University of the Basque Country (UPV/EHU)

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Computer Science and Artificial Intelligence Department
Faculty of Informatics

Ph.D. Thesis

Contributions to autonomous robust navigation of mobile robots in industrial applications

Iker Lluvia Hermosilla

1. Supervisor Elena Lazkano Ortega
Computer Science and Artificial Intelligence
University of the Basque Country (UPV/EHU)

2. Supervisor Ander Ansuategi Cobo
Autonomous and Intelligent Systems Unit
TEKNIKER

February 13, 2023

Contributions to autonomous robust navigation of mobile robots in industrial applications

Iker Lluvia Hermosilla

February 13, 2023

Iker Lluvia Hermosilla

Contributions to autonomous robust navigation of mobile robots in industrial applications

Ph.D. Thesis, February 13, 2023

Supervisors: Elena Lazkano Ortega and Ander Ansuategi Cobo

University of the Basque Country (UPV/EHU)

Computer Science and Artificial Intelligence Department

Faculty of Informatics

Manuel de Lardizabal Pasealekua, 1

20018 Donostia-San Sebastián

Abstract

The number of scenarios in which mobile robots can be seen performing a wide range of tasks is increasing, being the most common transport, cleaning and rescue operations. For example, the media has recently shown how robots can disinfect schools and hospitals to prevent people from becoming infected with a certain virus and stop its spread. The functions that are perhaps most impressive are those for public assistance, but the number of applications for process automation in industry is much higher.

Among the capabilities that a mobile system can have, flexibility, security and robustness are probably the most demanded. On the one hand, whatever the purpose for which it is designed, a mobile platform must be able to adapt to the changes that an environment may undergo over time, or to the different casuistry that may occur in it on a day-to-day basis. In addition, it must avoid damaging the rest of the machinery and, especially, harming the people it shares the space with. On the other hand, of course, the mobile platform must perform the tasks assigned to it effectively and within a given time interval. Although significant progress has been made in this field in the last decades, there are still challenges that these systems have not been able to overcome.

One aspect where today's mobile platforms cannot compete with the level already achieved in industry is accuracy. The fourth industrial revolution brought with it the implementation of machinery in most industrial processes, and a clear strength of these is their repeatability. Autonomous mobile robots, which offer the greatest flexibility, lack this capability, as the most successful ones are based on probabilistic algorithms that depend on the perception of the environment. For this reason, a large part of the research work presented in this document focuses on quantifying the error committed by the main mapping and localisation methods, offering different alternatives for improving their positioning.

To perceive this environment around the robot, the main sources of information are exteroceptive sensors, which measure the surroundings and not so much the state of the robot itself. Because of this, some methods are very dependent on the scenario for which they have been developed, and do not obtain the same results when moving to a different scenario. Most mobile platforms generate a map that represents the environment, and use it for many of their calculations for actions such as, e.g., navigation. Map generation is a process that, in most cases, requires human intervention and has a great impact on the subsequent performance of the robot. In the last part of this research work, a method to optimise this step in order to generate a richer model of the environment without requiring additional time is proposed.

Resumen

Cada vez son más los escenarios en los que se ven robots móviles realizando tareas muy diversas, siendo las más comunes labores de transporte, limpieza o rescate. Sin ir más lejos, los medios de comunicación han mostrado recientemente cómo los robots pueden desinfectar escuelas y hospitales para prevenir que las personas se infecten de un determinado virus y evitar que este propague. Las funciones que quizá más impresionan son las de ayuda a la ciudadanía, pero el número de aplicaciones destinadas a la automatización de procesos en la industria es mucho más elevado.

Entre las capacidades que puede ofrecer un sistema móvil, probablemente las más demandadas son la flexibilidad, seguridad y robustez. Por un lado, sea cual sea el objetivo para el que se diseñe, una plataforma móvil debe ser capaz de adaptarse a los cambios que pueda sufrir un entorno con el paso del tiempo, o a las diferentes casuísticas que se puedan dar en este en el día a día. Además, debe evitar cualquier daño que pueda producir al resto de maquinaria y, especialmente, a las personas con las que comparta el espacio. Por otro lado, como cabe esperar, la plataforma móvil debe realizar las tareas que se le asignen de forma efectiva y dentro de un intervalo de tiempo determinado. Pese a que las últimas décadas los avances conseguidos en este campo han sido importantes, todavía existen retos que estos sistemas no han conseguido superar.

Un aspecto en el que las plataformas móviles actuales se quedan atrás en comparación con el punto que se ha alcanzado ya en la industria es la precisión. La cuarta revolución industrial trajo consigo la implantación de maquinaria en la mayor parte de procesos industriales, y una fortaleza de estos es su repetitividad. Los robots móviles autónomos, que son los que ofrecen una mayor flexibilidad, carecen de esta capacidad, principalmente debido al ruido inherente a las lecturas ofrecidas por los sensores y al dinamismo existente en la mayoría de entornos. Por este motivo, gran parte de este trabajo se centra en cuantificar el error cometido por los principales métodos de mapeado y localización de robots móviles, ofreciendo distintas alternativas para la mejora del posicionamiento.

Asimismo, las principales fuentes de información con las que los robots móviles son capaces de realizar las funciones descritas son los sensores exteroceptivos, los cuales miden el entorno y no tanto el estado del propio robot. Por esta misma razón, algunos métodos son muy dependientes del escenario en el que se han desarrollado, y no obtienen los mismos resultados cuando este varía. La mayoría de plataformas

móviles generan un mapa que representa el entorno que les rodea, y fundamentan en este muchos de sus cálculos para realizar acciones como navegar. Dicha generación es un proceso que requiere de intervención humana en la mayoría de casos y que tiene una gran repercusión en el posterior funcionamiento del robot. En la última parte del presente trabajo, se popone un método que pretende optimizar este paso para así generar un modelo más rico del entorno sin requerir de tiempo adicional para ello.

Acknowledgements

Lehenik eta behin, eskerrak eman nahi dizkizuet Elena eta Ander nire zuzendariei lan hau egiteko aukera emateagatik eta behar izan zaituztedan guztietan bertan egon zaretelako.

Era berean, eskerrik asko UPV/EHUri eta Teknikerri aukera hau ematean nigan konfiantza izateagatik.

Thanks to the AIS Laboratory of Prof. Dr. Wolfram Burgard at the University of Freiburg, as a part of this research was done during an internship there. Special thanks to Tim Caselitz and Michael Krawez for sharing their knowledge and expertise with me there.

Thanks to the scientific - and not so "scientific" - community for sharing your knowledge, data, developments, etc., facilitating technological progress as a whole.

También me gustaría agradecer a mis compañeros de Tekniker por los múltiples consejos que me habéis dado y por haberme distraído con otros temas sin dejar que la robótica móvil fuera lo único en mi mente. Pero, principalmente, gracias por haber hecho que el día a día de los últimos años haya sido agradable y entretenido.

Como no, gracias a mi familia por el apoyo que he recibido a lo largo de toda mi vida y por haberme ayudado a ser la persona que soy. Agradezco especialmente a mi tía Idoia, mi tío Ernesto, Alexander, la amama Vega, y, sobre todo, a mis aitas, Carmen y Rober. Sin duda, siempre me habéis intentado dar los medios para cumplir mis objetivos y el ánimo para no sucumbir en el intento.

Por último, gracias a ti, Marina, por llenarme de ilusión cada día y por hacerme sentir que cualquier obstáculo es insignificante a tu lado.

Contents

List of Figures

List of Tables

Part I

Introduction

Introduction 1

Be the change that you wish to see in the world.

— **Mahatma Gandhi**

This chapter is an overall introduction to the work. First, we describe the context in which this research has been developed in Section 1.1. In Section 1.2, the motivation for doing it is explained, and Section 1.3 contains its objectives and scope. The scientific publications related are listed in Section 1.4. Finally, the main structure of the document is presented in Section 1.5.

1.1 Context

By the time of writing this thesis proposal, robots are more demanded than ever by society [@191]. Not only the industry, but also hospitals, schools and smaller businesses are using robots for an increasing variety of tasks. However, in order to understand the relevance of this event, it is interesting to make a brief tour through the history of robots first.

The word *robot* is relatively new in the English language. It was first introduced by the Czech playwright, novelist and journalist Karel Čapek in 1920 in his play *R.U.R. (Rossum's Universal Robots)* [33]. There, robots were some creatures capable of doing all the work that humans preferred not to do. In fact, the word comes from an Old Church Slavonic word, *rabota*, which means servitude of forced labour, although it also has cognates in German, Russian, Polish and Czech [@43]. Nevertheless, the concept of an entity capable of carrying out a complex series of actions automatically originates in the mythologies of many cultures around the world. Different inventors and engineers from ancient civilisations already attempted to build self-operating machines. In ancient Greece mythology, there was the idea of Talos, a giant automaton made of bronze that protected Europe in Crete from pirates and invaders. In the same 4th century BC, the Greek mathematician Archytas of Tarentum built a steam-powered mechanical bird, which he called "The Pigeon" [95].

The engineer Heron of Alexandria (10-70 AD) also created numerous automatic devices that users could modify, and described machines driven by air, steam and water pressure [179]. The Chinese scholar Su Sung erected a clock tower in 1088 with mechanical figures that chimed the hours [@218]. Ismael al-Jazari (1136-1206), a Muslim inventor of the Artuqid Dynasty, designed and built a series of automated machines, including kitchen utensils, water-powered musical automaton, and in 1206 an early programmable automata. The machines looked like four musicians aboard a boat on a lake, entertaining guests at royal feasts. Their mechanism contained a programmable drum with pegs that clashed with small levers that operated percussion instruments. The rhythms and patterns played by the drummer could be changed by moving the pins [89].

It is quite clear that the idea of automating tasks has always been present in the human mind, but it was not until the 20th Century that it was implanted on a massive scale. The Digital Revolution, or Third Industrial Revolution, brought to the factories and businesses the shift from mechanical and analogue electronic technology to digital electronics as a means of storing, transferring and utilising information. At the heart of this period is the mass production and widespread use of digital logic circuits, and their derivative technologies, including the digital computer, the digital mobile phone, and the Internet. At the same time, the first large industrial robots came into use, intended for heavy, repetitive tasks consisting of simple movements. Due to the speed, accuracy, and decreased cost with which they performed these functions, the 1980's saw an explosion in the development of robotics and the decade is known to be the beginning of the robotics era. It is considered that the seed that would later give rise to the intelligent robotics we know today was sown at this time. Since then, robots have evolved rapidly, increasing their capabilities and the number of scenarios in which they can outperform humans [93, 103, 120, 88]. Nevertheless, in order to perform many of these tasks autonomously, robots need to move around the environment.

It was not until 20 years after the first mobile robots appeared in the 1940's [249] that Shakey, a general purpose mobile robot capable of reasoning about its own actions, was built [172]. Mobile robots have the ability to move around the environment and are not fixed to a physical location. They may have guidance devices that allow them to follow a predefined navigation route in a relatively controlled space [140]. Alternatively, they can be autonomous mobile robots (AMRs) capable of navigating in a semi-structured environment without the need of physical or electromechanical guidance devices [201]. In the last years, mobile robots have become increasingly common in manufacturing and logistics applications, but they are also being used for search and rescue, hazardous material handling, security, and other applications where it is necessary or desirable to have a mobile robot that can navigate autonomously in a semi-structured environment.

The use of mobile robots in emergency situations could also help in search and rescue tasks, and is another context that has recently attracted a great deal of interest. Historically, the push for rescue robotics began in 1995 in the aftermath of the Great Hanshin Earthquake in Japan [46]. Rescue robots have been in use for a long time and their technology has advanced considerably over the years, such as their mechanical reliability when used in the field. Innovation is a key factor in robotic technology, which is seen as a next-generation industry with the potential to foster employment and economic development as well as contribute to advancing the 2030 Agenda for Sustainable Development with practical solutions [@61]. They may assist rescue efforts by searching, mapping, removing rubble, delivering supplies, providing medical treatment or evacuating casualties.

In order for mobile robots to be able to perform all these tasks adequately, a number of problems need to be tackled, such as:

- Perception and understanding of the surroundings.
- How to compute the absolute or relative localisation in the environment.
- Intelligent decision making.
- Robust and safe navigation and action execution.
- Error recovery behaviours.
- Interaction with other entities (humans, robots, etc.).

These problems have different solutions in the scientific literature, which are discussed further in this work, but most of them are mainly tested in setup environments where certain conditions are met for the system to work. A real environment is much more demanding than the situations to which these systems are exposed for evaluation, which means that they do not always meet the requirements for correct and continuous operation in such scenarios. In addition, industrial processes require a level of robustness and speed of execution that AMRs often do not achieve. Outside this realm, when robots are used in places where human presence or interaction is constant, they fall even further short of society's expectations and demands (mainly generated by science fiction). Therefore, there is a need for further research in order to develop techniques that can efficiently and robustly respond to the problems of real environments.

In line with the above, the aim of this research is to push the boundaries of mobile robot navigation towards more reliable and autonomous behaviour, with some emphasis on indoor industrial environments. There are several entities that have made this research work possible, the most important of which are described below.

1.1.1 Tekniker

Tekniker is a technology centre located in Eibar (Spain) specialised in Advanced Manufacturing, Surface Engineering, Product Engineering and ICT for manufacturing. Its mission is to deliver growth and well-being to society at large via R&D&I and to promote the competitiveness of the business fabric in a sustainable way. Throughout its 41-year history, Tekniker has participated in 268 European projects, leading 23% of them in the last decade, and only in 2021, it published 62 research articles, 41 of which were in Q1 journals. I started working with Tekniker in the final year of my Bachelor in Computer Science and, since then, this is where I have developed my career as a researcher.

1.1.2 University of the Basque Country (UPV/EHU)

The University of the Basque Country (UPV/EHU) is the public university of the Basque Autonomous Community (Spain). It is an institution with more than 42 years of history with state of the art facilities, which underpins much of the region's scientific and technological progress and success. Following their motto 'Give and spread knowledge', UPV/EHU is an integrating institution willing to produce knowledge, experience and research in order to forward them to the general public. UPV/EHU has given me the scientific background I need to develop as a researcher. Firstly, through the Bachelor's Degree in Computer Science and its emphasise in the mathematical and theoretical foundations of computing. Then, via the Master's Degree in Computational Engineering and Intelligent Systems with its focus on research and, finally, during the Doctoral Programme in Informatics Engineering.

1.1.3 Autonomous Intelligent Systems (AIS) Laboratory at the University of Freiburg

The Autonomous Intelligent Systems (AIS) laboratory belongs to the Department of Computer Science at the Albert Ludwig University of Freiburg (Germany). Its research, led by Prof. Dr. Wolfram Burgard, is focused on autonomous mobile robots, including: multi-robot navigation and coordination; environment modelling and state estimation; interaction; and applications in human-populated natural environments. In the third year of my Ph.D., I had the opportunity to do a stay in the AIS group as a guest researcher, where we worked on the design of an autonomous mapping system.

1.2 Motivation

I have always been fascinated by technology, and, particularly, by robots. But rather than just imagining a world full of futuristic machines, I wanted to understand how these devices were designed and programmed. However, it was not until 2012 that I really got into it. In the Bachelor's Degree in Computer Science, I learnt how computers work, the many possibilities they offer and what their limitations are. Three years later, I was programming my first mobile robot. It was a simple application in which a Kobuki-like[1] platform had to follow a line drawn on the ground and complete the course in the shortest possible time. Later, I had the opportunity to do an internship at Tekniker, and it is here where I really discovered the current state of the art and potential of mobile robotics.

In fact, a large part of my master's thesis was done in Tekniker, in which I rehabilitated a service robot while improving some of its functionalities [134]. The robot had to navigate through the hall of the building, provide information to visitors and guide them to the points they requested. I soon realised that it was a very demanding environment, as it had: large windows; dynamic and irregular objects; narrow places; and stairs. Besides, there was often a large presence of people, whose attention was captured by the robot, causing them to surround it. State of the art algorithms were not, and still are not, prepared for all the circumstances mentioned, so the robot's localisation, accuracy and trajectories were not as expected. Its behaviour differed from the tests conducted in the laboratory and in the same environment under controlled conditions, being far from ideal. In addition, we wanted the system to be operating continuously. In such a long period of time and with that unpredictable dynamism of the environment, it was nearly impossible to finish the day without any issues. While working on this project, I was also involved in other tasks related to mobile robotics, and worked side by side with colleagues who had both industrial and research experience.

[1] `http://kobuki.yujinrobot.com/about2/`

All these circumstances allowed me to learn about mobile robotics from different points of view, experiencing its limitations and knowing the requirements that each application and/or user has. I realised that there was a gap between what mobile robots could offer and what the world was demanding. Moreover, the use of these technologies continued to grow exponentially, and more and more areas were looking to implement them. So, there was a clear need to increase the capabilities of mobile robotics and its adaptation to different scenarios. As most of the research projects and applications focused on indoor environments, we wanted the work presented here to follow this line as well.

1.3 Objectives and Scope

In line with the ideas mentioned in the previous section, the overall objective of the thesis project is to increase the autonomy and reliability of mobile platforms in industrial environments. To achieve this, first of all, it is necessary to determine the state of the art of current technologies, in order to be able to take them a step further. Still, the scope of the research has to be bounded, as mobile robotics in general encompasses a large number of challenges, which cannot be addressed in a single project.

In this case, the developments focus on vision based systems, since we consider them to have the best balance among throughput, flexibility, accuracy and cost. They also offer the possibility to gather information of the surrounding without adding supplementary elements to it, as in the case of radio-frequency identification (RFID) technology. Furthermore, the work is limited to indoor environments, for being this where some of the circumstances we are most interested in occur, such as areas without a stable signal for global navigation satellite systems (GNSS). Besides, it is also easier to control external conditions in this kind of places, e.g., illumination. More specifically, the research work is oriented towards industrial indoor environments, as this is the area for which most applications are designed and, consequently, where our developments can have the greatest collaboration and impact. In one way or another, in all cases the experiments have been conducted with a certain level of dynamism in the environment. However, this problem is not directly addressed in our work. In addition, all the systems have been implemented over ground wheeled robots, discarding aerial, aquatic and hybrid platforms or any other type of mechanics (legged, tracked, etc.).

Three main objectives can be identified in this work:

- **Evaluate state-of-the-art methods for simultaneous localisation and mapping (SLAM).** Perform a rigorous qualitative and quantitative evaluation of state-of-the-art procedures for SLAM, in order to determine their requirements, strengths and limitations.
- **Improve accuracy and repeatability.** Increase the accuracy in the positioning of mobile platforms in indoor environments, increasing its repeatability and ensuring that when the robot reaches the destination, it always does in the same pose.
- **Increase autonomy for mapping.** Automate the process of mapping unknown environments to make it more efficient and increase the adaptability to new configurations. Similarly, facilitate the deployment of a mobile robot in any scenario and the management of changing environments.

1.4 Contributions

Regarding the scientific contributions generated during the research, the following list displays the articles that have been published and include contributions of the author:

(a) **I. Lluvia**, **A. Ansuategi**, C. Tubío, L. Susperregi, I. Maurtua and **E. Lazkano**, "Optimal Positioning of Mobile Platforms for Accurate Manipulation Operations", *Journal of Computer and Communications*, **2019**, DOI: 10.4236/jcc.2019.75001.

(b) M. Pattinson, S. Tiwari, Y. Zheng, M. Campo-Cossio, R. Arnau, D. Obregón, **A. Ansuategi**, C. Tubío, **I. Lluvia**, O. Rey, J. Verschoore, L. Lenža and J. Reyes, "GNSS precise point positioning for autonomous robot navigation in greenhouse environment for integrated pest monitoring", in *12th Annual Baška GNSS Conference*, **2019**, DOI: 10.5281/zenodo.2620125.

(c) S. Tiwari, Y. Zheng, M. Pattinson, M. Campo-Cossio, R. Arnau, D. Obregón, **A. Ansuategi**, C. Tubío, **I. Lluvia**, O. Rey, J. Verschoore, V. Adam and J. Reyes, "Approach for Autonomous Robot Navigation in Greenhouse Environment for Integrated Pest Management", in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, **2020**, DOI: 10.1109/PLANS46316.2020.9109895.

(d) **I. Lluvia**, **E. Lazkano**, and **A. Ansuategi**, "Active Mapping and Robot Exploration: A Survey", *Sensors*, **2021**, DOI: https://doi.org/10.3390/s21072445.

(e) **I. Lluvia**, **E. Lazkano** and **A. Ansuategi**, "Camera pose optimisation for 3D mapping", *IEEE Access*, **2023**, DOI: 10.1109/ACCESS.2023.3239657.

1.5 Thesis Outline

The thesis proposal presented here is organised as follows.

Chapter 2, *Theoretical Background*

Related works on the literature of mobile robotics are reviewed. From the classical but currently most used methods to the latest contributions, the relevant works related to the objectives of this research work are analysed. They define the canvas in which we want to paint.

Chapter 3, *Experimental Evaluation of Current Mapping Solutions*

As starting point, the most well-known mapping existing methods are evaluated and compared empirically in a concrete industrial environment. To develop our contributions, the effectiveness of current solutions must be evaluated in our workspace, as they may set our limitations, challenges and opportunities.

Chapter 4, *Accurate Positioning*

Our proposed approaches to improve the localisation accuracy of a mobile platform are explained. The experiments demonstrate that the position accuracy obtained with the current solutions is not enough for some tasks, thus we propose two different procedures to improve it.

Chapter 5, *Active Mapping and Exploration*

The system developed for autonomous mapping of unknown regions is defined. After a deep revision of the state of the art in indoor mapping, we propose a system that is capable of mapping an indoor environment autonomously.

Chapter 6, *Conclusions and Future Work*

Lessons learned in the course of this project are described. This chapter summarises the research performed, as well as its contributions to the scientific community, and registers the conclusions drawn from it. Finally, some ideas for further research are presented.

2 Robot Navigation: Concepts and Background

I have not failed. I've just found 10,000 ways that won't work.

— **Thomas Alva Edison**

This chapter provides a general review of the literature on different lines of research in mobile robotics. It explains in detail techniques that, although not particularly recent, are the basis of many of the methods being used at the present time. The theoretical framework on which the developments performed in the following chapters are based is defined in this chapter.

2.1 Introduction

Autonomous navigation is probably one of the most difficult skills expected of a mobile robot. In fact, it is the only one the general public thinks of when talking about, for example, an autonomous vehicle. However, its success depends on the correct performance of four main functions: environment interpretation, localisation, planning and motion control. First of all, the robot must be able to perceive the environment and represent it unequivocally, as well as store the information necessary to perform, at least, the tasks for which the robot is designed. Secondly, the system must be able to position itself in that environment and, likewise, be able to locate each element in it. In addition, it must have the ability to decide what decisions it should take to reach a certain destination safely. Last but not least, the robot must be able to command its locomotion system accordingly to follow the planned trajectory correctly.

The estimation of the change in position over time is called *odometry*, and is a core competency of a mobile robot. As data from motion sensors is used for this purpose, and sensors can give unexpected readings *per se*, odometry will also have an associated noise or uncertainty. Historically, odometry in mobile robotics has been estimated using wheel encoders, and one point to note is that the noise related to this calculations is cumulative. The error made in any of the measurements is carried over to all subsequent ones, which makes the problem of calculating and correcting that deviation much more complex. Nevertheless, if each measurement obtained were completely independent of the others, the error could be bounded and its effects reduced by taking more measurements. For example, a thermometer measures temperature, more or less accurately, but by taking many measurements one could calculate its distribution and apply a correction. As a consequence, exteroceptive sensors that provide absolute measurements such as cameras or satellite systems are useful in order to correct this drift and maintain a correct localisation.

As each scenario has its own requirements and limitations, the sensors used as a source of information also vary. Still, most of them can be classified into range sensors and vision-based systems, either in 2D or 3D systems. Many other sensors can be configured as primary information sources or act as additional ones, e.g. global navigation satellite systems (GNSS), rotary encoders, sonars or inertial measurement units (IMUs) [59, 14, 8, 200]. The wheels of a moving platform, for example, can slip or even rotate in the air depending on the surface, generating completely erroneous odometric information. For this reason, IMUs are incorporated in many vehicles as complementary sensors because, compared to other devices, they provide reliable information about the axes of rotation. For example, they are not as dependent as other sensors on the environment in which they operate, although they cannot perceive the surroundings. In fact, some cameras integrate IMUs in themselves [1] [2]. Many researchers opt for lasers or cameras because a large number of robots are intended to operate in indoor environments without GNSS signal, and other sensors are often not accurate enough.

With the expansion of cameras, the concept of *visual odometry* also started to get more attention. Briefly, it is the process of determining the position and orientation of a robot, i.e., odometry, through images, instead of wheel encoders. Whether visual odometry is monocular (one camera) or stereo (two or more cameras), it extracts relevant data from different images in a sequence to estimate the movement associated to the corresponding perspectives. Similarly, *visual-inertial odometry* refers to calculating odometry with the use of visual and inertial sensors, and *lidar odometry* corresponds to same process, but using a lidar.

In the end, many of these techniques end up considering sets of spatial points, also known as point clouds, specially those based on lidars or depth images. The process

[1] `https://www.intelrealsense.com/depth-camera-d435i/`
[2] `https://store.opencv.ai/collections/all/products/oak-d-poe`

of finding the spatial transformation that aligns two point clouds is known as *point cloud registration* or *scan matching*. Instead, if the point set is purely in 2D pixel coordinates, it is called *image registration*. However, usually, even features extracted from 2D images are represented as 3D points with respect to a global reference frame, simplifying the association of the same point in space captured at different pixel coordinates in different images.

2.2 Probabilistic Algorithms

The favourable outcomes of most robotic algorithms depend to a large extent on the use of precise sensors and on obtaining accurate models. But these two conditions are not enough to guarantee full success, since errors and uncertainties will always be present in any real robotic system. Faced with this problem, probabilistic methods have shown to offer the most reliable results, as they are based on models that represent the information through probability functions, being more robust against the sensor limitations and noise in the robot kinematics and in the model of the environment. On the other hand, the most cited disadvantages of probabilistic algorithms are twofold: the computational inefficiency of having to consider entire probability densities of the robot's position space, and the inherent need to discretely approximate the continuous reality of the robot context [226].

The basic principle underlying all successful probabilistic mapping algorithms is the Bayes' theorem. It states that the posterior probability of a hypothesis, given some evidence, is proportional to the prior probability of the hypothesis multiplied by the likelihood of the evidence given the hypothesis. Bayes' theorem understands probability inversely to the total probability theorem, which relates the conditional probability with the marginal probability, as in Equation 2.1.

$$P(A) = \sum_{n} P(A|B_n)P(B_n) \tag{2.1}$$

The total probability theorem makes inference about an event B from the outcomes of the events A. Bayes rule calculates the probability of A conditional on B. Let $\{A_1, A_2, ..., A_i, ..., A_n\}$ be a set of mutually exclusive and exhaustive events such that the probability of each of them is non-zero ($P[A_i] \neq 0$ for $i = 1, 2, ..., n$). If B is any event for which the conditional probabilities $P(B|A_i)$ are known, then the probability $P(A_i|B)$ is given by the expression:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{2.2}$$

The Bayes filter, also known as recursive Bayesian estimation, extends the Bayes rule of Equation 2.2 to deal with time domain estimation problems [99]. The Bayes filter is a general probabilistic approach for estimating an unknown probability density function recursively over time using incoming measurements and a mathematical process model. It allows to compute the sequence of a posterior probability distribution that cannot be directly observed.

In probabilistic robotics, the Bayes filter is used to calculate the posterior probability of a robot's position, taking into account both the measurement noise and the uncertainty in the robot model. The whole process can be described using a graph known as the Hidden Markov Model (HMM) [17], as shown in Figure 2.1.

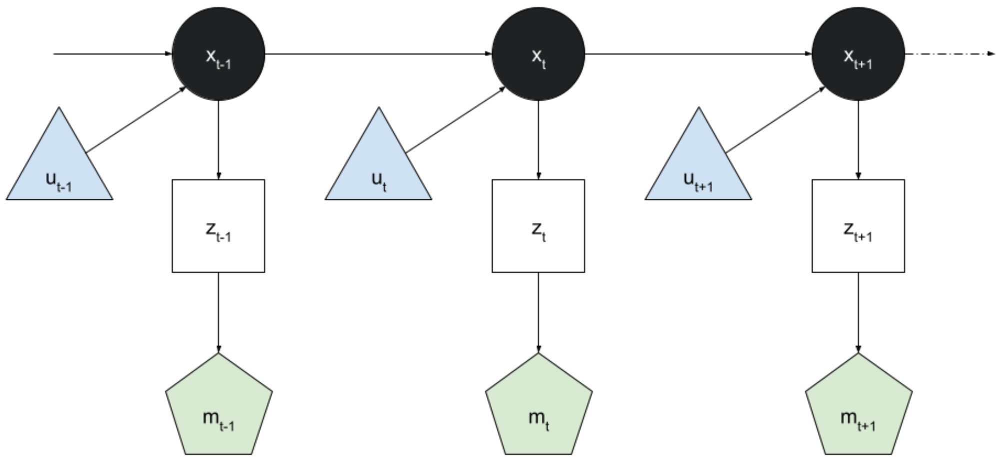


Fig. 2.1. – Bayesian Network of a Hidden Markov Model in the robotic mapping problem. x represents states of the robot, z observations, u actions, and m states of the map at each time step t.

Thus, with a Bayes filter the posterior probability on state x at time step t can be expressed as:

$$P(x_t|z_{0:t}, u_{0:t}) = \frac{P(z_t|x_t, z_{0:t-1}, u_{0:t})P(x_t|z_{0:t-1}, u_{0:t})}{P(z_t|u_{0:t})} \tag{2.3}$$

In Equation 2.3, the posterior distribution $P(x_t|z_t, u_t)$ is also called *belief* $Bel(x_t)$, and it represents the confidence or certainty of the state x_t, given the all the observations $z_{0:t}$ and actions $u_{0:t}$ until time t. The denominator $P(z_t|u_{0:t})$ is a normaliser that is necessary to ensure that the left hand side of Bayes rule is indeed a valid probability distribution, and it can be replaced by a constant η:

$$Bel(x_t) = \eta P(z_t|x_t, z_{0:t-1}, u_{0:t})P(x_t|z_{0:t-1}, u_{0:t}) \tag{2.4}$$

Applying the assumptions of Markov chain model and the law of the total probability to Equation 2.4, we obtain the following:

$$\begin{aligned} Bel(x_t) &= \eta P(z_t|x_t)P(x_t|z_{0:t-1}, u_{0:t}) \\ &= \eta P(z_t|x_t) \int P(x_t|x_{t-1}, z_{0:t-1}, u_{0:t})P(x_{t-1}|z_{0:t-1}, u_{0:t})dx_{t-1} \\ &= \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_t)P(x_{t-1}|z_{0:t-1}, u_{0:t})dx_{t-1} \\ &= \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_t)P(x_{t-1}|z_{0:t-1}, u_{0:t-1})dx_{t-1} \end{aligned} \tag{2.5}$$

In Equation 2.5, the second factor within the integral represents $Bel(x_{t-1})$, which is the confidence of the state at time $t-1$. A more compact way of expressing Equation 2.5 using this recursive form of the Bayes filter is:

$$Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}, u_t)Bel(x_{t-1})dx_{t-1} \tag{2.6}$$

To get the actual value of the belief $Bel(x_t)$, the two generative models involved must be defined. The probability $P(z_t|x_t)$ is often referred to as the *perceptual model*, as it describes how the observations z are generated in the different x states of the robot. The other model, $P(x_t|x_{t-1}, u_t)$, is known as the *motion model*, since it shows the probability with which the action u leads to the state x_t from the previous state x_{t-1}. It should be stressed that this explanation assumes that the map is static, the noise in each state is independent and the model has no approximation errors. It satisfies the Markov property.

There are many different probabilistic algorithms that have been developed for mobile robotic applications. In the next sections, the most relevant techniques in this area are described.

2.3 Localisation

If a robot could incorporate an exact global positioning system, the problem of localisation would be solved. The robot would be able to obtain its position and orientation with respect to a reference frame whenever necessary, and no estimation would be required. However, that technology does not exist yet. Current GNSS devices do not provide the accuracy needed to perform many tasks, nor do they work in all environments. Indeed, there are areas where they cannot receive the signal to provide an effective localisation, commonly known as *GPS-denied areas*.

In addition to the localisation in the environment, mobile robots must be able to calculate their relative position with respect to different frames of the workspace. For example, if there is a human in the surroundings of the robot, the spatial relation between both must be known in order to perform any action that may affect the person. Furthermore, to achieve a safe navigation, the robot must actively detect all the obstacles in its trajectory, which implies computing their position. Unless each element knows its exact position and morphology, and is able to transmit this information to the rest of the entities in real time, creating a perfectly communicated ecosystem, robotics systems need methods to perceive and interpret the environment on their own.

The three main localisation challenges are: pose tracking, global localisation and *kidnapped* robot. In pose tracking, the initial position of the robot is known within an uncertainty level and the objective is to track the motion of the robot for each point in time during navigation. Here, each position depends on the position of the robot in the previous state and the action performed, satisfying the Markov property, as the conditional probability distribution of future states of the process depends only upon the present state. This calculation is done continuously to monitor the whole trajectory of the robot, using odometry and sensor data to get that transformation between states. However, in case of high uncertainty, the calculation of the robot position can quickly degenerate and end with a completely incorrect localisation calculation, since the uncertainty is accumulative. Figure 2.2 depicts an approximation of how the pose uncertainty grows as the robot moves. For this reason, the error made by the sensors should be minimal. In addition, in pose tracking, the initial belief of the robot's position is a normal distribution, and it is not uncertainty-free either.

Global localisation, also known as relocation, refers to the problem of determining the position of a robot under initial global uncertainty. The robot is located at any point in the environment and, with no prior information about its pose, it must

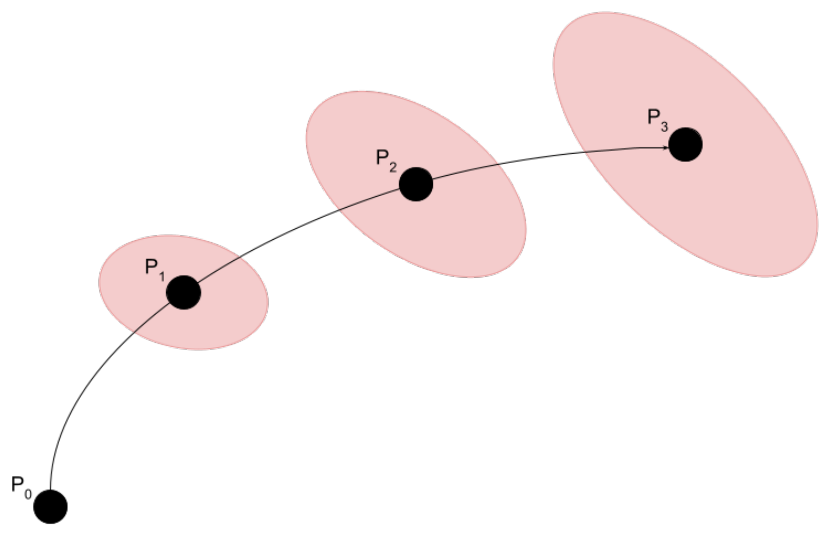


Fig. 2.2. – Approximation of the increment in the odometry uncertainty as the robot navigates. Black circles represent robot positions in different timestamps and red ellipses its possible locations in the space considering measurements noise.

deduce exactly where it is. There are two ways to solve this: by establishing sets of possible hypotheses or by discarding all but the correct possibility. Either way, the robot perceives the environment by means of sensors, with the difficulties that we have already mentioned. The last challenge cited, the kidnapped robot problem, could be considered as an extreme variant of the global localisation problem, as it states a situation where a localised robot is carried to an arbitrary location during its operation. This is commonly used to test a robot's ability to recover from catastrophic localisation failures. An autonomous robot must be able to handle pose tracking and relocation simultaneously, recognising when it is kidnapped and being able to recover from that situation.

It is clear that sensors play a critical role in all of the above forms of localisation, and it is due to the inaccuracy of these that localisation poses difficult challenges. The error committed by these devices is often referred as measurement noise. Sensors are the only source of information with which the environment is perceived, so their accuracy is completely decisive in the performance of the system. They are the entry point of a probably huge decision-making chain, and an unexpected reading at this point can propagate all the way down the chain. Hence, how the system interprets the environment depends on the sensors used to do so, and localisation algorithms must be able to cope with noise to avoid undesired behaviours.

2.3.1 Kalman Filter (KF)

One way to deal with the computational complexity of uncertainty over continuous spaces is by estimating the state with a parametrised function. The Kalman filter (KF) is a technique by means of which this type of uncertainty can be treated, provided that it is represented by probability densities of the Normal or Gaussian type [106]. It is a state estimator that works on the basis of prediction and subsequent correction. This means that it first calculates the uncertainty in a certain state, estimated by a prediction mechanism based on the dynamics of the system, and then corrects this prediction using measurements entering the system. This methodology emerged from the influential contribution of Smith, Self and Cheeseman [213, 212], who introduced the statistical theoretical framework for solving the SLAM problem.

The KF assumes that both the measurements and the state of the system can be described by a linear dynamic system that is affected by white noise. The set of equations that models the evolution of the system state over time, and describes how the measurements are related to it, can be modelled in the discrete state space according to the following state (x_t) and observation (z_t) equations:

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_t + v_t, & v_t &\sim \mathcal{N}(0, Q) \\ z_t &= Hx_t + w_t, & w_t &\sim \mathcal{N}(0, R) \end{aligned} \tag{2.7}$$

A represents the state-transition model, B the input-control model for the action u, and H the observation model. v and w represent noise and are assumed to be, respectively, a multivariate normal distribution with covariance Q and a zero mean Gaussian white noise with covariance R. For the simple case, the various matrices are constant with time, and thus the subscripts are not used, but KF allows any of them to change t each time step [252]. Figure 2.3 shows the model underlaying a KF.

KF can be seen as set of equations that fall into two groups. The first one is known as the *prediction step*, where the prior estimations are used to *predict* the state at the next time step through the idealised version of the dynamic system. These equations are responsible of updating the current state and error covariance based on time intervals, resulting in an *a priori* estimate. Equations belonging to the second group, called the *correction step*, are in charge of the *correcting* the estimation obtained in the other step. They incorporate sensor measurements into the *a priori* result to improve it an obtain a new *a posteriori* estimation.

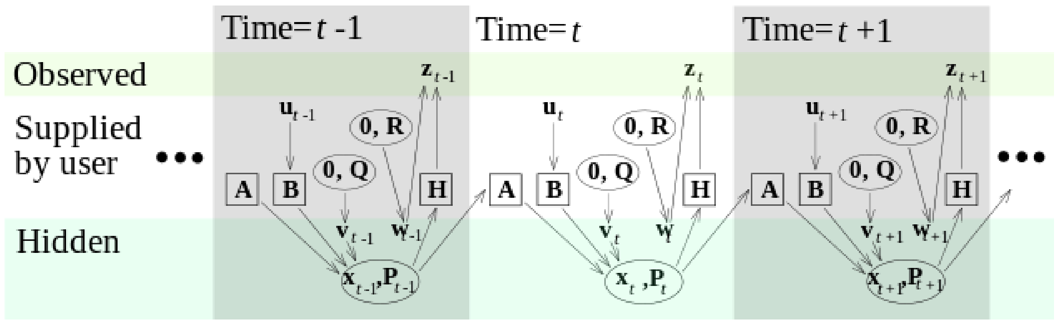


Fig. 2.3. – Model underlying the Kalman filter. Squares represent matrices, ellipses multivariate normal distributions (with the mean and covariance matrix enclosed) and unenclosed values are vectors. P is the *a posteriori* estimate covariance matrix. (Headlessplatter, Public domain, via Wikimedia Commons) [@253].

