrotor or similar structures, but some examples can be found in [11] where an L1 optimal control is implemented. The objective of this controller is to minimise effects of unknown plant parameters and disturbances. In [12] a mixed robust feedback linearisation with linear GH$_\infty$, controller is applied to a Quadrotor, analysing the worst case of control law design. Performance issues of the controller are illustrated in a simulation study that takes into account parameter uncertainties and external disturbances as well as measurement noise. In [13] tracking of trajectory and yaw is achieved using a feedback control system considering parametric uncertainly.

In this paper a model-based H$_\infty$ controller is designed for two kinds of helicopters Quad and Hex rotor. This design is validated using SSV ($\mu\Delta$) to test the robustness and performance of prototypes. Controllers are tested in simulation where one of its motors fails and the controller should maintain the prototype flying. This design is also tested on a real Hex rotor prototype.

The paper is organised as follows. In Section 2 the two prototypes are described including the sensorial system; in Section 3 a model of the Quad and Hexrotor is presented; in Section 4 the control structure is explained; in Section 5 the theoretical aspects of SSVs are presented and applied to calculate stability and performance of the prototypes; in Section 6 the design of the robust controller used in prototypes is presented. Section 7 presents comparative results between Quadrotor and Hexrotor, and Sections 8 and 9 describe different flight tests with different failures in the motors.

2 Prototype description

Hand crafted multirotors are composed of the following parts: cross-shaped tubes, four (Quadrotor) or six (Hexrotor) motors with blades placed at ends of the cross-shaped tubes and a board containing electronic components. Chassis are made of aluminium because
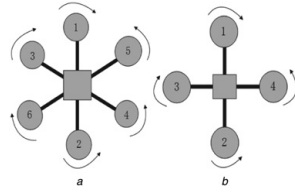
134

Fig. 1 *Multirotor helicopter that uses six motors in a star structure*
a Hexrotor structure
b Quadrotor structure

of its low weight. Total lift force for Quadrotor and Hexrotor is the same, hence motors of Quadrotor have more power and weight than Hexrotor.

The sensorial system of the helicopters is composed by accelerometers, gyroscopes, a multi-degree magnetometer and an ultrasonic device. Gyroscopes are incremental devices; consequently, errors produced by these sensors accumulate over time. In contrast to this, ultrasound devices, accelerometers and magnetometer are absolute sensors. A Kalman filter is used to merge the information coming from these devices to get a pose. Many authors have used Kalman filters as a standard sensor fusion technique [14]. In the case of multirotor micro-helicopters each type of sensor measures different state variables, hence a reduced-order Kalman filter is implemented.

Data from gyroscopes is combined to accelerometers to estimate pitch and roll angles. Accelerometer measurements are low pass filtered to reduce vibrational noise; this reduces the frequency response of sensors and increases accuracy. Procedure for calculating orientation (yaw) is analogous using a magnetometer. Prototype altitude is calculated using an ultrasonic sensor.

The Kalman filter provides to control system angular position and altitude and estimates accuracy. All this information is available at the highest frequency of 250 Hz, which corresponds to gyroscopes.

3 Micro-helicopter model

Taking into account all forces and effects that appear in helicopters, a non-linear dynamic model is obtained [14, 15]. To simplify this non-linear model, just for design purposes, some assumptions are made. The most important one concerns to the variations of the angles between the prototype and ground reference systems. These variations are assumed to be small enough hence this non-linear contribution can be neglected. In addition to this, some other non-linear effects with a negligible impact on the system's dynamic behaviour are ignored. Most relevant are ground and gyroscopic effects, and variations in battery charge. This simplified model is used only for controller design; a complete non-linear model is used in simulations and tests. The dynamic general behaviour of the micro-helicopters is described in (1). Equations without subscript are common to Quad and Hex, and the specific equations have the Hex or Quad subscript

$$
\begin{aligned}
\ddot{x} &= \frac{(F_z - k_x \dot{x})}{m} \sin(\theta)\cos(\psi) \\
\ddot{y} &= \frac{(F_z - k_y \dot{y})}{m} \sin(\phi)\cos(\psi) \\
\ddot{z} &= g - \frac{(F_z - k_z \dot{z})}{m} \cos(\phi)\cos(\theta)
\end{aligned}
\tag{1}
$$

where θ, φ, ψ represent the pitch, roll and yaw angles, x, y, z micro-helicopter position, m mass, g the gravitational force, F_z

force generated by propellers and k_x, k_y, k_z are air friction coefficients. The linear rotational dynamics corresponds to the (2)

$$
\begin{aligned}
I_\theta \ddot{\theta} &= dF_\theta - k_\theta \dot{\theta} \\
I_\phi \ddot{\phi} &= dF_\phi - k_\phi \dot{\phi} \\
I_\psi \ddot{\psi} &= dF_\psi - k_\psi \dot{\psi}
\end{aligned}
\tag{2}
$$

where $\boldsymbol{I}_{\theta,\varphi,\psi}$ is the moment of inertia and d the distance from the propellers to the centre of the prototype, k_θ, k_ϕ and k_ψ are air friction rotational coefficients and $F_{\theta,\varphi,\psi}$ are forces generated by propellers in each axis

$$
I = \begin{bmatrix} I_\theta & 0 & 0 \\ 0 & I_\varphi & 0 \\ 0 & 0 & I_\psi \end{bmatrix}; \quad
I_{\text{Quad}} = \begin{bmatrix} 2m_{\text{Q}}D^2 & 0 & 0 \\ 0 & 2m_{\text{Q}}D^2 & 0 \\ 0 & 0 & 4m_{\text{Q}}D^2 \end{bmatrix};
$$

$$
I_{\text{Hex}} = \begin{bmatrix} 3m_{\text{H}}D^2 & 0 & 0 \\ 0 & 3m_{\text{H}}D^2 & 0 \\ 0 & 0 & 6m_{\text{H}}D^2 \end{bmatrix}
\tag{3}
$$

Inertia matrix presented in (3) is taking from the axis placed in the centre of the prototype for Quad and Hex multirotor where m_{Q} is the mass of each propeller in the quad and m_{H} in the hex.

With small angular movements, the main non-linearity of the model is because of the rotor quadratic dynamics. For each of the micro-helicopter's propellers, the main force generated by propeller can be divided in two different components, drag (F_{id_i}) and thrust (F_{it_i}) force where i represents the index to indicate each rotor (4). Thrust force is the component that lifts the helicopter, and drag force is the component that spins the helicopter

$$
\begin{aligned}
F_{it} &= A_{t_i}(\omega_i)^2 + B_{t_i}(\omega_i) + C_{t_i} \\
F_{id} &= A_{d_i}(\omega_i)^2 + B_{d_i}(\omega_i) + C_{d_i}
\end{aligned}
\tag{4}
$$

where A, B and C represent blade efficiency constants for each force and ω_i is the rotational speed of motor i. These parameters are experimental measurement in the prototype and adjusted to a quadratic expression. Relationship between motor input voltage and developed force is given by (5)

$$
Gm_i(s) = \frac{\omega_i}{\text{Voltage}_i} = \frac{K_{m_i}}{T_{m_i}s + 1}
\tag{5}
$$

where K_{m_i} and T_{m_i} denote motor static gain and time constant, respectively. These parameters are slightly different for each motor.

For Quad and Hex multirotor F_θ, F_ϕ and F_ψ are forces applied on each axis. These forces are defined in (6), where α_{Hex} is the angle separation between the rotors in Hexrotor, in this case 30°

$$
\begin{aligned}
F_z(\text{Quad}) &= F_{1t} + F_{2t} + F_{3t} + F_{4t} F_z(\text{Hex}) \\
&= F_{1t} + F_{2t} + F_{3t} + F_{4t} + F_{5t} + F_{6t} \\
F_\theta(\text{Quad}) &= F_{1t} - F_{2t} F_\theta(\text{Hex}) \\
&= F_{1t} - F_{2t} + (F_{3t} + F_{5t} - F_{4t} - F_{6t})\sin(\alpha_{\text{hex}}) \\
F_\phi(\text{Quad}) &= F_{3t} - F_{4t} F_\phi(\text{Hex}) \\
&= (F_{3t} + F_{6t} - F_{4t} - F_{5t})\cos(\alpha_{\text{hex}}) \\
F_\psi(\text{Quad}) &= F_{1d} - F_{2d} + F_{3d} - F_{4d} F_\psi(\text{Hex}) \\
&= F_{1d} - F_{2d} + F_{3d} - F_{4d} + F_{5d} - F_{6d}
\end{aligned}
\tag{6}
$$

Size, weight and power of the Quadrotor propellers are bigger than Hex ones to obtain the same total lift force for Quad and Hex.