KF are another type of probabilistic algorithm that have been widely used in robotics. They are based on linear models and work by estimating the state of a system at any given time based on a series of measurements taken over time. The KF then uses these estimates to calculate the most likely state of the system at any given time, as well as its uncertainty. By the nature of the algorithm, it requires an initial state and a covariance matrix, although these can be initialised to any value. For example, if the initial state is unknown, a guess of the initial state can be provided as a starting point, setting the covariance matrix with large values.

The problem with the KF is that it considers a linear system with Gaussian noise, but there are cases where this conditions are not met. For instance, while a lidar produces linear measurements, a radar does not, as it contains angles that are non linear. As a consequence, there is a need to adapt the KF so that it can be used with nonlinear systems. The Extended Kalman filter (EKF) proposed by R. C. Smith and P. Cheeseman [213] is probably the most widely used estimator for nonlinear systems, and it simply approximates the nonlinear models using the first derivative of Taylor series, called Jacobian Matrix, before applying the KF equations. Then, at each time step, the Jacobian is evaluated with current predicted states. In some systems, the derivation of the Jacobian matrices can lead to significant complexities, but there is another method that attempts to simplify this step called unscented Kalman filter (UKF) [104]. In the UKF, the linearisation of the model is performed using the *unscented transformation*, which uses a set of weighted points to parameterise the means and covariances of probability distributions, calculating the statistics of a random variable which undergoes a nonlinear transformation. With the same order of computational complexity as EKF, UKF often provides better estimates for nonlinear systems and is equivalent to KF for linear ones.

2.3.2 Monte Carlo Localisation (MCL)

The Monte Carlo method is a non-deterministic technique used to approximate complex mathematical expressions that are costly to evaluate accurately. Applied to the problem of localisation, the algorithm is known as Monte Carlo Localisation (MCL) [55, 229] and it is one of the most popular techniques in mobile robotics. To localise the robot, the MCL algorithm uses a particle filter (PF) to estimate robot's position, which are a set of algorithms used to solve filtering problems arising in signal processing and Bayesian statistical inference [131]. A PF represents the posterior probability distribution, called *belief*, by considering a set of particles or samples, each one representing a possible state. The term "particle filter" was first coined in 1996 by Del Moral [51] in reference to the mean-field interacting particle methods used in fluid mechanics since early 1960s.

MCL requires a set of initial samples M to track a state vector $X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$. The algorithm is initialised by randomly assigning a state to each particle $x_0^{[m]}$ (with $1 \leq m \leq M$), being M the number of particles in the set X_t. For instance, X_t represents the belief of the robot's position at time t in the localisation problem. The state-space model can be nonlinear and the initial state and noise distributions nonparametric estimations. The particles are then updated using Bayes rule, based on the latest measurements z_t and actions u_t. Each particle has a likelihood weight w_t assigned to it that represents the probability of the system being in the state that particle represents. After this resampling step, the particles with negligible weights are replaced by new particles in the proximity of the particles with higher weights. There are different criteria for such resampling, such as the variance of the weights and the relative entropy with respect to the uniform distribution [54], but the aim is always for the particles to reflect the most recent information about the robot's position. Algorithm 1 shows the basic MCL procedure.

Particle filter techniques provide a well-established methodology [51, 52, 53] for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. Besides, they have the advantage of being very efficient computationally, since it only needs to consider a small number of particles to obtain a good approximation of the posterior distribution. However, these methods do not work well when applied to very high-dimensional systems and they are known to be less reliable than other probabilistic methods, as they are very sensitive to outliers in the data. Adaptive PF techniques aim to reduce this issue by bounding the error introduced by the sample-based representation of the PF. In this line, one of the most relevant approaches is the one proposed by D. Fox [69], where a sampling method based on Kullback-Leibler distance, or

Algorithm 1 Monte Carlo Localisation (MCL) algorithm.

```
1: procedure MCL($X_{t-1},\ u_t,\ z_t,\ m$)
2:     $\bar{X}_t = X_t = \emptyset$
3:     for $m = 1$ to $M$ do
4:         $x_t^{[m]} = MotionModel(u_t,\ x_{t-1}^{[m]})$
5:         $w_t^{[m]} = MeasurementModel(z_t,\ x_t^{[m]},\ m)$
6:         $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:     end for
8:     for $m = 1$ to $M$ do
9:         draw $i$ with probability $\propto w_t^{[i]}$
10:        add $x_t^{[i]}$ to $X_t$
11:    end for
12:    return $X_t$
13: end procedure
```

KLD-sampling, is used. The core idea behind this method is choosing a small number of samples if the density is focused on a small part of the state space, and it chooses a large number of samples if the state uncertainty is high. In this way, KLD-sampling yields similar or even better results while using considerably fewer particles than the original approach.

2.4 Mapping

The mapping problem in robotics consists of generating a spatial model of the environment. These models are mainly used to enable the robot to localise itself with respect to a known reference point and to plan trajectories between two points on the map. In short, the main purpose of creating a digital representation of the environment is to navigate autonomously through it. Besides, the generated models are useful for many other tasks, among which we would like to highlight the simulation of real environments and the acquisition of information from unknown areas or inaccessible to humans.

Although having a map allows for trajectory planning, its very creation also requires the robot to navigate through the environment. Sensors have range limitations that make it impossible for the robot to perceive the whole area from a single point of view, e.g., cameras cannot detect objects that are behind concrete walls. Therefore, sensors must see the environment from different perspectives, so that all the elements can be mapped. As sensors are normally fixed to the robot, it is necessary that the mobile platform navigates through the environment during the mapping process. The actuators that make this motion possible are also subject to errors, i.e., after executing a succession of consecutive movements, the platform may not be where it should theoretically be. If this is the case, and mapping sensor readings are added directly to the map, the map will now correspond to reality, as elements captured in that scan will be added in the wrong place. Sensor readings have their own reference frame, that are transformed into robot's frame. Thus, the error in localisation affects how sensor readings are projected. Sensor noise together with uncertainty in the robot pose gives raise to inaccurate maps or even to not usable maps.

It is worth recalling that in standard mapping processes the robot is normally guided by a human in order to ensure that the environment is fully covered and that the loop is closed. Loop closure refers to exploiting the detection of an already mapped area to minimise the accumulated error [254, 91, 122] during the navigation. It is also the responsibility of the operator to decide the trajectories of the robot and the termination criteria, factors that directly affect the quality of the resulting map and, in consequence, in the performance of the corresponding navigation. Consequent local drifts are accumulated during motion, increasing the uncertainty and leading to an inaccurate estimate. Loop closure enhances considerably the veracity of the map with respect to the traversed path, as the technique attempts to correct this divergence in the position of the robot. Loop closure detection methods differ from one another as they may be geared towards specific map representations [146, 75, 109, 76, 123, 74]. Despite all this, the resulting model usually requires a refining process to correct any erroneously added element. This post-processing can be done manually or automatically [25, 135, 2].

2.4.1 World Representation

The main environment representations used in mapping systems can be hierarchically classified in a few groups, as shown in Figure 2.4. Many other works already explain the different methods to model geometry in a robot readable format [227, 14, 216, 73, 30, 5] and the IEEE Map Data Representation working group developed a standard specification for representing a map used for robot navigation [266].

There are two predominant map representations: topological maps and metric maps. The first ones, similar to graphs, only consider places and relations between them, represent the environment as a set of nodes (locations) that are connected by edges. An edge between two nodes is labelled with a probability distribution over the relative locations of the two poses, conditioned to their mutual measurements [80].

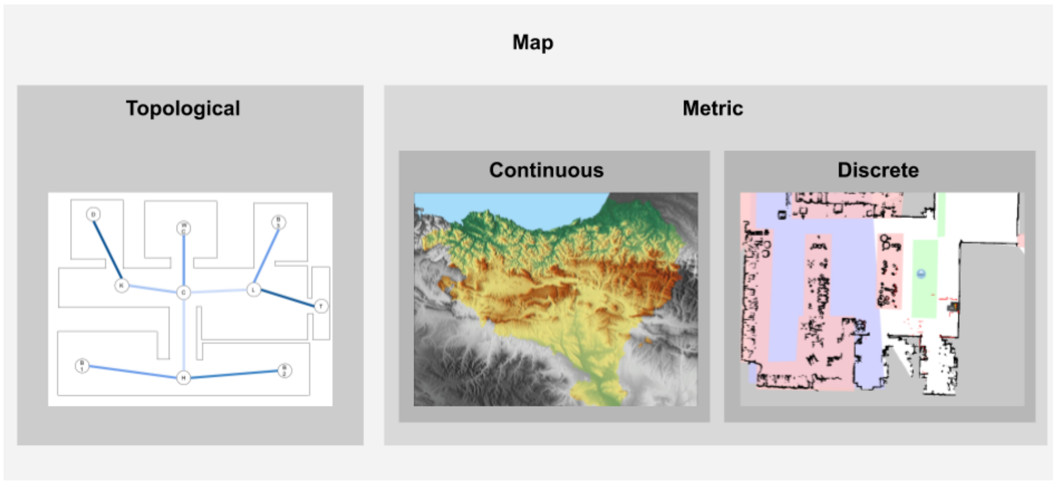


Fig. 2.4. – Map representation types.

A representation of the world based on this simplification eases mapping large extensions and provides all the necessary information for path planning [164]. Notwithstanding the simplification of the world model, the concept of vicinity is lost in topological representations not to mention the absence of explicit information about the occupancy of the space. To overcome this, many authors store additional information or use a combination with metric maps [71, 4, 23, 237, 39].

Alternatively, in metric maps the objects are placed with precise coordinates. They provide all the necessary information to apply a mapping or navigation algorithm, but the map size is directly proportional to the size of the working area, which makes, by itself, costly to model large areas specially concerning 3D mapping. Three are the common representations used in metric maps: landmark-based maps, occupancy grid maps and geometric maps.

Landmark-based representations, also called feature-based representations, identify and keep the poses of certain distinctive elements [126, 127]. A requirement that must be met in these representation is that the landmarks must be unique and distinguishable by the robot perception system. These landmarks can be points, lines or corners, or even faces in 3D mapping, forming a sparse representation of the scene. Landmarks can be more than raw sensor data measurements, such as complex descriptors which establish the corresponding data association with each measurement.

Instead, occupancy grid maps discretise the environment into so-called grid cells. Each cell stores information about the area it covers. It is common to store in each cell a single value representing the probability of being an obstacle there. Grid cells may contain 2D or 3D information [111, 145]. There is a variant referred as 2.5D that, without being a pure 3D grid map, stores height information in an extended 2D grid cell map [85]. Grid maps can be formed by regular grids or by sparse grids. Regular grids make a discretisation of the continuous space into cells that always have the same dimensions, while sparse grids extend the concept of the regular grid by grouping regions with same values, similar to a tree-like approach. In this occupancy grid classification, a 3D technique that it is worth mentioning is the Octree Encoding [152]. An octree is a hierarchical 8-ary tree structure that can represent objects of any morphology in any specified resolution. As it is shown later, it is widely used in systems that require 3D data storage due to its high efficiency, because the memory required for representation and manipulation is on the order of the surface area of the object. A well known implementation of this idea is the OctoMap framework [3] developed by Kai M. Wurm and Armin Hornung [94].

A widely used concept in grid maps is that of costmap. The costmap represents the difficulty of traversing different areas of the map. It takes the form of an occupancy grid with abstract values that do not represent any measurement of the world, as the cost itself is obtained by combining the static map, the local obstacle information and the inflation layer. It is usually used for path planning [150, 160, 144, 137].

Finally, there are geometric maps, which can be considered also discrete maps. In geometric maps all the objects detected by the sensors are represented by simplified geometric shapes, such as circles or polygons. This is an attempt to efficiently represent the environment without losing much information, but it hampers trajectory calculation and the overall management of the data. So, in practice, not many works make use of this method and the occupancy grid map option is preferred [101, 151].

2.5 Simultaneous Localisation and Mapping (SLAM)

The problem of mobile robot navigation has historically been tackled by breaking it down into three subproblems: environment mapping, localisation and path planning. These three subproblems have become broad areas of research for decades, approached separately and deterministically, ignoring uncertainty in robot sensing and motion. In the early 1990s, probabilistic robotics turned the classical approach to navigation on its head by developing algorithms capable of explicitly taking into account the intrinsic uncertainty of the robot's sensors [228, 47]. Probabilistic robotics focuses on representing uncertainty and information through probability

[3] `https://octomap.github.io/`

distributions rather than on the basis of a single guess. With the rise of these methods came the problem of SLAM, also known as concurrent mapping and localisation (CML) [59, 67]: the mapping of sensor readings to a global reference frame depends on the robot's location in that frame and the uncertainty in the robot's pose due to the accumulated odometry error that affects the map construction process. Visual odometry allows for improved navigation accuracy in robots or vehicles using any type of locomotion. Combining visual information from cameras or inertial sensors with information from wheel motion overcomes drift and provides much more accurate visual localisation [72]. However, errors in the constructed map and robot pose are correlated. Therefore, these two problems must be tackled together.

SLAM techniques vary depending on whether indoor or outdoor robots are used. Outdoor environments are more challenging as they do not have specific limits and the criteria of choosing the next navigation point or termination can be completely different. Besides, indoor and outdoor classic SLAM systems differ from each other in aspects such as the individual requirements, sensors provided and morphology of the robots, Markovian, Kalman filter-based and PF-based are the most common techniques used to solve the SLAM problem.

SLAM can be considered to be a mapping process in which robot localisation is uncertain. However, the planning task is put aside during the map building process. Although strategies like coastal navigation can be used [195], while building the map the robot is usually guided by a human by means of teleoperation. In this way, the wheels' drift can be minimised, ensuring as well the full coverage of the environment. Once the map is available, classic planers (A*, Dijkstra, ...) or stochastic planning techniques, e.g., Partially Observable Markov Decision Process (POMDPs), can be used for navigation. Moreover, teleoperating the robot for mapping is usually a highly time-consuming task, especially in large areas or when the movement of the robot is limited. In other cases, it is difficult or even impossible for the robot to be guided due to insufficient connectivity or dangerous conditions, such as in rescue operations of natural disasters.

The advent of SLAM techniques motivated huge advances and opened new possibilities for robot development, but there are still considerable challenges in performance when adding environment dynamism or increasing dimensionality. Subsequently, SLAM-based systems proliferated widely in other areas, such as vision-based online 3D reconstruction or self-driving cars.

SLAM is one of the most demanding competencies for a mobile robot, but successful navigation requires success in all three pillars: perception, the robot must interpret its sensors to extract meaningful data about the surroundings; localisation, the robot must determine its position in the environment; and path planning, the robot must decide how to act to achieve its goals and modulate the outputs of its motors to achieve the desired trajectory.

In the following sections, some of the most relevant methods proposed to solve the SLAM problem are described, with emphasis on pioneering methods leading to new lines of research.

2.5.1 FastSLAM

FastSLAM, proposed for the first time by Motemerlo et al. [155] is considered as the first efficient PF for SLAM, since the previous approaches were only feasible in small environments. FastSLAM utilises a Rao-Blackwellised representation of the posterior, integrating particle filter and Kalman filter representations. On the one hand, the algorithm employs a modified PF for estimating the path posterior $p(x^t \mid z^t, u^t, n^t)$, each particle x^t corresponding to a guess of the robot's pose at time t. z^t and u^t correspond to the measurements and actions, respectively, and n^t to the indexes of the landmarks perceived, all until time t. n_t is often referred to as *correspondence* and, in this formulation, the correspondences $n^t = n_1, \ldots, n_t$ are assumed to be known, and so is the number of landmarks K observed. On the other hand, KF is used to represent the conditional landmark estimates $p(\theta_k \mid x^t, z^t, u^t, n^t)$, using separate KFs for each different landmark. θ_k denotes the location in space of each landmark for $k = 1, \ldots, L$. Since the landmark estimates are conditioned by the trajectory, each sample in the PF has its own local landmark estimates, being a total of $L * M$ KFs for M particles and L landmarks. Therefore, the SLAM problem is divided into one problem of estimating the robot's path x^t and L problems of estimating this same number of landmarks:

$$p(x^t, \theta \mid z^t, u^t, n^t) = p(x^t \mid z^t, u^t, n^t) \prod_{L} p(\theta_k \mid x^t, z^t, u^t, n^t) \tag{2.8}$$

Even if FastSLAM recursively estimates the full posterior distribution over robot pose and landmark locations, it scales logarithmically with the number of landmarks in the map.

In this first version of the method, the proposal distribution representing the motion model is the following:

$$x_t^{[m]} \;\sim\; p(x_t \mid x_{t-1}^{[m]}, u_t) \tag{2.9}$$

Nevertheless, this formula is especially problematic if the vehicle motion noise is large relative to the measurement noise, which is the case of real-world robots. For this reason, the same authors proposed a year later FastSLAM 2.0 [156], which includes a modification that corrects this aspect. Here, both the motion u_t and the measurement z_t are considered for the pose sampling. Thus, the new proposal distribution can be formulated as follows:

$$x_t^{[m]} \sim p(x_t \mid x_{t-1}^{[m]}, u_t, z_t, n_t) \tag{2.10}$$

By the nature of the algorithm, the current estimate of the observed landmark n_t must also be included in order the inclusion of the measurement z_t to make sense.

2.5.2 Unscented Kalman Filter SLAM (UKF-SLAM)

We have already described Kalman filters in Section 2.3.1, and mentioned its expansion for nonlinear systems through EKF and UKF. Although EKF is perhaps the most widely used version of KF to solve the SLAM problem, different experiments show that UKF obtains better results [148], so this one is described in more detail here.

For the application of the KF it is necessary to describe the system by means of stochastic equations of state, which are given by

$$x_t = f(x_{t-1}, u_{t-1}) + v_t \tag{2.11}$$

$$z_t = g(x_t) + w_t \tag{2.12}$$

where x_t is the state of the system, u_t is the action, $v_t \sim N(0, Q_t)$ is the process noise, z_t the measurement, and $w_t \sim N(0, R_t)$ is the measurement noise, all at time t. The state vector $x = [r^T \; m^T]^T$ consists of the state of the robot $x = [x \; y \; \theta]^T$ and the state of the map $m = [m_1^T \; \cdots \; m_n^T]^T$, which is considered static $m_t = m_{t-1} = m$.

The objective of the KF is to estimate the state x_t from the measurements z_t, by means of the prediction and correction steps. This is where UKF differs from the standard KF method. An *unscented transform* (UT) is applied directly in the prediction step to obtain the state vector and the *a priori* covariance matrix using Equation 2.11. In the correction step, the UT is applied to predict the measurement obtained from Equation 2.12. Then, the state and the covariance matrix are corrected as in the classic KF procedure. The UT is designed on the basis that "*it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation*" [105]. In brief, the UT computes a subset of points, called *sigma points*, and propagates them through a nonlinear model function, to then estimate a Gaussian distribution. This same approach has been applied to FastSLAM [112], obtaining better results than those obtained by classic KF and UKF separately [121].

2.5.3 GraphSLAM

Thrun and Montemerlo [230] proposed GraphSLAM, where the SLAM problem is addressed as a graph problem, instead of as a set of independent particles, as in the previous approaches. In GraphSLAM, the SLAM problem compounded of a set of measurements $z_{1:t}$ with associated correspondence variables $c_{1:t}$, and a set of controls $u_{1:t}$, are turned into a graph. Thus, a node can correspond to either a robot pose x_t or a feature in the map m_t at a given point in time t. Consequently, there are two types of edges in this graph: motion arcs, which connect any two consecutive robot poses, and measurement arcs, which connect poses to the landmarks that were observed from there. Figure 2.5 shows how the SLAM problem is represented as a simple graph.

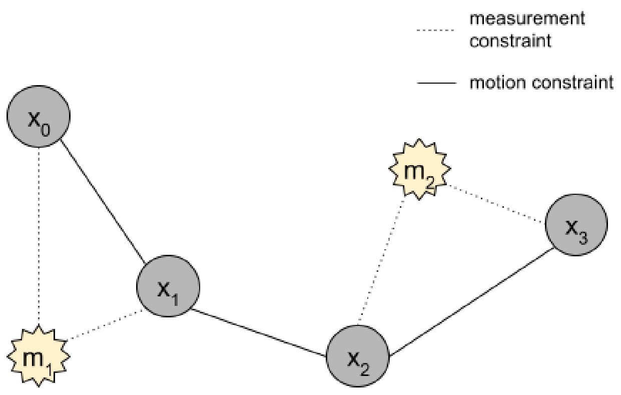


Fig. 2.5. – Graph-based SLAM.

Each edge in GraphSLAM is a nonlinear quadratic *constraint*, which are of the form of Equation 2.13 for motion constraints and Equation 2.14 for measurement constraints. In these equations, while g is the kinematic motion model of the robot and R_t the covariance of the motion noise, h is the measurement function and Q_t the covariance of the measurement noise.

$$(x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \tag{2.13}$$

$$(z_t^i - h(x_t, m_j, i))^T \; Q_t^{-1} \; (z_t^i - h(x_t, m_j, i)) \tag{2.14}$$

The target function in GraphSLAM is to sum all these constraints, which results in a nonlinear least squares problem:

$$\begin{aligned} J_{GraphSLAM} = {} & x_0^T \; \Omega_0 \; x_0 \\ & + \sum_t (x_t - g(u_t, x_{t-1}))^T \; R_t^{-1} \; (x_t - g(u_t, x_{t-1})) \\ & + \sum_t \sum_i (z_t^i - h(y_t, c_j^i, i))^T \; Q_t^{-1} \; (z_t^i - h(y_t, c_j^i, i)) \end{aligned} \tag{2.15}$$

Notice that Equation 2.15 also includes an *anchoring constraint* of the form $x_0^T \; \Omega_0 \; x_0$, corresponding to the absolute coordinates of the map by initialising the very first pose of the robot as $(0 \; 0 \; 0)^T$. Nodes and edges are represented by probability distributions because constraints are inherently uncertain, and thus minimising the objective function yields to the most likely map and robot path.

2.5.4 MonoSLAM

Davison et al. proposed the first SLAM algorithm based in monocular cameras only in the mid 2000s, called MonoSLAM [47, 50]. Up to that moment, the proposed methods used sensor measurements, normally obtained from lidars, as well as robot odometry as input data. The basis of MonoSLAM is building a real-time feature-based map that is represented by a state vector, composed of the stacked state estimates of the camera pose and all the image features, and the covariance matrix for the uncertainty of these estimates. Then, during execution, the state vector is updated using EKF in two alternating ways, when the camera moves between captures and after measurements of features have been achieved. The new state vector is calculated applying the corresponding velocities and accelerations to the previous state.

In order to start the updating steps, firstly, the system is initialised with a known target placed in front of the camera. Here, the state vector with the initial camera position is given an initial low level of uncertainty. On the one hand, this first step allows the system to assign a precise scale to the estimated map and motion and, on the other, it provides several features with known positions to start with the predict-measure-update tracking sequence.

However, not only the camera state must be updated, but also the probabilistic 3D map, as new features can be added to it or even remove some of the ones already mapped. To do this, MonoSLAM *actively predicts* the image position of each feature in the map, to efficiently decide which to measure. Briefly, using the estimates of camera position and features positions, the position of a point feature relative to the camera is calculated first. Then, the position at which the feature is expected to be found in a future state is estimated using the camera calibration parameters and the corresponding relative transformation of the pose of the camera. This calculation results in a Gaussian probability density function that defines an ellipsis in the image within a specific feature is likely to lie. Limiting the search to these areas can increase efficiency and minimise the chance of mismatches. Finally, the detections of the already known features are used to feed the EKF model and update the map accordingly.

Similar to the camera state vector, features have a special initialisation. As the three coordinates of an unknown element cannot be extracted from a single image, a semi-infinite 3D line is assigned to each feature the first time it is added observed, with its corresponding Gaussian uncertainty.

MonoSLAM is the first relevant algorithm removing robot odometry from the equation, giving rise to a new field of study based solely on visual information, known as visual SLAM (vSLAM). In fact, like many later vSLAM algorithms, MonoSLAM does not use a robot itself in their proposal, only a hand-held camera.

2.5.5 Parallel Tracking and Mapping (PTAM)

Parallel Tracking And Mapping (PTAM) algorithm, developed by Klein and Murray [114], is also one of the first vSLAM methods, pioneering the separation of tracking and mapping, and running them into two different threads. Besides, they propose performing the mapping process only when it is necessary, instead of with every new frame of the camera, introducing the concept of *keyframes*.

Firstly, the map is initialised, which is a collection of M point features located with respect to a common coordinate frame W, each one representing a patch in the world. Thus, the j^{th} point in the map p_j has coordinates $p_{jW} = (x_{jW}\ y_{jW}\ z_{jW})^T$ in the global coordinate frame W and a unit patch normal n_j. The map is initialised in a semi-automatic process, where the user has to smoothly translate and rotate the camera to allow the system to estimate the essential matrix of the camera and triangulate the base map. At the end of this step, the map contains two keyframes and describes a relatively small volume of space. New keyframes are then added when the tracking is good, enough time has passed since the last keyframe was added, and the camera pose has changed substantially. The corresponding pose of each keyframe with respect to the world reference frame is calculated using bundle adjustment (BA) [234, 87], which iteratively adjusts the map. As BA has a relatively high computational complexity, it needs too much time to converge as the map gets bigger. Thus, PTAM performs global and local adjustments independently and iteratively even if there is no new keyframe. Here, triangulation between two consecutive keyframes is used to calculate the point's depth information.

Each keyframe that is passed to the mapping module initially considers the tracking system's camera pose estimation and all feature measurements. However, the mapping thread can later re-project and measure new potentially visible features and add them to the list. For each new frame, the tracking system estimates an initial pose of the corners for performing the projection of the map points on the image. Similar to MonoSLAM, PTAM also reduces the search space for new features by transforming map points to the image plane using the camera projection model. Finally, the algorithm uses the correspondences to iteratively update the camera pose μ by minimising the Tukey's biweight objective function Obj [236], which considers the projection error e_j, the noise of the measurement σ_j, and σ_T, an estimate of the distribution's standard deviation derived from all the residuals. This minimisation process for the pose update is expressed by Equation 2.16, where S is a set of successful patch observations.

$$\mu' = \underset{\mu}{\operatorname{argmin}} \sum_{j \in S} Obj\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right), \tag{2.16}$$

2.5.6 Multi-State Constraint Kalman Filter (MSCKF)

In the last years, methods that give solution to the SLAM problem using not only cameras, but also IMUs, have gained popularity in the scientific community. The main reasons for this interest are the increasing use of unmanned aerial vehicles and the ability of inertial sensors not to be so dependent on external conditions. Moreover, in outdoor environments, elements of interest may be so far away that it is difficult to track or match landmarks accurately. The problem of SLAM through inertial and vision sensors is known as visual-inertial SLAM (viSLAM), and the multi-state constraint Kalman filter (MSCKF) [163] is one of the first relevant methods to address the problem in this manner.

MSCKF proposes an EKF-based estimator to track the 3D pose of an IMU frame I with respect to a global frame of reference G, whose state vector is given by:

$$x_{imu} = \begin{bmatrix} {}^{I}q_G^T & b_g^T & {}^{G}v_I^T & b_a^T & {}^{G}p_I^T \end{bmatrix}^T, \tag{2.17}$$

where ${}^{I}q_G$ is the unit quaternion describing the rotation from G to I, ${}^{G}p_I$ and ${}^{G}v_I$ are the position and velocity of the IMU with respect to G, and, finally, b_g and b_a are the biases affecting the gyroscope and the accelerometer, respectively. Accordingly, the state vector from Equation 2.17 has its corresponding error state vector $\tilde{x}_{imu}$. Thus, after reaching time step k and including N camera poses to the EKF state vector, x_{imu} has the following form:

$$\hat{x}_{imu} = \begin{bmatrix} \hat{x}_{imu_k} & {}^{C}\hat{q}_{G,1}^T & {}^{C}\hat{p}_{C,1}^T & \dots & {}^{C}\hat{q}_{G,N}^T & {}^{G}\hat{p}_{C,N}^T \end{bmatrix}^T \tag{2.18}$$

The filter propagation equations are derived by discretisation of the continuous time IMU system model. Upon recording a new image, the state augmentation is performed, computing the camera pose from the IMU estimate as:

$${}^{C}\hat{q}_G = {}^{C}\hat{q}_I \otimes {}^{I}\hat{q}_G \text{ and } {}^{G}\hat{p}_C = {}^{G}\hat{p}_I + C_{\hat{q}}^T\ {}^{I}\hat{p}_C, \tag{2.19}$$

and appending it to the state vector, augmenting the covariance matrix of the EKF accordingly. In MSCKF, the measurement model groups camera observations by feature, rather than per camera pose, avoiding to include the feature position in the filter state vector. Finally, the EKF is updated when a feature that has been tracked in a number of images is no longer detected and every time a new image is recorded.

2.5.7 ORB-SLAM

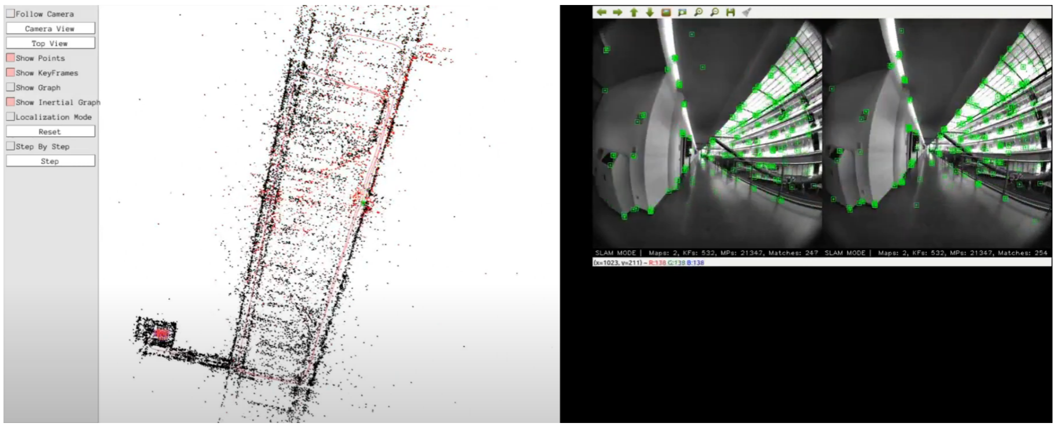
ORB-SLAM can already be considered as a family of methods, as since 2015 four different approaches have been published on the same basis [141], all of them with meaningful impact.

ORB-SLAM [165] emerges as a monocular SLAM system capable of performing tracking, mapping, relocalisation and loop closing using the same ORB features (described in Section 2.6). The method is based on the ideas of PTAM and, similar to this, performs tracking and local mapping in different threads. However, ORB-SLAM also includes loop closing, which is executed in a third thread, and the map initialisation step is done automatically. To accomplish it, the system actively searches for feature matches until a safe two-view configuration is found. If there is enough parallax, the adequate model between homography and fundamental matrix is selected, to then retrieve the associated motion hypotheses and, finally, perform a BA.

Once the map is initialised, it starts tracking and mapping continuously. The former is executed every time the camera captures a new frame, and consists of performing a motion-only BA to estimate the pose and deciding if that frame is considered as a keyframe for mapping or not. If so, the mapping module receives that keyframe and triangulates ORB with several keyframes to decide if the new frame and features are added to the map. Here, again, a local BA is performed to optimise the pose of all the connected elements in the map, represented as a covisibility graph. As a last step of the mapping process, an attempt to detect and remove redundant keyframes is made.

The third pillar of ORB-SLAM is loop closing. By representing the last keyframe as a bag of words, its similarity with all its neighbours in the covisibility graph is computed. Then, with those with the highest scores, their relative transformation is calculated as in [167] and, if the correspondence with any of them is supported by enough inliers, the loop is considered as valid. Detecting a loop closure implies fusing duplicate points, inserting new edges in the covisibility graph and performing a pose graph optimisation.

Following the same idea described here, Visual Inertial ORB-SLAM (VIORB) [168] extends this method to include IMU data in all four map initialisation, tracking, mapping and loop closing procedures. On the contrary, ORB-SLAM2 [166] was presented at about the same time as VIORB, but it focuses on adapting the method to stereo and RGB-D cameras. Recently, ORB-SLAM3 [32] has been presented, a system able to perform visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models. Its main novel contributions are a viSLAM system that fully relies on Maximum-a-Posteriori (MAP) and a multiple map scheme that allows it to recover from long periods of poor visual information. Besides, it is all implemented in a unique open-source library [4]. Figure 2.6 shows an example of the system mapping a university-like environment with a hand-held camera.



```
SLAM MODE |  Maps: 2, KFs: 532, MPs: 21347, Matches: 247  SLAM MODE |  Maps: 2, KFs: 532, MPs: 21347, Matches: 254
(x=1023, y=211) ~ R:138 G:138 B:138
```

Fig. 2.6. – ORB-SLAM3 running in the TUM-VI dataset [@243]. On the left-hand side, the points and graph of the map being built are shown. On the right-hand side, the current frame captured by the camera with its corresponding features. Image obtained from [@176].

2.6 Image Feature Detection and Matching

Some of the SLAM methods described in Section 2.5 are based on techniques that attempt to extract features from images in order to compare them with subsequent images and thus deduce the motion associated with the transition from one capture to another. For this reason, below, we describe briefly the main feature extraction methods used in mobile robotics and, in general, in computer vision.

Scale Invariant Feature Transform (SIFT)

The Scale Invariant Feature Transform (SIFT) [136] is an algorithm used in computer vision to detect and identify similar features between different digital images. It consists of calculating numerical information derived from the local analysis of an image, called descriptors, that characterises its visual content regardless of scale, framing, viewing angle and exposure. To get them, the first step of the algorithm

[4]https://github.com/UZ-SLAMLab/ORB_SLAM3

is the detection of key points, each of which is defined by its coordinates on the image and a characteristic scale factor. This is followed by a reconvergence and filtering step where the accuracy of the location of the key points is improved and a number of irrelevant ones eliminated. Each remaining key point is then associated with an intrinsic orientation that depends only on the local content of the image around the key point, at the scale factor considered. This ensures the invariance of the method to rotation and is used as a reference in the calculation of the descriptor, which is the final step in this process. Hence, two images of the same object are likely to have similar SIFT descriptors, especially if the shooting times and viewing angles are close, and vice versa in two photographs of very different subjects. This discrimination robustness is a fundamental requirement for most applications and explains much of the popularity of the SIFT method.

Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) [18] is a fast and robust algorithm for local, similarity invariant representation and comparison of images. In SIFT, a window is used to extract corners for potential key points, detecting changes in the direction of lines for that particular window size and, to detect larger corners, larger windows are used. Laplacian of Gaussians (LoG) is useful for detecting edges that appear at various image scales or degrees of image focus, but it is costly. Thus, SIFT approximates LoG with Difference of Gaussians (DoG) to do it faster. SURF goes a step further and approximates LoG with box filters, which approximate second-order Gaussian derivatives and can be evaluated at a very low computational cost using integral images and independently of size. Besides, this method relies on the determinant of the Hessian matrix for both space and scale selection, improving the performance even more. This fast computation of operators makes SURF a very interesting option for real-time applications such as tracking and object recognition.

Features from Accelerated Segment Test (FAST)

SURF improves the execution time of SIFT, but is still not fast enough for SLAM with limited computational resources. To overcome this issue, Features from Accelerated Segment Test (FAST) [194] is proposed, which focuses on the computational efficiency of the corner detection. FAST uses a 16 pixel circle around each pixel of the image to determine whether it is a corner or not. If, here, a set of N contiguous pixels are all brighter or darker than the intensity of the candidate pixel p plus minus a threshold value t, then it is classified as a corner. Thus, N and t values have a significant impact on the performance of the algorithm and they should be selected wisely, since the number of detected corner points should not be too many and the computational efficiency should not be sacrificed to achieve high performance. In this process, machine learning is applied to achieve superior computational efficiency, first performing corner detection with a given N on a set of training images and then creating a decision tree with them based on the ID3 algorithm [186] for fast detection in other images.

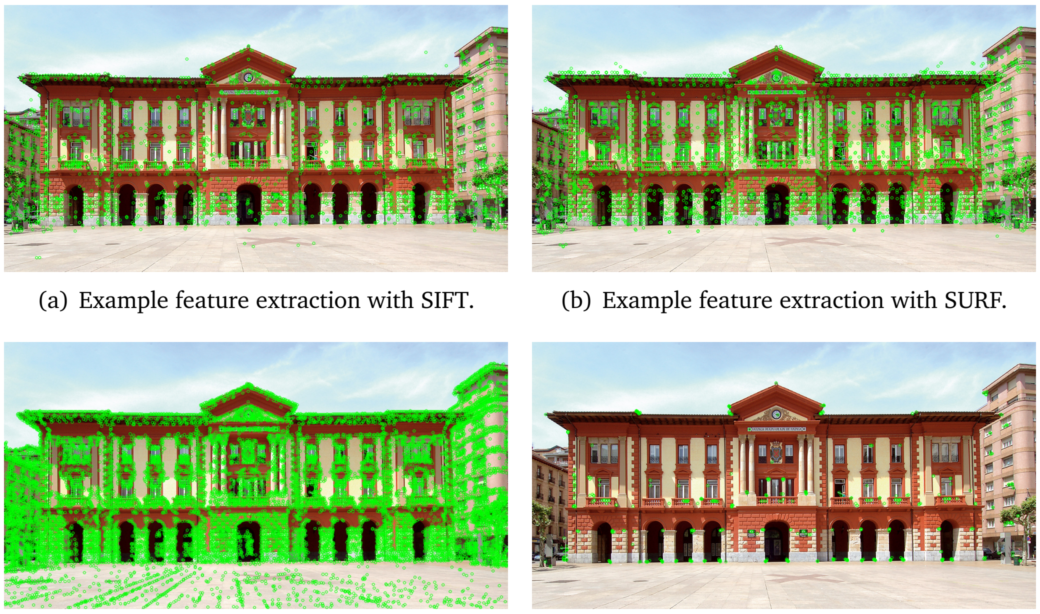
Binary Robust Independent Elementary Features (BRIEF)

The main difference of the Binary Robust Independent Elementary Features (BRIEF) [31] method is that it uses binary strings as an efficient feature point descriptor. The neighbourhood defined around a pixel or key point is known as a patch, which is a square of a certain width and height of pixels. BRIEF identifies the patches around the key point and converts them into a binary vector, so that they can represent an object. In short, it is a faster method for feature descriptor calculation and matching, but it does not provide a method to find the features, and any other feature detector must be used for this purpose. Besides, it also provides a high recognition rate unless there is a large rotation in the plane.