4 Micro-helicopter non-linear analysis

In this section, a control system for micro-helicopters is presented. The model presented in previous section is non-linear and with
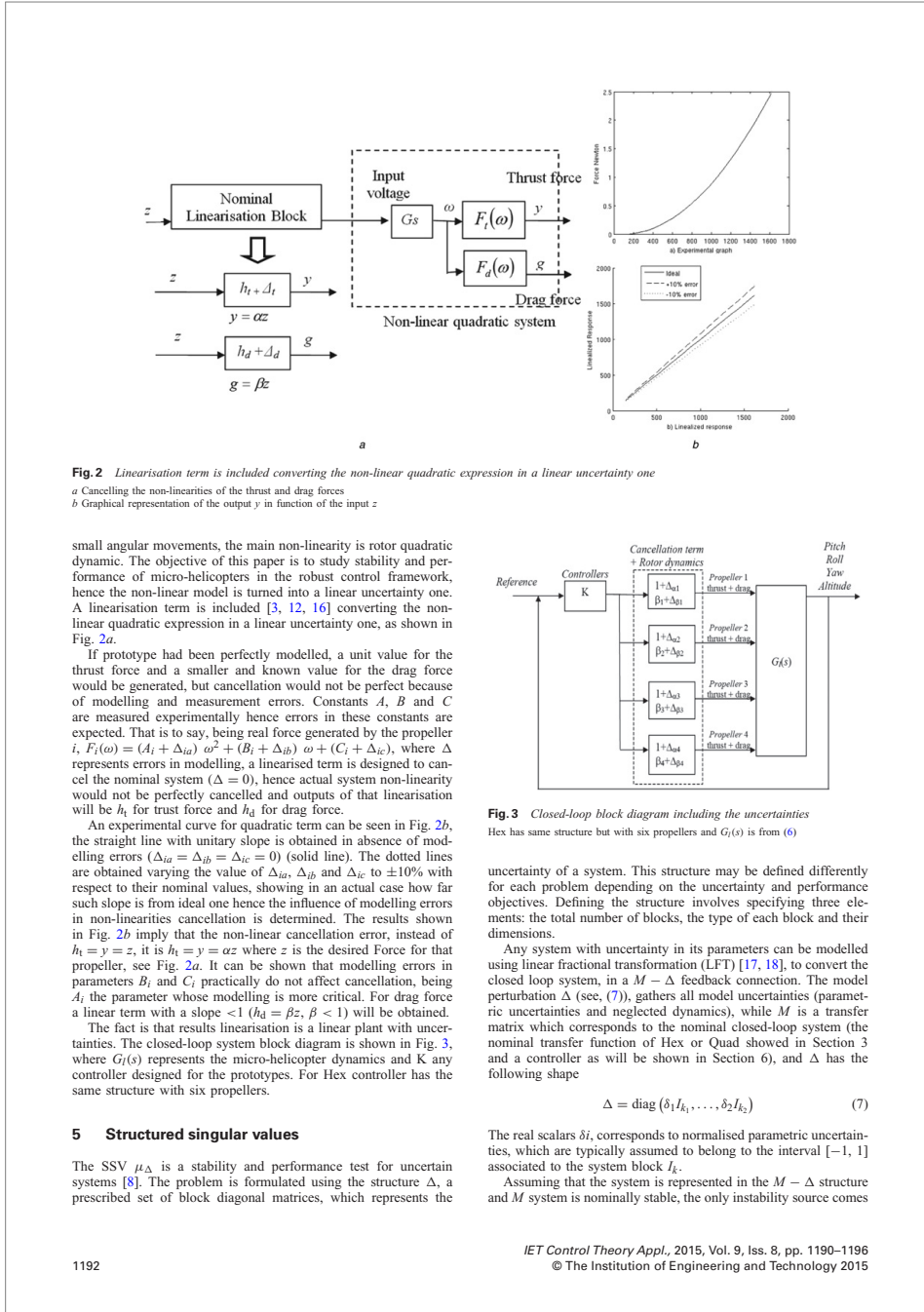
Fig. 2 *Linearisation term is included converting the non-linear quadratic expression in a linear uncertainty one*
a Cancelling the non-linearities of the thrust and drag forces
b Graphical representation of the output *y* in function of the input *z*

small angular movements, the main non-linearity is rotor quadratic dynamic. The objective of this paper is to study stability and performance of micro-helicopters in the robust control framework, hence the non-linear model is turned into a linear uncertainty one. A linearisation term is included [3, 12, 16] converting the non-linear quadratic expression in a linear uncertainty one, as shown in Fig. 2a.

If prototype had been perfectly modelled, a unit value for the thrust force and a smaller and known value for the drag force would be generated, but cancellation would not be perfect because of modelling and measurement errors. Constants A, B and C are measured experimentally hence errors in these constants are expected. That is to say, being real force generated by the propeller i, $F_i(\omega) = (A_i + \Delta_{ia})\ \omega^2 + (B_i + \Delta_{ib})\ \omega + (C_i + \Delta_{ic})$, where Δ represents errors in modelling, a linearised term is designed to cancel the nominal system ($\Delta = 0$), hence actual system non-linearity would not be perfectly cancelled and outputs of that linearisation will be h_t for trust force and h_d for drag force.

An experimental curve for quadratic term can be seen in Fig. 2b, the straight line with unitary slope is obtained in absence of modelling errors ($\Delta_{ia} = \Delta_{ib} = \Delta_{ic} = 0$) (solid line). The dotted lines are obtained varying the value of Δ_{ia}, Δ_{ib} and Δ_{ic} to $\pm10\%$ with respect to their nominal values, showing in an actual case how far such slope is from ideal one hence the influence of modelling errors in non-linearities cancellation is determined. The results shown in Fig. 2b imply that the non-linear cancellation error, instead of $h_t = y = z$, it is $h_t = y = \alpha z$ where z is the desired Force for that propeller, see Fig. 2a. It can be shown that modelling errors in parameters B_i and C_i practically do not affect cancellation, being A_i the parameter whose modelling is more critical. For drag force a linear term with a slope <1 ($h_d = \beta z$, $\beta < 1$) will be obtained.

The fact is that results linearisation is a linear plant with uncertainties. The closed-loop system block diagram is shown in Fig. 3, where $G_l(s)$ represents the micro-helicopter dynamics and K any controller designed for the prototypes. For Hex controller has the same structure with six propellers.

5 Structured singular values

The SSV μ_Δ is a stability and performance test for uncertain systems [8]. The problem is formulated using the structure Δ, a prescribed set of block diagonal matrices, which represents the
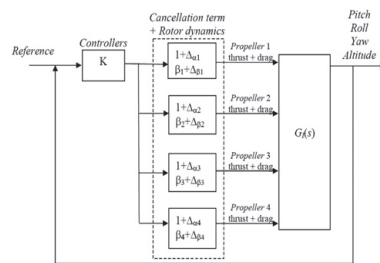
Fig. 3 *Closed-loop block diagram including the uncertainties*
Hex has same structure but with six propellers and $G_l(s)$ is from (6)

uncertainty of a system. This structure may be defined differently for each problem depending on the uncertainty and performance objectives. Defining the structure involves specifying three elements: the total number of blocks, the type of each block and their dimensions.

Any system with uncertainty in its parameters can be modelled using linear fractional transformation (LFT) [17, 18], to convert the closed loop system, in a $M - \Delta$ feedback connection. The model perturbation Δ (see, (7)), gathers all model uncertainties (parametric uncertainties and neglected dynamics), while M is a transfer matrix which corresponds to the nominal closed-loop system (the nominal transfer function of Hex or Quad showed in Section 3 and a controller as will be shown in Section 6), and Δ has the following shape

$$\Delta = \mathrm{diag}\left(\delta_1 I_{k_1}, \ldots, \delta_2 I_{k_2}\right) \tag{7}$$

The real scalars δ_i, corresponds to normalised parametric uncertainties, which are typically assumed to belong to the interval $[-1, 1]$ associated to the system block I_k.

Assuming that the system is represented in the $M - \Delta$ structure and M system is nominally stable, the only instability source comes

136

Fig. 4 *Design of the H∞ controller for the four-rotor micro-helicopter*
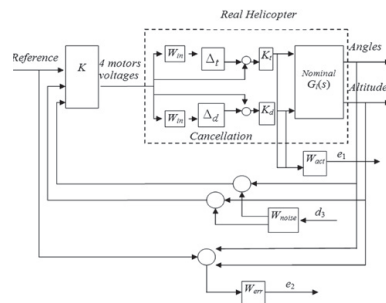Hex presents same structure but with six motors. $G_l(s)$ is from (6) for Quad or Hex

from the Δ closed loop, hence measuring the minimum change on Δ which makes unstable the whole system, a robust stability margin in function of uncertainty Δ can be calculated. To calculate this margin, μ_Δ is defined in the following way:

Definition: For $M \in C^{nxn}$, $\mu_\Delta(M)$ is defined as

$$\mu_\Delta(M) = \frac{1}{\min\{\bar{\sigma}(\Delta) : \Delta \in \Delta, \det(I - M\Delta) = 0\}} \quad (8)$$

unless no $\Delta \in \Delta$ makes $(I - M\Delta)$ singular, in which case $\mu_\Delta(M) = 0$.

$\mu_\Delta(M)$ is a measurement of the small perturbation size who converts the full system unstable, for a stable $M - \Delta$ structure, $\mu_\Delta(M) < 1$. An upper bound of μ_Δ gives a sufficient condition of robustness; a lower bound gives a sufficient condition for when robustness will not be met.

Change on performance generated by the uncertainty of a system can be measured using $\mu_\Delta(N)$, where N is an augmented uncertain system. The closed loop uncertain system is augmented with an output uncertainty Δ_p, placed at the output of the plant, to test the performance characteristics under internal and external perturbations. The change of S (sensitivity function) in function of those uncertainties is a good estimation of system performance. To calculate this change, $\mu_\Delta(N)$ is used, where the upper bound guarantees robustness performance of a LFT for perturbations up to a certain size, and a lower bound exhibits perturbations which cause performance degradation in the perturbed system.

The calculation of exact μ_Δ is a NP hard problem, hence usually some conservative considerations are made to get a solution in a polynomial time. Many algorithms have been proposed in the literature to calculate μ_Δ upper bound [17].

$\mu_\Delta(M)$ is not only a system robustness and performance analysis tool, it can be used as a controller synthesis method. The most used algorithm based on $\mu_\Delta(M)$ is D K-iteration [19], it combines H∞-synthesis and μ-analysis. The objective of this synthesis method is to minimise the peak value over frequency of μ, looking for the most robust controller. Usually a set of weighting functions are used to shape the behaviour of the closed loop system, setting the bandwidth and the characteristics of the controller.

6 Robust controller design

A design based on SSVs is used to implement the schema of Fig. 4. In addition to looking for the optimum controller, this methodology takes into account the expected disturbances to obtain the most stable controller in every possible situation [19]. An algorithm based

on D-K iteration is used to synthesise the controller, where W_{err}, W_{act}, W_{noise} and W_{in} are weighting functions and K the controller. Where

$$W_{err} = K_{err} \frac{1}{T_{err}s + 1}$$
$$W_{act} = K_{act} \frac{1}{T_{act}s + 1}$$
$$W_{noise} = K_{noise} \frac{(1/\text{freq}b_{noise})\,s + 1}{(1/\text{freq}a_{noise})\,s + 1} \quad (9)$$
$$W_{in} = K_{in} \frac{1}{T_{in}s + 1}$$

W_{err} weights the difference between the response of the closed-loop system and the reference value. It is very important to limit the system bandwidth to avoid possible instabilities. An upper frequency of 1 rad/s is used in this design.