Oriented FAST and Rotated BRIEF (ORB)

Oriented FAST and Rotated BRIEF (ORB) [196] is a fusion of FAST key point detector and BRIEF descriptor with some modifications to enhance the performance. ORB performs as well as SIFT on the task of feature detection, while being almost two orders of magnitude faster. Besides, it adds an analysis of variance and correlation of oriented BRIEF features and a learning method for decorrelating BRIEF features under rotational invariance, leading to better performance in nearest-neighbour applications. It is worth noting that the SLAM method described in Section 2.5.7, called ORB-SLAM, is based on this technique.

An example of applying the feature detection algorithms described here to an image can be seen in Figure 2.7. There are many other feature detection algorithms, but these are still the most used ones in vision-based SLAM systems, even in the latest advances [263, 265].



(a) Example feature extraction with SIFT.

(b) Example feature extraction with SURF.

(c) Example feature extraction with FAST.

(d) Example feature extraction with ORB.

Fig. 2.7. – Example feature extraction with the different methods described in Section 2.6. Each algorithm is applied with the parameters presented in their original articles, which may not be the optimal ones for this image.

2.7 Planning

There are different approaches to address the problem. On the one hand, there are deliberative or hierarchical systems [36], where the trajectory to be executed is calculated in advance. On the other hand, there are reactive systems, which rely on the current state of the system to decide the next action to take, repeating this process iteratively until the final goal is reached [132]. From the combination of these two systems, hybrid methods appear, where the deliberative system is above the reactive system. Here, the former is in charge of planning the main trajectory and the latter tries to follow it while avoiding obstacles [180]. This subdivision leads us to introduce two other key concepts in autonomous navigation systems, global planning and local planning. In short, global planning refers to the process of calculating the positions through which the mobile platform has to go in order to reach the destination. This is usually based on the static map, which includes the fixed elements of the environment. Local planning (or obstacle avoidance) is, in contrast, the task of calculating the velocity commands to be sent to the platform in order to follow the global trajectory. It is based on a local map that moves together with the position of the robot, as it is intended to represent a space around the robot. In this local map, all the elements captured by the sensors, both fixed and dynamic, are included, since the local planner is in charge of avoiding collision with any other element in the surroundings. In practice, in most cases global and local terms can be directly replaced by static and dynamic terms, as deliberative systems are predominant in global planning and reactive systems in local planning.

Additionally, it is noteworthy that, with the recent explosion of machine learning research, data-driven techniques are also being applied to the problem of autonomous navigation [238]. Work of this nature has resulted in systems that use end-to-end learning algorithms in order to find navigation systems that map directly from perceptual inputs to motion commands, thus bypassing the classical hierarchical paradigm.

2.7.1 Partially Observable Markov Decision Process (POMDP)

A Markov Decision Process (MDP) is a time-dependent random phenomenon for which the Markov property is fulfilled. This refers to the property of certain stochastic processes whereby the probability distribution of the future value of a random variable depends only on its present value, being independent with respect to the history of that variable. In a MDP, the state is fully observable, i.e., the agent has full information about the current state. However, as we have already seen, this is rarely the case in mobile robotics, as uncertainty is abundant in real-world planning scenarios. When the Markov model underlying the data is completely unknown for the agent, even if observable data is available, it is known as a HMM (already mentioned in Section 2.2). However, this is not generally the case in mobile robotics either, as localisation algorithms, for example, provide estimates of the robot's state. When information about the state is partially available in a MDP, the process is known as a Partially Observable Markov Decision Process (POMDP). In robot control, it addresses both the uncertainty measurement and the uncertainty in control effects.

A POMDP models a decision process in which a sensor model must be maintained, which is represented as a probability distribution of different observations Ω given the underlying state x. Unlike in a MDP, which maps the underlying states X to actions A, the POMDP policy π is a mapping of the history of observations, or belief states B, to actions A. In robot navigation, the goal is to find the policy π that specifies the motion command $\pi(b)$ that is executed in belief state b and maximises the value function V_T

$$V_T(b) = \gamma \max_u \left[r(b,u) + \int V_{T-1}(b')p(b'|u,b)db' \right], \tag{2.20}$$

which leads to the control policy

$$\pi_T(b) = \operatorname*{argmax}_u \left[r(b,u) + \int V_{T-1}(b')p(b'|u,b)db' \right] \tag{2.21}$$

The value function in POMDPs is defined over the space of all beliefs the robot might have about the state of the environment. Solving POMDPs exactly over the continuous space is often computationally intractable and, thus, the best known mobile robotics algorithms are approximate. In grid-based algorithms, for example, the value function is calculated for a set of points in the belief space, determining the optimal action to take for states that are not in the set of grid points by means of interpolation.

Although POMDPs are a general representation of motion control problems, the main way of dealing with a navigation problem is to combine global planning with local planning or obstacle avoidance. For this reason, the main idea underlying the most relevant algorithms in each of these two groups are detailed below, which are based on discrete map representations.

2.7.2 Global Path Planning

We have already mentioned that the most commonly used maps in mobile robotics are discrete maps, and one of the main reasons is that they simplify the calculation of trajectories considerably. More specifically, the most efficient discrete maps for this purpose are grid maps. These cell-based maps can be easily represented by graphs, where sections are equivalent to vertices - also called 'nodes' - that are connected by edges if the robot can navigate from one vertex to another. Thus, global path planning can be seen as a graph problem, being the objective to find the shortest path between two vertex. In this context, we should understand "shortest" as the optimum for the cost function we use. That is, if the edge weights contain only distance information, the calculated path will actually be the shortest. But, for instance, if they also include the electricity consumption of getting from one cell to another, the planned path will be the most energy-efficient.

One way or another, the meaning of the value of each edge is, in principle, irrelevant with respect to the path planning methods. In fact, the algorithms on which most of them are based were developed outside the field of autonomous navigation and even before it emerged, including those at the cutting edge [3, 220, 133]. In this sense, we will explain the core idea behind the main algorithms for the determination of the shortest path in a graph, which are: the breadth-first search, the depth-first search, Dijkstra's algorithm, and the A* algorithm.

Depth-First Search

Similarly, the depth-first search (DFS) [138] consists of expanding each and every one of the nodes that it locates, recursively, on a specific path. When there are no more nodes to visit on that path, it applies backtracking, and repeats the same process with each of the siblings of the node already processed. However, as in this case the last discovered state is the first to be considered, it is unsuitable for discrete maps, as it can be completely inefficient for destinations close to the origin, especially in the hypothetical case where the map is infinite, as it would not terminate. Its computational cost is $O(|V|+|E|)$, where $|V|$ and $|E|$ are the number of vertices and edges, respectively, assuming that all basic operations such as determining whether a point has been visited are performed in constant time.

Breadth-First Search

In a breadth-first search (BFS) [274], you start at the origin point and scan all neighbours of this node, i.e. all positions adjacent to it. Then, for each of the neighbours, its respective adjacent neighbours are examined, and so on until the target is reached. So all trajectories with k steps are always considered before any with $k + 1$ steps. As in the case of DFS, the computational cost of BFS is $O(|V| + |E|)$.

Dijkstra

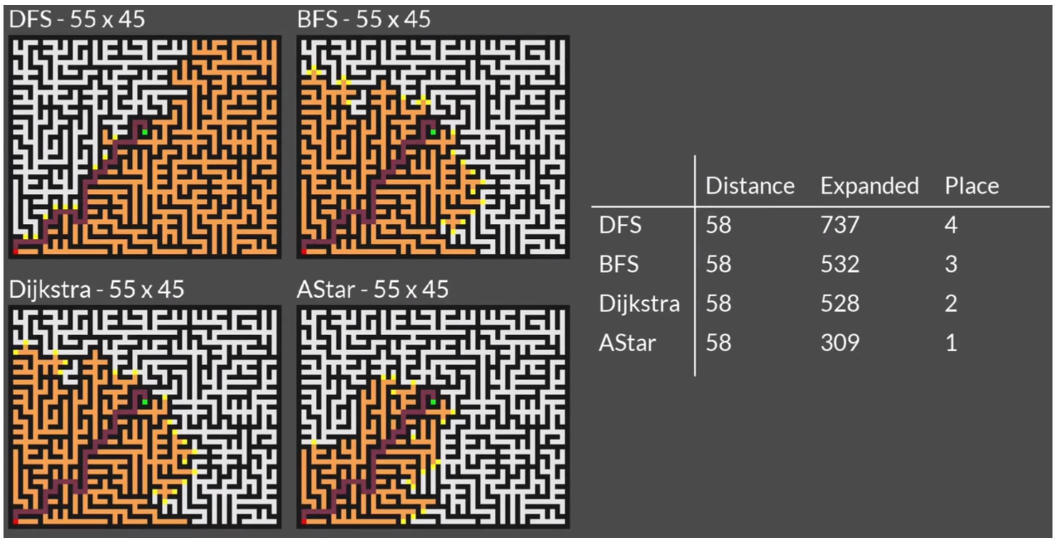
Dijkstra's algorithm determines the shortest path given an origin vertex to the rest of the vertices in a graph with weights on each edge [57]. The idea behind this algorithm is to explore all the shortest paths starting from the source vertex and leading to all the other vertices; when the shortest path from the source vertex to the rest of the vertices that make up the graph is obtained, the algorithm stops. Dijkstra's algorithm belongs to the group of greedy algorithms, since in each iteration the vertex with the smallest value is selected. This is a specialisation of the uniform cost search and, as such, it does not work on graphs with negative cost edges -

by always choosing the node with the shortest distance, nodes may be excluded from the search that in future iterations would lower the overall cost of the path by passing through a negative cost edge. The computational cost of Dijkstra's algorithm is $O(|E| + |V|log|V|)$ and can be lower depending on the problem [40].

A*

The A* or A-star search algorithm is an extension of Dijkstra's algorithm that aims to reduce the total number of states it explores by incorporating a heuristic estimate of the target performance given a state [86]. Thus, it uses an evaluation function $f(n) = g(n) + h'(n)$, where $h'(n)$ represents the heuristic value to go from node n to the end, and $g(n)$ the actual cost to reach that node from the start. The algorithm is a combination of breadth-first and depth-first searches: while $h'(n)$ tends to be depth-first, $g(n)$ tends to be breadth-first. In this way, the search path is changed whenever there are more promising nodes, using a priority queue ordered by the value of the function $f(n)$. Thus, A* is also a greedy algorithm, as it always selects the locally optimal option. Besides, it is a complete algorithm because if a solution exists, it will always find it. The time it takes to achieve this will depend on the quality of the heuristic estimation. In the worst case, the complexity will be exponential, while in the best case, the goal can be reached in linear time. To guarantee the optimisation of the algorithm, the function $h'(n)$ must be heuristically admissible, that is, it must not overestimate the real cost of reaching the target node. Moreover, if $h'(n) = 0$ is satisfied for every node n in the network, A* becomes an uninformed uniform cost search, behaving as Dijkstra's algorithm. The performance of the A* algorithm is the best of the four approaches presented in this section, with a worst-case computational cost $O(|E|)$.

Figure 2.8 shows a comparison of these four path-finding algorithms. It should be noted that a single example should not be taken as representative of its efficiency, as each method has particular circumstances in which it is the optimal one. Moreover, in BFS and DFS, the path found is the shortest if all the edges have the same weight - or they are unweighted - since they explore the graph based on the number of edges traversed from the origin. As the maps used do not always meet this condition, Dijkstra and A* are the most used methods which are, in addition, more efficient.



	Distance	Expanded	Place
DFS	58	737	4
BFS	58	532	3
Dijkstra	58	528	2
AStar	58	309	1

Fig. 2.8. – A comparison of the main path-finding algorithms in a 55x45 cell example maze and a random destination point. White colour depicts the free unexplored space, black are the obstacles (or walls), green is the destination point, red the origin (or robot position), orange colour represents the explored nodes, yellow the next nodes to analyse, and maroon is the trajectory found. On the right-hand side of the image, some metrics of their performance are shown [@215].

2.7.3 Obstacle Avoidance

Local obstacle avoidance focuses on obtaining an alternative trajectory to the main trajectory based on the information gathered from the sensors during the navigation of the vehicle. The resulting movement is calculated based on the most recent sensor readings, robot's target position and its position relative to the target position. Obstacle avoidance methods act as intermediaries between the global path planner and the robot control module. Using a map, the global planner creates a kinematic path for the robot to get from a starting point to a destination point. Until the target is reached, the obstacle avoidance system creates a local map around the robot, with an associated cost function. This area is depicted as a grid-map, whose values are calculated, generally, based on the risk of collision or distance to the nearest obstacle. The task of the local planner is to use this cost function to determine the linear and angular velocities to send to the robot. The obstacle avoidance approaches presented below depend to a greater or lesser extent on the existence of a global map, a target destination or trajectory and precise knowledge of the location of the robot in the map.

Dynamic Window Approach

The Dynamic Window Approach (DWA) [70] is a planner that calculates the optimal collision-free trajectory for a robot to reach its goal (see Figure 2.9). However, it differs from the rest in the sense that it only works in the velocity space, as it uses the Cartesian axes (x, y) and converts them into velocities (v, w) for the robot.

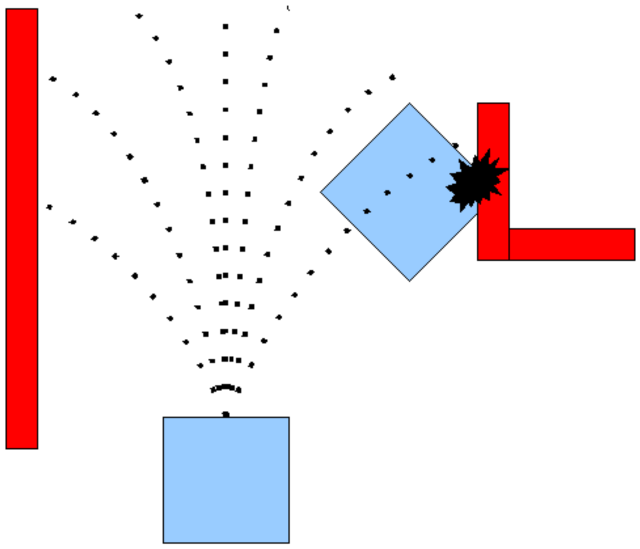


Fig. 2.9. – Illustration of the Dynamic Window Approach algorithm [@162]. The blue box is the robot, obstacles are in red, and the discontinued lines represent the calculated possible trajectories.

DWA algorithm focuses on two main goals: calculating a valid velocity space and choosing the optimal velocity. To achieve this, the search space is constructed from the set of velocities which produce a safe trajectory, called *admissible* velocities, allowing the robot to stop before colliding. The next step is to reduce these candidates using a *dynamic window*, which consists of all tuples (v, w) that can be reached within the next sample time period, given the acceleration capabilities and the current velocity of the robot. Finally, the tuple chosen will be that maximising (a) the alignment of the robot with the goal direction, (b) the distance between the estimated path and the nearest obstacle, and (c) the speed at which the robot moves towards the destination.

Rapidly-Exploring Random Trees

The underlying idea behind the algorithms based on Rapidly-Exploring Random Trees (RRTs) [125] is to generate random displacements until a succession of them form the desired trajectory (see Figure 2.10). Considering the starting point as the root, a tree is generated with nodes (positions) and branches (movements) connected to each other, which are added to the tree in case they are feasible, i.e., they pass entirely through free space.

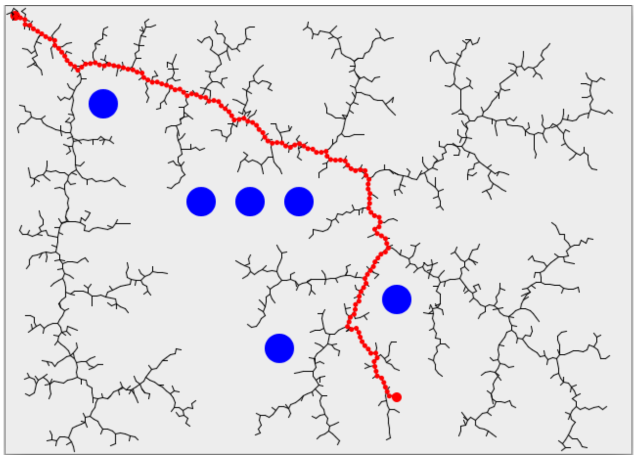


Fig. 2.10. – Example of path planning using Rapidly-Exploring Random Trees [7]. Blue circles are obstacles, black lines are the explored trajectories and red is the selected one.

With uniform sampling of the search space, the probability of expanding an existing node is proportional to the size of its Voronoi region. As the largest Voronoi regions belong to the states on the frontier of the search, this means that the tree preferentially expands towards large unsearched areas. Because of this feature, this type of algorithms have been widely used in the autonomous exploration of unknown environments.

Lazy Theta*

This algorithm is based on Theta* [45], which in turn is based on A*. Theta* is a variation of A*, which arises from the fact that the A* algorithm finds the shortest path in the graph, but this one does not have to coincide with the shortest path in the continuous environment. Theta* simply removes this restriction, making it an *any-angle* path planning algorithm (see Figure 2.11). Thus, the difference between the two algorithms is that Theta* allows the parent of a vertex to be any visible vertex, in contrast to A*, where the parent must necessarily be an unoccupied neighbour.

What happens is that if Theta* checks the visibility between a vertex and its parent, and this vertex is never expanded, i.e., its neighbours are never analysed, the check

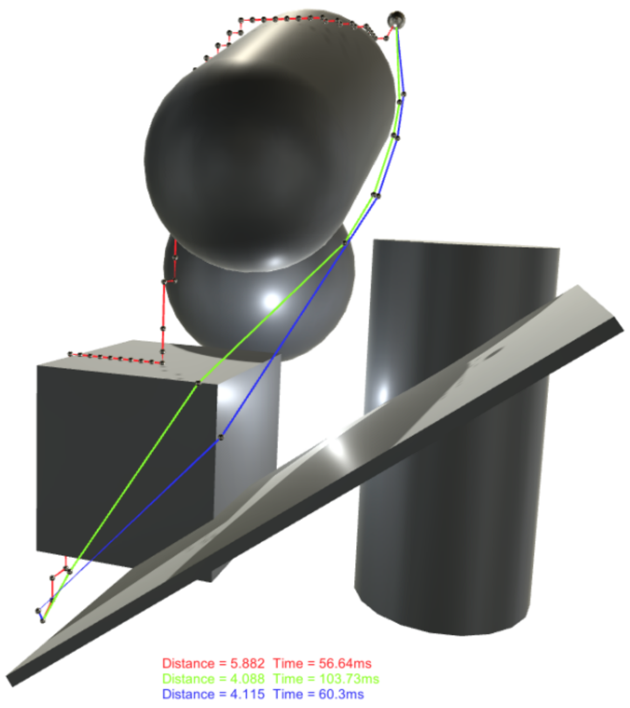

Distance = 5.882 Time = 56.64ms
Distance = 4.088 Time = 103.73ms
Distance = 4.115 Time = 60.3ms

Fig. 2.11. – A* (red), Theta* (green) and Lazy Theta* (blue) path planning comparison in an example 3D environment [@90].

is unnecessary. The *Lazy* variant, uses a different evaluation to perform only one line-of-sight (LOS) check per expanded vertex, at the expense of slightly more expanded vertices, returning practically the same paths as Theta*.

2.8 Active Mapping

Active mapping, also called Active SLAM (ASLAM), is the task of actively planning robot paths while simultaneously building a map and localising within it [164]. ASLAM goes a step further than the classic SLAM problem as it seeks the robot to move autonomously during the whole mapping process. Exploration while mapping could simplify the setting up of a navigation system in many applications, as the robot would be capable of building the map by itself with no human interaction.

ASLAM pretends to overcome the disadvantages and potential sources of incongruity previously mentioned by developing methods to perform the exploration step automatically, i.e., automatising the robot guiding process by selecting online the paths or navigation sub-goals that will lead to a more accurate map. The exploration field, extensively studied in the last decade [119, 209, 256, 221, 211], offers great advantages for mobile robots, especially in hostile environments.

It has also been referred to in scientific literature as automatic SLAM [203, 107], autonomous SLAM [38, 113, 110], adaptive CML [67], SPLAM (Simultaneous Planning, Localisation and Mapping) [216] or robot exploration [262].

This concept of active motion selection is addressed from two research areas: mobile robotics and computer vision. In robotics, the problem is known as robot exploration whether in computer vision the term active perception is used.

2.8.1 Robot Exploration

In robotics [255], exploration refers to the autonomous creation of an operational map of an unknown environment. Besides generating a world representation while dealing with uncertain localisation, the robot must control its motion. Namely, it must calculate the goals and the actions to take thereon to achieve those goals while actively reacting to unexpected situations.

According to the literature, an ASLAM algorithm consists of three iterative phases [5]: pose identification, goal selection, and navigation and checking, whose relationship is shown in Figure 2.12. These phases involve the classic tasks of localisation, path planning and mapping. The pose identification step identifies the possible destinations; the goal selection phase selects the next destination; and, in the last stage, the algorithm termination is evaluated based on the remaining candidate points. As it can be seen, the three phases depend on the candidate viewpoints. Those candidates are just possible destinations of the mobile robot, which are limited by the model used to manage the navigation component. Therefore, how the environment data is represented may have a huge impact.

Pose Identification

Given the partial map of the environment, the robot identifies a set of destinations. In theory, all the physically reachable and non-visited destinations should be evaluated, as they are potential gains of information and, thus, any of them could be the desired optimal viewpoint. However, in practice, the computational complexity of the evaluation grows exponentially with the search space which proves to be computationally intractable in real applications [26, 149, 216]. Thus, the implementation of certain filters reduce the number of candidates and the complexity of the

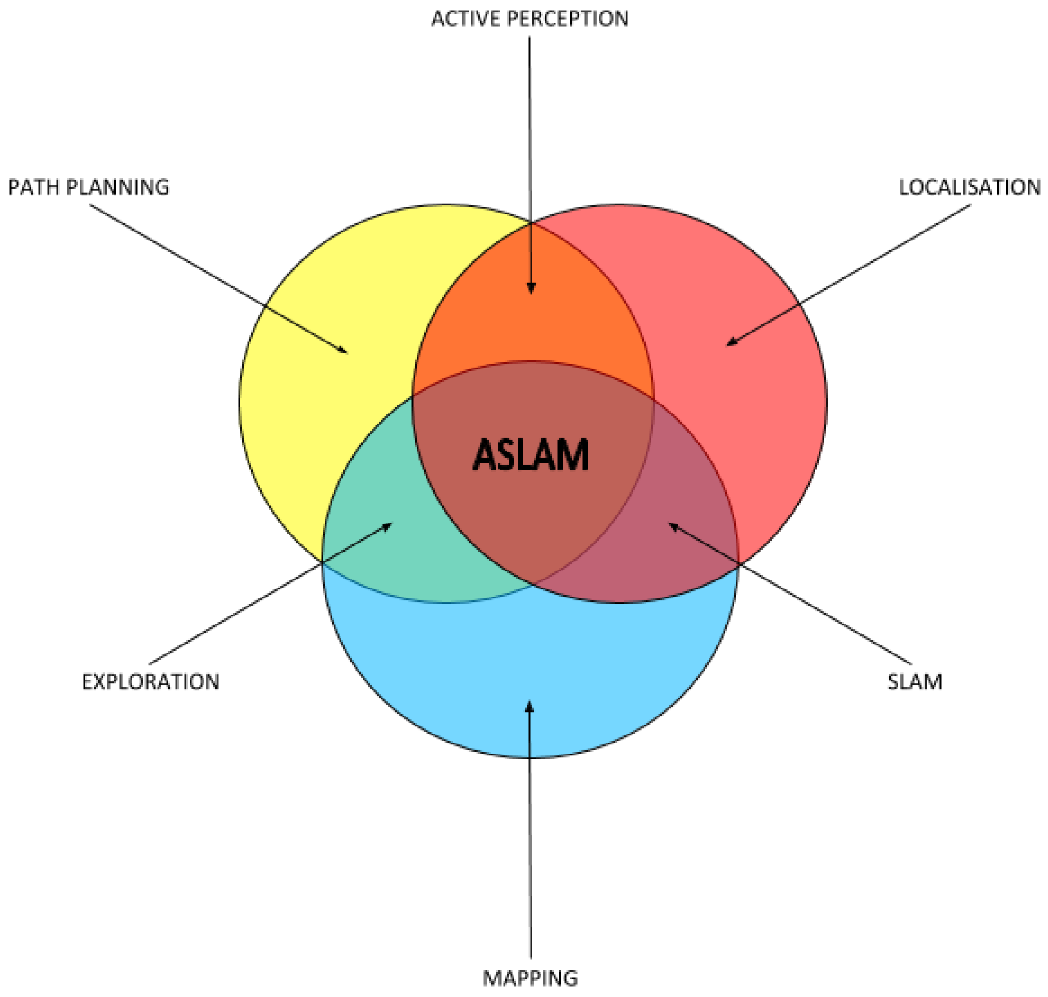


Fig. 2.12. – Components that form the Active Simultaneous Localisation And Mapping (ASLAM) problem, expressed as set theory. Overlapping areas represent the combination of individual tasks.

problem [241, 204]. Some poses can be discarded a priory, such as the ones that are already tested, the ones surrounded by already mapped points or the ones that are too far in the unknown space.

The most straightforward approach could be one that sets random destinations to an existing SLAM system until a certain condition is met. In this case, the ASLAM solution would not differ much from a SLAM technique. However, researchers have developed more elaborated algorithms to autonomously map an environment.

In terms of dimensionality reduction and mapped space, a widely used concept in the context of exploration is that of a frontier [260, 262, 261]. Frontiers are regions on the boundary between open space and unexplored space, i.e., points in the map between the free known space and the unknown space. The main idea behind these points is that, as they are in the mapped space, it is very probable that the robot can reach them. In addition, they ensure that unexplored areas next to them can be covered. Frontiers, thus, represent optimal sets of points to be reached in order to expand the explored environment. An example of a map with all the frontiers identified is shown in Figure 2.13.

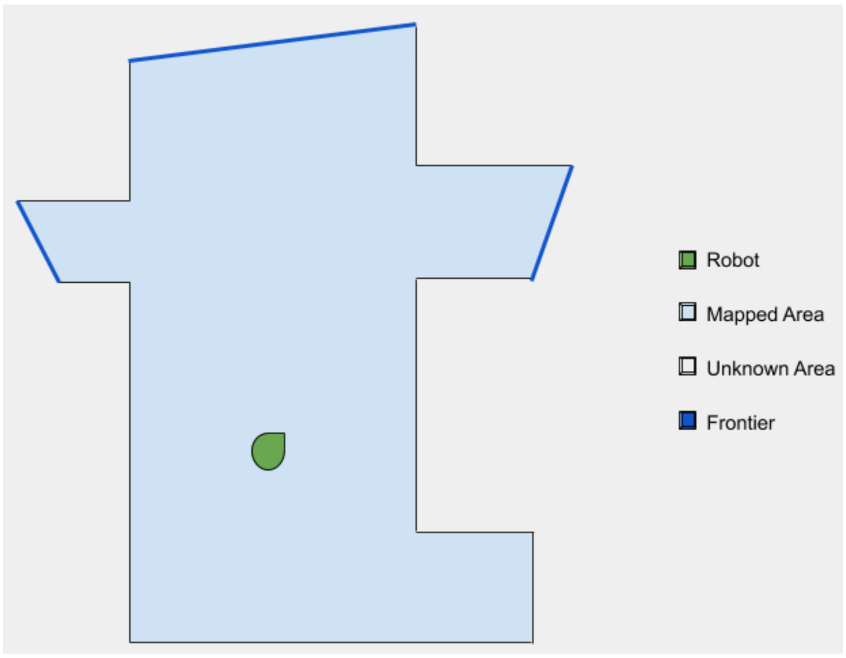


Fig. 2.13. – Example of a 2D map with frontiers.

Frontiers can be searched by applying computer vision techniques such as edge detection or region extraction [108]. However, the vast majority of the approaches find these points with algorithms based on Breadth-First Search (BFS) [274, 208, 233] and cluster these frontiers to be more efficient, as adjacent frontier points can lead to a similar exploration result proportional to the granularity of the map. Generally, the clustering is performed using k-means [214, 64, 77, 224], although some authors propose other alternatives such as histogram-based methods [154].

Optimal Goal Selection

Once the set of potential destinations is identified, the cost and gain of each of them need to be estimated. The cost represents the effort required to reach the actual goal, e.g., the distance between the actual position and the target pose. The gain corresponds to the difference between the information provided by the map after and before navigating to the selected goal. Information usually refers to the number of points discovered. Generally, gain and cost are variables of a function that results in a value called utility [102]. Utility serves as a metric to compare exploration trajectories [35, 192]. Ideally, to compute the utility of a given action, the robot should reason about the evolution of the posterior over the robot pose and the map, taking into account future (controllable) actions and future (unknown) measurements. However, computing this joint probability analytically is, in general, computationally intractable [34, 65, 217], and thus, it is approximated [34, 157].

In general, the information gain $I_b(u)$ of a belief b associated with an action u is represented as the difference between the entropy $H_p(x)$ of a probability distribution p and expected entropy of the state x' after action u and measurement z, as shown in Equation 2.22:

$$I_b(u) = H_p(x) - E_z[H_b[x'|z, u)] \tag{2.22}$$

Navigation and Checking

The robot navigates to the chosen optimal destination and updates the map during motion or upon attaining the goal. Updating a map means completing unknown data as well as improving the quality of the already known elements or correcting old or erroneous information. Originally, the navigation is performed with a classic technique, such us Dijkstra or A* [235], which is completely independent of robot exploration. However, some researchers propose adapting these trajectories for the autonomous mapping purpose, as it is described later on in Chapter 5. Then, the robot checks if the exploration procedure must continue. In that case, the process proceeds again with the pose identification step and another iteration is executed.

When exploring an unknown environment, there may always be potential areas of mapping or accuracy improvement, driving the system to an infinite loop. In order to perform the autonomous mapping in a finite period of time, a termination criteria must be defined. Thus, the procedure will be considered finished when that criteria is satisfied [216, 20]. Many exploration systems based on frontiers consider the exploration process as concluded when no frontier targets exist in the map [34, 44, 188]. Other works opt for more straight-forward solutions as the distance travelled, the time elapsed or the number of frames captured. Yet these conditions are independent of the quality or completeness of the model built, and their effectiveness is highly dependent on the optimal candidate selector. Kriegel et al. [118] propose a 3D reconstruction system in which the termination criteria is a predefined maximum number of scans mapped. This value is set based on an estimation of the quality and the completeness rate of the scene, and it is unique for each area due to the particular properties of different objects. Hence, there are stopping conditions that are environment-specific.

Uncertainty metrics from Theory of Optimal Experimental Design (TOED) [68] try to quantify utility from the variance of the variables of interest, that is, they try to minimise the covariance of the joint posterior based on the actions to be executed. Compared to information-theoretic metrics that are difficult to compare across systems, functions built upon TOED seem promising as stopping criteria. However, this decision is currently an open challenge [183].

2.8.2 Active Perception

In computer vision, the problem of exploration emerged as active perception [15] and it is defined as an intelligent and reactive process for gathering information from the environment to reduce the uncertainty around an element. It is considered intelligent because the sensor's state changes on purpose according to the sensing strategies. Active perception applied to mobile robotics describes the capability of the robot to determine particular movements in order to get a better understanding of the environment. In the context of localisation, the position and/or heading of the sensor is moved in an attempt to search for signs that reduce the position uncertainty. In consequence, the shape, geometries or appearance of the surrounding elements cannot be unknown and the robot must be able to recognise them with accuracy. This is called active localisation and, although the environment is assumed to be already mapped, many advances in this field can be extrapolated to the ASLAM problem [29, 6, 100]. A work that is especially worth highlighting is the one presented by Zhang et al. [268]. On the one hand, they present a perception-aware receding horizon planner for micro aerial vehicles that allows the robot to reach a given destination and avoid visually degraded areas at the same time [269]. On the other hand, they define a dedicated map representation for perception-aware planning that is at least one order of magnitude faster than the standard practice of using point clouds [270, 271]. Both contributions could push the research of mapping unknown areas in the right direction.

In contrast, active perception in the context of mapping refers to the challenge of modelling the environment with no prior information and calculating the motion required to achieve it. Thus, this idea includes two processes, the reconstruction of the environment itself and the motion control of the sensor. This method is referred to as active mapping, i.e., a map of the unknown environment must be built in a finite time, optimising the accuracy of the resulting model and the actions executed for that purpose. There are several methods for estimating the spatial transformation that corresponds to a specific motion. Approaches based on probabilistic filters [228, 171] give good results and are thus, used most commonly together with those that employ Structure from Motion (SfM) [240, 200]. SfM is a photogrammetric range imaging technique for estimating 3D structures from 2D image sequences, similar to estimating the structure from stereo vision.

Some of the problems ASLAM is facing have already been tackled by other research fields. For instance, active vision deals with the task of choosing the next optimal sensor pose for 3D object reconstruction or obtaining a complete model of a scene. This is known as Next Best View (NBV) and it has been studied since the 1980's [42]. Alternatively, computational geometry confronts the art gallery problem. This problem was first set out by Victor Klee in 1973 [175] as a matter of interest in the field of security. The art gallery problem for an area *A* is to find a minimum-cardinality covering viewpoint set *P* for *A*. It was called this way because one envisions the area *A* as the floor plan of an art gallery, and the points in *P* as locations to place guards, so that every part of the art gallery is seen by at least one guard. In summary, provided that the system already has a digital model of the environment, the objective is to find the minimum set of poses from which all the scene is seen. Transferred to computational geometry, the gallery corresponds to a simple polygon and each guard is represented by a point in the polygon. The goal is then to find a minimal cardinality set of points (guards) that can be connected by line segments without leaving the polygon. Many contributions have been done finding the optimal placement of surveillance devices [24, 173, 97, 79, 161].

Part II

Contributions

3 Experimental Evaluation of Current Mapping Solutions

Everything we hear is an opinion, not a fact.
Everything we see is a perspective, not the truth.

— **Marcus Aurelius**

In order to make contributions, a detailed understanding of the performance of current solutions in a specific workspace is necessary, as they can mark limitations, challenges and opportunities. This chapter presents the experimental evaluation of some methods for robust autonomous navigation of mobile robots in indoor environments.

3.1 Introduction

Autonomous navigation aims to design robotic systems capable of navigating without having to modify the environment, i.e. without adding additional elements to it. Currently, the performance of techniques in this line changes considerably depending on the environment, as their efficiency depends directly on the characteristics of the environment, such as morphology, lighting or size. During the operation of a mobile robot, sensor readings are matched to the map and the pose is estimated based on the similarity between them. Consequently, the greater the dissimilarity between different regions of the environment, the easier it is for the system to localise itself, increasing its accuracy, robustness and overall performance.

Shop floors and industrial environments are full of repetitive areas or long corridors where the system localisation critically depends on unreliable odometry data, due to high aliasing. As a result, the accuracy of localisation estimation methods is degraded. Moreover, these environments undergo frequent modifications that would require a remapping of the environment that is not always affordable.

Most of the developments to be carried out throughout this research work are put through the same industrial environment, so it is necessary to be aware of the limitations of current techniques in order to develop improvements or new methods. For this reason, the aim of this first phase is to empirically evaluate the main SLAM techniques in order to lay the foundations for further research. Using an external ultra-precision system, the aim is to quantify the error made in localisation in the SLAM process by these algorithms. In addition, the maps generated by each of the different methods are also compared, as the subsequent autonomous navigation is directly affected by the model of the environment on which it is based.

3.2 Sensors for SLAM

Probabilistic approaches have become popular because they give a robust solution to the SLAM problem. As the most relevant of them are already explained in Section 2.5, the following is a brief review of the most commonly used sensors for this task. The presented techniques are mainly based on two information sources: motion information, usually obtained from wheel encoders and IMUs, and perception information, gathered, in most cases, from cameras or lidars. Devices belonging to the first group are known as *proprioceptive* sensors, and they provide local information, in the sense that they are independent of the environment. An odometer, for example, calculates the total or partial distance travelled, but does not directly give any information about the surroundings. The measurements depend only on the previous state and the movement since then, which generates a cumulative error that is impossible to detect without additional information. For this reason, it is necessary to use sensors that are capable of sensing the environment, i.e., *exteroceptive* sensors. These kind of sensors allow the system to correct this accumulative error and make a more accurate and robust estimate.

Among the different sensors used for correcting the estimated odometry positions, lidar sensors are preferred nowadays, as they are quite reliable in most scenarios [189, 251, 187, 129]. Briefly, lidars send out laser pulses and measure the returned wavelength to obtain the distance to the first object in its path. They offer a high level of robustness in terms of light effects such as shadows or reflections, and visible light is not necessary for their operation. In some applications, it is even preferred to acquire readings in the dark. However, many of the methods proposed in the last decade are based on 2D images obtained from digital cameras. These are cheap, lightweight, and the 2D image stream they provide are a valuable source of information for visual feature detection [165, 63, 169]. Besides, in the last years there has been a proliferation of devices that integrate both an RGB camera and a depth module, which are known as RGB-D cameras. They use both sensors to collect data simultaneously and in a post-processing step the colour attribute is added to each point of the point cloud, obtaining a realistic coloured point cloud, which can better represent many aspects of a scene.

One robust approach to correct the localisation of a robot in an indoor environment using visual sensors is by means of artificial markers [10]. By this method, visual landmarks are placed in the environment through which the robot will navigate and those are calibrated so that their positions with respect to the map reference frame are known. When a landmark is identified, the robot's relative position with respect to it is computed and thus, the robot localisation in the map can be corrected [62, 222]. Although this method improves the quality of the pose estimation significantly, it has some important drawbacks. On the one hand, the environment has to be modified, which makes its installation costly and not always possible. On the other hand, each of the labels to be placed has to be accurately calibrated. If the calibration is not accurate, then the error is propagated to the robot pose estimate, which can lead to unexpected situations. Furthermore, these measurements will only be valid for a specific map. If the map changes, the reference frame also changes, and the coordinates of the reference points have to be recalculated to fit the new map.

Several strategies designed to work outdoors use GNSS to address the localisation aspect [1, 248]. However, those systems do not succeed in common factories due to the bad quality of the signals produced by two main reasons [202]:

- The metal structures of the thick walls of industrial buildings. These structures prevent signals from propagating properly, occasionally creating completely isolated areas that are unreachable.

- Factories often have a considerable number of machines, each equipped with its own communication devices. These devices can generate interferences or even loss of information that can affect wireless data transmission. [139].

These communication issues can be addressed in different ways. Some approaches propose to use signal repeaters or amplifiers to improve the quality of the signals [258] or replace the global positioning system with a local positioning system [210]. Either way, the GNSS signal is not guaranteed to be available at all times and it also requires a rigorous calibration between the GNSS data and the environment. In addition, these sensors cannot sense the environment, which makes it necessary to have another exteroceptive sensor. Optimising the use of sensors such as cameras and lidars so that they can solve the entire SLAM problem without the need for additional elements leads to more adaptable systems, less resources management, and reduced robot manufacturing and installation costs.

3.3 Proposed Approach

The algorithm used to build the map can make a big difference, so it is important to choose the right one. However, regardless of the map building method used, the procedure to follow is very similar. The robot must move around the environment allowing the sensors to gather information from the area. The quality of the map depends directly on the smoothness of the platform's movement. Therefore, it is highly recommended to limit speeds, especially rotational speeds, as rotations increase the cumulative error the most, and abrupt turns can result in a much harder scan matching problem. In addition, it is also a good practice to revisit mapped areas to facilitate the global loop closure that many methods perform during mapping. Loop closure consists of asserting that the vehicle has returned to a previously visited location and correcting the correspondence of the scan matches in the map being constructed to overlap the initial and final positions. Loop closure can be local, where one tries to correct the relationship between neighbouring elements, and global, where the correction is applied to the whole map.

Generally, the mapping process requires a post-modelling process in which undesirable elements, such as dynamic obstacles detected during map construction, are removed and loop closure is corrected. The decision to consider an obstacle as static or dynamic, and therefore to decide whether to keep it on the map or not, is still an open discussion. In theory, only static and immobile volumes should be part of the map, but in most cases this is unfeasible because, strictly speaking, only walls and pillars fulfil this condition. However, much relevant information would be lost and the map would have little similarity to the sensor readings the robot gathers from the environment as it navigates.

In this work, three map building strategies are studied: GMapping, Hector Mapping and Cartographer. All of them are available as open source implementations in ROS [@124], which makes our method easily replicable. GMapping [81] is a Rao-Blackwellized particle filter to learn grid maps from laser range data, using a particle filter in which each particle carries an individual map of the environment. It computes an accurate proposal distribution taking into account not only the movement of the robot but also the most recent observation. To perform the computation efficiently, a marginalisation of the probability distribution of the state space based on the Rao-Blackwell theorem is done [190]. On the contrary, Hector Mapping [115] is a scan matching approach that attempts to find the best alignment of each laser scan with the map. Based on a Gauss-Newton approach, inspired by the field of computer vision, it optimises the alignment of beam endpoints with the map learnt so far. Therefore, there is no need to perform a data association search between beam endpoints or an exhaustive pose search, as the correspondence is implicitly made with all the preceding scans. However, the Hector Mapping method does not perform loop closure. Finally, Cartographer [91] uses a branch-and-bound approach for computing scan-to-submap matches as constraints and achieve real-time loop closure detection and graph optimisation. To cope with the accumulative error, it regularly runs a pose optimisation, creating constant submaps that take part in scan matching for loop closure. Therefore, loops are closed immediately using branch-and-bound when a location is revisited.

As mentioned before, industrial environments are full of symmetries and the shape of the laser scan is, to a large extent, equal in spite of height variations when keeping the same coordinate values. For this reason, during the mapping process only 2D lidar information is used instead of the 3D point cloud. Although the 3D data is flattened to the ground plane, the conversion could also have been done by extracting the data in a range of heights and discarding the rest. This downsizing is done to be able to extrapolate the method to more robots because not many of them can incorporate a 3D lidar. Moreover, handling a complete point cloud involves a greater degree of computational complexity and the extra information does not increase accuracy proportionally.

Once a realistic map is created, the location of the origin is measured and saved with an external measurement unit, which serves as ground truth when comparing the localisation precision using that particular map. The Leica AT901 [1]laser tracker shown in Figure 3.1 is used for that purpose, as it is a device that measures distances with a micrometric accuracy.

The robot travels around the environment making runs with different styles of navigation, varying the lineal and rotational speeds and accelerations, the order and the visited areas of the map, the amount of turns done or the length and duration of the runs. The more realistic the performed trajectories the better the guess of the behaviour of the algorithm, and the more meaningful the experiment is. The pose estimations of the algorithms are tracked while the real position of the robot is saved with the laser tracker. Afterwards, both data are associated based on their timestamp. Statistical values are obtained from that analysis, which allows to establish a quantitative performance of each algorithm.

[1]`https://manchester-metrology.co.uk/leica-absolute-tracker-at901/`

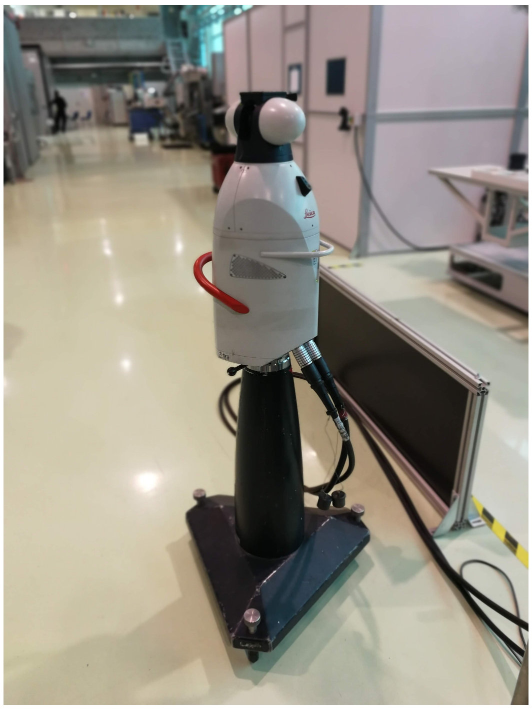


Fig. 3.1. – The laser tracker used to measure distances with a micrometric accuracy. It serves as ground truth to measure localisation errors.

3.4 Experiments

The experimentation is proposed to be performed in an industrial rectangular workshop of 90 m x 30 m, with a wall in the longitudinal axis. This wall divides the area in two halves that are communicated by two large sliding doors, which were completely opened during the experiment. This structure forms a sort of circuit with a width of 4 m - 8 m, where there are machines, benches and industrial robots next to the walls alongside the route. A part of the scenario, with the platform navigating though it, can be seen in Figure 3.2.

The experimentation is divided in two main sections: mapping and localisation. Besides, they are performed in that order, i.e., mapping before localisation evaluation, as the former has a direct impact on the latter.

3.4.1 Robotic Platform

The robotic platform used for the experiments is a Segway RMP 440 Flex Omni, able to drive in any direction due to its omnidirectional wheels, at a speed of 2.2 m/s (5 mph) with a maximum payload of 450 kg (1,000 lbs). However, its velocities

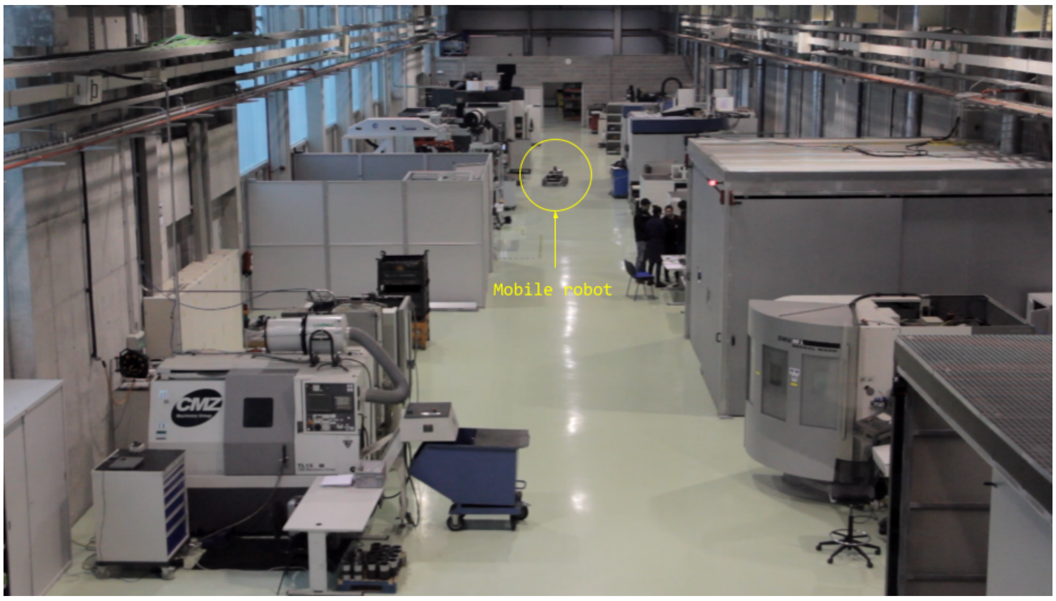


Fig. 3.2. – West half of the workshop. At the top, the mobile platform navigating.

are limited to 0.5 m/s (1.1 mph) and 1 rad/s during the experiments for improved control and safety.

It includes the following main elements:

– **PC with Intel i7 Processor**: Core i7-3517UE (1.7-2.8 GHz + HD 4000), 16GB (2x8GB) DDR3-1333/1600 SDRAM, 128GB high performance SSD.

– **CH-Robotics UM7 IMU**: Attitude and Heading Reference System (AHRS) for precise pose estimation.

– **Velodyne VLP-16 lidar PUCK**: Real-time, 360º, 3D distance and calibrated reflectivity measurements with a 150 m range, 360º horizontal field of view and a 30º vertical field of view, with a 15º up and down.

– **RGB camera**: Optical format of 1/1.8" and a focal length of 12 mm. It is not used in these experiments.

3.4.2 Mapping

The robot needs a map for navigation and the map must contain information relevant to how the data will be processed by the system. This means that the map must represent the environment according to how the robot's sensors capture it, which does not necessarily correspond to a human concept of realism. During this experimentation, first of all, a location is established to be the starting point of the mapping process and the origin of coordinates of the map. This point normally corresponds to the place where the mapping process starts. Then, with the help of a joystick, the robot is guided through the environment, moving the robot smoothly and covering the whole area of the workshop. In order to evaluate the different algorithms under the same conditions, the three maps are built simultaneously, during the same teleoperated trajectory and with the same sensor readings.

It is impossible to determine at a glance which of the maps obtained with the three algorithms is better, at least not without having a real map that serves as ground truth. This is quantified to some extent in Section 3.4.3, but a first comparison is made in Figure 3.3, where the overlap of all the maps in the same image is shown.

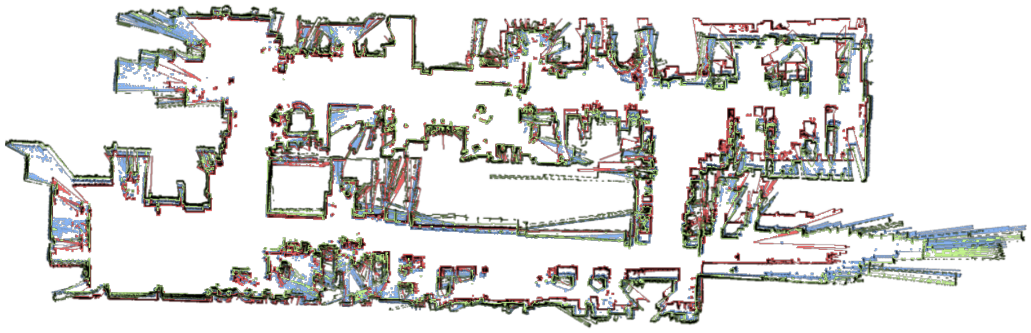


Fig. 3.3. – Overlapping of the maps obtained with each algorithm: *gmapping*, blue; *hector_slam*, red; *cartographer*, green.

Apparently, *hector_slam* has a significant deviation in the orientation with respect to *gmapping* and *cartographer* in the north part of the map (right-hand side of Figure 3.3), which is the region that corresponds to the last phase of the trajectory of the mapping process. It may be because *hector_slam* does not have loop closing ability and therefore it does not apply the corresponding transformation in the data to rectify the graph. Besides, *gmapping* and *cartographer* seem to cover a wider range, i.e., there are elements that they add to the map while *hector_slam* does not, as can be clearly seen in the bottom right of Figure 3.3. It should be noted that all three algorithms have been configured so that parameters such as valid sensor range, update rate, or map resolution are the same. Moreover, the process has been carried out several consecutive times and the differences between the maps and the trend of each technique have been analogous. Although this first comparison is not meaningful in determining which algorithm is better, it can be useful as a first approximation of the effectiveness of each algorithm.

3.4.3 Localisation

Strictly speaking, we are evaluating the combination of a localisation method with each of the mapping algorithms. However, the only aspect that changes from one combination to another is the map itself, which is generated with the corresponding mapping technique, so it can be deduced that the error measured is a result of the latter. Indeed, localisation estimations may vary depending on the map being used, as it represents how the robot perceives the environment. In these terms, if the map built is mediocre, the whole system will perform poorly. As neither *gmapping* nor *hector_slam* have a localisation-only mode, the probabilistic Adaptive MCL (AMCL) [69] algorithm has been used for that purpose driving the localisation tests.

In order to measure the magnitude of the localisation error produced by the use of the different maps, teleoperated trajectories are recorded to afterwards reproduce the data in each map, while AMCL is running for localisation purposes. Thus, no planning algorithm is used to fulfil the routes; predefined action commands are used instead.

First of all, we must fix a point with the laser tracker from which all trajectories will start. This will also be the origin of coordinates of all the maps that are generated in this manner. Indeed, this point must be common to all reference systems to be able to directly compare the values of one system with another. Once this is done, the mapping process can begin and be performed as described in Section 3.3.

12 tests were done, in which 35 different poses were recorded. Among these, 7 were long routes executing each of them a complete loop of the workshop, and the other 5 were shorter trajectories that do not visit the whole environment, but all of them finish in the origin. Besides, in 6 out of the 12 navigations the driving is smooth, whereas in the other 6 the robot rotates 360° every 10 m. These sudden rotations were done with the aim of adding systematic uncertainty to the system and see how each configuration responds to them. As expected, jerky navigation does add uncertainty in localisation as the errors in translations were bigger in those cases, as shown in Table 3.1. Translational mean errors in smooth navigation were 0.19 m, 0.34 m and 0.17 m for each system, smaller compared with the 0.24 m, 0.50 m and 0.25 m of the abrupt navigation. Nevertheless, while the same increment of the error occurs with *hector_slam*, 1.61° against 2.20°, this is not the case with the rotation values of *gmapping* and *cartographer*, as shown in Table 3.2.

In the smooth navigation, the rotational mean errors obtained with *gmapping* and *cartographer* are, respectively, 1.08° and 0.94°, whereas values of 0.24° and 0.80°

	Translational error					
	Smooth navigation			**Abrupt navigation**		
	gmapping	hector_slam	cartographer	gmapping	hector_slam	cartographer
Mean	0.191	0.340	**0.170**	**0.242**	0.498	0.254
Min.	0.063	0.041	**0.001**	0.019	0.019	**0.005**
Max.	**0.506**	1.099	0.543	**0.544**	1.509	0.997
Std. dev.	**0.130**	0.317	0.185	**0.176**	0.482	0.317

Tab. 3.1. – Translational error in metres made with the three mapping algorithms, calculated on the basis of the laser tracker measurements.

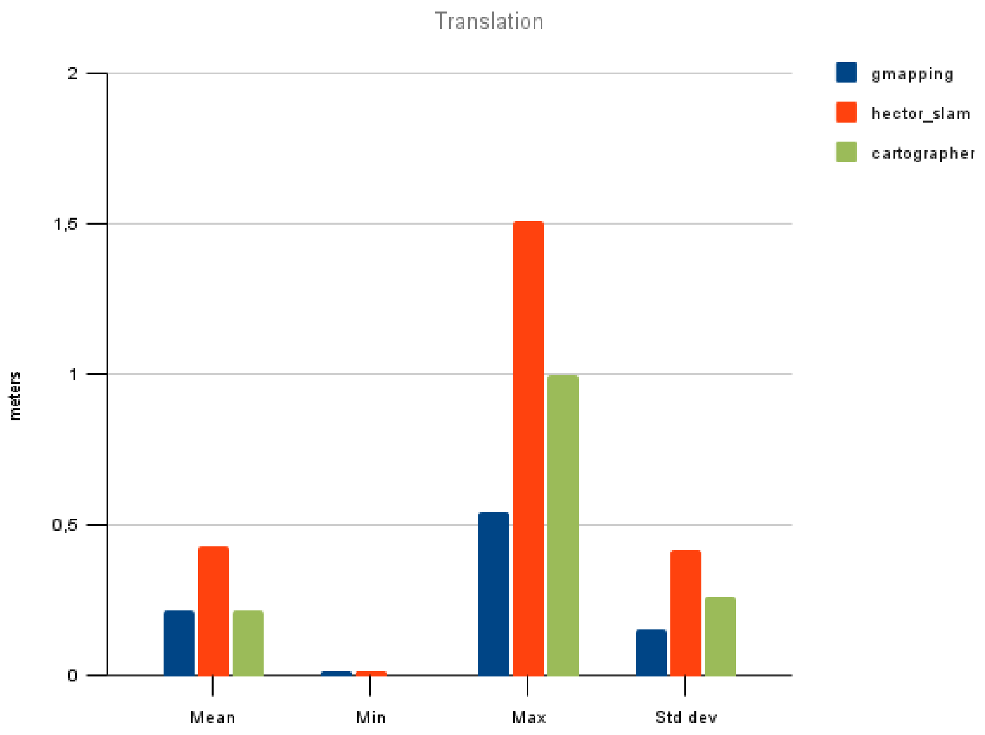
	Rotational error					
	Smooth navigation			**Abrupt navigation**		
	gmapping	hector_slam	cartographer	gmapping	hector_slam	cartographer
Mean	1.088	1.606	**0.939**	2.048	2.196	**0.800**
Min.	0.074	0.122	**0.037**	**0.021**	0.024	0.028
Max.	3.305	5.467	**2.884**	10.185	10.423	**4.393**
Std. dev.	1.077	1.847	**0.963**	2.782	2.700	**1.210**

Tab. 3.2. – Rotational error in degrees made with the three mapping algorithms, calculated on the basis of the the laser tracker measurements.

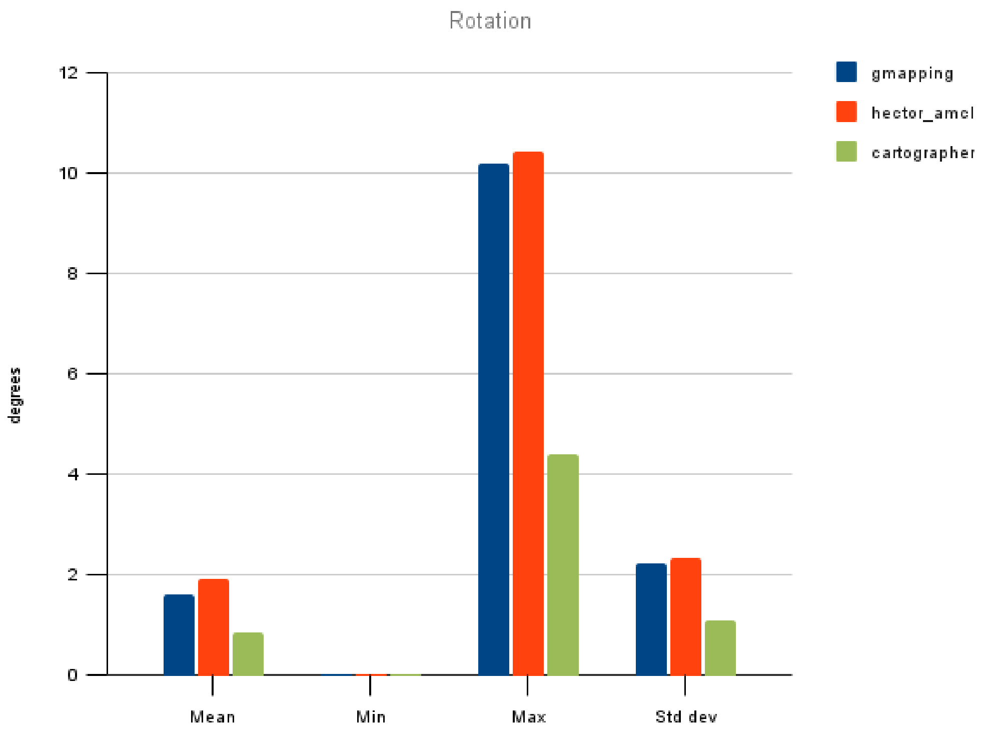
are obtained navigating with sudden rotations. The overall values, including translational and rotational errors in both smooth and abrupt navigation, are depicted in Figure 3.4.

3.5 Conclusions

The main conclusion this experiment leads to is that relative small differences in the generated maps are directly related with the localisation procedure accuracy in each one. Therefore, using an appropriate environment representation for the task in hand is of high importance to obtain good results during pose estimation. To determine if a map is good enough, it is mandatory to experimentally test it in real situations. Based on the navigation done with the three maps, *gmapping* and *cartographer* obtain the best results. But, based on our experience and the applications we are working on, we consider that the system using *gmapping* is the preferred one for general use. It obtains significantly lower maximum and standard deviation values in translation (see Figure 3.4(a)) and only slightly higher ones in rotation (see Figure 3.4(b)). This may be, to some extent, an indication of robustness and stability. However, if one could ensure that navigation will be always smooth, *cartographer* would be a better option according to the results.



(a)



(b)

Fig. 3.4. – Overall error made with the three mapping algorithms, including both smooth and abrupt navigation. While Figure 3.4(a) depicts translational errors in meters, Figure 3.4(b) rotational errors in degrees.

As expected, rough movements increase the error in localisation, so the robot should have both angular and linear velocities and accelerations limited to cautious values during the general usage. Even though navigating smoothly, the accuracy of the positioning in the destination point is not enough for some operations and an additional system may be required in some applications.

4 Accurate Positioning

If you tell the truth, you don't have to remember anything.

— **Mark Twain**

The positioning accuracy achieved in mobile robotics using state-of-the-art navigation systems may not be sufficient depending on the task or may be deteriorated by the scenario in which the robot is located. In this chapter, two different systems are proposed to reduce the error made in these situations.

4.1 Introduction

Localisation is a core competency for any autonomous robot and many developments assume the localisation as known. The robot location is given with respect to a specific reference frame. However, depending on the task, the localisation accuracy might not be enough. Here, we refer to *accurate positioning* as estimating the location of the platform with a smaller tolerance than a general purpose localisation algorithm in the same scenario.

Some applications do not require the robot to maintain an accurate localisation along the entire path, but do require it at certain locations [147]. A typical example is a mobile manipulator that performs inspection or handling tasks of some elements in a plant. This type of systems can be more tolerant to deviations during navigation between destinations, as long as the task is completed safely and within a reasonable time interval. On the contrary, it requires a precise knowledge of the positioning of the platform at the places where mobile manipulation is to be performed, due to the nature of the task. Indeed, a lack of coordination between the robotic arm and the mobile platform or low precision in either positions can lead to misleading results.

Here, we study the positioning accuracy of a platform in two different environments, proposing and evaluating a localisation improvement method for each case. The two cases are as follows:

1. In the first one, the robot must navigate autonomously inside a factory and then position itself precisely in front of an aircraft aileron to perform an inspection operation. The proposed solution is based on artificial landmarks placed in the inspection area that serve as a reference to improve localisation accuracy.

2. In the second case, the platform operates in a semi-indoor environment, such as a greenhouse, where it must be able to travel from plant to plant to perform inspection and spraying tasks. In this case, the proposed improvement consists of fusing the information obtained from the system's local sensors with the location provided by a GNSS module.

In the following sections, both scenarios are explained in detail independently.

4.2 Artificial Landmark-based Localisation for Aileron Inspection

Highly integrated composite parts have begun to enter the market. Their inherent complex structure makes more difficult inspection activities being crucial the development of more flexible, more reliable and faster non-destructive testing solutions. Currently, most inspections are performed manually, which entail two key disadvantages: an excessive amount of time and the risk of potential safety failures due to human errors. To solve these issues, the automation of inspection processes has been envisioned as a strategic solution. The integration of robots in manufacturing lines has improved process reliability, reduced manufacturing times and, therefore, costs.

This work aims to improve robotic capabilities, enabling faster and more reliable operations. Specifically, the robot needs to navigate through an industrial plant to reach an area where there is an aircraft aileron. Once there, it needs to position itself precisely with respect to the aileron in order to perform an inspection process using advanced ultrasonic techniques, as shown in Figure 4.1. Afterwards, the platform must be able to precisely position itself at some pre-defined points, so that the robotic arm mounted on it performs fast and robust ultrasonic scanning of the aileron to accurately follow the complex curvatures of the composite parts. As the working space of the arm is limited, the arm cannot scan the entire surface of the same aileron at once, so the mobile platform must reach different positions during a single inspection. Consequently, there must be robust coordination between the base and the arm, since the ultrasonic sensor is particularly sensitive to slight variations.

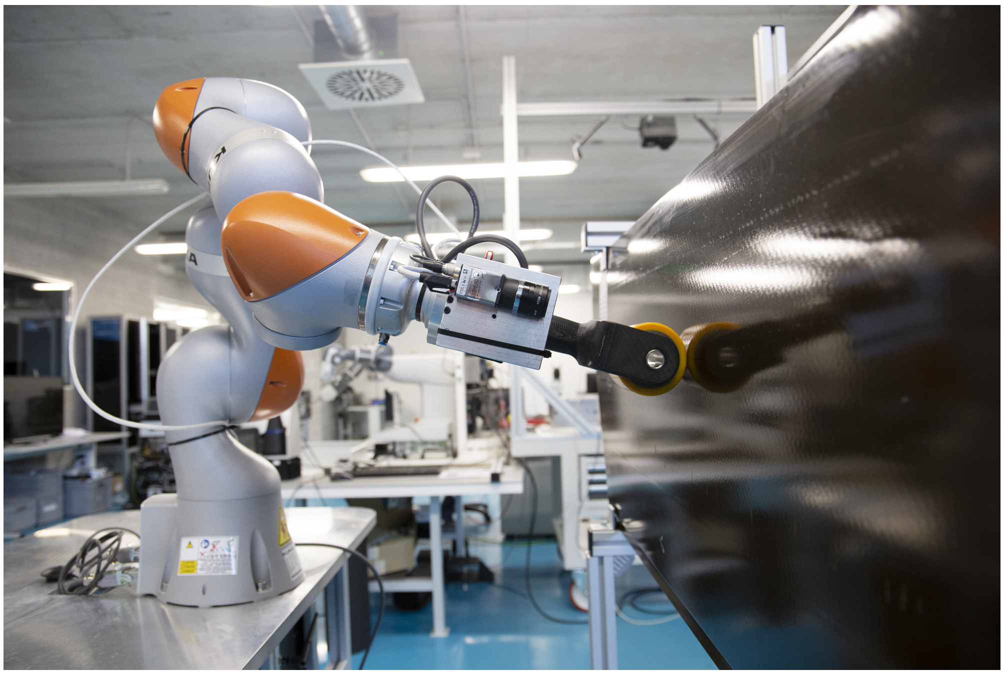


Fig. 4.1. – The aileron inspection process. On the right-hand side, the aileron to be scanned. The robotic arm with the scan is in front of it. In the lower left corner of the image, the top part of the mobile platform is seen.

Based on the results obtained from our previous experiments described in Chapter 3.4.3, the localisation accuracy obtained with the state of the art methods for navigation is not enough for the task in hand. Therefore, another positioning approach is required, which must be able to correct the deviation.

4.2.1 Visual Landmarks

Landmarks are defined as characteristic elements, regions or points in the environment that can provide reliable localisation when they are within the robot's field of view. Their uniqueness allows them to be identified unequivocally from different perspectives, allowing an association between their corresponding scans and robot positions. While some methods actively search for natural landmarks in the environment, others opt to place artificial markers or labels in it so that the robot use them as known references to, for example, correct odometry error. One of the main advantages of the artificial landmark-based approaches is that they can be placed in critical areas and reduce system failures. It has been shown that, when certain assumptions and conditions are satisfied, these procedures guarantee the robot the possibility to localise itself precisely [9, 259].

Setting artificial landmarks in the environment is simply a way of ensuring that this provides relevant and easily distinguishable features. Indeed, one of the main problems of localisation is dealing with environmental aliasing in symmetrical or highly ambiguous scenarios.

4.2.2 Setting Up the Scenario

We present an approach where,after navigating, the platform arrives at the destination point and switches its localisation system to one with higher accuracy. Specifically, we propose a system based on artificial vision, which uses fixed tags located in the target area as a reference point to correct the position.

The whole operation requires to ensure the precision of the camera, to create the map for autonomous navigation, and to calculate the optimal position of the robot with respect to each artificial marker. In addition, the location of these markers on the created map must be known, which is what determines the destinations prior to the precise positioning process, e.g., for autonomous navigation. These procedures are explained below.

Ground Truth Generation

Before anything else, we must verify that the additional localisation system for accurate positioning can actually provide a smaller tolerance than required. In our case, the proposed vision system is validated using the TRITOP photogrammetric unit [19]. TRITOP is an optical mobile measurement system, which accurately defines the 3D coordinates of object points at quasi-static conditions. Based on this information, TRITOP is capable of calculating 3D displacements and deformations of objects and components. It consists of one digital single lens camera (DSLR), contrast coded and uncoded bars, scale bars and the corresponding software - in this case, TRITOP v6.2 has been used. As in the work proposed by Koutecký et. al. [117], TRITOP is used for the validation of the vision system incorporated in the robotic platform. This measuring procedure achieves a theoretical accuracy of 0.015 mm in an area of 1 m x 0.5 m x 0.5 m, which fits in the settings defined for the validation.

First, the coded plate used in the robot vision system is fixed in the same configuration as it will later be used in the real application. Afterwards, several foils with coded points of the TRITOP system are placed around it, so as everything can be seen together. Then, some images are taken with the camera, varying the point of view, but trying to keep as much markers as possible in the FoV. The relation between the plate and the coded foils must be the same in all the images, so they must be placed at stationary locations. A total of 14 images are taken in this particular evaluation, similar to the one shown in Figure 4.2. Then, both the TRITOP system and the robot vision system find the labels in the images and each of them calculates the pose of the camera with respect to a common reference frame. In addition, the TRITOP system also computes the intrinsic parameters of the camera. The results estimate an average image point deviation of 0.02 pixels and an average object point deviation of 0.01 mm. Accordingly, the vision positioning system assumes this error at all times, which can be considered as negligible.

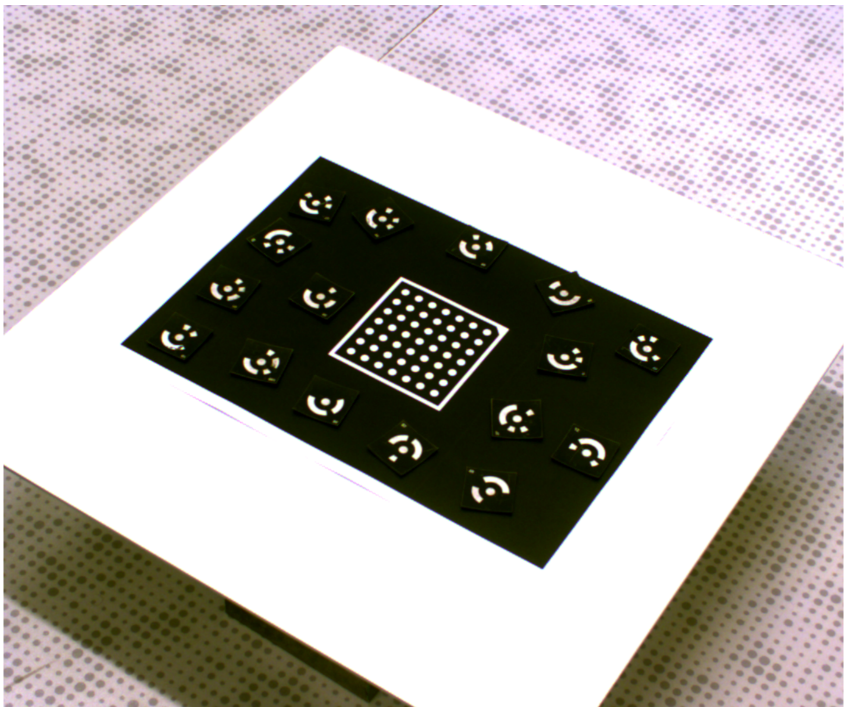


Fig. 4.2. – Example of an image taken for the TRITOP calibration.

Once it is verified that the additional localisation system has the potential to correct the error made in autonomous navigation, it has to be evaluated in the real scenario. Factors such as platform motion control, system cycle time, motion inertia forces, lighting changes, or even the printing of the labels itself, can alter the accurate positioning result. The mobile robot used in these experiments is a modified version of the one described in Section 3.4.1, as a KUKA LBR iiwa 7 arm has been mounted on top of it, with all the necessary components to ensure its correct operation. In addition, the motion control of the platform has been re-calibrated, due to the increase in weight and volume of the whole structure.

Mapping

As the robot has to navigate autonomously before reaching the inspection area, it needs a map of the environment. The SLAM method used for this task is GMapping, based on the results obtained from our previous experimentation. Briefly, we can get lowest translational and rotational deviations during the autonomous navigation. The mapping process is performed in the traditional way and as explained in Section 3.4.2, the robot is teleoperated, guiding it smoothly through all the environment and using data obtained from the lidar. The 2D map obtained is post-processed before using it for autonomous navigation.

Optimal Position Calculation

In the inspection area, the references are used to determine where the platform must be positioned so that the arm on it can perform the execution. This position is calculated empirically, i.e., the platform is positioned in different configurations until a complete correct inspection of the aileron can be performed. In invalid configurations, the arm cannot achieve all points of the spoiler because they are out of reach, and the optimal inspection is the one with the least number of stops. Once the whole trajectory can be fulfilled by the arm, three actions must be taken:

(a) Save the localisation on the map. Having the localisation method active and correctly located, the pose of the platform in the global frame is saved. This will be the destination for the autonomous navigation.

(b) Place the reference. The label is fixed to a structure where its spatial relationship relative to the aileron does not change, and where the robot's camera used for precise positioning can see it.

(c) Calculate and save the transformation between the camera and the label. Using the accurate vision system, the pose of the label with respect to the camera is obtained and saved, as it is this relation which connects the chain of transformations between the part to be inspected and the inspection tool.

For the specific aileron inspection task, the mobile platform cannot remain stationary as the arm mounted on it would not be able to reach the entire piece. Thus, there is more than one label in the working area, as shown in Figure 4.3. In this way, the platform can move to different scanning positions without losing the accuracy offered by the vision system. In these cases, the spatial relation between different labels must be calculated as well.

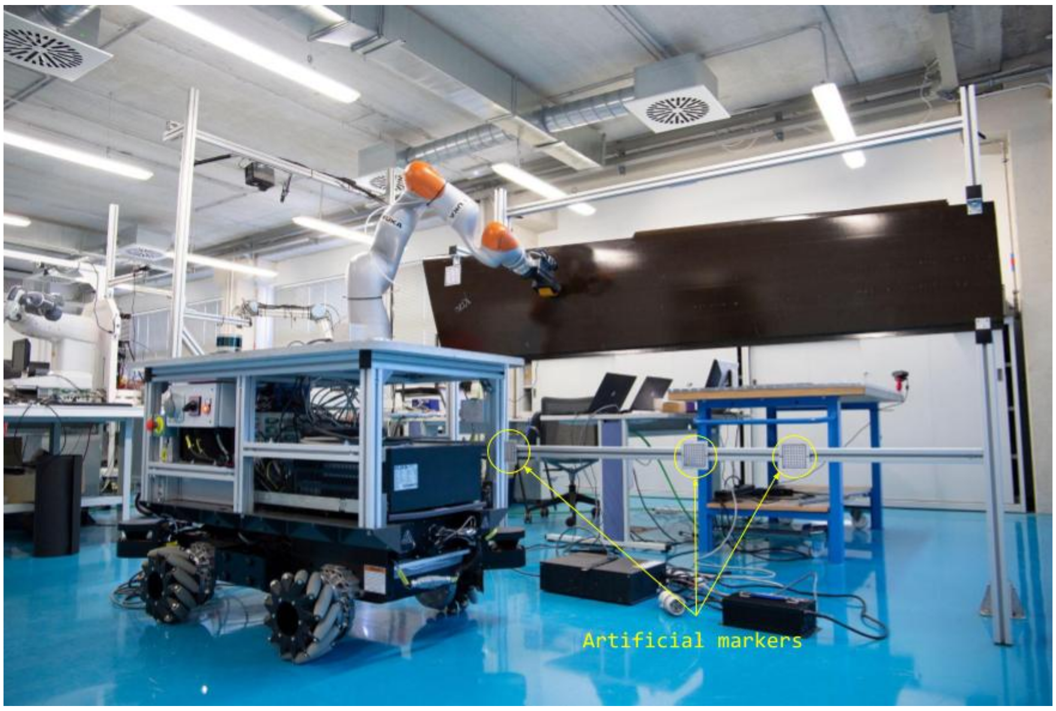


Fig. 4.3. – The accurate positioning and inspection scenario. The platform is positioned in front of the first label, and the arm is performing the inspection of the first part. Under the aileron, the label for the next scanning position can be seen.

Noticeably, a requisite is that the label must be located within the field of view of the sensor that will afterwards be used for correcting the final positioning error. Ergo there is a maximum error tolerance in natural navigation determined by the features of the camera and lens chosen. Aspects like the focus, the depth of field or the angle of view should be calculated beforehand as those features mark the boundaries of the region inside which the vision system must fall after the navigation.

4.2.3 Proposed Approach

The robot navigates autonomously through the shop floor until the optimal position from the inspection area is set as the goal. Figure 4.4 shows a snapshot of how the environment is represented to perform the autonomous navigation. Once at the destination, the system must find the artificial marker to be able to proceed with the correction. Based on the chosen vision configuration, the autonomous navigation must reach the destination within a certain area, outside of which the robot is not able to detect the marker. If there is no error in the robot orientation, the variability is an area of approximately 20 cm in radius. On the contrary, if there is no translational deviation, a maximum error of 16° (0.28 rad) is allowed. These values vary if both translation and orientation are not precise in the last pose of the navigation, which is what usually happens. If so, the accurate positioning process starts.

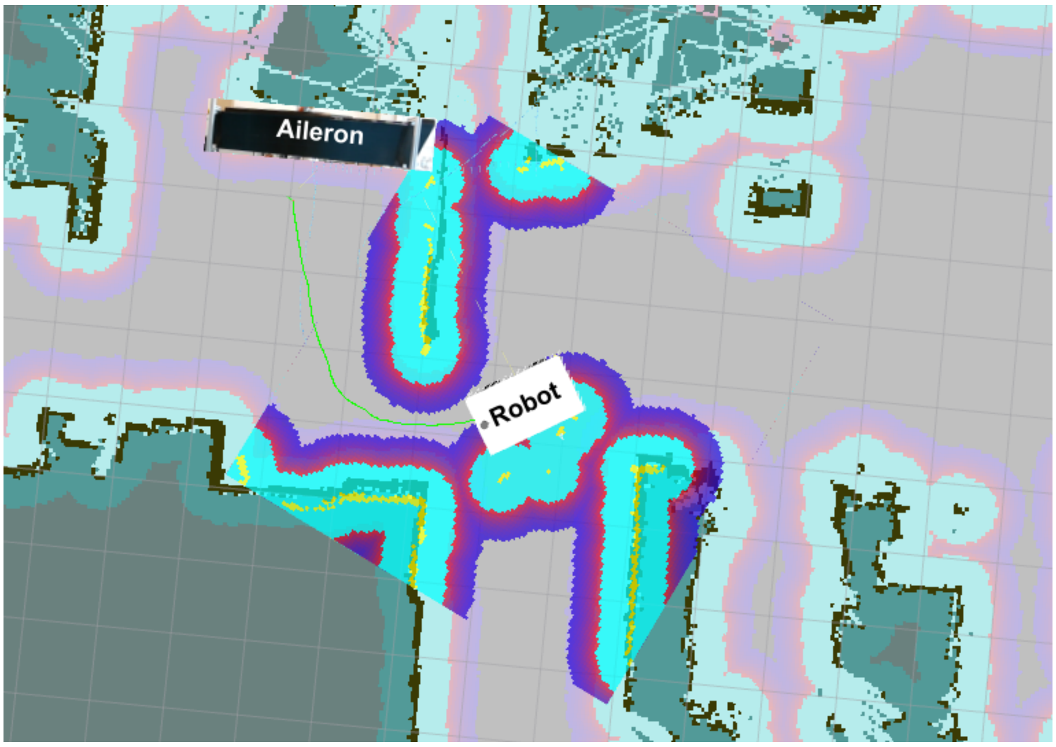


Fig. 4.4. – The robot navigating autonomously to the inspection area. It is about to pass through the door that connects the large shop floor with the laboratory where the aileron is placed. Free mapped space is in grey, obstacles are in black and blue/red colours are the inflation of obstacles in the costmap. The green line represents the trajectory planned to reach the goal.