W_{act} weights the commands applied to the plant. A low-pass filter is used to restrict the effect of the commands over the plant oscillations.

W_{noise} weights the sensor response. This term is chosen in such a way that it properly models the sensor noise behaviour. It is designed as a high-pass filter defined between 10 and 100 rad/s.

W_{in} weights the uncertainty to which the nominal linear plant $G_l(s)$ is subjected.

To find the appropriate values for the weight function parameters, a first valid controller is designed using the micro-helicopter model. Taking this controller as the starting point, a sweep of the parameters is carried out so as to find those which yield the best closed-loop system behaviour. The minimum value of the error corresponds to the following H∞ controller parameters: $K_{err} = 25$, $T_{err} = 0.063$ rad/s; $K_{act} = 10$, $T_{act} = 0.063$ rad/s; $K_{noise} = 0.3$ rad/s, freq $a_{noise} = 10$ rad; freq$b_{noise} = 100$ rad; $K_{in} = 0.3$, freq$a_{in} = 10$ rad, freq$b_{in} = 30$ rad.

7 Hex Quad comparative

The SSV is used as a robust stability and performance test to compare the two structures previously defined (Quadrotor and Hexrotor). The uncertain closed-loop system generated using the controller synthesis presented in Section 6, is tested using $\mu_\Delta(M)$ to calculate what structure is more robust. D-K iteration generates, based on the model, a robust controller for each micro-helicopter structure, hence the difference between robustness and performance depends only on the structure itself, represented by the transfer function of Hex and Quad showed in Section 3.

The two helicopters structure (Hex and Quad) have the same equivalent total force in their motors (the total force of all the motors is 1.8 Kg). Average force of each Hex motor is: 310 and 465 g. for Quad, both structures have the same motor power conditions as a starting point.

To carry out the robust stability and performance study for the Quadrotor, the first to be tested is nominal stability. Thus, the closed-loop system poles are computed testing that all of them are in the left-hand side of the complex plane. The SSVs of the Quadrotor are computed and shown in Fig. 5a. As said in Section 5, for a stable closed loop system (M), the only source of instability is uncertainty (Δ) which is set to 40%, and the stability margin can be calculated using structured singular vaules $\mu_\Delta(M)$. The robust stability margin is the reciprocal of the upper bounds of $\mu_\Delta(M)$. The maximum of the largest $\mu_\Delta(M)$ is 0.90 at a frequency of 7.5 rad/s as seen in Fig. 5a, corresponding to an uncertainty tolerance of 111%. The uncertainty is set to 40%, hence the controller is stable to an uncertainty of the 44% in a change of parameters.

In Fig. 5b the SSVs for the Hexrotor are presented. All poles of the closed-loop system are placed on the left-hand side half
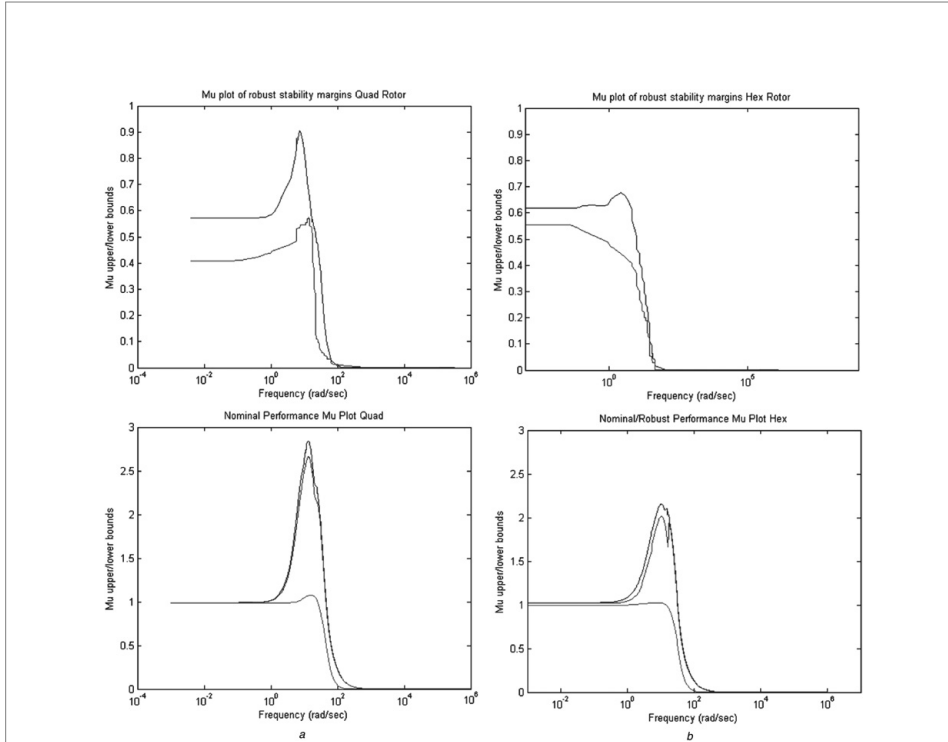
1193

137

Fig. 5 *Largest SSV of the system is in solid, and the smallest in dashed*
a Quadrotor
b Hexrotor

of the complex plane, hence the system is nominally stable. The maximum value of the SSV is 0.69 at a frequency of 3.18 rad/s, corresponding to an uncertainty tolerance of 147%. The initial uncertainty is set to 40%, hence the system is stable to a total change of 58.8% in its parameters. Results show that hex rotor is more robustly stable than quad rotor in all frequencies, hence hex rotor is able to tolerate higher perturbations and remain on flight.

The frequency response of largest and smallest SSVs of performance matrix $\mu_{\Delta}(N)$, presented in Section 5, is used to test the system performance. In Fig. 5 down, dotted line represents the nominal performance as a sensitivity function, as is expected performance is close to 1 at low frequency, which means good reference tracking. In solid and dashed lines, the largest and the smallest singular values are presented, showing the performance change in the sensitivity function because of uncertainty. At high frequencies $\mu_{\Delta}(N)$ increase decreasing performance because of uncertainty.

The peak value is close to 2.8 in Quadrotor, meaning that sensitivity function can be about 2.8 times larger than required because of changes in parameters. In Hexrotor the peak value is of 2.1, showing loose of performance but better than Quadrotor.

The result based on the μ_{Δ} analysis shows that Hexrotor presents better robustness and performance with respect to Quadrotor. The controllers generated by both micro-helicopters are based on a D-K iteration getting the best controller in a robustness sense, hence results are independent to the controller itself, it depends only on helicopter structure.

8 H_{∞} controller efficiency tests

To determine results of controllers, different types of motor failure during a stable flight are analysed. As starting point, a test flight is made from taking off to hovering, in this and the following tests all non-linearities presented in the model are covered, including quadratic propeller force, actual euler angle behaviour, ground effect, gyroscopic effect of propellers and structure, air friction, saturation of commands and so on. In Fig. 6 micro-helicopters output is shown, where the objective is hovering in the position ($X = 0$, $Y = 0$, $Z = -1$). This is the nominal behaviour for prototypes and it will be used as comparative. To perform the motor failure study, two adverse situations are considered, in the first one, micro-helicopter is in stable flight and suddenly one of the motors fails for a short time. In the second one of the motors fails indefinitely. In these two cases the controller should be able to recover the system from the failure.

8.1 Case 1: one of the motors fails in stable flight for a short period of time

The first case considers failure of one helicopter motor for a short period of time. The system recovers from the temporary loss of motor power between seconds 15 and 16. As expected, the highest amplitude oscillations in the system response occur during
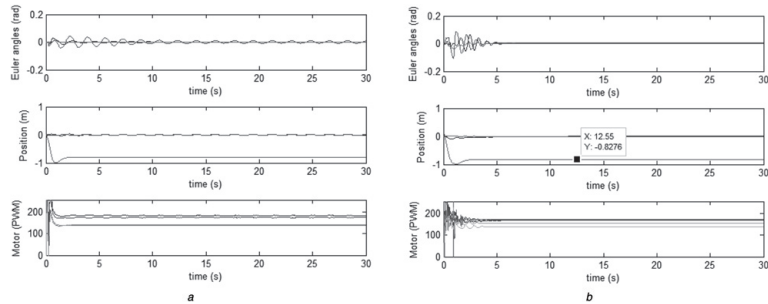
138

Fig. 6 *Stable flight from taking off to hovering*

a Quadrotor and
b Hexrotor
The top figure represent the Euler angles evolution, the middle one the Cartesian coordinates (*X*, *Y*, *Z*) and the bottom one the command applied to the motors for both prototypes Quad and Hexrotor
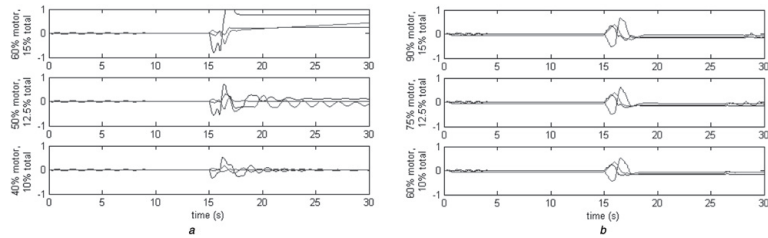

Fig. 7 *Failure of one motor for 1 s with different failure level*

a Quadrotor and
b Hexrotor
Each sub-plot represents the orientation (roll solid, pitch dash, yaw dot)
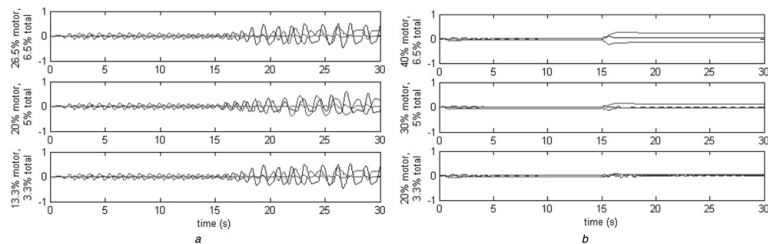
Fig. 8 *Failure of one motor for a long time*

a Quadrotor and
b Hexrotor

failure but that oscillations don't drive the system to instability. The main problem associated is command saturation, in fact, this problem appears in all the helicopter stability tests. The Cartesian position is controlled too and just a displacement of less

1 m is generated by this failure. When the motor recovers its power, the system recovers its pose. Hexrotor shows little oscillation and less time to recover than Quadrotor, as theoretical results predict.

1195

139

Appendix F. Stability and performance analysis of unmanned aerial vehicles: Quad vs. Hex

Table 1 Test resume

maximum tolerable failure %	Quadrotor		Hexrotor	
	motor failure	total failure	motor failure	total failure
short motor failure	50 00%	12.5%	90 00%	15%
long motor failure	30 00%	7.5%	50 00%	8.3%

Figure 7 shows, how helicopter behaviour deteriorates when motor 4 loses power, fail level (as total percentage fail and specific motor fail) is shown in the graph for each simulation. Hexrotor is more robust to motor fail, it can even resist a 90% fail for 1 s in one motor (equivalent to 15% of total power fail). In opposite, motor fail bigger than 60% in the Quadrotor (15% of total) drive the system to instability.

8.2 Case 2: one of the motors fails permanently in stable flight

In the second case, a permanent failure in one of the helicopter motors occurs while hovering.

The prototype is more sensitive to this kind of failure, allowing for less motor faults than the first case. In Fig. 8 a test with multiple failures is shown. Hexrotor is more robust to this kind of motor fail than Quadrotor. The maximum motor failure before instability in Quadrotor is 30% (7.5% total) and in Hexrotor is 50% (8.3% total). A final test result resume is shown in Table 1.

9 Real experiments of the micro-helicopter robust stability

Two videos of the Hexrotor in flight can be seen in the following URLs: http://www.youtu.be/sGKO-9acEZA and http://www.youtu.be/lfOC9WtcGWs. In video 1, multiple short failures occur and the controller maintains the prototype in flight. In the video 2 a long failure in one of the motors occurs and the prototype is able to come back to hovering. The same failures in Quadrotors will be irrecoverable.

10 Conclusions

In this paper a comparative of a Quadrotor and Hexrotor micro-helicopter is made. This comparative is made in the robust control framework, hence a linear uncertainty model is calculated based on the actual non-linear model of the prototypes. Sensorial reconstruction is based on an extended Kalman filter to obtain an accurate orientation representation. A robust control system based on H_∞ optimisation is made and the SSV is used as stability and performance test to compare the two structures described. Result of SSV shows that Hexrotor is intrinsically more stable than Quadrotor. This result is tested in a series of flights in which a motor fails. These tests confirm the theoretical result of better stability of Hexrotor than Quadrotor.

Some applications of multirotors include flying for a long time hence fails in propellers happen. Stability and robustness

characteristics of different types of multirotors can be the key factor to select a multirotor structure. Hexrotor presents better stability and performance characteristics than Quadrotor, and other advantages like more lift power using the same propellers than Quadrotor. It is a very interesting structure for Unmanned Aerial Vehicles with similar size, weight and building price than Quadrotor but better performance and stability characteristics.

11 Acknowledgements

12 References

1 Bouabdallah S., Becker M., Siegwart R.: 'Autonomous miniature flying robots: coming soon! – research, development, and results', *IEEE Robot. Autom. Mag.*, 2007, 14, (3), pp. 88–98
2 Tzes A., Nikolakopoulos G., Alexis K.: 'Model predictive quadrotor control: attitude, altitude and position experimental studies', *IET Control Theory Appl.*, 2012, 6, (12), pp. 1812–1827
3 Das A., Subbarao K., Lewis F.: 'Dynamic inversion with zero-dynamics stabilisation for quadrotor control', *IET Control Theory Appl.*, 2009, (August 2008), pp. 303–314
4 García-Delgado L., Dzul A., Santibáñez V., Llama M.: 'Quad-rotors formation based on potential functions with obstacle avoidance', *IET Control Theory Appl.*, 2012, 6, (12), pp. 1787–1802
5 Michael N., Mellinger D., Lindsey Q., Kumar V.: 'The GRASP multiple micro-UAV testbed', *IEEE Robot. Autom. Mag.*, 2010, 17, (3), pp. 56–65
6 Li Y., Sun Y., Dai X.: 'μ-synthesis for frequency uncertainty of the ICPT system', *IEEE Trans. Ind. Electron.*, 2013, 60, (1), pp. 291–300
7 Sariyildtz E.: 'Robust stability anaylsis of the system with disturbance observer: the real parametric uncertainty approach II'. IECON 2012 – 38th Annual Conf. IEEE Industrial Electronics Society, 2012, pp. 2470–2475
8 Engelken S., Patra S., Lanzon A., Petersen I.R.: 'Stability analysis of negative imaginary systems with real parametric uncertainty – the single-input single-output case', *IET Control Theory Appl.*, 2010, 4, (11), p. 2631
9 Garcia Carrillo R.L.L.R., Dzul A.: 'Hovering quad-rotor control?: a comparison of nonlinear controllers using visual feedback', *IEEE Trans. Aerosp. Electron. Syst.*, 2012, 48, (4), pp. 3159–3170
10 Ton C., Mackunis W.: 'Robust attitude tracking control of a quadrotor helicopter in the presence of uncertainty'. IEEE Conf. on Decision and Control, 2012, pp. 937–942
11 Satici A.C., Poonawala H., Spong M.W.: 'Robust optimal control of quadrotor UAVs', *IEEE Access*, 2013, 1, pp. 79–93
12 Mokhtari B.D.A., Banallegue A.: 'Robust feedback linearization and GH controller for a quadrotor unmanned aerial vehicle'. IEEE Int. Conf. on Intelligent Robots and Systems, 2005, pp. 1009–1014
13 DongBin Lee B.X., Burg T.C., Dawson D.M., Shu D.: 'Robust tracking control of an U nderactuated Quadrotor aerial-robot based on a parametric uncertain model'. IEEE Int. Conf. on Systems, Man and Cybernetics, 2009, no. October, pp. 3187–3192
14 Zhou Q.-L., Zhang Y., Qu Y.-H., Rabbath C.-A.: 'Dead reckoning and Kalman filter design for trajectory tracking of a quadrotor UAV'. Proc. of 2010 IEEE/ASME Int. Conf. on Mechatronic and Embedded Systems and Applications, 2010, pp. 119–124
15 Hoffmann G.M., Huang H., Waslander S.L., Tomlin C.J.: 'Precision flight control for a multi-vehicle quadrotor helicopter testbed', *Control Eng. Pract.*, 2011, 19, (9), pp. 1023–1036
16 Toledo J., Acosta L., Sigut M., Felipe J.: 'Stabilisation and altitude tracking of a four-rotor microhelicopter using the lifting operators', *IET Control Theory Appl.*, 2009, 3, (4), p. 452
17 Packard A., Doyle J.: 'The complex structured singular value', *Automatica*, 1993, 29, (1), pp. 71–109
18 Postlethwaite I., Kim J., Bates D.G.: 'A geometrical formulation of the μ-lower bound problem', *IET Control Theory Appl.*, 2009, 3, (4), pp. 465–472
19 Skogestad S., Postlethwaite I.: 'Multivariable feedback control: analysis and design' (Wiley, 2007), p. 608

140

VERDINO, prototipo eléctrico de vehículo autoguiado

This appendix includes the full text for the following article:

141

VERDINO, PROTOTIPO ELECTRICO DE VEHICULO AUTOGUIADO

L. Acosta, J. Toledo, A. Hamilton, R. Arnay J. Espelosin, N. Morales, D. Perea, L. Moreno
lacosta@ull.es, jttoledo@ull.es, rafa@isaatc.ull.es, jespelosin@isaatc.ull.es, nestor@isaatc.ull.es, dani@isaatc.ull.es, lorenzo@isaatc.ull.es
Dpto. Ing. de Sistemas y Automática, Arq. y Tec. de Comp., Univ. de La Laguna

Resumen

En este artículo se describe el prototipo Verdino, un vehículo eléctrico auto guiado. Se presenta la adaptación necesaria para convertir el vehículo eléctrico convencional, en una plataforma preparada para ser controlada por un ordenador. También se explica el sistema sensorial del que se ha dotado al prototipo y los algoritmos principales que se utilizan. El prototipo está basado en el sistema operativo ROS (Robotic Operative System), una plataforma de desarrollo de aplicaciones específicamente diseñada para robótica, lo que acelera el tiempo de desarrollo del prototipo. Por último se muestran una serie de resultados que muestran las capacidades actuales del prototipo.

Palabras Clave: Vehículo auto guiado, telémetros laser, reconstrucción del entorno.

1   INTRODUCCION

El siguiente artículo describe el prototipo "Verdino" un carrito eléctrico de los utilizados en los campos de golf y modificado para ser capaz de conducir de manera autónoma. El desarrollo se enmarca en el proyecto de investigación SAGENIA, DPI2010-18349. Este proyecto trata de resolver el transporte interno de una urbanización promovida por el Cabildo Insular de Tenerife y el Instituto Tecnológico de Energías Renovables (ITER http://www.iter.es). Se trata de 25 viviendas unifamiliares ecológicas http://casas.iter.es/, diseñadas para ser respetuosas con el medio ambiente, toda la energía necesaria proviene de fuentes renovables y el diseño de las viviendas permite el ahorro en calefacción y aire acondicionado. Se muestra una fotografía de la urbanización en la Figura 1. El objetivo final de la urbanización es que las viviendas sean totalmente autónomas en todos los sentidos, incluyendo el transporte.