First, the system detects the marker with the camera and calculates the spatial relation between the marker and the robot. The ideal inspection pose of the platform with respect to the marker is already known, as it is calculated beforehand. Hence, the system derives the relative movement to be performed from this relation, until the objective pose is achieved. These movements are slight corrections, and they are accomplished by sending linear and angular velocity commands directly to the platform's motion controller. To travel between consecutive inspection poses of a unique aileron, the same technique is used. During this movement phase, it must be noted that the system does not make use of the map and that its localisation system totally depends on the platform's odometry and on seeing the labels. Therefore, ideally, at least one marker should always be in the field of view of the camera. If no markers are detected, the platform moves with *dead reckoning*.

Once the second label is detected, the same accurate positioning and inspection process is performed. In principle, this positioning process can be repeated as many times as necessary without worsening the accuracy, as each marker is calibrated independently with respect to the aileron.

To check that the platform actually achieves the desired accuracy, the vision system itself is used (validated as explained in Section 4.2.2). With this same method, the pose of the platform before and after the correction is measured, in order to quantify the improvement. In practice, though, the way to verify that the accuracy has been sufficiently improved is ensuring the arm can reach every point of the inspection trajectory.

4.2.4 Experimental results

The tests conducted focued on demonstrating that the proposed vision system is capable of correcting the error made in localisation during autonomous navigation, enabling the efficient inspection of an aileron. The robot started in a in a large shop floor and navigated autonomously to a laboratory next to it, passing through a narrow entrance and travelling a distance of approximately 20 m to its final destination. For autonomous navigation, the entire store floor was mapped with GMapping, which was 30 m wide and 90 m long, and the robot used AMCL for localisation and A* for trajectory planning. A total of 25 experiments were performed, each of them with the corresponding accurate positioning process. As each experiment's starting point varied in position and orientation, the time navigating autonomously and the length of the trajectories also varied from one test to another, with the aim of adding variability to the experimentation and increasing the number of situations covered.

If the tolerance configured for the correction of the motion control is too small, the platform will hardly achieve it and, as a consequence, it can keep oscillating trying to accomplish the correction. After some tests with the platform, a tolerance of 5 mm in the x axis, 3 mm in the y axis, and 0.12° (0.002 rad) in rotation was established for all the experiments. However, as these values are relatively low, we considered that the error made by the autonomous navigation at the destination point was fully corrected if the robot succeeded to complete the accurate positioning step. Besides, in order to position the platform within this range, the speeds at which it moves were also reduced to a minimum. After several tests, we concluded that these values were: 0.012 m/s in x axis, 0.012 m/s in y axis, and an angular velocity of 2.3 °/s (0.040 rad/s).

Figure 4.5 shows the positions achieved after autonomous navigation, which are then corrected. Note that autonomous navigation is only used to reach the first marker, as for the subsequent markers a relative movement is made without taking into account the map-based localisation. For this reason, to calculate the correction that the proposed system achieves, only the positions with which the robot reaches the first marker have been considered. In all the tests conducted, autonomous navigation reached the inspection area so that the camera could detect the artificial marker. Similarly, in all cases, the platform successfully achieved precise positioning. Therefore, thanks to the vision system, the platform could always reach the goal with the accuracy mentioned above.

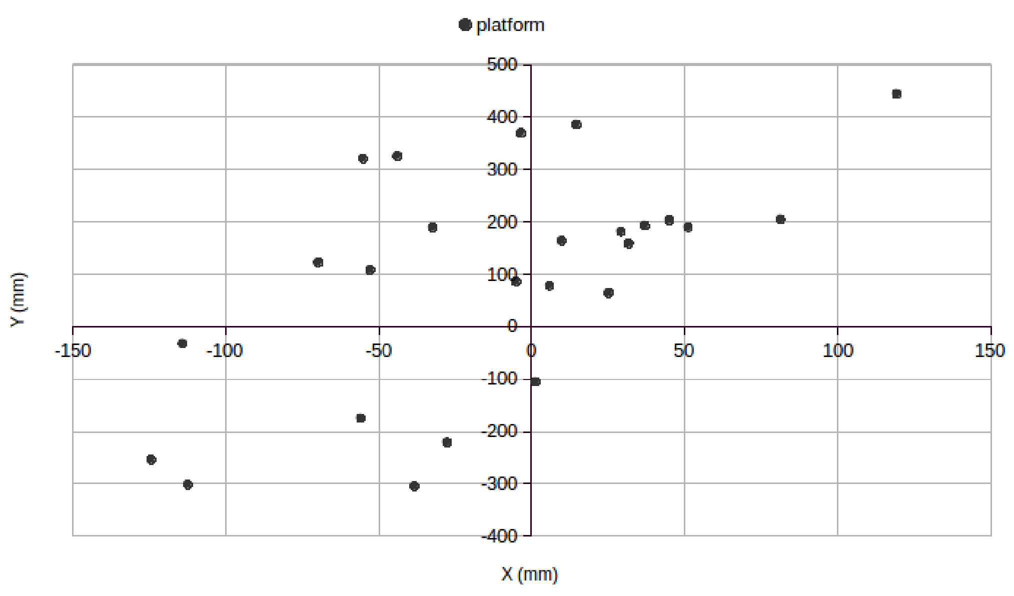


Fig. 4.5. – Positions of the robot after the autonomous navigation. If we understand the graph as the horizontal plane, each point P represents the coordinates of the position at which the robot arrives after the autonomous navigation and prior to correction. Being the origin O the target position, which is achieved after the correction, the modulus of $\overrightarrow{PO}$ is equivalent to the distance corrected.

In the 25 positions recorded, an average error in translation of 217 mm was corrected with a standard deviation of 104 mm. Even in the most accurate autonomous navigation (closest to the target position), a minimum improvement of 70 mm was achieved. On the contrary, in the worst case, an error of 456 mm was corrected with the accurate positioning method. Note that, as shown in Figure 4.6, the errors on the y axis were much larger, on the order of two to three times the ones made on the x axis. This is probably due to the fact that the odometry in the lateral movement, y axis, of the used platform is less precise than when moving in the longitudinal axis, because of the specific configuration of the mecanum wheels.

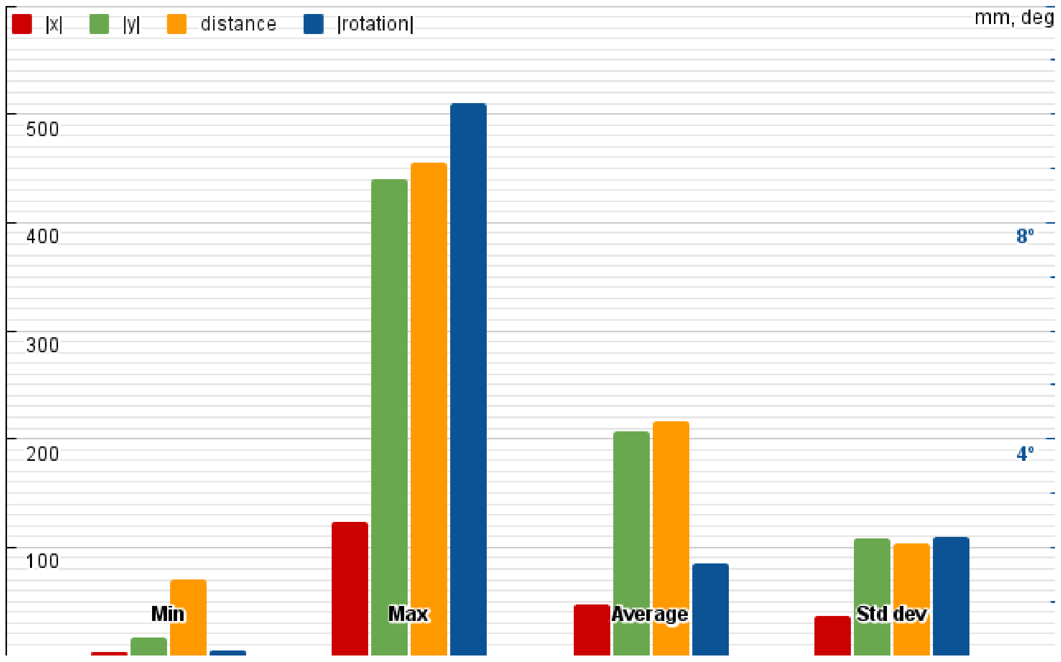


Fig. 4.6. – Errors corrected with the precise positioning procedure after autonomous navigation.

Furthermore, the tests showed that the deviation tended equally to one side of the x axis as to the other, i.e., the positive magnitudes were similar to the negative ones. On the contrary, regarding the y axis, there was a tendency towards positive values (commonly left). This behaviour is shown in Figure 4.7. With respect to rotation, the inaccuracies were proportionally much smaller than in translation, if we compare them with the maximum error of 16º admissible to perform the correction (see Section 4.2.3). Besides, the measurements were more homogeneous as, on average, 1.7º were corrected, with a standard deviation of 2.2º. Still, the greatest improvement was 10.2º, which is a significantly large error, although it could be considered as an outlier.

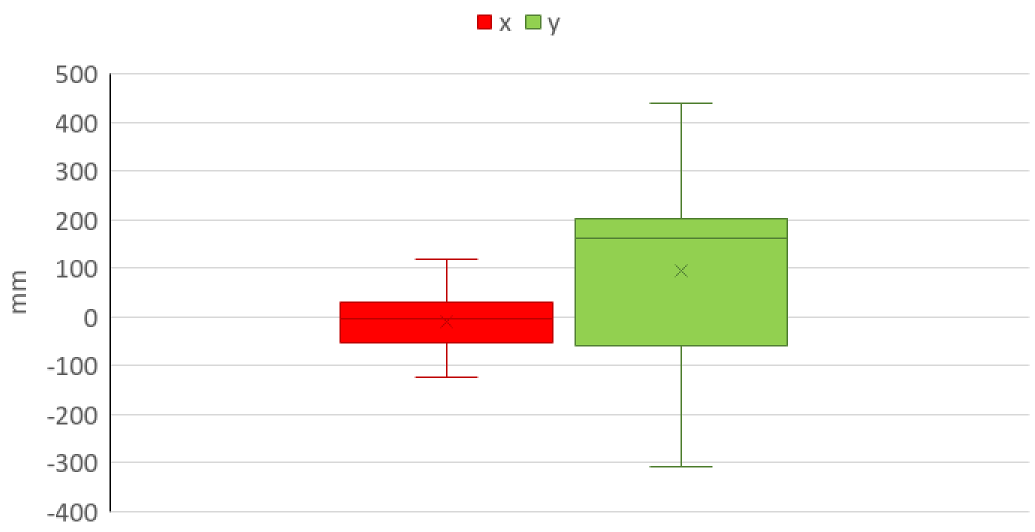


Fig. 4.7. – Tendency of the error on each axis.

4.3 GNSS-based Localisation in Greenhouse Crops

Issues such as population growth, climate change, resource scarcity and increased competition, a key challenge for agriculture today is to increase production while reducing resources. Greenhouses can protect crops from adverse weather conditions, allowing year-round production and environmental consistency to increase yields, while modern crop management approaches can significantly reduce water use. However, in a closed environment such as a greenhouse, any infection can spread rapidly if pests are not detected and treated in time, which will negatively affect production.

To prevent this, crop inspection for pest identification must be performed thoroughly, so that the appropriate treatment can be applied as soon as possible and keep damage to a minimum. In addition, treatment must be applied in the right place, and only there, so as not to use more resources than necessary. Traditionally, manual inspection methods have been used, but these are labour-intensive and inefficient, making them unfeasible as the area of glasshouses increases. As an alternative, automatic inspection by computer vision has become increasingly popular. This method is well suited to repetitive tasks and can also improve detection accuracy, improving productivity and reducing the use of pesticides.

The main objective of this research is to design and develop an innovative robotic solution for Integrated Pest Management (IPM) in greenhouse crops, with the ability to navigate inside greenhouses whilst performing pest detection and control tasks in an autonomous way. Although some automatic IPM tools and techniques have already been proposed [239, 257, 60, 16], none of them presents a completely autonomous mobile manipulator as part of the pest treatment system. The main advantage of using a mobile robotic platform is that it can cover a large greenhouse, or greenhouses, with just a few sensors, as the same robot can move through different sections and inspect any number of plants in the path. Furthermore, unlike systems that rely on fixed guide rails, for example, a mobile platform does not require a specific fixed infrastructure, being able to cope with changes in the layout that may arise, making it much more flexible and easier to deploy.

Precise localisation is a fundamental capability for this research, as it enables autonomous navigation of the robot inside the greenhouse and also makes it possible to register accurately the location of the detected pests. Given that the use of pesticides is subject to regulation, the robot needs two working modes: detection and treatment. During detection mode, the detected pests must be registered in the generated map with enough accuracy to guarantee that the robot will be able to come back to the issued areas when working in treatment mode. Pesticide spraying cannot be done as soon as a pest is detected because the points to be sprayed are determined by a complex decision algorithm that requires knowledge of the state of the entire greenhouse.

The high level of symmetries and repeated scenes in greenhouses make traditional localisation algorithms drastically worsen the pose estimation. Moreover, the rough terrain of the working area affects the odometry significantly. On the one hand, the planar motion assumptions are not directly applicable when the ground is ondulated. On the other hand, bumps on the ground can cause motion abruptness and make visual registration harder. Therefore, to effectively perform the task, an additional positioning system is required.

Artificial landmark-based vision systems like the one presented in Section 4.2 are not valid in greenhouses for several reasons:

- It is not efficient to cover the whole greenhouse with artificial labels. They are large environments with narrow corridors, which means that not many elements fall within the camera's field of view, due to the closeness of these. Blurring of images in motion or at long distances and occlusions can lead to complete loss of the robot's location. Besides, calibrating all of them with respect to a common reference to obtain a global localisation value is a complex task.

- A fixed structure on which to attach the labels is needed. Greenhouses are changing scenarios due to the nature of the plants that compose them, in which it is not possible to add static labels that serve for the localisation of the platform.

- Accuracy and robustness in outdoor conditions is not tested. The accurate landmark-based positioning has been designed and evaluated in an indoor environment, where the lighting conditions are stable.

Most SLAM research studies are conducted in indoor environments with a large number of features that are always under similar conditions. Outdoor environments present changes such as seasons, weather, or time of day that are difficult to reflect on maps and pose a challenge to SLAM methods [223, 197, 27]. Although GNSS can help solve the location problem in these situations, it requires a constant and reliable signal.

A greenhouse, the scenario of the present study, presents the disadvantages of both worlds. On the one hand, its infrastructure can block the direct reception of GNSS signals and generate multipath interference [250], which has disastrous consequences on the provided localisation. On the other hand, its visual appearance is conditioned by climate. Plants evolve over time, mainly due to seasons, and indoor lighting is directly influenced by outdoor conditions. Figure 4.8 shows the state of the greenhouse at the beginning and at the end of the project, while the Segway platform described in Section 3.4.1 adapted for this scenario navigates through one of the rows.

For these reasons, an additional localisation system is needed that does not have the mentioned disadvantages of visual methods. In our approach, we study the capabilities of combining GNSS and lidar data with probabilistic techniques in a semi-indoor environment. It is stated that the localisation module should prove an accuracy of 30 cm in position and 5º in heading at least 95% of the time, mainly given by the spraying algorithm.

4.3.1 Current State of GNSS for Mobile Robotics

GNSS refers to a constellation of satellites that provide signals from space and deliver position and timing data to GNSS receivers. The receivers then use this data to determine location which has, by definition, global coverage. GNSS positioning techniques provide real-time measurements that can be used as the primary sensor in some agricultural robot navigation scenarios [181]. The signals provided by

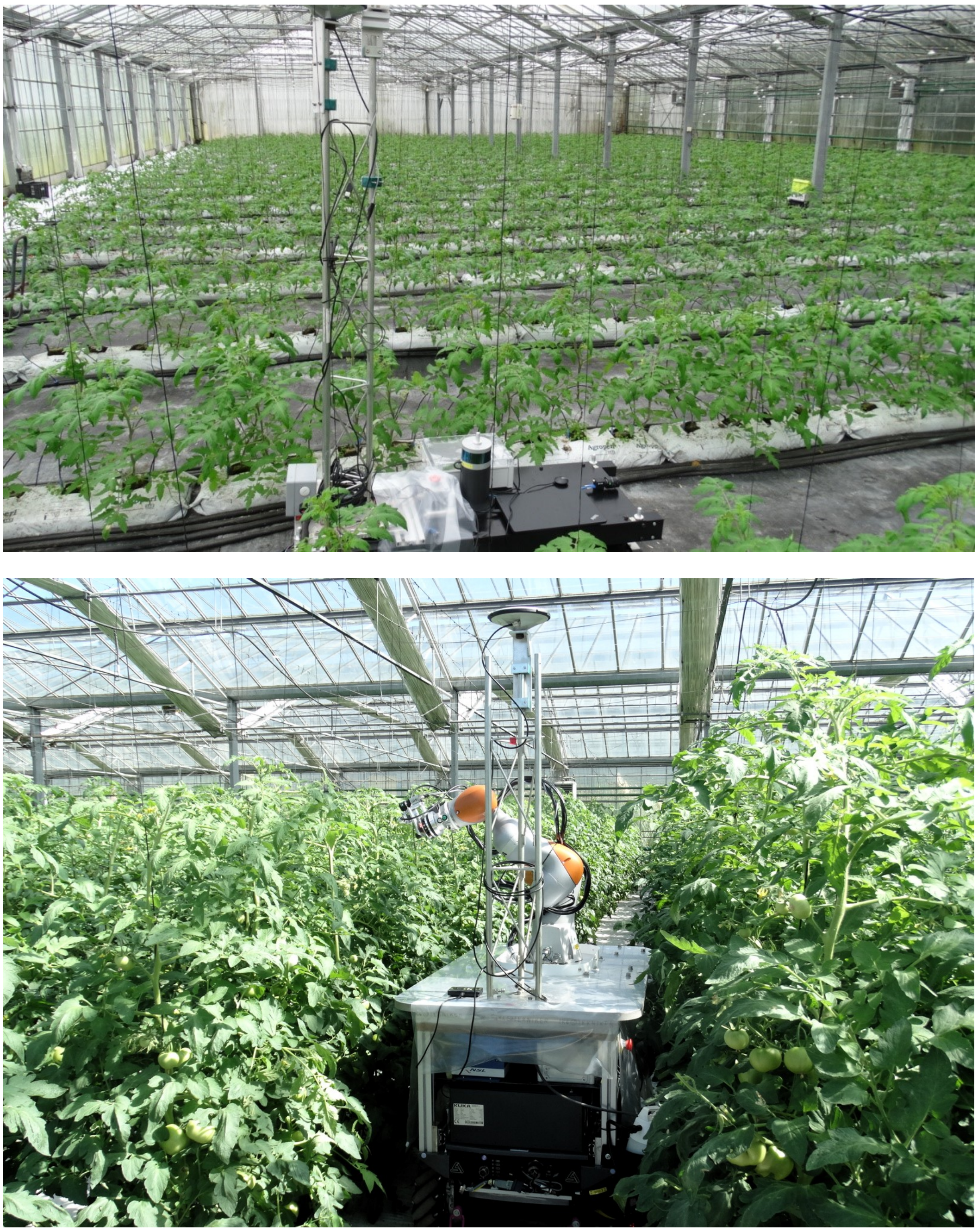


Fig. 4.8. – The platform inspecting plants while navigating through the greenhouse. Note the difference of the plants at the beginning of the development (top) compared to some months later (bottom). In that period, the platform was modified and *grew* as well.

Galileo [225], the European GNSS, allows users to know their exact position with greater precision than what is offered by other available satellite systems, making it a valuable tool for tackling this challenge.

However, due to the difficulties mentioned in Section 4.3, mobile robotics applications cannot rely solely on GNSS-based geolocalisation. Therefore, it is usually coupled with the dead reckoning (DR), based on inertial navigation systems (INS) and encoders' odometry [82]. In fact, GNSS and dead reckoning (DR) techniques are complementary in nature, since the incremental error of the latter can be corrected by the former, whose error is not cumulative. Furthermore, while one requires a satellite signal, that can be lost, the other does not require any communication with an external element.

To perform this GNSS-DR combination, the most common strategies are tightly-coupled and loosely-coupled integration. Briefly, the basic difference between them is the type of data shared by the GNSS receiver and the DR module [41]. In the loosely-coupled technique, the positions and velocities estimated by the GNSS receiver are blended with the INS navigation solution [84], while in the case of the tightly-coupled method, GNSS raw measurements are processed through a unique Kalman filter with the measurements coming from the DR sensors [159]. The heading angle provided by GNSS-DR can help to correct drift issues, as well as to control big errors in relative positioning [184].

Therefore, problems and errors arising from separate GNSS and DR methods can be reduced by exploiting the complementary qualities of both through GNSS-DR fusion techniques. However, in practice, this solution is not robust enough [231], as both components may lead to a bad outcome at the same time. SLAM methods have proven to provide relatively good robustness and accuracy when the environment does not vary much with respect to the map. Although we have already said that a greenhouse changes over time, it does in the mid to long term, so SLAM methods can be a valuable source of information in the short term [182]. Thus, SLAM techniques can help to robustify the system by, for example, correcting the drift error of the DR when the GNSS signal is not available [37].

The following is a brief description of the GNSS techniques that have been assessed for analysis in this scenario, which are the most widely used ones in the literature.

Single Point Positioning (SPP)

SPP is a positioning technique characterised by a single receiver that receives signals from at least four different satellites [245]. It does not use any post-processing technique, and its position is not relative to any point with known position (e.g. a nearby base station). Although SPP is robust and easy to compute, it suffers from several limitations. First, measurements of this technique's code - or pseudo-distance - experiences noise and errors that limit performance and make the accuracy of positioning solutions around 1-2 m at best [199]. This tolerance is valid for some operations, but higher precision is required for the geo-referencing of photographs or treatments and autonomous navigation corresponding to our application.

Real Time Kinematic (RTK)

RTK is a GNSS correction technology that provides real-time centimetre-level positioning [170]. It consists of two receivers: one is the rover receiver placed in the mobile platform, and the other one is the base station located over a known control point. Both elements receive satellite signals and communicate between them via Internet or radio, so the rover can apply base corrections and improve its accuracy. Note that a specific base station is not always needed, as there may be local service providers who share their base corrections over the Internet. However, in both cases, the performance decreases as the distance between the base station and the user increases [83].

Post-Processing Kinematic (PPK)

PPK is an alternative technique to RTK, where positional corrections are applied in a post-processing step [232]. Here, the rover and the base station collect raw GNSS data, which are then processed to get an accurate positioning track. Thus, there is no need for continuous communication between both receivers, but the corrected position is not available in real time either. The accuracy obtained is, as in the case of RTK, within a few centimetres.

Precise Point Positioning (PPP)

PPP stands out as an optimal approach for providing static and kinematic geodetic point positioning solutions using a single receiver. Combining GNSS satellite clock and orbit corrections generated from a network of global reference stations, PPP is able to provide position solutions at centimetre-level precision without requiring a base station [22]. This absence of the need for a local reference station means that PPP is applicable anywhere in the world, as the accuracy does not depend on the distance to a reference station. In addition, the user's position can be calculated directly in a global reference frame instead of being positioned relative to a single reference station. Nevertheless, the main limitation of the PPP technique is its requirement of a relatively long convergence time for precise positioning.

4.3.2 Proposed Approach

The approach proposed in this work is based on a GNSS-DR-SLAM localisation strategy that attempts to combine the strengths of the three methods to perform the inspection and treatment procedures successfully. The target environment in which the system must operate is a single rectangular greenhouse of 52 x 30 m composed of a main longitudinal corridor and 39 transverse corridors or rows.

To manage the different data sources, the architectural design of the proposed system considers localisation at two different levels: absolute and relative. Figure 4.9 shows how the proposed architecture combines different sources of information for localisation estimation and commands the mobile platform to perform navigation.

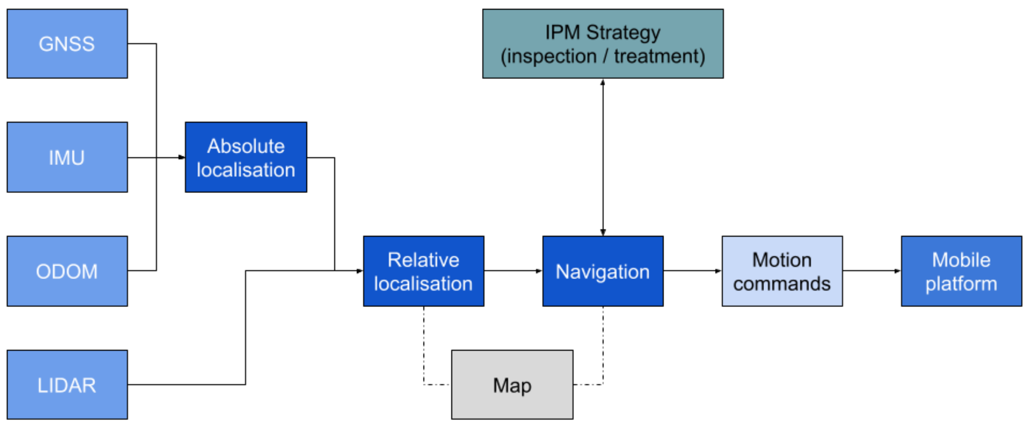


Fig. 4.9. – Proposed system architecture diagram in the greenhouse scenario.

Absolute Localisation

The purpose of the absolute localisation function is to provide real-time absolute position and heading information to the relative localisation module. Due to the nature of the greenhouse, different robot poses that are far from each other can generate the same SLAM posterior. Thus, limiting the area of possible poses can significantly increase the robustness and precision of the estimation.

The absolute localisation module combines GNSS, inertial and odometry data to generate accurate real-time position and heading information. The GNSS processing uses the PPP approach to obtain the required absolute position performance, and the data obtained from the odometer and the IMU are combined with it. In particular, the proposed PPP solution takes advantage of additional signals provided by the Galileo system, such as the E5 AltBOC signal [206].

First, the output of the PPP module is transformed into the robot's world coordinates, i.e., geographic coordinate expressed as a latitude/longitude pair and an earth-reference heading into x, y and heading with respect to the robot's origin. To do this, the geolocalisation of the robot's world frame must be known, as the transformation is calculated based on this relation. In our case, as the system uses a map for the operation, the robot's world frame is the origin of the map itself, whose corresponding geolocalisation is known and constant. Then, PPP information is combined with that from the robot's sensors to provide a better estimation of the robot's position in real time. The fusion is performed with a UKF, using the implementation proposed by Moore and Stouch [158], as it accepts measurements from an arbitrary number of pose-related sensors, among which IMUs, odometers, and GNSSs are included. In addition, if there is a long period in which no data is received from any of the sensors, the filter will continue to estimate the state of the robot via an internal motion model, allowing continuous estimation.

Relative Localisation

The purpose of the relative localisation function is to capture the robot sensor information at the right level of abstraction and use it to estimate the robot position on the map of the environment. This relative position and heading of the robot within its local environment is provided to the navigation function so that the platform is able to travel to the inspection and spraying destinations.

The relative localisation layer takes the output of the absolute localisation layer as input and combines it with a local localisation module. This second module takes lidar readings and attempts to match them with the map through a KLD-sampling MCL [69] (see Section 2.3.2). The 2D map that represents the environment is created using GMapping [81] while the platform is teleoperated, as described in Section 3.4.2. However, in this work, the map used in the experiments is obtained giving to GMapping the output of the absolute localisation module, instead of using raw odometry.

The final position and heading information is given in local coordinates in the ENU (East, North, Up) reference frame, i.e., tangent to the earth's surface with axes pointing *east* for x, *north* for y and *up* for z, and whose origin is defined by a specific coordinate point in the greenhouse.

4.3.3 Experimental results

To validate both the absolute and relative localisation estimations, a reference is needed for comparison. In these greenhouse tests we did not have an accurate external system to use as a ground truth, as in the case of Section 3.3. Alternatively, we evaluate if the robot moves through the corridors in a straight line and pointing in the direction of the travel. Therefore, for these parts of the route where the robot moves in a straight line along the corridors, a reference path is calculated using the PPP coordinates of the start and end of the corridor. Then, the localisation obtained at each step is compared to this reference value, considering the difference between the two as the heading deviation. This method can be used to identify positions with a poor heading and to get a general idea of the quality of the estimation.

However, using this reference as ground truth for the evaluation of the heading accuracy has several limitations. Firstly, it only covers periods when the robot is moving in a straight line and does not give an indication of the performance when the robot is turning. In addition, this reference line assumes that the robot has maintained the same heading at every position along the route, which is not always the case, as can be seen in Figure 4.10. Therefore, the results do not consider those areas where the robot made small turns and changes of direction, which correspond to rows 1 and 39, and the main corridor. These alterations are consequence of navigation trajectories, and not of localisation accuracy, which is what we want to evaluate. Finally, even in the other rows where the space was limited and the robot could not vary its heading much, it was not always heading forwards, so we do not expect this analysis to determine accuracy with complete reliability, but it does provide an approximation of the overall level of performance.

Data logs collected in the real greenhouse scenario have been used for the verification of the localisation layers. During the experiment, the robot performed a total of 150 different trajectories, navigating along the main corridor and some of the rows among the plants 10 times. It started at the entrance of the greenhouse, near

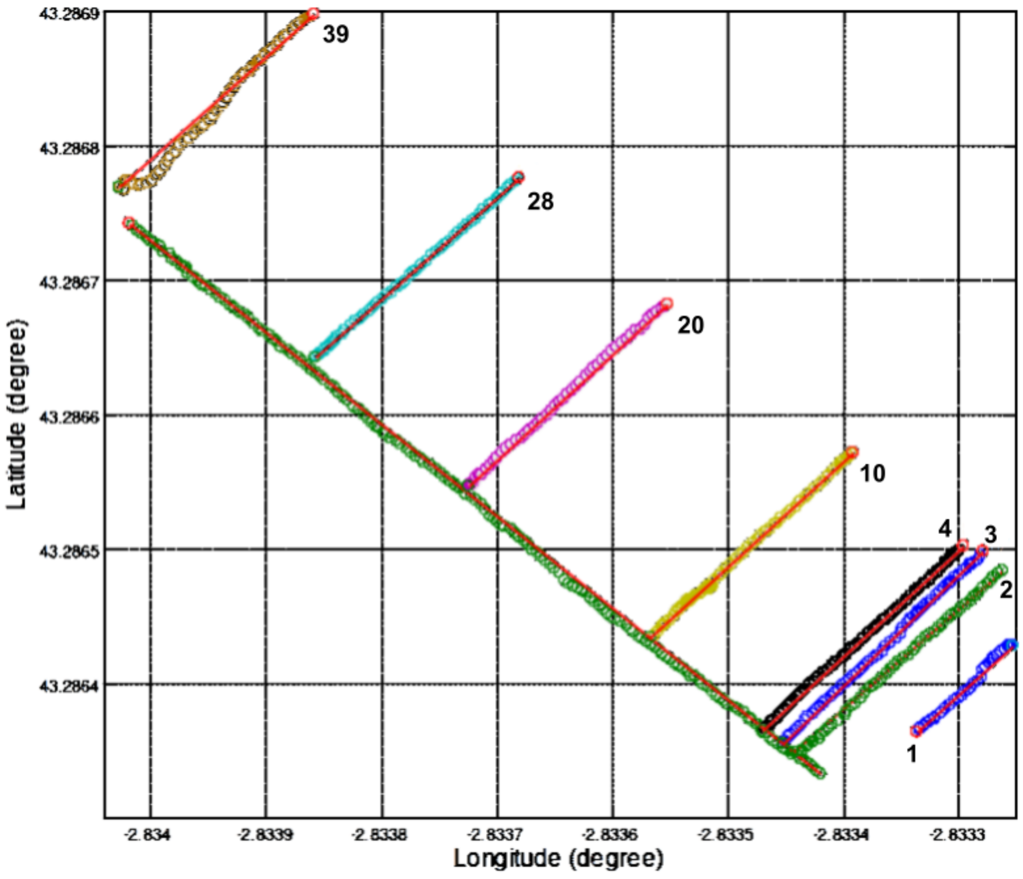


Fig. 4.10. – Robot positions along the greenhouse corridors. The red line shows the straight line between the starting point and the end point of the corridor, i.e., the reference path. Circles show the positions of the robot obtained by the absolute localisation module.

corridor 1, and all the autonomous navigation destinations were selected by the IPM strategy, which was restricted to the corridors shown in Figure 4.10 to reduce the time of the validation.

Mapping

Although the actual creation of the map is not the subject of study in this work, it is necessary to estimate the localisation with AMCL and a good map is crucial to navigate successfully, as noted in Chapter 1. To do this, several tests were carried out using different configurations. With regard to the generation of the map, it is important that the result does not depend too much on the growth phases of the plants or other elements in the greenhouse that may change size or shape over time. For example, in vegetated environments, mapping the stems may help to robust the robot's behaviour, but not the leaves, as the these ones may vary too much.

The method used to build the map was GMapping, as we ranked it as the best after the tests described in Chapter 3. Initially, the map was built with the same configuration used in Section 3.3, that considers 100 particles and odometry errors with 0.075 rho/rho and theta/rho, and 0.15 rho/theta and theta/theta. It is worth mentioning that, in this approach, each particle includes a particular state of the map. With respect to *odometry errors*, these parameters aim to quantify the extent to which these values diverge from the actual movement of the robot. In this line, rho/rho and rho/theta correspond to the odometry error in translation as a function of translation and rotation, respectively. Thus, theta/rho and theta/theta are just the opposite, how translation and rotation affect to the odometry error in rotation. Using these parameters for GMapping, the resulting map is not able to reflect features such as straight lines in the main corridor, rows among plants or right angles, as shown in Figure 4.11.

To obtain a configuration that overcomes these drawbacks, an iterative *trial and error* process is followed, modifying the number of particles and reducing the parameters related to odometry errors - taking into account that these errors with a PPP+DR solution are significantly lower. The final configuration considers 200 particles and odometry errors 100 times smaller (0.00075 rho/rho and theta/rho, and 0.0015 rho/theta and theta/theta).

Figure 4.11 shows the difference between the maps obtained using the initial configuration for GMapping and the one selected. Ten maps have been generated with each configuration, in which the strengths and weaknesses corresponding to each of them are maintained. With the final parameters, both the main corridor and the rows among the plants are straight lines, there are right angles between them and the last three rows maintain the expected parallelism. In addition, the dimensions represented on the map match the measurements in the real greenhouse, which are 123 m long and 27 m wide.

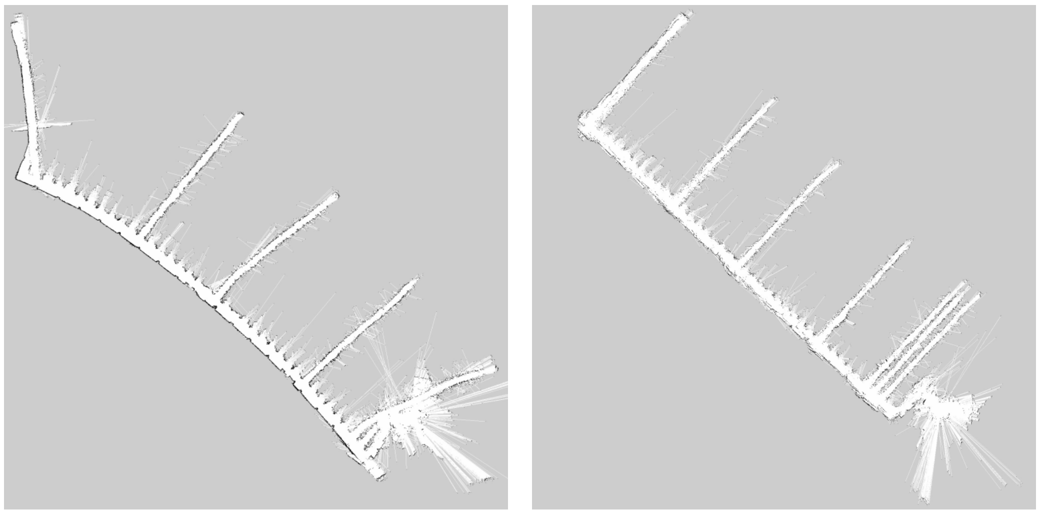


Fig. 4.11. – Maps generated with different GMapping configurations. On the left-hand side, the map built with the initial configuration. On the right-hand side, the one used in the experiments generated with the configuration obtained after an iterative optimisation process. Note the straight lines, the right angles and the number of corridors.

Absolute Localisation Error

The heading error has been calculated individually for each corridor, without taking into account those where the robot did not go in a straight line, as mentioned above. Furthermore, these values have been discretised into three groups or conditions, as the robot should maintain an error of less than 5º 95% of the time.

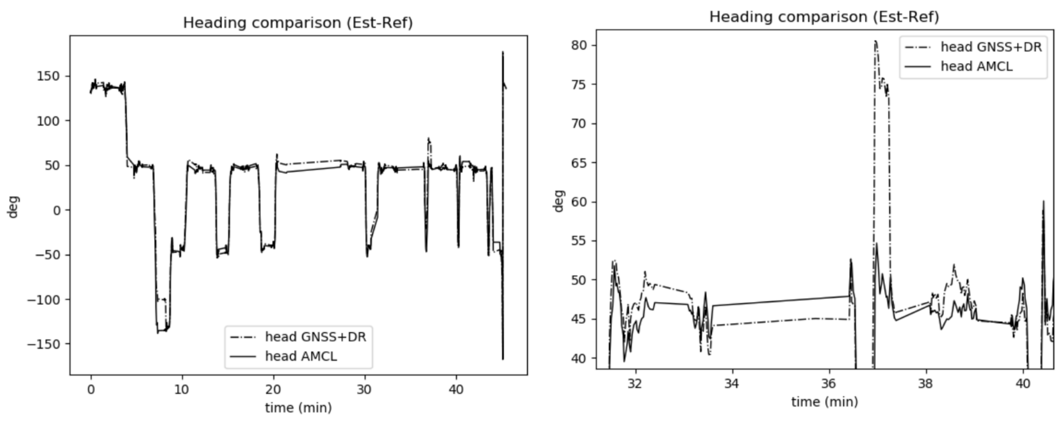
Table 4.1 shows the percentage of measurements where the error is less than a certain value, where the heading deviation is less than 5º in 91.12% of the positions evaluated, and less than 6º in 94.48%. Based on the tests performed, 95% of the time an error of less than 7° can be assured, so that the proposed system offers a robust accuracy between 1° and 2° lower than the required one. Although this is not a formal validation, it is a good approximation, as it has proven to be sufficient to be able to perform the IPM process correctly in subsequent experiments.

Row	<5º	<6º	<7º
2	92,00%	92,00%	92,00%
3	91,80%	100,00%	100,00%
4	92,30%	96,15%	96,15%
10	96,35%	97,01%	97,67%
20	86,79%	92,45%	100,00%
28	87,50%	89,29%	91,07%
Mean	**91,12%**	**94,48%**	**96,15%**

Tab. 4.1. – Deviation of the heading estimation of the system, with respect to the straight line representing the corridor.

Relative Localisation Error

Figure 4.12 shows the difference between the heading calculated by the absolute and the relative localisation modules. It can be seen that they have a similar profile and that the values vary slightly. However, a maximum difference of 26.037º is reached in a region, which is considerable.



(a)

(b)

Fig. 4.12. – Heading difference between absolute and relative localisation modules, where the 0 value corresponds to the East. Figure 4.12(b) zooms in on the interval between minutes 32 and 40 of Figure 4.12(a) to represent the difference with greater precision.

Analysing the heading estimation provided by the two modules for each straight path followed by the robot in the greenhouse, it can be seen that the relative module is able to improve the solution provided by the absolute location module for all cases, as shown in Table 4.2.

Row	Abs. error (°)	Rel. error (°)	Error reduction %
2	3.232	2.916	9.78%
3	2.084	2.073	0.54%
4	2.686	2.062	23.25%
10	2.677	2.121	20.76%
20	2.005	1.841	8.21%
28	3.175	2.205	30.55%
Mean	**2.643**	**2.203**	**16.66%**

Tab. 4.2. – Mean reduction of the heading estimation error when applying the relative localisation method to the result of the absolute localisation module.

4.4 Conclusions

In this chapter, we have presented the development and evaluation of two methods for improving localisation accuracy in real-world scenarios.

On the one hand, the platform must navigate autonomously through an indoor scenario, requiring a higher precision than the one obtained in this operation with state-of-the-art methods at the destination point. This improvement is achieved by detecting artificial tags placed in the destination area, which are used to directly control the robot's movement. Using an external micrometric calibration system, it has been validated that the relative localisation of these tags is estimated with millimetric accuracy. However, the platform does not offer such a level of control, so the proposed positioning system achieves a final accuracy of approximately 5 mm in the x axis, 3 mm in the y axis, and 0.12° (0.002 rad) in rotation around the z axis.

On the other hand, we address the problem of localisation using the same platform, but this time in a greenhouse. This semi-indoor scenario poses other challenges, mainly due to its symmetry, repetition of elements, and variation of the plants. The proposed method is based on the use of GNSS PPP, which serves as an additional input to the localisation system. With this fusion, the results obtained are better than those offered by each of them independently, as it has been proven in the generation of the map and the estimation of the platform's heading.

Compared to the results obtained in Section 3.4.3, these errors are slightly larger due to the features of the environment described above, but remain in the same order of magnitude, making the two studies consistent with each other.

These experiments have shown once again that localisation based on sensory information and the model of the environment previously generated by it may not be sufficient for the performance of certain tasks. This is highly dependent on the environment in which the operation is performed, and it is for this same reason that there is no universal solution.

5 Active Mapping and Exploration

There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.

— **Albert Einstein**

In this chapter, we present a system that is capable of building an uncertainty 3D map of the environment by optimising the pose of the sensor that is feeding it. Our algorithm estimates online the next optimal pose of the camera in order to maximise the information provided by the map in the next step, balancing revisiting and exploration. It is combined with a 2D mobile robot exploration method to obtain a richer 3D model of the environment. The system is also compared with a 3D exploration procedure to evaluate its performance and the quality of the obtained maps.

5.1 Introduction

Having a virtual model of the environment is necessary for the transition to the smart industry. When creating the digital twin of a robot, not only the robot itself must be digitalised, but also its surroundings. Modelling the environment using visual sensors is an efficient way of doing this task, with a result that can be very realistic and equivalent to how the robot perceives the environment. However, the process of teleoperating a robot to map an environment is usually a highly time consuming task, especially in large areas or when the movement of the robot is limited. In other cases, it is difficult or even impossible for the robot to be guided due to insufficient connectivity or dangerous conditions, such as in rescue operations of natural disasters. Also, a technician should be available and present so the robot completes the mapping process efficiently, which is not always possible in real scenarios. Besides, environments where mobile robots end operating in are often unstructured and chaotic, and it leads to a fast decay of the performance over time.

This issue makes necessary to create or edit the map regularly. Therefore, there is not yet a method to make the robot capable of mapping an environment and start navigating through it effectively without human intervention.

Most of the SLAM techniques share the guidance of the robot by a human during the mapping process to ensure the full coverage of the environment and to ease the detection of loop closures as well. In fact, loop closure detection is the process of recognising an already mapped area and it is still one of the biggest problems in SLAM [76]. The seed of loop closure is to minimise the accumulated error [91, 74] during mapping and correct the map being built, increasing the coherence between the digital representation and the real scenario. Despite this, the resulting model usually needs a post-processing refining process to correct any erroneously added element. Altogether, it is not a straightforward process to perform all these actions automatically without no human intervention. As we have already mentioned, ASLAM consists of an online search for exploration destinations for mapping, instead of the traditional "passive" and teleoperated procedure. It simplifies the setting up of a navigation system in many applications, as the robot is capable of building the map by itself with no human interaction.

In this chapter, we present an ASLAM system that optimises the pose of the sensor for the 3D reconstruction of an environment, while a 2D algorithm controls the motion of a mobile platform. A Rapidly-Exploring Random Tree (RRT) [242] algorithm is used for the ground exploration strategy, which calculates the optimal exploration destination and sends it to the navigation unit. Then, the navigation system calculates an obstacle-free trajectory and the mobile platform executes it. Simultaneously, the pose of the camera is optimised based on the state of the 3D map and the position of the platform, in order to capture the most relevant information possible in each iteration. The information gathered from the camera is fed back into the 3D map. The main contributions of this work are as follows:

1. A system to create a 3D model of the environment that provides information about the quality of each mapped area. This model serves as a metric to compare different reconstruction approaches.

2. An active vision method that optimises the pose of a camera to explore the environment and increase the quality of the resulting map.

5.2 Related Work

ASLAM approaches may focus on improving any of the phases described in Section 2.8.1 (*pose identification*, *goal selection* and *navigation and checking*). One relevant aspect that differentiates one algorithm from another is whether it optimises the complete trajectory of the robot during exploration or it only estimates an optimal viewpoint [34, 207]. In the latter case, only the navigation destination is calculated, and a generic algorithm that does not address exploration is in charge of path planning. Another common criterion to group these methods is the localisation uncertainty. Exploration can assume perfect positioning [142] or, on the contrary, it can address localisation uncertainty too [246], setting destinations not only to broaden the knowledge about the environment, but also to improve localisation estimates. Also, the search space for the pose identification step may vary as well. Candidates can be found in the entire map, or they can be limited to a part of it. Additionally, the candidate evaluation can be performed sequentially before and after the navigation step, or it can be performed continuously while the robot moves through the environment. In this case, the complete trajectory can be fulfilled, or, instead, it can be interrupted and a trajectory towards a new destination executed. As we can observe, the problem of ASLAM can be faced with many different perspectives, and the there are many trends in optimisation. In the following paragraphs, a brief review of the different approaches relevant to the current literature is done.

Even though it is one of the first contributions to ASLAM and not a recent work, we consider that the approach of Brian Yamauchi [260] must be highlighted, as many methods are still based on the same strategy. He proposes a solution based on frontiers, the key idea behind which is: "to gain the most new information, move to the boundary between open space and uncharted territory". To do so, he uses an evidence grid map where each point is free, occupied or unknown. Free points adjacent to unknown points are potential exploration destinations and they are grouped into frontier regions. Then, the robot navigates iteratively to the nearest reachable frontier. From there, the robot is able to get observations of unexplored space and add them to the map, in addition to seeing new potential goals. The path planner uses a depth-first search to calculate the shortest obstacle-free path from the robot's current position to the goal location. Navigating to each vantage point and discarding inaccessible ones, the robot can map every reachable point in the environment.

Senarathne and Wang [205] extend Yamauchi's work to 3D volumes and present a strategy based on the concept of surface frontiers. They detect surfaces in a given

3D occupancy grid map and extract their edges. Then, voxels that do not have their six faces exposed to unmapped space are discarded and the rest are considered as frontier voxels. The remaining edges are processed to remove noisy data and their contours segmented at corners. Finally, surface frontier representatives are generated that indicate the direction in which the surface is projected. These vectors are used to determine the navigation goals of the robot. In their approach, valid semi-random view configurations are generated and selected based on the the distance to them and the length of the surface frontier seen from there. The exploration finishes when no frontier regions are detected. Similarly, Dornhege et al. [58] present a frontier-based ASLAM of a 3D environment, but they seek frontiers in the complete volumetric space, and not only using surfaces. Besides, they set a specific number of poses as a termination criteria.

Some ASLAM methods propose to optimise the trajectory itself for mapping purposes and not only for the exploration goal of each iteration. In this vein, using Rapidly Exploring Random Trees (RRTs) [125] with a Receding Horizon (RH) strategy are well-known sampling-based techniques [21]. Umari and Mukhopadhyay [242] use RRTs to grow towards unknown regions and passively detect frontiers. The tree is not used to define the robot trajectory itself, but to search for frontier points. It runs independently of the robot movement. Likewise, Papachristos et al. [178] present an RH-based ASLAM strategy that considers the uncertainty of the robot localisation as well. In fact, the RRT is only used to find exploration destinations, and it is a second planning layer the one that aims to optimise the probabilistic mapping behaviour of the robot and minimise the root's belief uncertainty. In an attempt to get the benefits of different exploration methods, some approaches propose combining them in a single ASLAM solution [204, 177, 66].

As during the exploration a complete map is not available, some approaches concur that ASLAM strategies should include explicit place revisiting actions to reduce the localisation uncertainty [219]. In that vein, Carlone et al. [34] present a method that evaluates the particle-based SLAM posterior approximation using the Kullback–Leibler divergence to decide between exploration and place revisiting. More recently, Lehner et al. [128] integrate this same concept upon a submap-based 6D ASLAM system. Similarly, Valencia et al. [244] evaluate the utility of exploration and place revisiting sequences to choose the one that minimises the overall map and path entropy. On the other hand, Sadat et. al. [198] consider feature-richness during path planning to direct the sensor of a drone towards high-density areas and avoid visually-poor sections, as the latter can increase the uncertainty in pose estimation and make SLAM fail. The candidate viewpoints are also ranked based on their surface normals and the viewing distance.

Concerning dimensionality, nowadays most of the mobile robots are designed for 2D navigation, albeit there can be slopes, stairs or elevators that make the robot operate at different heights. In these situations, mobile systems use multi-level maps [193] and alternate among different layers, but the motion is still performed in a plane. Here, robot poses, velocity commands, and trajectories include only x and y positional values and rotations around the z axis (*yaw*). Indeed, 3D motion calculations increase the complexity of the problem considerably, and often it is not necessary. In this line, Micro Aerial Vehicles (MAVs) have been widely used in surveillance, search and rescue, exploration and mapping applications. Their reduced size and high manoeuvrability make them ideal for moving in cluttered environments. The algorithms developed in this area go beyond the limitations of ground robots, but there are several interesting approaches to take into consideration. For example, Kompis et al. [116] present an informed sampling approach that takes advantage of surface frontiers to sample viewpoints only where high information gain is expected. Potential Next-Best-Views (NVBs) are sampled from the MAV's configuration space using surface frontiers, and ranked by their expected information gain. Since the computational power of the MAV is limited, they define a heuristic to decide which proposed viewpoint to evaluate next.

Davison and Murray [48, 49] implement a general system for autonomous localisation using active vision, where a stereo head is controlled in real time during SLAM to improve localisation accuracy. They head the cameras towards certain areas to ensure that persistent features are rematched, reducing the motion drift. To decide which feature the vision system should target, they calculate uncertainties for all visible features, and choose the one with the largest uncertainty. Marchand and Chaumette [143] do consider unknown space in their work, as they propose a 3D scene reconstruction system based on an information gain function that represents either the observation of a new object or the certainty that a given region is object-free. Since the reconstruction is performed via a camera mounted on a robotic arm and it is limited to geometric primitives, they avoid unreachable viewpoints and positions in the vicinity of the robot joint limits. Isler et al. [98] go a step further, and propose mounting the arm on a mobile platform to achieve the complete 3D reconstruction of more complex objects. They employ an information gain function that considers both unknown voxels and their entropies, being these last ones derived from the occupancy probability. Delmerico et al. [56] also propose a set of volumetric information formulations and evaluate them together with recent formulations in the literature [118, 247].

Even though many more different methods have been proposed for active sensor control [78, 272, 267], most of them focus on the optimisation of the handled task once the scans have been added to the map, without intervening in the map building process itself. Instead, these approaches make use of common occupancy grid maps, which may include colour data or not, but they are not optimised for the posterior information estimation of already seen areas. We believe that the performance of 3D reconstruction algorithms is dependant upon the data stored in the map and, for this reason, our proposal includes additional information for uncertainty calculations in the map creation step.

5.3 Proposed Approach

Our approach makes use of active perception to search for viewpoints of the sensor that reveal areas of the environment that increase coverage together with the quality of the map being built. Our method differs from the ones mentioned above in the fact that it relies on the characteristics of the acquisition sensor itself and their effect on the data obtained to calculate the information gain and the NBV, instead of considering the size of the area to be mapped or the localisation uncertainty. In addition, it is proposed as a complementary method to an exploration process, optimising the perception capabilities of the robot used, and not a complete ASLAM method. The proposed approach is based on its own octree-like structure, which makes it independent and compatible with other mapping and navigation methods. Broadly, the system is an iterative algorithm that: captures a frame from the sensor; updates the 3D model according to it; calculates the next best sensor pose; moves the sensor; and repeats the process again capturing a frame from the new pose. In the generated 3D model, an uncertainty value together with spatial and occupancy information are stored for each point. In the next two sections, the uncertainty estimation and the algorithm itself are explained in detail.

5.3.1 Uncertainty Estimation

As we want the system to consider not only the completeness or coverage of the map but also the quality, we need a value that represents this aspect. It is hard to define or measure the quality of a map when there is no ground truth to compare it with, that is what happens most of the times in robotic mapping scenarios. In the absence of this option, the quality of a map can be seen as the reliability of its data. As uncertainty is a common term in mobile robotics and it is directly related to the reliability, it has been adopted for the representation built in our method.

The *uncertainty map* presented here is an *octree*-based [153] representation. Octrees are tree data structures where each branch node has eight children, as it represents an octant. Tree nodes store information regarding the space they represent. In our proposal, leaf nodes are voxels of the size of the resolution of the map, and they include uncertainty values u which are calculated as a function of the pose they have been seen from. Each time a cell is seen, the new value and the stored are combined to update the uncertainty value accordingly. The u value of non leaf nodes is thus inferred from leaf nodes. Every 3D point observed can be tracked, whether it represents free or occupied space.

We consider that points in the scene at a certain distance are more likely to be correctly captured by the sensor than elements that are closer or farther from it. Cameras are configured to see optimally a specific plane in the scene, called the focal plane. The closer an object is to the focal plane, the sharper it is seen. This attribute is degraded gradually until an element is too far from this location and it is completely blurred. The depth of field is then defined as the distance between the closest and farthest objects in a photograph that appear to be acceptably sharp [264]. This fact is modelled on our uncertainty map in two ways. On the one hand, from the set of points P captured by the sensor, only the points p whose Euclidean distance to the lens $d(p)$ falls in the range $[D_{min}, \dots, D_{max}]$ are added to the map M. Hence, elements that lay outside the depth of field are not taken into account. Therefore, being $p(x, y, z)$ a point whose spacial coordinates in the Euclidean space are x, y and z:

$$p(x,y,z) \in M \iff p(x,y,z) \in P : \{D_{min} < d(p) < D_{max} \mid d(p) = ||\overrightarrow{(x,y,z)}||_2\}$$

On the other hand, the distance $d(p, F)$ from each gathered point $p(x, y, z)$ to the focal plane F is measured as the difference between the distance from the lens to the point $d(p)$ and the distance from the lens to the focal plane D_F:

$$d(p,F) = |d(p) - D_F|, \tag{5.1}$$

Finally, the distance-based uncertainty of each point p in M is estimated according to $d(p, F)$, as shown in Equation 5.2.

$$u_d(p,F) = 1 - \frac{2}{1 + e^{d(p,F)}} \tag{5.2}$$

Equation 5.2 is a sigmoid function that represents the gradual degradation of the sharpness of the objects as explained above. As it only receives positives values, it does not have its characteristic "S"-shaped curve, as shown in Figure 5.2.

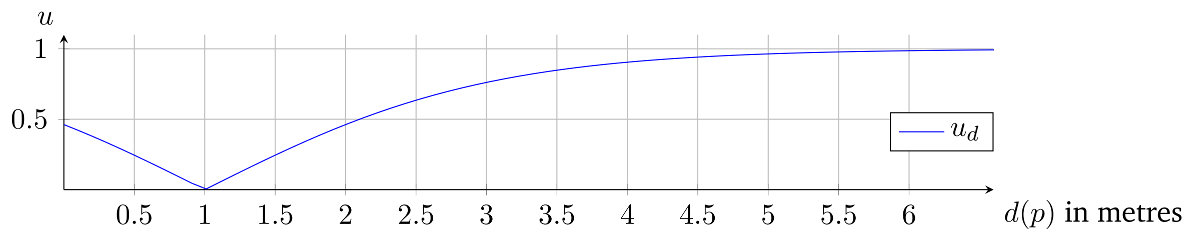


Fig. 5.1. – Plot of the function used to calculate the uncertainty of a point based on the distance to the lens in metres, being the distance to the focal plane $D_F = 1$.

Similarly, we assume that, given the FOV of a lens, points that lay on the boundary of the frame have a higher uncertainty due to the inherent distortion of the lens [28]. Although there are methods such as camera calibration to correct this deviation, they never reduce the error to zero. To represent this phenomenon, we define a function that calculates an uncertainty value for each point $p(x, y, z) \in P$, with respect to the sensor origin. The angles the point $p(x, y, z)$ forms with the horizontal plane XZ and vertical plane YZ, α_h and α_v respectively, are calculated as follows:

$$\alpha_h = \arctan(\frac{x}{z}) \tag{5.3}$$

$$\alpha_v = \arctan(\frac{y}{z}) \tag{5.4}$$

Besides, the horizontal FOV_h and vertical FOV_v angles determined by the sensor itself are required. They are used to perform the unity-based normalisation and bring the angles from Equations 5.3 and 5.4 into the range $[0, 1]$:

$$\alpha'_h = |\frac{\alpha_h}{FOV_h}| \tag{5.5}$$

$$\alpha'_v = |\frac{\alpha_v}{FOV_v}| \tag{5.6}$$

Then, the angle-based uncertainty u_α value of a point is the average between α'_h and α'_v:

$$u_\alpha(p) = \frac{\alpha'_h + \alpha'_v}{2} \tag{5.7}$$

Lastly, distance-based uncertainty u_d obtained from Equation 5.2 and angle-based uncertainty u_α from Equation 5.7 are combined into a unique u value, as shown in Equation 5.8.

$$u(p) = \frac{u_d + u_\alpha}{2} \tag{5.8}$$

u is the final uncertainty estimation calculated for each point of a scan. Figure 5.2 shows the impact one function or another has in the uncertainty map built.

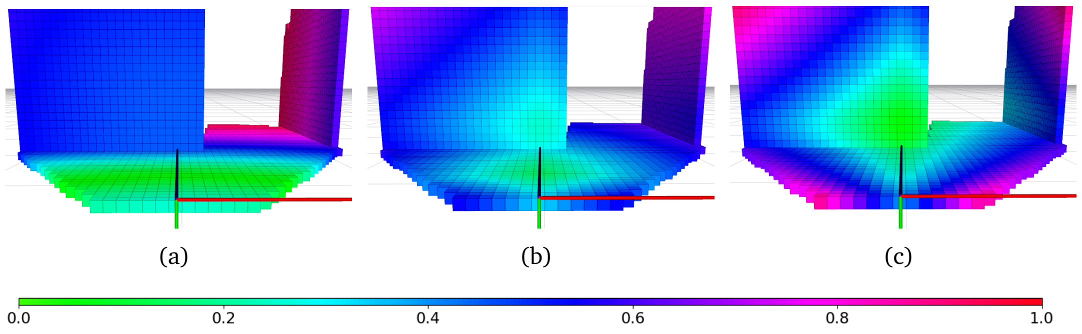


Fig. 5.2. – Initialisation of the uncertainty map in the same environment for each of the three uncertainty estimation Functions 5.2, 5.8 and 5.7. Showing 5.2(a) the distance-based uncertainty map (u_d), 5.2(b) the combined uncertainty map (u), and 5.2(c) the angle-based uncertainty map (u_α). A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

Nevertheless, once this value is calculated for an input point, it must be checked whether it is already mapped or not. If not, it is considered as a new unmapped point and it is added to the octree directly with its u value. This action increases the mapped area, i.e. the coverage. On the contrary, if the input point is already known and it has an uncertainty value from a previous iteration, it must be updated instead, enhancing the map quality. Here, we propose to store the uncertainty value that corresponds to the best viewpoint from which a certain point has been captured. That is, if a voxel v is captured for the t-th time, its new uncertainty $u_t(v)$ value is the minimum between the value $u(v_t)$ calculated with Equation 5.8 and the uncertainty value stored $u(v_{t-1})$ (resultant from the previous iteration):

$$u_t(v) = min(u(v_t), u(v_{t-1})) \tag{5.9}$$

It is worth mentioning that, although Figure 5.2 only shows occupied voxels, the uncertainty map is also capable of storing information about free points.

5.3.2 Camera Pose Optimisation

The camera pose optimisation algorithm is in charge of estimating the optimal viewpoint of the camera for a certain state of the uncertainty map. Given a partially completed map and a position on the mobile platform, the optimal pose of the sensor is calculated. The final objective is to maximise the knowledge about the environment. Mapping new areas and improving the reliance of already acquired data can both be considered as increasing the knowledge about the environment, and the uncertainty do represent both aspects. Thus, we can assume that reducing the uncertainty of the map implies progressing towards the final objective. It should be noted that the sensor and map used for this task may also contribute to the estimation of the global localisation. For this reason, the optimisation algorithm not only focuses on exploration, but also selects viewpoints to observe already mapped areas that may contribute to maintain a low localisation uncertainty.

As the system has no prior knowledge about the environment being mapped, it only stores information about the occupancy value of visited points (nodes), but not about the quality of the exploration itself. So, a new measure is needed to assess if a map is better than another: information.

Information attempts to quantify the amount of knowledge we have about an environment, and it is calculated with the uncertainty value we have already estimated. As Equation 5.8 maps the inputs into the range (0, 1), it is consistent to say that the uncertainty value that would correspond to unknown cells is 1. Accordingly, their information value can be quantified as 0. Since the information value cannot be negative, the result from the information function i must be inversely proportional to the result of the uncertainty function u. Besides, a cell with uncertainty 0 corresponds to the maximum information. Thus, the information function i must satisfy the following statements:

$$\forall v \in M : i(v) \geq 0$$

$$\forall v, w \in M : \{u(v) \geq u(w) \implies i(v) \leq i(w)\}$$

$$\forall v, w \in M : \{u(v) = 0 \implies \nexists w \mid i(w) > i(v)\}$$

That being said, the result of the information function i is a growing function from 0 to a maximum, as the uncertainty for the same element decreases from 1 to 0. Taking this into account, the proposed information function i for a single cell v is:

$$i(v) = k - u(v) \mid k > 1 \tag{5.10}$$

k must be greater than 1 to avoid having a negative information value. Considering what information represents, we propose calculating the information of a set of cells or map simply adding the information value of each of the cells inside it. Besides, we present a function whose maximum is the least possible maximum, because greater values would favour the amount of cells in the map. We consider that, at this point, the uncertainty of each cell itself should receive more attention, and that the system can be biased towards exploration in the next-best view selection process. Hence, the information value of a set A is calculated according to Equation 5.11.

$$I(A) = \sum_{v \in A} (1 - u(v)) \tag{5.11}$$

The most relevant aspect about the information value of a map is that it allows the comparison between two maps of the same environment in different phases. At any moment t, the system has a particular map M_t built and the camera has a specific pose q_t. The objective is to move the sensor to a pose q_{t+1} such that the map M_{t+1} provides the maximum information possible based on the state of the system in the previous step t. In other words, the pose optimisation algorithm has to maximise the information gain from one iteration to another, moving the sensor accordingly. In addition, the information value I obtained from Equation 5.11 can also be used to compare maps that, for example, have been built with different approaches.

The challenge here is how to predict the information gain of an unknown point. If a point is already mapped and, thus, its x, y, z coordinates are known, if that point would fall in the sensor's FOV for any of its possible poses can be calculated, together with the exact position at which that point would be seen. Besides, in most of the environments, it is done with a very high accuracy. Using these inputs in the functions presented above, the system can predict how the uncertainty value stored in each cell would be updated and the information gain it would provide, if any. In this case, the information gain of that cell $g(v)$ would be the difference between its actual information value $i(v_t)$ and the hypothetical future value $i(v_{t+1})$. However, when the camera's FOV includes unmapped points, the future estimated information value cannot be calculated mathematically. It is impossible to ensure where the ray would find an object, adding the corresponding cell, and setting it as occupied. To tackle this problem, we propose a "positive" approach, as we imagine that every unknown point will be occupied. The information provided by an unknown point is quantified as 0, i.e., $i(v_t) = 0$, and, with the previous assumption, its information in the potential next iteration $i(v_{t+1})$ is calculated as for any already mapped cell. In summary, the information gain of an unknown point is equal to the information value it has when it is occupied, which as mathematically expressed as in Equation 5.12.

$$g(v) = \begin{cases} i(v_{t+1}) - i(v_t) & \text{if } i(v_{t+1}) > i(v_t) \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

Again, being coherent with Equation 5.11, the information gain of a set of cells A is the sum of the gain of every element inside it:

$$G(A) = \sum_{v \in A} g(v) \tag{5.13}$$

Once the information gain provided by any future camera pose can be estimated, it is possible to optimise it. The optimisation algorithm simulates the FOV of each pose q of all the viable poses Q of the sensor, calculating the information gain for each of them. Then, the best one is selected and the sensor is moved accordingly. Taking all calculations into account, the pseudocode of the whole sensor pose optimisation process is shown in Algorithm 2.

The camera optimisation process proposed has two key functions that form the core of the algorithm, as they determine the poses to be considered in each iteration and the value - or ranking - assigned to each of them, in order to finally choose only one, the one to be adopted by the sensor. These functions are $GetPossiblePoses$ and $EstimateGain$ (Line 3 and 7 of Algorithm 2, respectively).

GetPossiblePoses The potential optimisation poses in each iteration are calculated based on: the sensor's current position, $q_{current}$; its degrees of freedom (DOF), ranges and velocity limits, $SensorDOF$; and a t value, which determines the maximum time the sensor can be in motion between one pose and another, i.e., the maximum

Algorithm 2 Camera pose optimisation

```
Require: $M$ : 3D uncertainty map
Require: $SensorFOV$ : Sensor's FOV specification
Require: $SensorDOF$ : Sensor's movement's degrees of freedom (DOF), ranges
    and velocities
 1: while True do
 2:     $q_{current} = GetCurrentPose()$
 3:     $Q = GetPossiblePoses(q_{current},\ SensorDOF,\ t)$
 4:     $g_{max} = 0$
 5:     $q_{opt} = q_{current}$
 6:     for each $q \in Q$ do
 7:         $g_q = EstimateGain(q,\ SensorFOV,\ M)$
 8:         if $g_q > g_{max}$ then
 9:             $g_{max} = g_q$
10:             $q_{opt} = q$
11:         end if
12:     end for
13:     if $q_{opt} \neq q_{current}$ then
14:         $MoveSensor(q_{opt})$
15:     end if
16: end while
```

time it can take to reach the estimated optimal pose. While the first two parameters are strictly necessary to estimate new poses, the t parameter is added for two main reasons. First, since the platform continues navigating while the sensor motion calculations are performed, the sensor pose selected with respect to the world and the one reached may differ. This difference will be bigger the longer the time between sensor poses and the higher the speed of the platform. Thus, a balance between these two values must be maintained. Second, larger values of t imply more possible poses to consider, leading to increased computation time. Note that the elapsed time between consecutive sensor poses is the sum of the computation time plus the sensor movement time, and the t value affects both, contributing to reduce the difference between the initial state of the environment and the final state of each iteration. In this line, there is also a configuration parameter q_{step} that determines the step between poses, which serves not only to discretise the sensor positions, but also to modify the number of them and, consequently, the computational cost of the optimisation. The pseudocode of the complete procedure is shown in Algorithm 3.

EstimateGain Each of the candidate sensor poses Q must be evaluated in order to estimate the optimal one. To this end, what the camera would capture in each of the poses is predicted using a ray-tracing technique. Based on the FOV, range and

Algorithm 3 GetPossiblePoses

```
Require: $q$ : Relative pose of the sensor
Require: $SensorDOF$ : Sensor's movement's degrees of freedom (DOF), ranges
    and velocities
Require: $t$ : Maximum movement time (0 = no limit)
 1: $Q = \{q\}$
 2: if $t = 0$ then
 3:     $q_{ranges} = SensorDOF_{ranges}$
 4: else
 5:     $q_{ranges} = GetRanges(q, SensorDOF, t)$
 6: end if
 7: $n = GetNumberOfPoses(q_{ranges}, q_{step})$
 8: for $i$ in $[1..n]$ do
 9:     $q_{candidate} = GetPose(q_{current}, q_{step}, i)$
10:     add $q_{candidate}$ to $Q$
11: end for
12: return Q
```

resolution of the sensor, a virtual ray is cast from the pose of the sensor through each pixel to determine what is visible along the ray in the 3D scene. If the ray reaches the maximum range without hitting any non-free voxel, a value of 0 gain is assigned to it. On the contrary, if it hits a non-free voxel, either occupied (mapped) or unknown (unmapped), the corresponding gain is calculated according to Equation 5.12. Then, the values of the rays with the same FOV are summed up to get the information gain of the corresponding sensor pose. Repeating this procedure for all the candidate poses, as in Algorithm 2, the optimal pose is obtained, which is sent to the sensor motion module. In principle, the optimisation algorithm has no termination criteria, as it is intended to run in parallel and independently of the rest of the mobile robot system.

5.4 Description of the Implemented System

Our algorithm is implemented in the well-known Robot Operating System (ROS) framework [185], which is accessible to many other developers and users. Besides, it provides the tools to integrate our functionalities into a robotic platform.

For the experiments described hereunder, we have integrated our algorithm in a modified Turtlebot3 Waffle platform. It is a small omnidirectional wheeled robot with a fixed RGB-D camera at the front and a lidar in the centre with 360º vision. To test our camera optimisation algorithm correctly, a telescopic arm with a RGB-D camera on top of it has been added to the platform, which makes possible to pan, tilt, and move the sensor along the vertical axis independently, as shown in Figure 5.3. However, as the objective is to see the viability of the proposed approach, in these first trials, the optimisation has been limited to the pan movement, i.e., one degree of freedom.

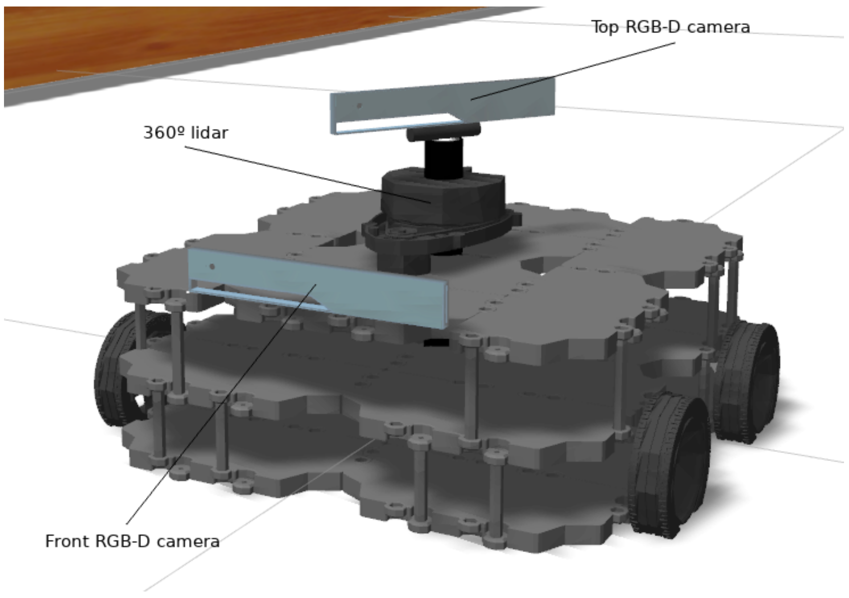


Fig. 5.3. – Turtlebot3 Waffle platform with a second RGB-D mobile camera added on top of it. In the image, it is rotated 60º left, as it has a 360º pan range.

As our sensor pose optimisation algorithm attempts to be modular and generic without being restricted to a certain type of robot, it does not control the motion of the platform, and publicly available ROS packages are used for this purpose. Our 3D exploration is combined with a package that implements an RRT-based 2D exploration algorithm [242], which finds exploration goals in the horizontal plane and sends them to the navigation system. These destinations are managed by the ROS Navigation Stack, configured for obstacle-free trajectory planning and execution. Thus, while a mobile robot exploration system controls the motion of the platform and builds a 2D map, our system optimises the pose of the camera on the fly to build a dense 3D map. Although both systems run in parallel, they are independent and there is no communication between them.

With this configuration, where the camera pose optimisation algorithm does not take into account the motion of the robot for the camera poses, an adequate t value must be set as it has a significant impact. Essentially, it reduces the main issues of the lack of coordination between platform and camera motion. Since the sensor is attached to the platform, when the latter moves, the former moves implicitly in the same direction. Consequently, the sensor is not in the global pose the optimisation algorithm decided to move to. This effect may be exacerbated when the platform moves faster or the system needs more time to estimate the optimal camera pose. Setting a low t value makes the transformations between consecutive poses smaller and their overlapping regions larger, which eases the 3D map merging [130]. If the same camera is used for localisation, it also improves position estimation [174].

As described before, the uncertainty map simply gets the colour image and the point cloud provided by the top RGB-D camera and creates a 3D octomap accordingly. The implementation has been done on the standard octomap, and a detailed explanation can be found in the original paper [94]. Apart from occupancy and colour, our version adds the possibility to store uncertainty values with the functions described above. We have also implemented the corresponding Rviz plugins to visualise these values[1].

In brief, the implementation for testing consists of three independent systems that live together in the same mobile robot: 2D mapping, uncertainty 3D mapping and navigation. They gather information of the environment from different sensors and may communicate with each other using ROS. Besides, they can send commands to the platform and control its motion, which closes the cycle and enables a fully autonomous behaviour. Figure 5.4 shows a general overview of the logic of the system and its communications.

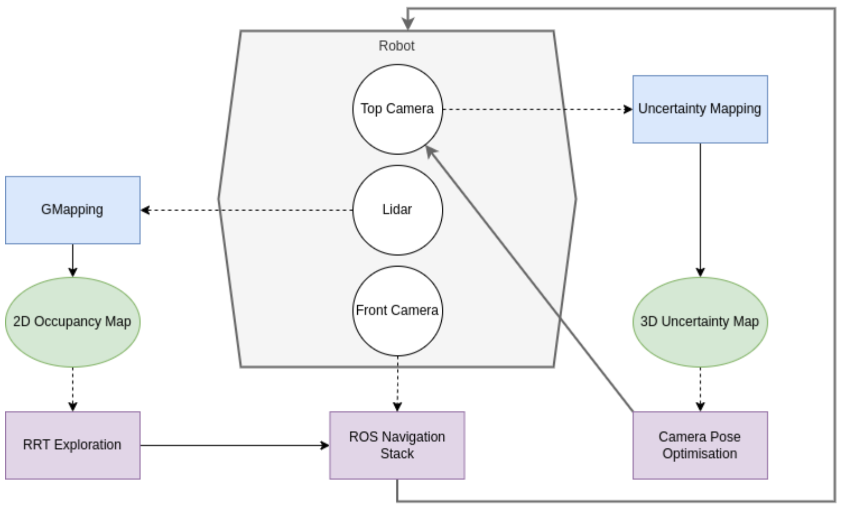


Fig. 5.4. – Complete ASLAM system's diagram implemented in the mobile robot.

The experimental method followed is slightly different in simulation and in real-world tests. Moreover, the data obtained in simulation are not always extrapolable to

[1]The code of the optimisation algorithm and the octomap related functionalities is available online here: `https://github.com/fundaciontekniker/aslam-system`

a real scenario, as the uncertainties from the sensors, for example, vary greatly. Thus, the experiments performed and their results are shown separately in the following two sections.

5.5 Evaluation in simulation

The system has been tested in diverse simulated and real environments where the size, the number of elements and the level of detail of these vary.

The models used for the experiments in simulation are the following:

- "simple house". It is provided by the Robotis group in the official manual of the TurtleBot3 platform, in the simulation section (`https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/`). It contains six rooms, with a few basic elements, covering a total area of 15 x 10 m. It has been chosen for its simplicity.

- "apartment". It is also crated by the AWS Robotics team, and it is a very detailed and realistic representation of an apartment of 19 x 11 m. It is available here: `https://github.com/aws-robotics/aws-robomaker-small-house-world`. It offers a good idea of how the mapping system would perform in a domestic environment.

- "bookstore". It is a 3D model of a bookstore developed by the AWS Robotics team (see Figure 5.5). It has many and very detailed elements, such as furniture and books, which makes it a suitable environment to test a 3D mapping system. It can be acquired from: `https://github.com/aws-robotics/aws-robomaker-bookstore-world`. The size of the bookstore is 15 x 14 m.

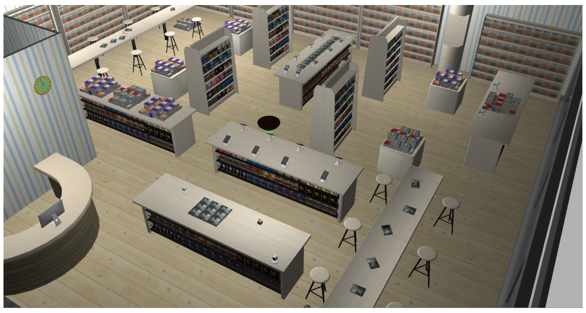


Fig. 5.5. – "bookstore" test environment.