Siguiendo la filosofía de este proyecto, el sistema de transporte debe ser respetuoso con el medio ambiente, por eso se ha elegido un vehículo eléctrico cuyas baterías pueden ser recargadas utilizando energías renovables, convirtiendo el vehículo en no contaminante.

El modo de funcionamiento del transporte será bajo demanda, los habitantes de la urbanización bioclimática serán los usuarios de este transporte público, existirá una parada en cada vivienda y se podrá solicitar el vehículo como si fuera un taxi. Una vez el usuario esté en el vehículo se le podrá indicar el destino, que puede ser otra vivienda o el centro de visitantes donde se encuentra el aparcamiento de los vehículos convencionales de la urbanización.

Figura 1: Mapa del ITER en Tenerife con la urbanización de viviendas bioclimáticas.

En este artículo se describirá el vehículo elegido para cumplir esta función, las modificaciones mecánicas que se le han realizado, la electrónica desarrollada, los sensores de los que dispone y el software necesario para permitir su control en tiempo real.

2   DESCRIPCIÓN DEL VEHÍCULO

Debido al entorno en el que se desenvolverá el vehículo autoguiado que se presenta en este trabajo: una urbanización de viviendas bioclimáticas, y al espíritu no contaminante del ITER, se ha optado por utilizar un vehículo eléctrico. De entre los que se encuentran en el mercado, se ha llegado a la conclusión de que los que mejor se adaptan a las

729

142

características requeridas en este proyecto son los empleados en los campos de golf. Por este motivo, se ha elegido el modelo TXT-2 de EZGO. En la figura 2 se presenta una fotografía del vehículo utilizado como transporte.
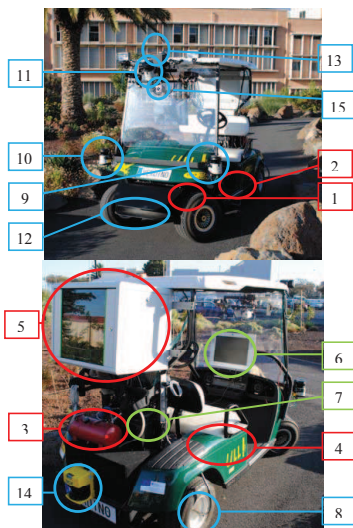

Figura 2: Fotografía del prototipo Verdino, un carrito de golf TXT-2 de EZGO.

Algunas de las características más significativas del EZGO TXT-2 son las que se relacionan a continuación. En la tabla I se describen brevemente las características principales del prototipo incluyendo su sistema de actuación, su sistema sensorial y el ordenador instalado en él.

- Velocidad Max: 19 – 23 km/h
- Peso: 293 kg, sin baterías
- Largo: 2.67 m
- Ancho: 1.18 m
- Altura: 1.23 m hasta el volante.

Tabla I: Resumen de modificaciones realizadas en el prototipo

Índice Imagen	Tipo	Características
1	Sistema de actuación	Motor de CC con encoder acoplado a la dirección.
2	Sistema de actuación	Sistema de frenado basado en pistón neumatico
3	Sistema de actuación	Compresor neumático
4	Sistema de actuación	Electrónica de control de tracción.
5	Sistema de actuación	Armario donde se sitúa la electrónica del vehículo
6	Ordenador de a bordo	Pantalla ordenador a bordo.
7	Ordenador de a bordo	Ordenador central.
8	Sistema Sensorial	Sistema odométrico para el cálculo de posición y velocidad
9	Sistema Sensorial	Telémetro láser corto alcance izquierdo
10	Sistema Sensorial	Telémetro láser corto alcance derecho
11	Sistema Sensorial	Telémetro láser corto alcance vertical
12	Sistema Sensorial	Telémetro láser largo alcance
13	Sistema Sensorial	GPS diferencial
14	Sistema Sensorial	Telemétro laser detección obstáculos trasero
15	Sistema Sensorial	Cámaras visible e infrarojo

3 SISTEMA DE ACTUACIÓN DEL PROTOTIPO

El primer paso que se ha dado para automatizar el vehiculo es adaptar los controles básicos del coche para que puedan ser manipulados por un ordenador. Se ha modificado la dirección, el control de la tracción y el freno, convirtiendo el vehículo comercial en un sistema totalmente controlable desde un ordenador. Otro objetivo buscado con esta modificación es el de no perder la capacidad de conducir el vehículo de manera manual. El prototipo aquí descrito permite conducirse con normalidad por un usuario, o cambiando de modo, puede ser conducido por un ordenador. A continuación se describirán las principales modificaciones mecánicas en el prototipo.

3.1 MODIFICACIÓN DE LA DIRECCIÓN

En el control de la dirección se utiliza un motor de corriente continua que se ha acoplado a la columna de la dirección integrando un encoder incremental para conocer su posición. De esta manera se logra mantener operativo el volante, pudiéndose así manejar el vehículo de forma manual si fuera necesario. Para conmutar el vehículo a modo manual simplemente se inhibe la electrónica de control,

730

143

venciendo la resistencia mecánica que ofrece el motor.

3.2 MODIFICACIÓN DEL FRENO

En el frenado automático del vehículo se emplea un sistema neumático que garantiza una respuesta de frenado muy rápida. Se ha instalado un compresor en la parte trasera del vehículo que proporciona el aire necesario al circuito. En la parte inferior del vehículo se ha instalado un cilindro neumático acoplado directamente al cable de freno que acciona las zapatas del coche. El diseño se ha realizado buscando que el pedal de freno no se deshabilite nunca, siempre será posible frenar el vehículo mecánicamente de forma manual ante una emergencia.

La fuerza de frenado es controlada a través de dos electroválvulas proporcionales que permiten controlar la presión en la cavidad de salida del cilindro neumático, regulando así la fuerza de la frenada.

3.3 MODIFICACIÓN DE LA TRACCIÓN

Para completar la adaptación mecánica y electrónica del prototipo se ha intervenido el sistema de control que viene de serie en el vehículo para poder generar esa señal desde el ordenador de a bordo. Mediante un conmutador se puede seleccionar entre el modo de conducción normal, o el modo automático en el que el conductor es un ordenador.

4 ELECTRÓNICA DE CONTROL

La electrónica de control diseñada es la encargada de gestionar todas las modificaciones mecánicas realizadas en el vehículo. Se trata de una placa con un microcontrolador PIC18f de la casa Microchip, que se comunica con el ordenador de a bordo mediante un enlace RS-232. Gracias a esta electrónica se puede controlar la posición del volante, la acción del freno o el control de la tracción.

El microcontrolador implementa un control PID para fijar la posición final del volante. La consigna de este controlador PID es recibida mediante el enlace RS-232 en un protocolo de comunicaciones creado a este efecto. La parte de adaptación de potencia se realiza mediante un puente en H de mosfet que permite manejar la intensidad necesaria el motor (la intensidad del motor que controla el volante ronda los 14 Amp.).

El control de las válvulas que permiten la acción de frenado y de la electrónica encargada de la tracción se realiza también mediante este microcontrolador a través del protocolo de comunicaciones preestablecido.

Inicialmente la electrónica estaba instalada en el interior del vehículo, pero debido a las dificultades de reparación y mantenimiento se decidió trasladar toda la electrónica a un armario de comunicaciones instalado en la parte trasera del vehículo. Este armario incluye un panel de parcheo instalado con el objetivo de facilitar y ordenar los múltiples cables y conexiones que componen el prototipo. En este armario también están instaladas varias fuentes de alimentación que proporcionan los voltajes necesarios de funcionamiento para los múltiples sistemas que tiene instalado el prototipo. El vehículo tiene 6 baterías de 6 voltios conectadas en serie, lo cual proporciona un voltaje cercano a los 36 voltios y a través de las fuentes de alimentación se consiguen las salidas de 12, 5, 9 y 24 voltios necesarias para el resto de los sistemas.

El prototipo tiene un ordenador instalado a bordo y alimentado a través de las baterías del vehículo. La electrónica también informa al ordenador del estado del vehículo a través del enlace RS-232. Entre la información facilitada se encuentra la velocidad actual de avance, la posición actual del volante, el estado del freno, etc. Gracias a esta información el ordenador puede aplicar la estrategia de control necesaria para fijar la velocidad de avance. El resultado de esta estrategia puede ser acelerar o frenar según las condiciones de la vía y la velocidad deseada.

Por razones de seguridad se ha instalado un sistema de alarma junto al pedal del freno del vehículo. Si se pisa esta alarma, la electrónica detectará esta acción y parará automáticamente el vehículo inhibiendo todo tipo de control automático sobre el prototipo.

La electrónica está integrada en el vehículo convirtiendo el prototipo en un sistema dual que puede ser conducido tanto de forma manual por un usuario como de forma automática por el ordenador de a bordo. El vehículo se convierte en un periférico para el ordenador de a bordo pudiendo controlar completamente su comportamiento a través de la conexión RS232.

5 SISTEMA SENSORIAL

Para poder realizar una conducción automática el prototipo cuenta con un completo sistema sensorial que le permita localizarse en el entorno y detectar obstáculos. Esta información se utilizará para construir un mapa del entorno por el que navegar utilizando distintos algoritmos. Los principales sensores con los que cuenta el prototipo son:

731

144

5.1 ODOMETRIA

Un sistema odométrico, a pesar de su sencillez, aporta muchas ventajas a la localización global de un robot. La primera ventaja es su total autonomía, sin necesidad de referencias externas, para calcular movimientos incrementales. Otra gran ventaja es su gran precisión para movimientos pequeños, aportando una fuente de localización que proporciona un resultado estable.

En la implementación se ha colocado un encoder de 1024 pulsos acoplado a cada rueda trasera del vehículo, las ruedas tienen un diámetro de 40 cm, con lo que la precisión lineal teórica del desplazamiento calculado es menor de 1 mm. Utilizando la información de desplazamiento de cada rueda, se puede calcular la posición (X,Y) en coordenadas cartesianas a partir del punto de origen.