- "cafe". It is a spacious area of 25 x 10 m, with detailed elements along the walls but just a few tables in the middle. This configuration allows the robot to build the 2D map without navigating too much, because the lidar covers nearly all the area from any location. As the FoV of the camera has a reduced scope, the objective of the tests in this environment is to check if our system makes a significant difference in the resulting 3D map in this kind of scenarios. Indeed, the range of lidar for 2D mapping is set to 10 m with a 360º view, while the camera has a 4 m range and a horizontal angle of 59º. Besides, two people are walking through the cafe, adding dynamism to the problem. Here, the behaviour of the system is also evaluated in the presence of dynamic elements. The "cafe" model can be obtained from: `https://automaticaddison.com/how-to-load-a-world-file-into-gazebo-ros-2/`.

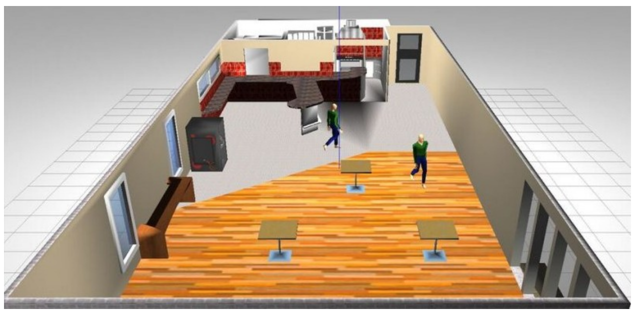


Fig. 5.6. – "cafe" test environment.

- "warehouse". It is the usual warehouse of the majority of the factories nowadays. It is another model created by the AWS Robotics team (`https://github.com/aws-robotics/aws-robomaker-small-warehouse-world`), and it is useful to make an idea of the result the system may obtain in an industrial use case. Its dimensions are 14 x 21 m.

- "willowgarage". It is a model available in Gazebo simulator itself by default, already used in many research works. It represents a set of rooms in an office-like area without objects, just walls (see Figure 5.7). It is the largest environment tested, with an area of 65 x 45 m. As our algorithm solves a local optimisation problem, the size of the environment does not increase the complexity of each iteration. But, it does the size of the 3D uncertainty map being built. It is relevant to test our system in such a large area to see if the results obtained in small environments are extrapolable.

Note that all the models used in the tests presented here are publicly available and compatible with the Gazebo simulator. Besides, they are all indoor environments where the mobile platform navigates through a unique flat floor.

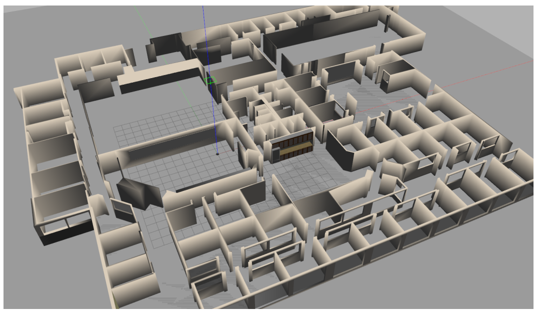


Fig. 5.7. – "willowgarage" test environment.

In all the environments, the RGB-D sensor simulated is similar to the Inter RealSense R200 camera, with a vertical FOV of 46º, horizontal FOV of 59º, a maximum range of 4 m and a 480x360 depth resolution. On the other hand, the 360º lidar is simulated with a range of 100 m and 0.5º of horizontal resolution, which provides 720 points in each scan. Nevertheless, the 2D mapping system is limited to a range of 4 m in all the environments but in "willowgarage", where it is limited to 10 m. Larger ranges increase the computational cost of the 2D exploration system, however, 4 m is too low for this environment, because many times, the sensor does not find any obstacles, the localisation has nothing to match, and the robot gets lost. This beam cropping is done for efficiency purposes, as there are no such large free obstacle distances in the selected scenarios.

5.5.1 Experimental Procedure

The system has been compared with two different exploration methods from the literature, one based on 2D [242] and another one that considers the complete 3D space [11, 13, 12]. Both algorithms are based on RRT mainly because it is one of the best performing techniques in exploration. We believe that, by using methods with the same basis, our algorithm gets more isolated and its effect is more clearly observable. The tests are as follows:

- **2D**. In each environment, the 2D autonomous exploration is performed first using a strategy based on the use of multiple RRTs, proposed by Umari and Mukhopadhyay[2]. It calculates the trajectories to explore the entire area and builds the corresponding map using a 360º lidar. During this phase, the RGB-D camera builds a 3D uncertainty map, while the robot moves through the environment keeping the camera pose constant. While exploration is performed, the entire trajectory of the robot is stored in order to replicate it in

[2]https://github.com/hasauino/rrt_exploration

subsequent tests. In this way, the robot can keep the same pose and velocities in the same timestamps among different tests of the same environment. This allows to compare under equal conditions this first 3D map with results where the 3D camera pose optimisation is performed.

- **2D+optimisation**. After a 2D exploration where a 3D map is built keeping the camera static, the same process is repeated with our pose optimisation algorithm running. As the trajectory executed in these tests is the one recorded in the previous case, the time needed to perform the exploration is exactly the same. However, as the pose of the camera with respect to the robot is continuously modified, the 3D map obtained differs. This type of test has been performed with three different configurations, where the t value varies between 0.2 s, 0.5 s and 1 s.
- **3D**. Finally, the algorithm proposed by the Robust Field Autonomy Lab at Stevens Institute of Technology [3] that performs complete 3D exploration of complex environments by means of a ground mobile robot is executed in the same environment. As this system calculates the trajectories of the mobile platform itself to perform the 3D exploration, its positions may be different from those achieved in the rest of the tests. In this last case, as in the first 2D exploration, the camera is static during execution.

In all tests, the 2D map for trajectory planning and navigation is built using OpenSLAM's GMapping algorithm, which is a Rao-Blackwellized particle filer to learn grid maps from laser range data [81]. These five procedures - one 2D exploration, three 2D exploration with our optimisation, one 3D exploration - are repeated three times in all the six environments described above, with three different starting points in each of them, making a total of 90 experiments. At the end of each of them, the time needed, the number of voxels of the 3D map and the total information provided by this one are calculated. The numbers shown in this section are average values.

Table 5.1 shows the comparison of 3D mapping results with the three different systems explained before. As can be seen, there is an improvement in the obtained 3D map by performing a 2D exploration with our optimisation algorithm compared to not using it. Our approach has been tested with three different values of t, and all three configurations help to build a denser and richer 3D model of any of the environments. Note that the value t limits the maximum time the sensor has to move from one pose to the next in each iteration. Thus, greater values evaluate more pose candidates, increasing the optimisation possibilities, at the expense of higher computational cost. For this reason, larger values t generally get slightly

[3]`https://github.com/RobustFieldAutonomyLab/turtlebot_exploration_3d`

Tab. 5.1. – Comparison of the effectiveness of the obtained 3D maps considering all simulation tests. The metrics shown are: number of voxels, increase in the number of voxels compared to the 2D test in percentage terms, information provided, information gain compared to the 2D test, information gain compared to 2D test in percentage terms.

ENVIRONMENT	TEST TYPE	VOXELS	VOXELS %	INFO	GAIN	GAIN %
simple house (15 x 10 m)	2D	63,894		38,898		
	2D+optimisation (t=0.2)	73,291	14.71%	42,705	3,808	9.79%
	2D+optimisation (t=0.5)	78,217	22.42%	46,386	7,489	19.25%
	2D+optimisation (t=1.0)	77,007	20.52%	45,984	7,086	18.22%
	3D	**79,248**	**24.03%**	**46,967**	**8,070**	**20.75%**
apartment (19 x 11 m)	2D	86,020		45,664		
	2D+optimisation (t=0.2)	113,218	31.62%	60,465	14,801	32.41%
	2D+optimisation (t=0.5)	118,294	37.52%	63,442	17,778	38.93%
	2D+optimisation (t=1.0)	**131,533**	**52.91%**	**70,747**	**25,083**	**54.93%**
	3D	106,592	23.91%	68,271	22,607	49.51%
bookstore (15 x 14 m)	2D	114,651		70,770		
	2D+optimisation (t=0.2)	130,635	13.94%	79,911	9,141	12.92%
	2D+optimisation (t=0.5)	137,217	19.68%	84,014	13,244	18.71%
	2D+optimisation (t=1.0)	139,970	22.08%	85,238	14,467	20.44%
	3D	**153,125**	**33.56%**	**91,666**	**20,896**	**29.53%**
cafe (25 x 10 m)	2D	69,566		35,585		
	2D+optimisation (t=0.2)	90,968	30.77%	48,345	12,759	35.86%
	2D+optimisation (t=0.5)	103,998	49.50%	55,636	20,051	56.35%
	2D+optimisation (t=1.0)	102,414	47.22%	56,058	20,473	57.53%
	3D	**136,490**	**96.20%**	**74,196**	**38,611**	**108.50%**
bookstore (14 x 21 m)	2D	158,786		85,558		
	2D+optimisation (t=0.2)	177,206	11.60%	95,257	9,700	11.34%
	2D+optimisation (t=0.5)	**193,576**	**21.91%**	104,291	18,734	21.90%
	2D+optimisation (t=1.0)	190,987	20.28%	104,012	18,454	21.57%
	3D	186,610	17.52%	**107,573**	**22,015**	**25.73%**
bookstore (65 x 45 m)	2D	835,622		461,921		
	2D+optimisation (t=0.2)	1,045,753	25.15%	556,748	94,827	20.53%
	2D+optimisation (t=0.5)	1,129,001	35.11%	610,689	148,767	32.21%
	2D+optimisation (t=1.0)	1,144,487	36.96%	617,163	155,242	33.61%
	3D	**1,371,774**	**64.16%**	**762,823**	**300,902**	**65.14%**

better results in the tests. Although, based on the experiments, the optimal t value seems to be relative to the environment being explored.

Tab. 5.2. – Efficiency comparison of the obtained 3D maps considering all simulation tests. The metrics shown are: duration of the exploration, increase in the time needed compared to 2D test in percentage terms, average number of mapped voxels per second, average information mapped per second.

ENVIRONMENT	TEST TYPE	DURATION (s)	TIME %	VOXEL / S	INFO / S
simple house (15 x 10 m)	2D	431		148	90
	2D+optimisation (t=0.2)	431	0.00%	170	99
	2D+optimisation (t=0.5)	431	0.00%	**181**	**108**
	2D+optimisation (t=1.0)	431	0.00%	179	107
	3D	1,668	287.01%	48	28
apartment (19 x 11 m)	2D	378		228	121
	2D+optimisation (t=0.2)	378	0.00%	300	160
	2D+optimisation (t=0.5)	378	0.00%	313	168
	2D+optimisation (t=1.0)	378	0.00%	**348**	**187**
	3D	3,018	698.32%	35	23
bookstore (15 x 10 m)	2D	497		231	142
	2D+optimisation (t=0.2)	497	0.00%	263	161
	2D+optimisation (t=0.5)	497	0.00%	276	169
	2D+optimisation (t=1.0)	497	0.00%	**282**	**172**
	3D	2,041	310.66%	75	45
cafe (25 x 10 m)	2D	227		306	157
	2D+optimisation (t=0.2)	227	0.00%	400	213
	2D+optimisation (t=0.5)	227	0.00%	**457**	245
	2D+optimisation (t=1.0)	227	0.00%	451	**247**
	3D	2,274	900.44%	60	33
warehouse (14 x 21 m)	2D	446		356	192
	2D+optimisation (t=0.2)	446	0.00%	398	214
	2D+optimisation (t=0.5)	446	0.00%	**434**	**234**
	2D+optimisation (t=1.0)	446	0.00%	429	233
	3D	4,477	904.49%	42	24
willowgarage (65 x 45 m)	2D	1,615		517	286
	2D+optimisation (t=0.2)	1,615	0.00%	647	345
	2D+optimisation (t=0.5)	1,615	0.00%	699	378
	2D+optimisation (t=1.0)	1,615	0.00%	**709**	**382**
	3D	40,271	2393.05%	34	19

Surprisingly, in some cases, the combination of a 2D exploration with our camera pose optimisation algorithm gets better results than the 3D exploration system. This is the case of the tests carried out in "warehouse" and "apartment" environments, obtaining a map with more voxels in the former and one with both more voxels and information in the latter (see Figure 5.9). It is worth mentioning that in some tests, the 3D exploration algorithm obtains worse result than even the basic 2D exploration because it does not visit some narrow areas where the 2D algorithm does and, indeed, the robot is able to map them. On the contrary, in the "cafe" environment, the 2D algorithm is able to map the entire environment with a short trajectory. As a consequence, the camera can only gather information from a small proportion of the environment. As the 3D exploration system calculates its own trajectory for this purpose, it maps the whole area and there is a huge improvement in the resulting map. Figure 5.8 shows the 3D uncertainty maps obtained with the three different systems in the "cafe" test environment.

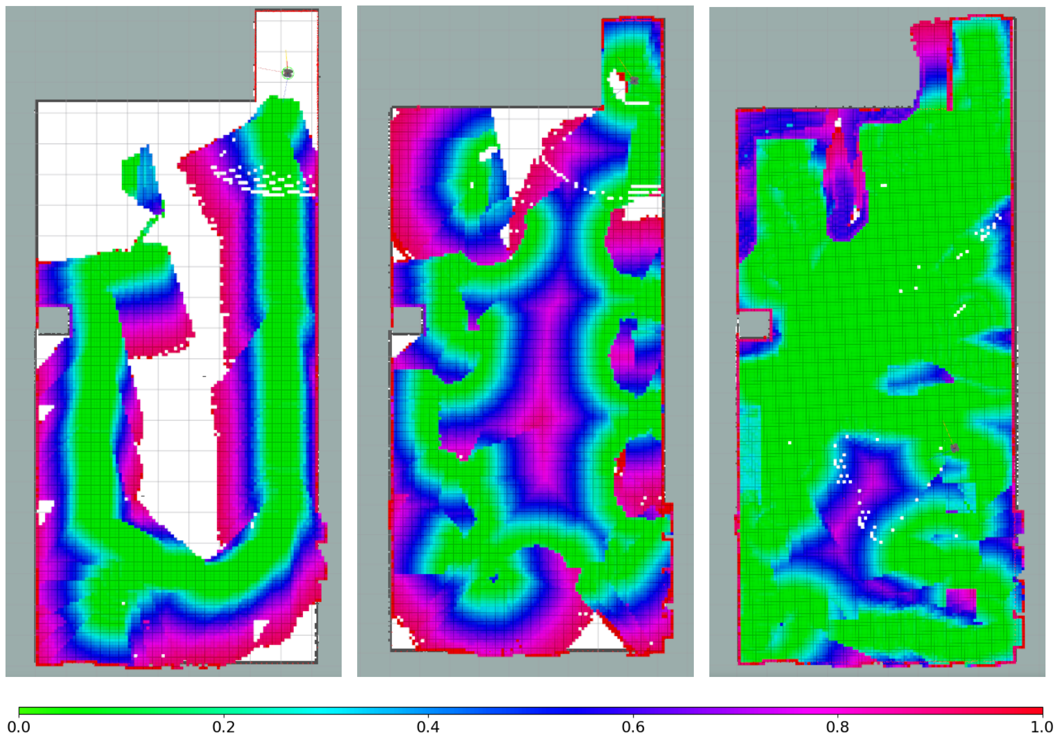


Fig. 5.8. – "cafe" environment mapping result after 2D basic exploration (left), combination of 2D exploration and camera pose optimisation with $t = 0.5$ (middle) and 3D exploration (right). The 3D uncertainty map is shown with colours, while the 2D map is in greyscale. A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

In 5 of the 6 environments, the map with the largest amount of information is achieved with the 3D scanning system, which is quite logical as it is the only procedure of those evaluated here that focuses entirely on that purpose. While the resulting model may differ from one scenario to another, the 3D exploration needs more time to complete the mapping process in all the tests, with an average duration of ten times the duration of the other exploration processes. So, although this system achieves the best result in most cases, the difference is not proportional to the time needed to do so. In this aspect, the combination of 2D exploration with our camera pose optimisation algorithm is the most efficient in all cases, regardless of the value *t* used. Table 5.2 shows how this procedure obtains the best ratio between both the number of voxels and the information value of the resulting model and the time

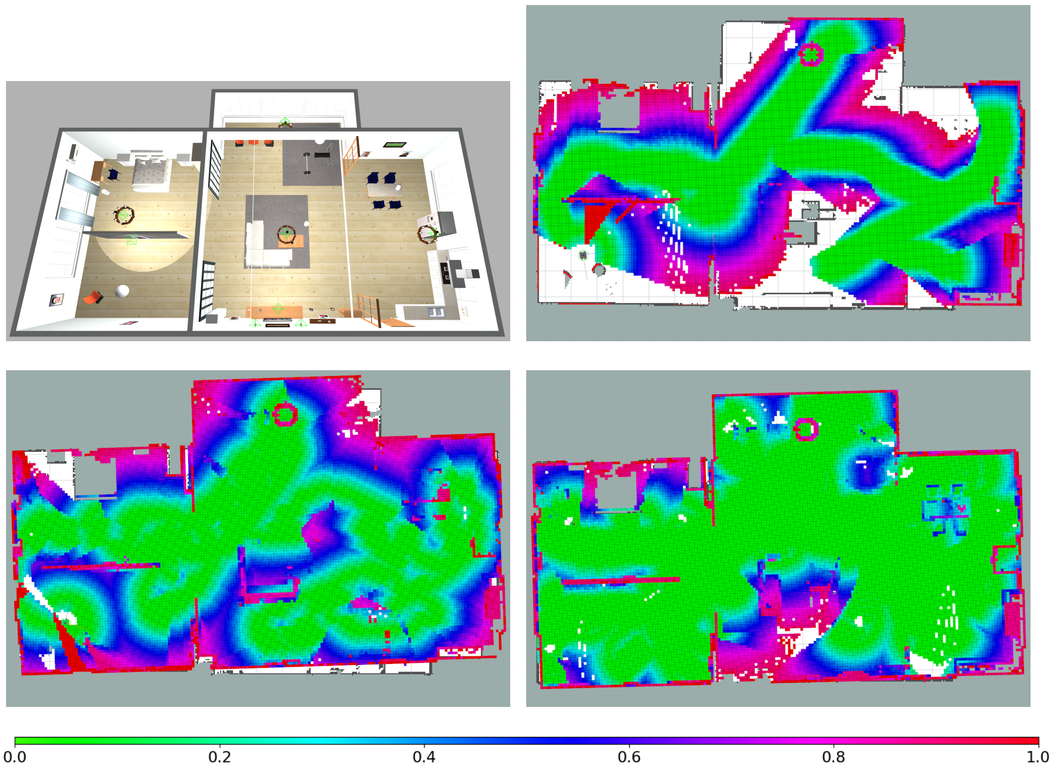
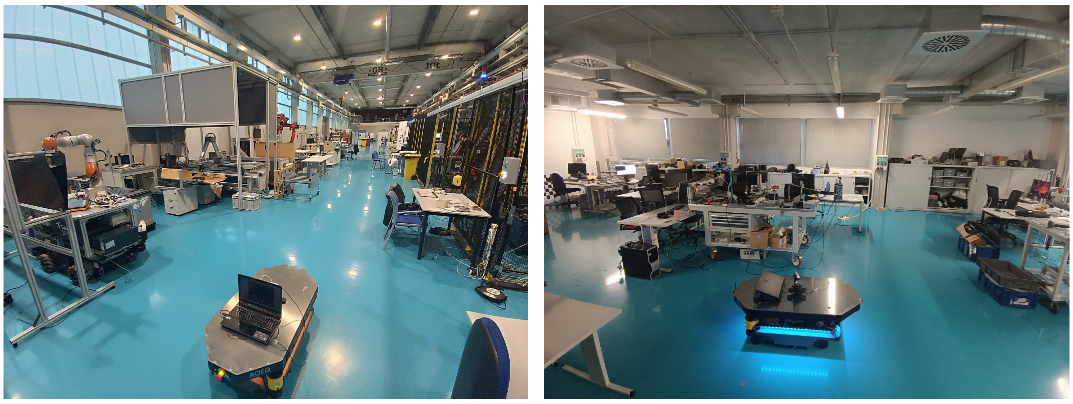


Fig. 5.9. – "apartment" environment (top left), mapping result after 2D basic exploration (top right), combination of 2D exploration and camera pose optimisation with $t = 0.5$ (bottom left) and 3D exploration (bottom right). The 3D uncertainty map is shown with colours, while the 2D map is in greyscale. A reduced HSV colour gradient is used to represent uncertainty values, where the most uncertain voxels are in red and the most reliable ones in green.

needed to build it. More specifically, in the tests carried out, *t* values of 0.5 or 1.0 are those that achieve the optimal result.

5.6 Evaluation in a real environment

With respect to the real-world experiments, the system has been tested in two different environments, *workfloor* and *lab*, the first one representing an uncluttered environment and the latter a cluttered one. A picture of each of them can be seen in Figure 5.10.



(a) (b)

Fig. 5.10. – Real scenarios in which the system has been tested. On the left, the *workfloor* environment, which is 20 x 10 m big and relatively uncluttered. On the right, the *lab* scenario, whose size is 15 x 8 m and represents a cluttered environment. The mobile platform used is shown at the bottom centre of both images.

Here, the mobile robot used is slightly different for that used in simulation. In this case, we have attached an Intel RealSense D435 RGB-D camera to a turret that enables pan movement and mounted this module on top of a MiR200 platform. However, the rest of the hardware components, sensor ranges, software and configurations are almost identical.

5.6.1 Experimental Procedure

The real-word performance of the algorithm has been tested following a strategy similar to that used in simulation. First, the map is built teleoperating the robot through the environment, ensuring the 360º lidar captures correctly all the elements of interest and the 2D map obtained with GMapping has no reachable unknown areas. In half of the experiments, at the end of the process, the robot is guided to the starting point and the first metres of the mapping trajectory are repeated, to ease the loop closure. But there is one thing to be aware of with this strategy. Most exploration algorithms do not return to the starting point and do not repeat these first movements at the end of the process, as is the case for the algorithm used to define the robot's trajectory in simulation. So, in the other half of the tests, the robot has travelled the minimum distance to cover the area to be mapped, without returning to the starting point before finishing. We may refer to it as a *one-way* trajectory. As in the experiments described in Section 5.5.1, during this first mapping process the 3D mapping camera is static with respect to the mobile platform, as the optimisation algorithm is not running. In addition, the entire trajectory is recorded, so that the robot performs exactly the same movements on subsequent runs that do have our proposed pose optimisation algorithm active.

In this case, no 2D exploration algorithm is used to build the map in this first step and the robot is manually guided to increase the speed and safety of the experimentation. Similarly, no 3D exploration method has been added to the evaluation for the same reason, but also because the mapping system is running on an MSI GL65 9SEK laptop whose battery lasts about 30 minutes with everything running, and tested 3D ASLAM methods require more time. Nevertheless, the tests we propose to perform on real scenarios are valid enough to evaluate the difference in the resulting 3D model between using our optimisation process and not using it.

As in the simulation experiments, each of the four configurations - one teleoperated 2D mapping, three with our optimisation - is repeated six times in each environment, *workfloor* and *lab*. Out of these six runs, three of them try to simplify loop closure detection, while the other three perform a *one-way* trajectory. Altogether, 48 runs have been conducted, the results of which can be seen in Table 5.3, where data from experiments where only the starting point varies have been averaged.

In both real-world scenarios, the map with the highest number of voxels and information is always the one obtained using our camera pose optimisation algorithm, achieving improvements of up to 131.67% and 142.48%, respectively. Regarding the optimal t-value, it is again specific to each mapping process, as all three values achieve the best result in some of the tests, being $t = 0.5$ the only one to be optimal twice and never obtaining the worst result. It should be noted that in the *lab* environment, $t = 0.2$ even gets a worse result than keeping the camera static. This configuration is prone to fall to a local minima, not taking advantage of the full range of the sensor's panning action. Moreover, this usually occurs at values between $\pm$ 180º and $\pm$ 120º, i.e., looking almost backwards in the direction of travel.

Tab. 5.3. – Comparison of the effectiveness of the obtained 3D maps considering all real-world tests. The metrics shown are: number of voxels, increase in the number of voxels compared to the 2D test in percentage terms, information provided, information gain compared to the 2D test, information gain compared to 2D test in percentage terms.

ENVIRONMENT	TEST TYPE	VOXELS	VOXELS %	INFO	GAIN	GAIN %
Workfloor (20 x 10 m)	2D	40,553		16,209		
	2D+optimisation (t=0.2)	50,024	23.35%	20,811	4,601	28.39%
	2D+optimisation (t=0.5)	**60,000**	**47.96%**	**25,592**	**9,383**	**57.89%**
	2D+optimisation (t=1.0)	49,894	23.04%	22,076	5,867	36.20%
Workfloor (20 x 10 m) (one-way)	2D	14,510		5,727		
	2D+optimisation (t=0.2)	**33,616**	**131.67%**	**13,887**	**8,160**	**142.48%**
	2D+optimisation (t=0.5)	31,001	113.65%	12,933	7,206	125.83%
	2D+optimisation (t=1.0)	27,785	91.49%	11,586	5,859	102.30%
Lab (15 x 8 m)	2D	42,289		17,922		
	2D+optimisation (t=0.2)	40,784	-3.56%	17,063	-859	-4.79%
	2D+optimisation (t=0.5)	43,788	3.54%	18,763	841	4.70%
	2D+optimisation (t=1.0)	**47,829**	**13.10%**	**21,006**	**3,084**	**17.21%**
Lab (15 x 8 m) (one-way)	2D	21,507		8,349		
	2D+optimisation (t=0.2)	20,754	-3.50%	8,398	49	0.59%
	2D+optimisation (t=0.5)	**30,325**	**41.00%**	**13,107**	**4,758**	**56.99%**
	2D+optimisation (t=1.0)	28,755	33.70%	12,542	4,193	50.22%

Another point to highlight is the difference in the improvement of our algorithm between repeating the first movements before the end of the mapping process and the *one-way* trajectories. For instance, information gain increases from 58.89% to 142.48% in *workfloor*, and from 17.21% to 56.99% in *lab*.

5.7 Conclusions

In this chapter, we presented a system that associates uncertainty values with the mapped points based on where they have been captured, and an algorithm that exploits this information to optimise the sensor pose for exploration. We use an octree-based 3D uncertainty map to estimate the camera's NBV online, which is built during motion. Adding this algorithm to a 2D exploration, a denser and richer model is acquired, without requiring additional time for it. Besides, in some cases, it has demonstrated to obtain similar or even better results to a complete 3D exploration system, but in much less time.

It is obvious that an ASLAM approach significantly reduces the human effort required to operate an autonomous navigation system in any indoor environment, as the map generation is automated. But, furthermore, it can also help to improve the accuracy and robustness of navigation. 3D models provide a greater amount of information than 2D maps, and can be very useful in resolving ambiguities, so common in human-made structures. The proposed method generates a partial 3D map of the environment, which can also be used for localisation, performing both processes without spending additional time. Similarly, the 3D model can be updated and/or completed during the autonomous navigation of the robot. It could be used both as a continuous, stand-alone 3D exploration system and as a *more confident* localisation system. The generated model contains information about the uncertainty of each spatial point, potentially allowing to focus the field of view on the *less uncertain* area, which *a priori* leads to a more reliable matching and a more accurate localisation. However, these theories need to be tested empirically.

Although the proposed method has been designed and tested on a terrestrial mobile platform, the algorithm can be extrapolated to any aerial or underwater vehicle, with whatever degrees of freedom to be optimised, making it a highly flexible solution.

Part III

Conclusions and Further Research

Conclusions and Future Work 6

Don't cry because it's over, smile because it happened.

— **Dr. Seuss**

This chapter summarises the work done throughout the research process, outlines the main conclusions drawn from it, and proposes some possible future steps and open questions that have yet to be solved.

6.1 Summary and Conclusions

As our particular main objective in mobile robotics is to strengthen, simplify, automate and, in general, improve the setting up and operation of mobile platforms in real indoor applications, it is necessary to know the current state of the art of the corresponding technologies. For this reason, the first of the three branches into which the work done in this research process is grouped consists of empirically assessing the capabilities and limitations of existing algorithms in the literature.

We have started by experimentally evaluating three known SLAM algorithms. The results obtained in these first tests prove that the trajectory followed by the robot during the mapping process has a direct influence on the morphology of the generated map. Likewise, even the smallest difference in the map makes the subsequent localisation estimation in that area differ when using one model or another. This effect is quite logical, and it is not a new discovery, but verifying it in our environment is relevant for the proposed improvements. What is an added value of our experimentation is the fact that the error made in the position estimation with each of the three algorithms is methodically calculated. By means of an external micrometric measuring device and the execution of multiple runs, the accuracy of this location has been quantified, which turns out not to be sufficient for the performance of certain tasks, e.g., inspection tasks.

The second branch builds on the limitations observed in the experimental evaluation part, and proposes two different methods to improve the localisation accuracy. On the one hand, we propose a vision-based method to improve the positioning of a mobile platform in an indoor environment. It can achieve an accuracy of up to 0.1 mm, validated with the TRITOP photogrammetric unit. However, when this value is extrapolated to the platform, the error increases up to 6 mm and 0.12º, as the platform control itself has a higher tolerance. Even so, an improvement of approximately 20 cm is achieved, increasing the range of possibilities considerably. Even so, an improvement of approximately 20 cm is achieved, a considerable step forward that enables some tasks to be performed correctly. The system is validated by navigating autonomously through an industrial plant before positioning precisely in front of an aircraft aileron. At this point, a robotic arm mounted on the platform performs a scan with an ultrasonic sensor, which cannot be carried out if the platform is not positioned with the necessary precision. The main limitation of this method is that the vision plate must be visible and within the field of view of the camera in order to perform the accurate positioning, which may not always happen in uncontrolled environments. On the other hand, the localisation accuracy of the same mobile platform is improved in a semi-indoor scenario by means of a GNSS. It navigates through a greenhouse, which is composed of a main corridor, rows perpendicular to it and a large number of plants in between. It is both a changing and symmetrical environment, due to the very nature of the greenhouse, and traditional localisation algorithms fail in estimating the position. The exploitation of the GNSS signal using the PPP technique is proposed to address these difficulties, which is used as input for the absolute localisation system for its subsequent fusion with the relative localisation module. By doing so, an error in the platform heading of less than 6° is achieved approximately 95% of the time, improving the estimation by up to 30% without the use of the GNSS signal. In addition, the map generated with the proposed system is much more realistic and faithful to the greenhouse morphology, although it has not been measured and the comparison is only done visually.

Finally, the last part differs to some extent from the previous ones, as it proposes a camera pose optimisation system for autonomous 3D mapping. This contribution aims to orient an RGB-D camera towards the area that will provide most information to the map. The optimisation is performed online, iteratively, and is completely independent of the other processes, although it is logically limited by the degrees of freedom set for the sensor in question. For this calculation, a 3D model of the environment containing an uncertainty estimation for each explored voxel is generated. This uncertainty value is calculated from the distance and angle from which it has been captured, and is updated every time the corresponding point is viewed. The system is tested with a mobile robot that mounts an RGB-D camera on it. In simulation, a RRT algorithm is used for 2D exploration while our method controls the pose of the sensor and builds the 3D map, while in real-world scenarios the platform is teleoperated to generate that trajectory. The results show that our method considerably improves the generated 3D map compared to not optimising the camera pose, without requiring additional time or platform movements. Furthermore, the information this map provides is not that far from that generated by a full 3D scanning algorithm, which does require more resources. While this development contributes to automating the generation of 3D indoor representations, it also aims to quantify and increase the quality of these. These models can be used for different purposes, but also for localisation, and higher quality models of reality contribute to better localisation estimations, as we have already seen. So, in this last part, we not only contribute to the mapping autonomy, but also to the improvement of the localisation accuracy of mobile robots in indoor environments.

Altogether, the contributions presented here improve the basic capabilities expected from a mobile platform, such as mapping, localisation and, as a consequence, navigation. These developments are combined with algorithms from the literature to meet the requirements of certain applications. Chapters 3 and 4 are directly related to real applications demanded by industry, and their effectiveness is demonstrated by the actual accomplishment of these tasks. Although the focus in the latter is mainly centred on the inspection tasks to be performed, the proposed procedure can be extrapolated to other scenarios, taking a step forward in this line in general. Chapter 5, in contrast, introduces a general purpose method that can be implemented not only on mobile platforms, but also on any element with a mobile 3D sensor whose position can be optimised.

6.2 Future Work

The precise positioning procedures described here are intended to inspect an element of the environment using a robotic arm mounted on the mobile platform. During this process, platform and robot are coordinated to ensure that the task is performed successfully, but the actions of both are performed sequentially. That is, if it is necessary to move the platform so that the arm can continue the inspection, first, the former pauses its trajectory and moves to a safe position. Next, the platform is moved to the desired location and, then, the arm can proceed with its execution. Depending on the size of the item to be inspected and the reach of the arm, this process can be repeated iteratively until the whole element is analysed. Ideally, both should be able to know each other's pose in real time in order to adapt their trajectories accordingly. The scientific community is making advances in this direction [96, 92], but there is not a solution that maintains the same precision that a robotic arm can offer independently when the platform on which it is mounted is moving. Overcoming this challenge will be a major breakthrough in mobile robotics, especially in control and localisation, and will offer the possibility of optimising many processes in industry, which, in principle, would reduce costs. For these reasons, we propose as a next step the task of achieving accurate positioning while the platform is in motion and not only when it is standing still facing the vision plate. The two devices will then need to be able to communicate with each other and modify their respective trajectories in real time, without the movement of one having an unintended effect on the actions of the other, so that each can perform its task effectively.

The combination of 2D exploration with our sensor pose optimisation has shown good results for 3D exploration and the system is highly configurable and adaptable to different sensors. However, this first version of the algorithm has been implemented with some limitations. On the one hand, the camera pose optimisation should be tested with more degrees of freedom. In the performed evaluation, only the pan movement has been considered, but our goal is to optimise, at least, the tilt and height coordinates too in every iteration. However, due to the high computational cost of such extension, we would need to parallelise the code first to take advantage of the increasing computing power of multicore architectures. On the other hand, we want to calculate the angle-based uncertainty taking into account not only the direction of the camera, but also the normal of the corresponding point or surface. Besides, although a set of optimal poses for a certain platform trajectory is hard to estimate correctly as how the map will grow is unknown, the system can consider the next pose of the robot and calculate the optimal pose accordingly. Our system runs independent from the robot motion to make it more generic and because the robot used is omnidirectional, it does not need to rotate to reach any objective. Nevertheless, with other motion systems, there can be a big divergence between the NBV expected and the point of view achieved, and taking it into account and going one step ahead would make a big difference in these situations.

Related to the mapping of the environment, latest advances in 3D reconstruction build realistic models of static environments, but dynamic scenes need more complex solutions [273]. Semantic mapping consists of building a map that represents spatial volumes with concepts or labels, i.e., associating a chair in the scene with a chair object in the model, and not just with a set of points. There are several powerful applications of semantic mapping in robotics. For example, a robot could use this information to self-localise in the environment, to improve the accuracy or to approach a previously detected object instead of a specific metric position. Furthermore, we would like to extend the state of the art of the environment by developing a system capable of building a dense 3D model of the environment autonomously with the use of visual sensors and physical manipulation of objects. This model would not only store semantic information of the elements in the scene, but also additional characteristics that improve their management and manipulation, such as materials, shapes and physical properties.

Some more ambitious and others not so much, mobile robotics offers a wide range of possibilities to explore, with the potential application in a variety of areas. In fact, whatever task we think of, it is very likely that a mobile robot can help to automate it or chain it with another, so that the whole process is done faster, safer or more efficiently. Although much progress has been made in this line in the last decades, the incoming advances are perhaps even more exciting and may have an even greater impact in the society.

Bibliography

[1] Motilal Agrawal and Kurt Konolige. “Real-time localization in outdoor environments using stereo vision and inexpensive gps”. In: *18th International Conference on Pattern Recognition (ICPR)*. Vol. 3. 2006, pp. 1063–1068 (cit. on p. 57).

[2] Alexander Albrecht and Nina Heide. “Mapping and automatic post-processing of indoor environments by extending visual SLAM”. In: *International Conference on Audio, Language and Image Processing (ICALIP)*. 2018, pp. 327–332 (cit. on p. 22).

[3] Ali Alyasin, Eyad I Abbas, and Sundus D Hasan. “An efficient optimal path finding for mobile robot based on dijkstra method”. In: *4th Scientific International Conference Najaf (SICN)*. 2019, pp. 11–14 (cit. on p. 40).

[4] Adrien Angeli, Stephane Doncieux, Jean-Arcady Meyer, and David Filliat. “Incremental vision-based topological SLAM”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 1031–1036 (cit. on p. 23).

[5] María Luisa Rodríguez Arévalo. “On the uncertainty in active SLAM: representation, propagation and monotonicity”. PhD thesis. Zaragoza, Spain: Department of Computer Science and Systems Engineering, University of Zaragoza, 2018 (cit. on pp. 22, 46).

[6] Artur Arsenio and M Isabel Ribeiro. “Active range sensing for mobile robot localization”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2. 1998, pp. 1066–1071 (cit. on p. 50).

[7] Mario Arzamendia Lopez. “Reactive Evolutionary Path Planning for Autonomous Surface Vehicles in Lake Environments”. PhD thesis. Seville, Spain: Department of Electronic Engineering, University of Seville, 2019 (cit. on p. 44).

[8] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Llado. “The SLAM problem: a survey”. In: *Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence (CCIA)*. Vol. 184. Jan. 2008, pp. 363–371 (cit. on p. 12).

[9] Izwan Azmi, Mohamad Syazwan Shafei, Mohammad Faidzul Nasrudin, Nor Samsiah Sani, and Abdul Hadi Abd Rahman. “ArUcoRSV: Robot localisation using artificial marker”. In: *International Conference on Robot Intelligence Technology and Applications (RiTA)*. 2018, pp. 189–198 (cit. on p. 70).

[10] Andrej Babinec, Ladislav Jurisica, Peter Hubinský, and Frantisek Duchon. “Visual localization of mobile robot using artificial markers”. In: *Procedia Engineering* 96 (2014), pp. 1–9 (cit. on p. 57).

[11] Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. “Information-theoretic exploration with Bayesian optimization”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1816–1822 (cit. on p. 111).

[12]Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. “Information-theoretic exploration with Bayesian optimization”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1816–1822 (cit. on p. 111).

[13]Shi Bai, Jinkun Wang, Kevin Doherty, and Brendan Englot. “Inference-enabled information-theoretic exploration of continuous action spaces”. In: *Robotics Research*. Springer, 2018, pp. 419–433 (cit. on p. 111).

[14]Tim Bailey and Hugh Durrant-Whyte. “Simultaneous Localisation and Mapping (SLAM) Part 2: State of the Art”. In: *IEEE Robotics and Automation Magazine*. 2006 (cit. on pp. 12, 22).

[15]Ruzena Bajcsy. “Active perception”. In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005 (cit. on p. 50).