El sistema odométrico aporta un error en la localización muy bajo cuando el movimiento es pequeño, lo cual es muy útil para la convergencia de otros sensores y algoritmos de localización global que no proporcionan una salida estable en el calculo de la posición. Por ejemplo la salida de posición que proporciona el GPS va cambiando con el tiempo debido al error, aunque el prototipo no se mueva, lo que se compensa utilizando la salida del sistema odométrico

5.2 GPS

El GPS es el sistema universal de posicionamiento en exteriores, en nuestro caso contamos con un GPS diferencial JAVAD TRIUMPH-1. El GPS diferencial esta compuesto en realidad de dos dispositivos independientes, un GPS instalado en una posición fija y conocida, al se denomina Base y otro móvil denominado Rover que será el encargado de sensar la posición. La base está monitorizando continuamente su posición y es capaz de detectar los cambios en posición producidos por cambios en la atmosfera, variaciones en los satélites, etc. Esta información que caracteriza el error es enviada a través de un enlace inalámbrico al Rover para que así corrija estos errores y aumente la precisión. El dispositivo de Javad incluye una mejora sobre el sistema DGPS estándar, se trata del protocolo RTK capaz de proporcionar precisiones de 1cm en tiempo real, gracias a la detección y envió de las características atmosféricas de manera interactiva. Las características principales del dispositivo se pueden ver en la tabla II.

Tabla II: Características del GPS instalado en el prototipo

	Horizontal cm	Vertical cm
Precisión estática	0.3	0.5
Precisión cinemática	1cm	1.5cm
Precisión RTK	1mc	1.5cm
Precision DGPS	0.25 Post proceso	0.5 Tiempo real

Para el correcto funcionamiento del GPS se ha instalado una red de comunicaciones inalámbricas basadas en el protocolo Zigbee que permite generar una red mallada en tiempo real que de cobertura a toda la zona de acción del prototipo. De esta manera solamente es necesario colocar repetidores de señal en los lugares donde no exista cobertura para asegurarnos que las correcciones enviadas desde la Base lleguen siempre al Rover.

Sin embargo el GPS tiene asociado una serie de problemas intrínsecos a su funcionamiento y que lo limitan como dispositivo de localización del prototipo. El principal es que para conseguir precisiones menores de 1 metro, es necesaria una visión clara y completa del cielo, lo cual limitaría el uso de este sensor únicamente en exteriores, desde que el vehículo se acerque a un edificio o pierda calidad de señal por efectos meteorológicos, el error crecerá muchísimo pudiendo ser de unos 5m. El GPS informa junto con la posición de una estimación del error en forma de una elipse de posicionamiento, donde podemos calcular el semieje mayor y menor, indicando la fiabilidad del posicionamiento. Esto nos va a servir de indicador para saber qué sistema utilizar en cada momento.

5.3 TELEMETROS LASER

Los telémetros laser se han convertido en uno de los sensores más importantes en la robótica móvil actual. Permiten una medición rápida, fiable y fácil de procesar del entorno. En el caso de Verdino tiene instalado un total de 5 telémetros laser colocados en distintos lugares del prototipo y con distintas orientaciones para sensar distintas características del entorno.

El primero es un SICK LMS-221 instalado en la parte baja del prototipo, tiene una distancia máxima de medida de 30m, una resolución espacial de 10mm y una angular de 0.5º. La función principal de este telémetro es la de detectar obstáculos en la media y larga distancia, así el software de navegación detectará estos obstáculos y buscará una ruta para esquivarlos

732

145

XXXIII Jornadas de Automática. Vigo, 5 al 7 de Septiembre de 2012

Se dispone de dos SICK LMS-111 instalados en los dos laterales del vehículo. Son sensores de distancia media con una medida máxima de 18m, una resolución espacial de 12mm y un ángulo de visión de 270º. Estos sensores son utilizados para los algoritmos de SLAM, que utiliza el vehículo para generar el mapa y localizarse en él. Se han instalado dos para no generar ángulos muertos y poder cubrir todo el perímetro del prototipo.

Otro LMS-111 está instalado en el techo del vehículo con una inclinación de 30º hacia el suelo. Este telémetro sirve para ir escaneando en 3 dimensiones el entorno y poder detectar obstáculos con volumen, aceras, posibles socavones en el área de conducción etc. Gracias a este telémetro se pueden conocer las características reales del entorno cercano del prototipo, no limitándose a una interpretación 2D que puede no ser válida debido a algunos obstáculos con volumen.

Hay un último PLS-211 instalado en la parte trasera del prototipo cuya misión es detectar obstáculos cuando el vehículo circula en marcha atrás.

5.4 UNIDAD DE MEDIDA INERCIAL

El prototipo también dispone de una unidad de medida inercial, concretamente el moduelo MTi de Xsense. Este modelo incluye giróscopos, acelerómetros y magnetometros para realizar una estimación de orientación en 3D. La resolución de inclinación es de 0.5º y la orientación con respecto al norte magnético es de 1º en ambientes homogéneamente magnéticos. Este sensor se utiliza principalmente en combinación con la odometría para calcular la orientación del vehículo, y poder calibrar la orientación inicial.

5.5 CAMARAS

El prototipo también cuenta con una serie de cámaras instaladas en la parte delantera que son utilizadas para distintos algoritmos de visión, desde seguimiento de la carretera en tiempo real para asegurarnos que el prototipo está bien localizado en ella [1], detección de obstáculos basado en imágenes anteriores [7], detección de peatones, etc. Además estas cámaras tienen la posibilidad de modificar su orientación para aumentar el campo de visión [6]. Toda esta información se utiliza combinada junto a los telémetros laser para mejorar la detección de obstáculos y la navegación por el entorno.

Se utiliza también una cámara térmica en el infrarrojo lejano para ayudar a la detección de obstáculos que sean dificilmente detectables en el visible. Una cámara térmica ayuda mucho a detectar a personas y animales, ya que segmentar a estos por su temperatura temporal característica es relativamente sencillo. También nos sirve como filtro para mejorar la segmentación en el visible y así eliminar sombras temporales debido a ocultamientos y condiciones similares [2]. En la figura 3 podemos ver un ejemplo de peatón capturado por la cámara térmica donde destaca claramente del fondo de la carretera y facilita su segmentación.
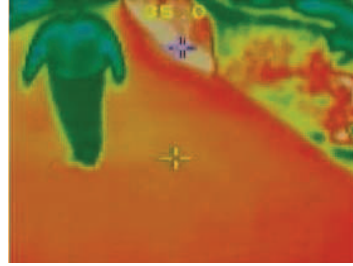

Figura 3: Imagen ejemplo de la cámara termografica.

5 SOFTWARE DE CONTROL

Para integrar todo el sistema sensorial previamente descrito y facilitar el desarrollo del proyecto, se ha decido implementar el software del proyecto utilizando ROS (Robot Operating System). ROS es un entorno de desarrollo orientado a la robótica que proporciona librerías y herramientas para ayudar a los desarrolladores de software a crear aplicaciones para controlar robots. Proporciona abstracción del hardware, controladores de dispositivos, librerías, visualizadores, sistemas de paso de mensajes, mantenimiento de paquetes y más aplicaciones útiles para el desarrollo de software y hardware. Además se trata de un sistema de software libre con distintos repositorios, con lo que se puede utilizar código desarrollado por otros miembros de la comunidad robótica con relativa facilidad.

Una aplicación ROS está basada en el paradigma de paso de mensajes, donde existe una serie de nodos independientes que realizan distintas acciones y publican la información al resto de los nodos para que puedan utilizarla a voluntad. Este esquema es muy flexible ya que permite dinámicamente modificar el comportamiento del sistema cambiando cada nodo por separado. Por ejemplo, cada telémetro láser está procesado por un nodo, que recibe los datos directamente del dispositivo, los procesa y los publica. De esta manera el algoritmo AMCL [5], por ejemplo se suscribe a estos datos y los utiliza para calcular la mejor posición sobre el mapa.

733

146

Otra gran ventaja es la facilidad para cambiar el comportamiento del sistema, donde únicamente es necesario cambiar o añadir un nodo y suscribirlo a los tópicos de interés para cambiar alguna política de comportamiento del vehículo. En la figura 4 podemos ver el esquema de nodos generada por ROS para la aplicación concreta de navegación.
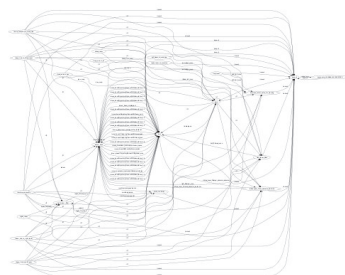

Figura 4: Esquema global de nodos en ROS

El desarrollo del software se hace de manera modular, centrándose el desarrollador en diseñar un modulo concreto con unas interfaces de entrada salida bien definidas sin necesidad de preocuparse del resto del proyecto. Este modulo una vez terminado puede ser utilizado en múltiples robots con cambios mínimos y reaprovechar así el código desarrollado e incluso código publicado por otros desarrolladores. Desarrollar un proyecto se simplifica mucho ya que se pueden reutilizar los nodos de otros robots similares, centrándonos únicamente en las partes específicas de nuestro robot, o en los algoritmos que deseemos probar o mejorar.
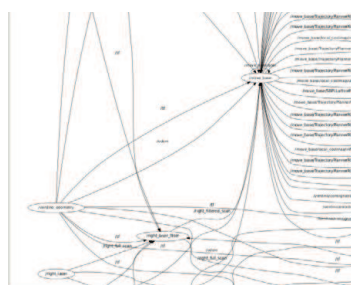
Figura 5: Detalle de los nodos relacionados con el control del vehículo

En el caso de Verdino, existe un nodo principal que representa el prototipo, que podemos ver representado en la figura 5, que recibe como entrada la posición y la velocidad de avance, devolviendo la posición y velocidad actual del prototipo. Por cada sensor especificado en la tabla I, existe uno o varios nodos encargados de recibir la información de ese sensor y publicar esta información procesada junto con la fiabilidad del sensor. Esta información la reciben los nodos encargados de la construcción del mapa, estimación de la posición actual, planificador global, planificador local, que son los encargados de definir un destino para el prototipo y la velocidad y ángulo de volante necesario para alcanzar este destino. Esta información se envía al nodo que controla el prototipo.

6 ESTIMACION DE LA POSICIÓN Y ORIENTACIÓN

La posición y orientación del vehículo se obtiene utilizando como base el sensor de odometría que está instalado en las ruedas traseras del prototipo. Esta posición es corregida con cada medida de los telémetros laser laterales utilizando una correlación entre escaneos sucesivos del telémetro. La construcción del mapa está basada en el algoritmo GMapping [5], un algoritmo muy eficiente basado en un filtro de partículas "Rao-Blackwellized" para resolver el problema de la localización y construcción simultanea de mapas (SLAM). Esta aproximación utiliza un filtro de partículas en las que cada partícula almacena un mapa individual del entorno. El algoritmo incluye una técnica adaptativa para reducir el número de partículas durante el aprendizaje de mapas basados en rejilla.

El sistema calcula una posición muy precisa basándose en la máxima probabilidad observada por la información de los sensores, la odometría y el proceso de correlación de escaneos del telémetro laser. Esto permite colocar las partículas de una manera más precisa lo que reduce drásticamente el número de medidas necesarias para lograr la convergencia del algoritmo.

Después de la generación del mapa, el carrito es localizado dentro del mapa previamente generado utilizando un algoritmo AMCL. El algoritmo está basado en un método de muestreo KLD [4], un análisis Montecarlo para estimar la posición y orientación basándose en la representación del filtro de partículas. La implementación y el computo necesario para esta localización es pequeño y la posición y orientación del robot estimada es muy precisa.

Esta información de posición y orientación calculada se fusiona con la información que proviene del GPS,

734

147

XXXIII Jornadas de Automática. Vigo, 5 al 7 de Septiembre de 2012

consiguiendo un sistema de posicionamiento robusto. En los casos en los que el GPS falle, debido a falta de visibilidad del cielo, el sistema es capaz de localizarse de manera precisa a través de sensores totalmente autónomos. De esta manera se consigue una localización precisa tanto en interiores, donde el telemetro laser proporciona gran cantidad de información con lo que el algoritmo de localización nos da una precisión alta, como en exteriores, donde el GPS suple la falta de datos que nos proporcionan los telémetros.

7   RECONSTRUCCIÓN DEL ENTORNO

La información de los sensores junto a la estimación de la posición se utiliza para la construcción del mapa de entorno. En este mapa se buscará la mejor ruta que lleve de manera segura al destino a través de un planificador global, que fija la dirección global al destino y un planificador local, encargado de esquivar los obstáculos que se sitúan en el entorno cercano.
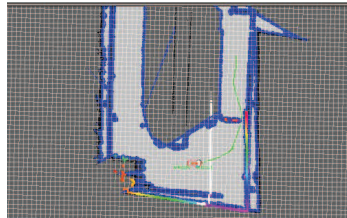
Figura 6: Mapa de navegación del prototipo.

La reconstrucción básica del entorno se lleva a cabo a través de la construcción de un mapa en 2D, en este quedan reflejadas las zonas válidas de paso para el prototipo, y los obstáculos fijos que se encuentran en la ruta. Este mapa es enriquecido en tiempo real con la información de nuevos obstáculos dinámicos que aparecen en el entorno como peatones u otros coches circulando. De esta manera la búsqueda del camino local se trata como una búsqueda del camino más corto plausible en el mapa.

7.1   RECONSTRUCCIÓN 3D DEL ENTORNO

Muchas veces un mapa en dos dimensiones del entorno no es suficiente para una correcta navegación del vehículo [3]. En este mapa no van a quedar reflejados posibles obstáculos que no sean detectados por los telémetros láser al estar situados a distinto nivel, socavones en el suelo, aceras, etc. Para solventar este problema se ha instalado un telémetro laser en la parte superior del vehículo orientado hacia

el suelo, para realizar un escaneo 3D del entorno cercano.

El telémetro instalado en el techo del vehículo y orientado hacia el suelo, descrito en la sección 2 se utiliza para registrar no solo la distancia hasta el punto más cercano, sino también la intensidad reflejada del laser. La orientación del telémetro permite escanear el entorno basándose en la posición conocida a través del algoritmo AMCL descrito en la sección 6, junto con el ángulo de inclinación y altitud conocida de este telémetro. De esta manera cada perímetro capturado por el telémetro se proyecta en el espacio de 3D en función de la posición actual del vehículo.
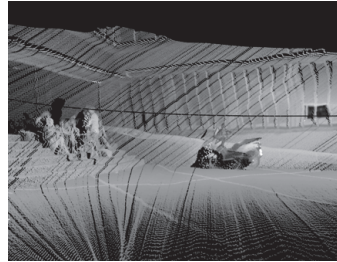

Figura 7: Reconstrucción 3D en exteriores.

Esta reconstrucción 3D es almacenada como una nube de puntos que puede ser consultada, procesada y visualizada. La reconstrucción tiene muy buena resolución, mostrando un error muy pequeño con el entorno real. En la figura 7, se observa una reconstrucción donde se puede ver una escena bastante compleja, para cada punto de la escena tenemos, además de la intensidad del laser que representa el color, la distancia a este punto. En una escena tan compleja como esta con distintos objetos a distintas distancias e incluso personas, se puede apreciar la resolución de la técnica aplicada.

Esta reconstrucción permite la navegación por zonas en las que existe muy poco espacio, o hay obstáculos que tienen un volumen que puede provocar colisión con el vehículo, permitiendo en estas condiciones realizar una navegación más precisa y robusta.

CONCLUSIONES

En este artículo se ha presentado la adaptación de un vehículo comercial con el objetivo de que sea totalmente controlado por un ordenador de a bordo. Se presenta el sistema sensorial del prototipo, la

735

148

estructura del software que se utiliza y los algoritmos utilizados.

Cabe destacar el uso de ROS como sistema operativo del prototipo, permitiendo así generar software compatible con otros sistemas, reduciendo sensiblemente el tiempo de desarrollo. ROS simplifica todas las labores de diseño definiendo una serie de interfaces y mensajes prediseñados que facilitan el desarrollo global del producto y permitiendo utilizar algoritmos estándar previamente desarrollados

Por último se presentan una serie de ejemplos de construcción de mapas 3D y 2D para la navegación utilizando el sistema sensorial como ejemplo de capacidades actuales del prototipo.

Agradecimientos

Referencias

[1] Arnay R., Acosta L., Sigut M.,Toledo J. "Ant colony optimisation algorithm for detection and tracking of non-structured roads" Electronics Letters 44 (12), 725-727 8 2008

[2] Arnay R, Acosta L., Sigut M., Toledo J. "Asphalted Road Temperature Variations Due to Wind Turbine Cast Shadows" Sensors 9 (11), 8863-8883

[3] Arnay R., Acosta L., Sigut M., Toledo J. "Detection of Non-structured Roads Using Visible and Infrared Images and an Ant Colony Optimization Algorithm" Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), 37-47 2009

[4] Fox D, "KLD-sampling: Adaptive particle filters and mobile robot localization", Advances in Neural Information Processing Systems (NIPS), 26-32

[5] Grisetti G, Stachniss C., and Wolfram Burgard: "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters", IEEE Transactions on Robotics, 2006

[6] Marichal GN, Toledo J, Acosta L, González EJ, Coll G. A neuro-fuzzy method applied to the motors of a stereovision system Engineering Applications of Artificial Intelligence 20 (7), 951-958 4 2007

[7] Morales N, Toledo JT, Acosta L., Arnay R. "Real-time adaptive obstacle detection based on an image database" Computer Vision and Image Understanding 115 (9), 1273-1287

736

149

Bibliography

The following references are author-ordered.

L. Acosta, J. Toledo, A. Hamilton, R. Arnay, J. Espelosin, N. Morales, D. Perea, and L. Moreno. Verdino, prototipo eléctrico de vehículo autoguiado, Sep 2012. XXXIII Jornadas Nacionales de Automática.

R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta. Safe and reliable path planning for the autonomous vehicle verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32, Summer 2016. ISSN 1939-1390. doi: 10.1109/MITS.2015.2504393.

E. Battesti, S. Bazeille, and D. Filliat. Qualitative localization using vision and odometry for path following in topo-metric maps. In *Proc. of the European Conference on Mobile Robots (ECMR)*, pages 303–308, 2011.

P. Beeson, N. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

I. Bogoslavskyi, O. Vysotska, J. Serafin, G. Grisetti, and C. Stachniss. Efficient traversability analysis for mobile robots using the kinect sensor. In *Proc. of the European Conference on Mobile Robots (ECMR)*, 2013.

I. Bogoslavskyi, M. Mazuran, and C. Stachniss. Robust homing for autonomous exploration robots operating in unknown environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015. Currently under review.

151

Bibliography

F. Bourgoult, A. Makarenko, S. Williams, B. Grocholsky, and F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

C. Cappelle, M. El Badaoui El Najjar, F. Charpillet, and D. Pomorski. Outdoor Obstacle Detection and Localisation with Monovision and 3D Geographical Database. In *IEEE Intell. Transportation Systems Conference*, pages 1102–1107, Oct. 2007.

T. Cerdena, Y. Callero, D. Perea, P. Betancor, D. Lutzardo, J. Toledo, and L. Acosta. Design and development of a fully autonomous decimeter-scale humanoid robot. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, April 2009. doi: 10.1109/ICMECH.2009.4957211.

H. Chang, J. Choi, and M. Kim. Probabilistic Localization of Service Robot by Sensor Fusion. In *Int. Joint Conf.*, pages 3626–3631, Oct. 2006a.

H. Chang, J. Choi, and M. Kim. Experimental Research of Probabilistic Localization of Service Robots using Range Image Data and Indoor GPS System. In *IEEE Conf. Emerging Technologies and Factory Automation*, pages 1021–1027, Sept. 2006b.

H. Chang, C. Lee, Y. Lu, and Y. Hu. P-slam: Simultaneous localization and mapping with environmental-structure prediction. *IEEE Transactions on Robotics*, 23(2):281–293, 2007.

M. Cummins and P. Newman. Highly scalable appearance-only slam – fab-map 2.0. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

I. Dryanovski, R. G. Valenti, and J. Xiao. An open-source navigation system for micro aerial vehicles. *Autonomous Robots*, 34(3):177–188, 2013a. ISSN 1573-7527. doi: 10.1007/s10514-012-9318-8. URL http://dx.doi.org/10.1007/s10514-012-9318-8.

I. Dryanovski, R. G. Valenti, and J. Xiao. Fast visual odometry and mapping from rgb-d data. In *2013 IEEE International Conference on Robotics and Automation*, pages 2305–2310, May 2013b. doi: 10.1109/ICRA.2013.6630889.

152

T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Journal of Autonomous Robots*, 12(3):287 – 300, 2002.

M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis andd automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

D. Fox. Adapting the Sample Size in Particle Filters Through KLD-Sampling. *International Journal of Robotics Research*, 22(12):985–1004, 2003.

D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *XVI Nat. Conf. Artificial Intell., XI Innovative Applications of Artificial Intell. Conf.*, pages 343–349. American Association for Artificial Intelligence, 1999.

D. Fox, J. Ko, K. Konolige, and B. Stewart. A hierarchical bayesian approach to the revisiting problem in mobile robot map building. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, 2003.

E. Frontoni, A. Ascani, A. Mancini, and P. Zingaretti. Robot Localization in Urban Environments using Omnidirectional Vision Sensors and Partial Heterogeneous apriori Knowledge. In *IEEE/ASME Int. Conf. Mechatronics and Embedded Systems and Applications*, pages 428–433, July 2010.

P. Furgale and T. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal on Field Robotics*, 27(5):534–560, 2010.

P. Furgale, P. Krüsi, F. Pomerleau, U. Schwesinger, F. Colas, and R. Siegwart. There and back again—dealing with highly-dynamic scenes and long-term change during topological/metric route following. In *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*, 2014.

M. Hentschel, O. Wulf, and B. Wagner. A GPS and Laser–Based Localization for Urban and Non–Urban Outdoor Environments. In *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, pages 149–154, Sept. 2008.

153

Bibliography

G. Hitz, A. Gotovos, F. Pomerleau, M.-E. Garneau, C. Pradalier, A. Krause, and R. Siegwart. Fully autonomous focused exploration for robotic environmental monitoring. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2014.

D. Holz, N. Basilico, F. Amigoni, and S. Behnke. A comparative evaluation of exploration strategies and heuristics to improve them. In *Proc. of the European Conference on Mobile Robots (ECMR)*, pages 25–30, Örebro, Sweden, 2011.

A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34, 2013.

T. Khalid, Z. Mourad, C. Jean-Bernard, and B. Mohammed. Bayesian Bootstrap Filter for Integrated GPS and Dead Reckoning Positioning. In *IEEE Int. Symposium on Industrial Electronics*, pages 1520–1524, June 2007.

J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.

S. Koenig and C. Tovey. Improved analysis of greedy mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, 2005.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3607–3613, 2011.

154

T. Li, S. Sun, and J. Duan. Monte Carlo Localization for Mobile Robot using Adaptive Particle Merging and Splitting Technique. In *IEEE Int. Conf. Information and Automation*, pages 1913–1918, June 2010.

M. Likhachev, G. Gordon, and S. Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS 16)*. MIT Press, 2004.

A. Makarenko, S. Williams, F. Bourgoult, and F. Durrant-Whyte. An experiment in integrated exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2002.

M. Mazuran, G. Tipaldi, L. Spinello, W. Burgard, and C. Stachniss. A statistical measure for map consistency in slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Hong Kong, China, 2014.

M. Moreira, H. Machado, C. Mendonca, and G. Pereira. Mobile Robot Outdoor Localization using Planar Beacons and Visual Improved Odometry. In *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, pages 2468–2473, Nov. 2007.

C. Mostegel, A. Wendel, and H. Bischof. Active monocular localization: Towards autonomous monocular exploration for multirotor mavs. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Hong Kong, China, 2014.

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

P. Newman, D. Cole, and K. Ho. Outdoor SLAM using Visual Appearance and Laser Ranging. In *IEEE Int. Conf. Robotics and Automation*, pages 1180–1187, May 2006.

M. Nitsche, T. Pire, T. Krajnik, M. Kulich, and M. Mejail. Monte carlo localization for teach-and-repeat feature-based navigation. In M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, editors, *Advances in Autonomous Robotics Systems*, volume 8717 of *Lecture Notes in Computer Science*, pages 13–24. Springer International Publishing, 2014. ISBN 978-3-319-10400-3.

155

E. Olson. Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 2009.

J. Peng, M. El Najjar, C. Cappelle, D. Pomorski, F. Charpillet, and A. Deeb. A Novel Geo–Localisation Method using GPS, 3D–GIS and Laser Scanner for Intelligent Vehicle Navigation in Urban Areas. In *Int. Conf. Advanced Robotics*, pages 1–6, June 2009.

D. Perea, J. Hernández-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta. Mcl with sensor fusion based on a weighting mechanism versus a particle generation approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 166–171, Oct 2013. doi: 10.1109/ITSC.2013.6728228.

D. Perea, F. Nenci, and C. Stachniss. Predictive exploration considering previously mapped environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2761–2766, May 2015a. doi: 10.1109/ICRA.2015.7139574.

D. Perea, J. Toledo, A. Morell, R. Arnay, L. Acosta, N.Morales, A. Hamilton, and J. Hernandez-Aceituno. Demonstration of the autonomous vehicle verdino (invited talk), 2015b. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2015).

D. Perea, I. Bogoslavskyi, and C. Stachniss. Robust exploration and homing for autonomous robots. *Robotics and Autonomous Systems*, 90:125 – 135, 2017. ISSN 0921-8890. doi: 10.1016/j.robot.2016.08.015. Special Issue on New Research Frontiers for Intelligent Autonomous Systems.

M. Pivtoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pages 3231 – 3237, August 2005.

R. Rocha, J. Dias, and A. Carvalho. Exploring information theory for vision-based volumetric mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2409–2414, Edmonton, Canada, 2005.

D. B. Rubin. Using the SIR algorithm to simulate posterior distributions. In M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 3*, pages 395–402. Oxford University Press, 1988.

S. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, and R. Vaughan. Feature-rich path planning for robust navigation of mavs with mono-slam. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Hong Kong, China, 2014.

A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proc. of Robotics: Science and Systems (RSS)*, volume 2, 2009.

J. Serafin and G. Grisetti. NICP: Dense normal based point cloud registration and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

R. Shade and P. Newman. Choosing where to go: Complete 3d exploration with stereo. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

S. Shen, N. Michael, and V. Kumar. 3d indoor exploration with a computationally constrained mav. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

D. Silver and A. Stentz. Monte Carlo Localization and Registration to Prior Data for Outdoor Navigation. In *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, pages 510–517, Sept. 2011.

R. Sim, G. Dudek, and N. Roy. Online control policy optimization for minimizing map uncertainty during exploration. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004.

C. Smaili, M. El Najjar, and C. Francois. A Chained Form State Representation for Outdoor Vehicle Localisation. In *XIV Int. IEEE Conf. Intell. Transportation Systems*, pages 1386–1391, Oct. 2011.

A. F. M. Smith and A. E. Gelfand. Bayesian Statistics without Tears: A Sampling–Resampling Perspective. *The American Statistician*, 46(2):84–88, may 1992. ISSN 00031305. doi: 10.2307/2684170.

157

Bibliography

C. Sprunk, G. Tipaldi, A. Cherubini, and W. Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

C. Stachniss and W. Burgard. Mapping and exploration with mobile robots using coverage maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 476–481, 2003.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

C. Stachniss, O. Martinez Mozos, and W. Burgard. Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52:205ff, 2009.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, 2005.

G. Tipaldi and K. Arras. Flirt–interest regions for 2d range data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3616–3622, 2010.

J. Toledo, L. Acosta, D. Perea, and N. Morales. Stability and performance analysis of unmanned aerial vehicles: Quadrotor against hexrotor. *IET Control Theory Applications*, 9(8):1190–1196, 2015. ISSN 1751-8644. doi: 10.1049/iet-cta.2014.1032.

R. G. Valenti, I. Dryanovski, C. Jaramillo, D. Perea, and J. Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5233–5238, May 2014. doi: 10.1109/ICRA.2014.6907628.

L. Wei, C. Cappelle, and Y. Ruichek. Unscented Information Filter Based Multi–Sensor Data Fusion using Stereo Camera, Laser Range Finder and GPS Receiver for Vehicle Localization. In *XIV Int. IEEE Conf. Intell. Transportation Systems*, pages 1923–1928, Oct. 2011a.

158

L. Wei, C. Cappelle, and Y. Ruichek. Localization of Intelligent Ground Vehicles in Outdoor Urban Environments using Stereovision and GPS Integration. In *XV Int. Conf. Advanced Robotics*, pages 192–197, June 2011b.

P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

K. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 47–53, 1998.

L. Zhang, R. Zapata, and P. Lepinay. Self–Adaptive Monte Carlo for Single–Robot and Multi–Robot Localization. In *IEEE Int. Conf. Automation and Logistics*, pages 1927–1933, Aug. 2009.

Firmado por:	JOSE DANIEL PEREA STRÖM	Fecha: 30/06/2017 00:15:39
	UNIVERSIDAD DE LA LAGUNA	
	JONAY TOMAS TOLEDO CARRILLO	30/06/2017 02:34:46
	UNIVERSIDAD DE LA LAGUNA	
	LEOPOLDO ACOSTA SANCHEZ	30/06/2017 08:37:26
	UNIVERSIDAD DE LA LAGUNA	
	ERNESTO PEREDA DE PABLO	06/07/2017 13:51:10
	UNIVERSIDAD DE LA LAGUNA	