[16]William S. Barbosa, Adalberto I. S. Oliveira, Gustavo B. P. Barbosa, et al. “Design and Development of an Autonomous Mobile Robot for Inspection of Soy and Cotton Crops”. In: *12th International Conference on Developments in eSystems Engineering (DeSE)*. 2019, pp. 557–562 (cit. on p. 79).

[17]Leonard E Baum and Ted Petrie. “Statistical inference for probabilistic functions of finite state Markov chains”. In: *The annals of mathematical statistics* 37.6 (1966), pp. 1554–1563 (cit. on p. 14).

[18]Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European Conference on Computer Vision (ECCV)*. 2006, pp. 404–417 (cit. on p. 35).

[19]Dirk Behring, Jan Thesing, Holger Becker, and Robert Zobel. “Optical coordinate measuring techniques for the determination and visualization of 3D displacements in crash investigations”. In: *SAE SP* (2003), pp. 225–232 (cit. on p. 70).

[20]Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. “Receding horizon" next-best-view" planner for 3d exploration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1462–1468 (cit. on p. 50).

[21]Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. “Receding horizon path planning for 3D exploration and surface inspection”. In: *Autonomous Robots* 42.2 (2018), pp. 291–306 (cit. on p. 96).

[22]S Bisnath and Y Gao. “Current state of precise point positioning and future prospects and limitations”. In: *Observing our changing earth*. Springer, 2009, pp. 615–623 (cit. on p. 84).

[23]Jose-Luis Blanco, Juan-Antonio Fernández-Madrigal, and Javier Gonzalez. “Toward a unified Bayesian approach to hybrid metric–topological SLAM”. In: *IEEE Transactions on Robotics* 24.2 (2008), pp. 259–270 (cit. on p. 23).

[24]Robert Bodor, Andrew Drenner, Paul Schrater, and Nikolaos Papanikolopoulos. “Optimal camera placement for automated surveillance tasks”. In: *Journal of Intelligent and Robotic Systems* 50.3 (2007), pp. 257–295 (cit. on p. 51).

[25] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. “Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework”. In: *The International Journal of Robotics Research* 23.12 (2004), pp. 1113–1139 (cit. on p. 22).

[26] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. “Information based adaptive robotic exploration”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 1. 2002, pp. 540–545 (cit. on p. 46).

[27] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. “Simultaneous localization and mapping: A survey of current trends in autonomous driving”. In: *IEEE Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220 (cit. on p. 80).

[28] Julie Buquet, Jinsong Zhang, Patrice Roulet, Simon Thibault, and Jean-François Lalonde. “Evaluating the Impact of Wide-Angle Lens Distortion on Learning-based Depth Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3693–3701 (cit. on p. 100).

[29] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. “Active mobile robot localization”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 1997, pp. 1346–1352 (cit. on p. 50).

[30] Cesar Cadena, Luca Carlone, Henry Carrillo, et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332 (cit. on p. 22).

[31] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. “Brief: Binary robust independent elementary features”. In: *European Conference on Computer Vision (ECCV)*. 2010, pp. 778–792 (cit. on p. 36).

[32] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. “Orb-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890 (cit. on p. 33).

[33] Karel Čapek and Iosif Kallinikov. *Rossum’s Universal Robots*. Fr. Borový, 1940 (cit. on p. 3).

[34] Luca Carlone, Jingjing Du, Miguel Kaouk Ng, Basilio Bona, and Marina Indri. “Active SLAM and exploration with particle filters using Kullback-Leibler divergence”. In: *Journal of Intelligent & Robotic Systems* 75.2 (2014), pp. 291–311 (cit. on pp. 49, 50, 95, 96).

[35] Henry Carrillo, Ian Reid, and José A Castellanos. “On the comparison of uncertainty criteria for active SLAM”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 2080–2087 (cit. on p. 49).

[36] Ozan Çatal, Tim Verbelen, Toon Van de Maele, Bart Dhoedt, and Adam Safron. “Robot navigation as hierarchical active inference”. In: *Neural Networks* 142 (2021), pp. 192–204 (cit. on p. 37).

[37]Le Chang, Xiaoji Niu, Tianyi Liu, Jian Tang, and Chuang Qian. “GNSS/INS/LiDAR-SLAM integrated navigation system based on graph optimization”. In: *Remote Sensing* 11.9 (2019), p. 1009 (cit. on p. 82).

[38]Fernando A Auat Cheein, Juan M Toibero, Fernando di Sciascio, Ricardo Carelli, and F Lobo Pereira. “Monte Carlo uncertainty maps-based for mobile robot autonomous SLAM navigation”. In: *IEEE International Conference on Industrial Technology (ICIT)*. 2010, pp. 1433–1438 (cit. on p. 46).

[39]Yongbo Chen, Shoudong Huang, Robert Fitch, et al. “On-line 3D active pose-graph SLAM based on key poses using graph topology and sub-maps”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 169–175 (cit. on p. 23).

[40]Boris V Cherkassky, Andrew V Goldberg, and Tomasz Radzik. “Shortest paths algorithms: Theory and experimental evaluation”. In: *Mathematical programming* 73.2 (1996), pp. 129–174 (cit. on p. 41).

[41]Kai-Wei Chiang, Hsiu-Wen Chang, Yu-Hua Li, et al. “Assessment for INS/GNSS/Odometer/Barometer Integration in Loosely-Coupled and Tightly-Coupled Scheme in a GNSS-Degraded Environment”. In: *IEEE Sensors Journal* 20.6 (2020), pp. 3057–3069 (cit. on p. 82).

[42]Cl Connolly. “The determination of next best views”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 1985, pp. 432–435 (cit. on p. 51).

[44]Anna Dai, Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, and Stefan Leutenegger. “Fast Frontier-based Information-driven Autonomous Exploration with an MAV”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9570–9576 (cit. on p. 50).

[45]Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. “Theta*: Any-angle path planning on grids”. In: *Journal of Artificial Intelligence Research* 39 (2010), pp. 533–579 (cit. on p. 44).

[46]Angela Davids. “Urban search and rescue robots: from tragedy to technology”. In: *IEEE Intelligent systems* 17.2 (2002), pp. 81–83 (cit. on p. 5).

[47]Andrew J Davison. “Real-time simultaneous localisation and mapping with a single camera”. In: *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*. 2003, p. 1403 (cit. on pp. 24, 29).

[48]Andrew J Davison and David W Murray. “Mobile robot localisation using active vision”. In: *European Conference on Computer Vision (ECCV)*. 1998, pp. 809–825 (cit. on p. 97).

[49]Andrew J Davison and David W. Murray. “Simultaneous localization and map-building using active vision”. In: *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), pp. 865–880 (cit. on p. 97).

[50] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. "MonoSLAM: Real-time single camera SLAM". In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067 (cit. on p. 29).

[51] Pierre Del Moral. "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 325.6 (1997), pp. 653–658 (cit. on p. 20).

[52] Pierre Del Moral. "Measure-valued processes and interacting particle systems. Application to nonlinear filtering problems". In: *The Annals of Applied Probability* 8.2 (1998), pp. 438–495 (cit. on p. 20).

[53] Pierre Del Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Vol. 88. Springer, 2004 (cit. on p. 20).

[54] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "On adaptive resampling strategies for sequential Monte Carlo methods". In: *Bernoulli* 18.1 (2012), pp. 252–278 (cit. on p. 20).

[55] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "Monte carlo localization for mobile robots". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 1999, pp. 1322–1328 (cit. on p. 20).

[56] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. "A comparison of volumetric information gain metrics for active 3D object reconstruction". In: *Autonomous Robots* 42.2 (2018), pp. 197–208 (cit. on p. 97).

[57] Edsger W Dijkstra et al. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 40).

[58] Christian Dornhege and Alexander Kleiner. "A frontier-void-based approach for autonomous exploration in 3d". In: *Advanced Robotics* 27.6 (2013), pp. 459–468 (cit. on p. 96).

[59] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics and Automation Magazine* 13.2 (2006), pp. 99–110 (cit. on pp. 12, 25).

[60] MA Ebrahimi, Mohammad Hadi Khoshtaghaza, Saeid Minaei, and Bahareh Jamshidi. "Vision-based pest detection based on SVM classification method". In: *Computers and Electronics in Agriculture* 137 (2017), pp. 52–58 (cit. on p. 79).

[62] Yoichiro Endo, Jonathan C Balloch, Alexander Grushin, Mun Wai Lee, and David Handelman. "Landmark-based robust navigation for tactical UGV control in GPS-denied communication-degraded environments". In: *Unmanned Systems Technology XVIII*. Vol. 9837. 2016, 98370F (cit. on p. 57).

[63] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. 2014, pp. 834–849 (cit. on p. 56).

[64] Jan Faigl and Miroslav Kulich. "On determination of goal candidates in frontier-based multi-robot exploration". In: *European Conference on Mobile Robots (ECMR)*. 2013, pp. 210–215 (cit. on p. 49).

[65] Nathaniel Fairfield and David Wettergreen. “Active SLAM and loop prediction with the segmented map using simplified models”. In: *Field and service robotics*. 2010, pp. 173–182 (cit. on p. 49).

[66] Margarida Faria, Ivan Maza, and Antidio Viguria. “Applying frontier cells based exploration and Lazy Theta* path planning over single grid-based world representation for autonomous inspection of large 3D structures with an UAS”. In: *Journal of Intelligent & Robotic Systems* 93.1-2 (2019), pp. 113–133 (cit. on p. 96).

[67] Hans Jacob S Feder, John J Leonard, and Christopher M Smith. “Adaptive mobile robot navigation and mapping”. In: *The International Journal of Robotics Research* 18.7 (1999), pp. 650–668 (cit. on pp. 25, 46).

[68] Valerii Fedorov. “Optimal experimental design”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.5 (2010), pp. 581–589 (cit. on p. 50).

[69] Dieter Fox. “KLD-sampling: Adaptive particle filters”. In: *Advances in neural information processing systems* 14 (2001) (cit. on pp. 20, 63, 85).

[70] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics and Automation Magazine* 4.1 (1997), pp. 23–33 (cit. on p. 43).

[71] Friedrich Fraundorfer, Christopher Engels, and David Nistér. “Topological mapping, localization and navigation using image collections”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2007, pp. 3872–3877 (cit. on p. 23).

[72] Friedrich Fraundorfer and Davide Scaramuzza. “Visual odometry: Part ii: Matching, robustness, optimization, and applications”. In: *IEEE Robotics and Automation Magazine* 19.2 (2012), pp. 78–90 (cit. on p. 25).

[73] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. “Visual simultaneous localization and mapping: a survey”. In: *Artificial intelligence review* 43.1 (2015), pp. 55–81 (cit. on p. 22).

[74] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. “LDSO: Direct sparse odometry with loop closure”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2198–2204 (cit. on pp. 22, 94).

[75] Emilio Garcia-Fidalgo and Alberto Ortiz. “On the use of binary feature descriptors for loop closure detection”. In: *Proceedings of the IEEE Emerging Technology and Factory Automation (ETFA)*. 2014, pp. 1–8 (cit. on p. 22).

[76] Emilio Garcia-Fidalgo and Alberto Ortiz. “ibow-lcd: An appearance-based loop-closure detection approach using incremental bags of binary words”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3051–3057 (cit. on pp. 22, 94).

[77] Avinash Gautam, J Krishna Murthy, Gourav Kumar, et al. “Cluster, allocate, cover: An efficient approach for multi-robot coverage”. In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2015, pp. 197–203 (cit. on p. 49).

[78] Jonathon Ashley Gibbs, Michael Pound, Andrew French, et al. “Active vision and surface reconstruction for 3D plant shoot modelling”. In: *IEEE/ACM transactions on computational biology and bioinformatics* (2019) (cit. on p. 97).

[79] Jose-Joel Gonzalez-Barbosa, Teresa García-Ramírez, Joaquín Salas, Juan-Bautista Hurtado-Ramos, et al. “Optimal camera placement for total coverage”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 844–848 (cit. on p. 51).

[80] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. “A tutorial on graph-based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43 (cit. on p. 22).

[81] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved techniques for grid mapping with rao-blackwellized particle filters”. In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46 (cit. on pp. 58, 85, 112).

[82] Paul D Groves. “Principles of GNSS, inertial, and multisensor integrated navigation systems, [Book review]”. In: *IEEE Aerospace and Electronic Systems Magazine* 30.2 (2015), pp. 26–27 (cit. on p. 82).

[83] Jing Guo, Xingxing Li, Zhenhong Li, et al. “Multi-GNSS precise point positioning for precision agriculture”. In: *Precision agriculture* 19.5 (2018), pp. 895–911 (cit. on p. 83).

[84] Joong-hee Han and Chi-ho Park. “Performance evaluation on GNSS, wheel speed sensor, yaw rate sensor, and gravity sensor integrated positioning algorithm for automotive navigation system”. In: *E3S Web of Conferences*. Vol. 94. 2019, p. 02003 (cit. on p. 82).

[85] Seung-Jun Han, Juwan Kim, and Jeongdan Choi. “Effective height-grid map building using inverse perspective image”. In: *IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 549–554 (cit. on p. 24).

[86] Peter E Hart, Nils J Nilsson, and Bertram Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 41).

[87] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. on p. 31).

[88] Mostafa Hassanalian and Abdessattar Abdelkefi. “Classifications, applications, and design challenges of drones: A review”. In: *Progress in Aerospace Sciences* 91 (2017), pp. 99–131 (cit. on p. 4).

[89] Salim Al-Hassani. “Al-Jazari: The Mechanical Genius”. In: *published on February 9th* (2001) (cit. on p. 4).

[91] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. “Real-time loop closure in 2D LIDAR SLAM”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278 (cit. on pp. 22, 59, 94).

[92] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. “Learning kinematic feasibility for mobile manipulation through deep reinforcement learning”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6289–6296 (cit. on p. 125).

[93] Gregory S Hornby, Seiichi Takamura, Jun Yokono, et al. “Evolving robust gaits with AIBO”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. 2000, pp. 3040–3045 (cit. on p. 4).

[94] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous robots* 34.3 (2013), pp. 189–206 (cit. on pp. 24, 108).

[95] Carl Huffman. *Archytas of Tarentum: Pythagorean, philosopher and mathematician king*. Cambridge University Press, 2005 (cit. on p. 3).

[96] Aitor Ibarguren and Paul Daelman. “Path Driven Dual Arm Mobile Co-Manipulation Architecture for Large Part Manipulation in Industrial Environments”. In: *Sensors* 21.19 (2021), p. 6620 (cit. on p. 125).

[97] S Indu, Santanu Chaudhury, Nikhil R Mittal, and Asok Bhattacharyya. “Optimal sensor placement for surveillance of large spaces”. In: *3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*. 2009, pp. 1–8 (cit. on p. 51).

[98] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. “An information gain formulation for active volumetric 3D reconstruction”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3477–3484 (cit. on p. 97).

[99] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007 (cit. on p. 14).

[100] Patric Jensfelt and Steen Kristensen. “Active global localization for a mobile robot using multiple hypothesis tracking”. In: *IEEE Transactions on Robotics and Automation* 17.5 (2001), pp. 748–760 (cit. on p. 50).

[101] Rui Jiang, Shuai Yang, Shuzhi Sam Ge, Han Wang, and Tong Heng Lee. “Geometric map-assisted localization for mobile robots based on uniform-Gaussian distribution”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 789–795 (cit. on p. 24).

[102] Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. “Autonomous exploration for 3D map learning”. In: *Autonome Mobile Systeme*. Springer, 2007, pp. 22–28 (cit. on p. 49).

[103] Joseph L Jones. “Robots at the tipping point: the road to iRobot Roomba”. In: *IEEE Robotics and Automation Magazine* 13.1 (2006), pp. 76–78 (cit. on p. 4).

[104] Simon J Julier and Jeffrey K Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal processing, sensor fusion, and target recognition VI*. Vol. 3068. 1997, pp. 182–193 (cit. on p. 19).

[105] Simon J Julier and Jeffrey K Uhlmann. “Unscented filtering and nonlinear estimation”. In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422 (cit. on p. 28).

[106] Rudolph Emil Kalman. “A new approach to linear filtering and prediction problems”. In: *Transactions of the ASME Journal of Basic Engineering* 82 (Series D) (1960), pp. 35–45 (cit. on p. 18).

[107] Kamarulzaman Kamarudin, Syed Muhammad Mamduh, Ali Yeon Md Shakaff, and Ammar Zakaria. “Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques”. In: *Sensors* 14.12 (2014), pp. 23365–23387 (cit. on p. 46).

[108] Matan Keidar and Gal A. Kaminka. “Efficient frontier detection for robot exploration”. In: *International Journal of Robotics Research* 33 (2014), pp. 215–236 (cit. on p. 48).

[109] Nishant Kejriwal, Swagat Kumar, and Tomohiro Shibata. “High performance loop closure detection using bag of word pairs”. In: *Robotics and Autonomous Systems* 77 (2016), pp. 55–65 (cit. on p. 22).

[110] Daanish Allen Khan. “Observable autonomous SLAM in 2D dynamic environments”. PhD thesis. Ottawa, Ontario, Canada: Ottawa-Carleton Institute for Electrical, Computer Engineering (OCIECE), Department of Systems, and Computer Engineering, Carleton University, 2011 (cit. on p. 46).

[111] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, et al. “Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network”. In: *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 399–404 (cit. on p. 23).

[112] Chanki Kim, Rathinasamy Sakthivel, and Wan Kyun Chung. “Unscented FastSLAM: A robust algorithm for the simultaneous localization and mapping problem”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 2439–2445 (cit. on p. 28).

[113] Soo-Hyun Kim, Jae-Gi Kim, and Tae-Kyu Yang. “Autonomous SLAM technique by integrating Grid and Topology map”. In: *International Conference on Smart Manufacturing Application (ICSMA)*. 2008, pp. 413–418 (cit. on p. 46).

[114] Georg Klein and David Murray. “Parallel tracking and mapping for small AR workspaces”. In: *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234 (cit. on p. 30).

[115] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. “A Flexible and Scalable SLAM System with Full 3D Motion Estimation”. In: *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE. Nov. 2011 (cit. on p. 58).

[116] Yves Kompis, Luca Bartolomei, Ruben Mascaro, Lucas Teixeira, and Margarita Chli. “Informed sampling exploration path planner for 3d reconstruction of large scenes”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7893–7900 (cit. on p. 97).

[117] Tomáš Koutecký, David Paloušek, and Jan Brandejs. “Method of photogrammetric measurement automation using TRITOP system and industrial robot”. In: *Optik* 124 (Sept. 2013), pp. 3705–3709 (cit. on p. 70).

[118] Simon Kriegel, Christian Rink, Tim Bodenmüller, and Michael Suppa. “Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects”. In: *Journal of Real-Time Image Processing* 10.4 (2015), pp. 611–631 (cit. on pp. 50, 97).

[119] Benjamin Kuipers and Yung-Tai Byun. “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations”. In: *Robotics and Autonomous Systems* 8.1-2 (1991), pp. 47–63 (cit. on p. 46).

[120] Rainer Kümmerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss, and Wolfram Burgard. “Autonomous robot navigation in highly populated pedestrian zones”. In: *Journal of Field Robotics* 32.4 (2015), pp. 565–589 (cit. on p. 4).

[121] Zeyneb Kurt-Yavuz and Sirma Yavuz. “A comparison of EKF, UKF, FastSLAM2. 0, and UKF-based FastSLAM algorithms”. In: *IEEE 16th International Conference on Intelligent Engineering Systems (INES)*. 2012, pp. 37–43 (cit. on p. 28).

[122] Manohar Kuse and Shaojie Shen. “Learning whole-image descriptors for real-time loop detection and kidnap recovery under large viewpoint difference”. In: *Robotics and Autonomous Systems* 143 (2021), p. 103813 (cit. on p. 22).

[123] Mathieu Labbe and François Michaud. “Online global loop closure detection for large-scale multi-session graph-based SLAM”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 2661–2666 (cit. on p. 22).

[125] Steven M. LaValle. “Rapidly-exploring random trees : a new tool for path planning”. In: *The annual research report* 129 (1998) (cit. on pp. 43, 96).

[126] Anthony Lazanas and J-C Latombe. “Landmark-based robot navigation”. In: *Algorithmica* 13.5 (1995), pp. 472–501 (cit. on p. 23).

[127] Changmin Lee, Seung-Eun Yu, and DaeEun Kim. “Landmark-based homing navigation using omnidirectional depth information”. In: *Sensors* 17.8 (2017), p. 1928 (cit. on p. 23).

[128] Hannah Lehner, Martin J Schuster, Tim Bodenmüller, and Simon Kriegel. “Exploration with active loop closing: A trade-off between exploration efficiency and map quality”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 6191–6198 (cit. on p. 96).

[129] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. “Map-Based Precision Vehicle Localization in Urban Environments.” In: *Robotics: Science and Systems*. Vol. 4. 2007, p. 1 (cit. on p. 56).

[130] Pan Li, Zhi Liu, and Dandan Huang. “3D Map Merging Based on Overlapping Region”. In: *3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*. 2019, pp. 1133–1137 (cit. on p. 108).

[131] Jun S Liu and Rong Chen. “Sequential Monte Carlo methods for dynamic systems”. In: *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044 (cit. on p. 20).

[132]Yanan Liu, Laurie Bose, Colin Greatwood, et al. “Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array”. In: *IET image processing* 15.9 (2021), pp. 1883–1892 (cit. on p. 37).

[133]Zhihai Liu, Hanbin Liu, Zhenguo Lu, and Qingliang Zeng. “A dynamic fusion pathfinding algorithm using delaunay triangulation and improved a-star for mobile robots”. In: *IEEE Access* 9 (2021), pp. 20602–20621 (cit. on p. 40).

[134]Iker Lluvia Hermosilla. “Mejora de estrategias de navegación en el robot de servicio kTBot/Teknibot”. MA thesis. Donostia, Spain: Faculty of Informatics, University of the Basque Country UPV/EHU, 2017 (cit. on p. 7).

[135]Pierre Lothe, Steve Bourgeois, Fabien Dekeyser, Eric Royer, and Michel Dhome. “Towards geographical referencing of monocular SLAM reconstruction using 3d city models: Application to real-time accurate vision-based localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 2882–2889 (cit. on p. 22).

[136]David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. on p. 34).

[137]David V Lu, Dave Hershberger, and William D Smart. “Layered costmaps for context-sensitive navigation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 709–715 (cit. on p. 24).

[138]Édouard Lucas. *Récréations mathématiques: Les traversees. Les ponts. Les labyrinthes. Les reines. Le solitaire. la numération. Le baguenaudier. Le taquin*. Vol. 1. Gauthier-Villars et fils, 1882 (cit. on p. 40).

[139]MC Lucas-Estañ, JL Maestre, B Coll-Perales, J Gozalvez, and I Lluvia. “An Experimental Evaluation of Redundancy in Industrial Wireless Communications”. In: *IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. 2018, pp. 1075–1078 (cit. on p. 57).

[140]Liam Lynch, Thomas Newe, John Clifford, et al. “Automated ground vehicle (agv) and sensor technologies-a review”. In: *12th International Conference on Sensing Technology (ICST)*. 2018, pp. 347–352 (cit. on p. 4).

[141]Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. “A comprehensive survey of visual SLAM algorithms”. In: *Robotics* 11.1 (2022), p. 24 (cit. on p. 33).

[142]Sebastian Mai, Christoph Steup, and Sanaz Mostaghim. “Simultaneous Localisation and Optimisation for Swarm Robotics”. In: Nov. 2018, pp. 1998–2004 (cit. on p. 95).

[143]Eric Marchand and François Chaumette. “Active vision for complete scene reconstruction and exploration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.1 (1999), pp. 65–72 (cit. on p. 97).

[144]Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. “The office marathon: Robust navigation in an indoor office environment”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 300–307 (cit. on p. 24).

[145]Pablo Marın-Plaza, Jorge Beltrán, Ahmed Hussein, et al. “Stereo vision-based local occupancy grid map for autonomous navigation in ros”. In: *11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*. Vol. 2016. 2016 (cit. on p. 23).

[146]Fernando Martín, Rudolph Triebel, Luis Moreno, and Roland Siegwart. “Two different tools for three-dimensional mapping: DE-based scan matching and feature-based loop detection”. In: *Robotica* 32.1 (2014), pp. 19–41 (cit. on p. 22).

[147]Humberto Martínez-Barberá and David Herrero-Pérez. “Autonomous navigation of an automated guided vehicle in industrial environments”. In: *Robotics and Computer-Integrated Manufacturing* 26.4 (2010), pp. 296–311 (cit. on p. 67).

[148]Ruben Martinez-Cantin and José A Castellanos. “Unscented SLAM for large-scale outdoor environments”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2005, pp. 3427–3432 (cit. on p. 27).

[149]Ruben Martinez-Cantin, Nando De Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. “A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot”. In: *Autonomous Robots* 27.2 (2009), pp. 93–103 (cit. on p. 46).

[150]Larry Matthies and Alberto Elfes. “Integration of sonar and stereo range data using a grid-based representation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 1988, pp. 727–733 (cit. on p. 24).

[151]Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. “Real-time semantic mapping for autonomous off-road navigation”. In: *Field and Service Robotics*. 2018, pp. 335–350 (cit. on p. 24).

[152]Donald Meagher. “Geometric modeling using octree encoding”. In: *Computer Graphics and Image Processing* 19.2 (1982), pp. 129–147 (cit. on p. 24).

[153]Donald Meagher. “Geometric modeling using octree encoding”. In: *Computer Graphics and Image Processing* 19.2 (1982), pp. 129–147 (cit. on p. 99).

[154]Amir Mobarhani, Shaghayegh Nazari, Amir H Tamjidi, and Hamid D Taghirad. “Histogram based frontier exploration”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011, pp. 1128–1133 (cit. on p. 49).

[155]Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. “Fast-SLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”. In: *18th National Conference on Artificial Intelligence (AAAI/IAAI)*. 2002, pp. 593–598 (cit. on p. 26).

[156] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 3. 2003, pp. 1151–1156 (cit. on p. 27).

[157] Luis V Montiel and J Eric Bickel. “Exploring the Accuracy of Joint-Distribution Approximations Given Partial Information”. In: *The Engineering Economist* 64.4 (2019), pp. 323–345 (cit. on p. 49).

[158] T. Moore and D. Stouch. “A Generalized Extended Kalman Filter Implementation for the Robot Operating System”. In: *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS)*. Springer, July 2014 (cit. on p. 85).

[159] Joshua J Morales and Zaher M Kassas. “Tightly coupled inertial navigation system with signals of opportunity aiding”. In: *IEEE Transactions on Aerospace and Electronic Systems* 57.3 (2021), pp. 1930–1948 (cit. on p. 82).

[160] Hans P. Moravec. “Sensor fusion in certainty grids for mobile robots”. In: *Sensor devices and systems for robotics*. Springer, 1989, pp. 253–276 (cit. on p. 24).

[161] Yacine Morsly, Nabil Aouf, Mohand Said Djouadi, and Mark Richardson. “Particle swarm optimization inspired probability algorithm for optimal camera network placement”. In: *IEEE Sensors Journal* 12.5 (2011), pp. 1402–1412 (cit. on p. 51).

[163] Anastasios I Mourikis, Stergios I Roumeliotis, et al. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation.” In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 2007, p. 6 (cit. on p. 32).

[164] Beipeng Mu, Matthew Giamou, Liam Paull, et al. “Information-based active SLAM via topological feature graphs”. In: *IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 5583–5590 (cit. on pp. 23, 45).

[165] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163 (cit. on pp. 33, 56).

[166] Raul Mur-Artal and Juan D Tardós. “Orb-SLAM2: An open-source SLAM system for monocular, stereo, and rgb-d cameras”. In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262 (cit. on p. 33).

[167] Raúl Mur-Artal and Juan D Tardós. “Fast relocalisation and loop closing in keyframe-based SLAM”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 846–853 (cit. on p. 33).

[168] Raúl Mur-Artal and Juan D Tardós. “Visual-inertial monocular SLAM with map reuse”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803 (cit. on p. 33).

[169] Tayyab Naseer, Wolfram Burgard, and Cyrill Stachniss. “Robust Visual Localization Across Seasons”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 289–302 (cit. on p. 56).

[170]KM Ng, J Johari, SAC Abdullah, A Ahmad, and BN Laja. “Performance evaluation of the RTK-GNSS navigating under different landscape”. In: *18th International Conference on Control, Automation and Systems (ICCAS)*. 2018, pp. 1424–1428 (cit. on p. 83).

[171]Adam Niewola and Leszek Podsędkowski. “PSD–probabilistic algorithm for mobile robot 6D localization without natural and artificial landmarks based on 2.5 D map and a new type of laser scanner in GPS-denied scenarios”. In: *Mechatronics* 65 (2020), p. 102308 (cit. on p. 51).

[172]Nils J Nilsson. *Shakey the robot*. Tech. rep. SRI INTERNATIONAL MENLO PARK CA, 1984 (cit. on p. 4).

[173]Ulrik Nilsson, Petter Ogren, and Johan Thunberg. “Optimal positioning of surveillance UGVs”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 2539–2544 (cit. on p. 51).

[174]Simona Nobili, Raluca Scona, Marco Caravagna, and Maurice Fallon. “Overlap-based ICP tuning for robust localization of a humanoid robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4721–4728 (cit. on p. 108).

[175]Joseph O’rourke. *Art gallery theorems and algorithms*. Oxford University Press Oxford, 1987 (cit. on p. 51).

[177]Christos Papachristos, Mina Kamel, Marija Popović, et al. “Autonomous exploration and inspection path planning for aerial robots using the robot operating system”. In: *Robot Operating System (ROS)*. Springer, 2019, pp. 67–111 (cit. on p. 96).

[178]Christos Papachristos, Shehryar Khattak, and Kostas Alexis. “Uncertainty-aware receding horizon exploration and mapping using aerial robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4568–4575 (cit. on p. 96).

[179]Evangelos Papadopoulos. “Heron of Alexandria (c. 10–85 AD)”. In: *Distinguished figures in mechanism and machine science*. Springer, 2007, pp. 217–245 (cit. on p. 4).

[180]BK Patle, Anish Pandey, DRK Parhi, A Jagadeesh, et al. “A review: On path planning strategies for navigation of mobile robot”. In: *Defence Technology* 15.4 (2019), pp. 582–606 (cit. on p. 37).

[181]Manuel Pérez Ruiz and Shrini Upadhyaya. *GNSS in precision agricultural operations*. Intech, 2012 (cit. on p. 80).

[182]Marek Pierzchała, Philippe Giguère, and Rasmus Astrup. “Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM”. In: *Computers and Electronics in Agriculture* 145 (2018), pp. 217–225 (cit. on p. 82).

[183]Julio A Placed, Jared Strader, Henry Carrillo, et al. “A survey on active simultaneous localization and mapping: State of the art and new frontiers”. In: *arXiv preprint arXiv:2207.00254* (2022) (cit. on p. 50).

[184]Chuang Qian, Hui Liu, Jian Tang, et al. “An integrated GNSS/INS/LiDAR-SLAM positioning method for highly accurate forest stem mapping”. In: *Remote Sensing* 9.1 (2016), p. 3 (cit. on p. 82).

[185] Morgan Quigley, Ken Conley, Brian Gerkey, et al. “ROS: an open-source Robot Operating System”. In: *IEEE International Conference on Robotics and Automation (ICRA), Workshop on "Open Source Software"*. Vol. 3. 2009, p. 5 (cit. on p. 106).

[186] J. Ross Quinlan. “Induction of decision trees”. In: *Machine learning* 1.1 (1986), pp. 81–106 (cit. on p. 36).

[187] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. “BigDog, the Rough-Terrain Quadruped Robot”. In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 10822–10825 (cit. on p. 56).

[188] Ankit A Ravankar, Abhijeet Ravankar, Yukinori Kobayashi, and Takanori Emaru. “Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors”. In: *Multidisciplinary Digital Publishing Institute Proceedings*. Vol. 4. 2018, p. 44 (cit. on p. 50).

[189] Ioannis Rekleitis, Jean-Luc Bedwani, Erick Dupuis, Tom Lamarche, and Pierre Allard. “Autonomous over-the-horizon navigation using LIDAR data”. In: *Autonomous Robots* 34.1-2 (2013), pp. 1–18 (cit. on p. 56).

[190] Christian P Robert and Gareth Roberts. “Rao–Blackwellisation in the Markov Chain Monte Carlo Era”. In: *International Statistical Review* 89.2 (2021), pp. 237–249 (cit. on p. 58).

[192] María L Rodríguez-Arévalo, José Neira, and José A Castellanos. “On the importance of uncertainty representation in active SLAM”. In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 829–834 (cit. on p. 49).

[193] Vinicio Alejandro Rosas-Cervantes, Quoc-Dong Hoang, Soon-Geul Lee, and Jae-Hwan Choi. “Multi-Robot 2.5 D Localization and Mapping Using a Monte Carlo Algorithm on a Multi-Level Surface”. In: *Sensors* 21.13 (2021), p. 4588 (cit. on p. 97).

[194] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *European Conference on Computer Vision (ECCV)*. 2006, pp. 430–443 (cit. on p. 35).

[195] N. Roy, W. Burgard, D. Fox, and S. Thrun. “Coastal navigation-mobile robot navigation with uncertainty in dynamic environments”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 1999, pp. 35–40 (cit. on p. 25).

[196] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *International Conference on Computer Vision (ICCV)*. 2011, pp. 2564–2571 (cit. on p. 36).

[197] Joseph Ryding, Emily Williams, Martin J Smith, and Markus P Eichhorn. “Assessing handheld mobile laser scanners for forest surveys”. In: *Remote Sensing* 7.1 (2015), pp. 1095–1111 (cit. on p. 80).

[198]Seyed Abbas Sadat, Kyle Chutskoff, Damir Jungic, Jens Wawerla, and Richard Vaughan. “Feature-rich path planning for robust navigation of MAVs with mono-SLAM”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 3870–3875 (cit. on p. 96).

[199]Rock Santerre, Lin Pan, Changsheng Cai, and Jianjun Zhu. “Single point positioning using GPS, GLONASS and BeiDou satellites”. In: (2014) (cit. on p. 83).

[200]Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. “Visual SLAM and structure from motion in dynamic environments: A survey”. In: *ACM Computing Surveys (CSUR)* 51.2 (2018), pp. 1–36 (cit. on pp. 12, 51).

[201]Ismail Hakki Savci, Abdurrahman Yilmaz, Sadettin Karaman, Hakan Ocakli, and Hakan Temeltas. “Improving Navigation Stack of a ROS-Enabled Industrial Autonomous Mobile Robot (AMR) to be Incorporated in a Large-Scale Automotive Production”. In: *International Journal of Advanced Manufacturing Technology* 120.5 (2022), pp. 3647–3668 (cit. on p. 4).

[202]Davide Scaramuzza, Michael C Achtelik, Lefteris Doitsidis, et al. “Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments”. In: *IEEE Robotics and Automation Magazine* 21.3 (2014), pp. 26–40 (cit. on p. 57).

[203]David Schleicher, Luis M Bergasa, Manuel Ocaña, Rafael Barea, and María Elena López. “Real-time hierarchical outdoor SLAM based on stereovision and GPS fusion”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.3 (2009), pp. 440–452 (cit. on p. 46).

[204]Magnus Selin, Mattias Tiger, Daniel Duberg, Fredrik Heintz, and Patric Jensfelt. “Efficient autonomous exploration planning of large-scale 3-D environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1699–1706 (cit. on pp. 48, 96).

[205]PGCN Senarathne and Danwei Wang. “Towards autonomous 3D exploration using surface frontiers”. In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2016, pp. 34–41 (cit. on p. 95).

[206]Nagaraj C Shivaramaiah and Andrew G Dempster. “The Galileo E5 AltBOC: understanding the signal structure”. In: *International global navigation satellite systems society IGNSS symposium*. 2009, pp. 1–3 (cit. on p. 85).

[207]Rakesh Shrestha, Fei-Peng Tian, Wei Feng, Ping Tan, and Richard Vaughan. “Learned map prediction for enhanced mobile robot exploration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 1197–1204 (cit. on p. 95).

[208]Robert Sim and Nicholas Roy. “Global A-Optimal Robot Exploration in SLAM”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2005, pp. 661–666 (cit. on p. 48).

[209]Reid G. Simmons, David Apfelbaum, Wolfram Burgard, et al. “Coordination for Multi-Robot Exploration and Mapping”. In: *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*. 2000, pp. 852–858 (cit. on p. 46).

[210] Stewart A Skomra and Julian Durand. *Method and apparatus for a local positioning system*. US Patent 9,258,797. 2016 (cit. on p. 57).

[211] Andrew J Smith and Geoffrey A Hollinger. “Distributed inference-based multi-robot exploration”. In: *Autonomous Robots* 42.8 (2018), pp. 1651–1668 (cit. on p. 46).

[212] Randall Smith, Matthew Self, and Peter Cheeseman. “Estimating uncertain spatial relationships in robotics”. In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193 (cit. on p. 18).

[213] Randall C Smith and Peter Cheeseman. “On the representation and estimation of spatial uncertainty”. In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68 (cit. on pp. 18, 19).

[214] Agusti Solanas and Miguel Angel Garcia. “Coordinated multi-robot exploration through unsupervised clustering of unknown space”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 1. 2004, pp. 717–721 (cit. on p. 49).

[216] Cyrill Stachniss. *Robotic mapping and exploration*. Springer, 2009 (cit. on pp. 22, 46, 50).

[217] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. “Information gain-based exploration using rao-blackwellized particle filters.” In: *Robotics: Science and Systems*. Vol. 2. 2005, pp. 65–72 (cit. on p. 49).

[219] Sudharshan Suresh, Paloma Sodhi, Joshua G Mangelson, David Wettergreen, and Michael Kaess. “Active SLAM using 3D submap saliency for underwater volumetric exploration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3132–3138 (cit. on p. 96).

[220] Rafal Szczepanski and Tomasz Tarczewski. “Global path planning for mobile robot based on Artificial Bee Colony and Dijkstra’s algorithms”. In: *IEEE 19th International Power Electronics and Motion Control Conference (PEMC)*. 2021, pp. 724–730 (cit. on p. 40).

[221] Lei Tai and Ming Liu. “Mobile robots exploration through cnn-based reinforcement learning”. In: *Robotics and biomimetics* 3.1 (2016), p. 24 (cit. on p. 46).

[222] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: A survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), p. 16 (cit. on p. 57).

[223] Jian Tang, Yuwei Chen, Antero Kukko, et al. “SLAM-aided stem mapping for forest inventory with small-footprint mobile LiDAR”. In: *Forests* 6.12 (2015), pp. 4588–4606 (cit. on p. 80).

[224] Yujie Tang, Jun Cai, Meng Chen, Xuejiao Yan, and Yangmin Xie. “An autonomous exploration algorithm using environment-robot interacted traversability analysis”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4885–4890 (cit. on p. 49).

[225] Youssef Tawk, Cyril Botteron, Aleksandar Jovanovic, and Pierre-André Farine. "Analysis of Galileo E5 and E5ab code tracking". In: *GPS solutions* 16.2 (2012), pp. 243–258 (cit. on p. 82).

[226] Sebastian Thrun. "Probabilistic algorithms in robotics". In: *AI Magazine* 21.4 (2000), pp. 93–93 (cit. on p. 13).

[227] Sebastian Thrun et al. "Robotic mapping: A survey". In: *Exploring artificial intelligence in the new millennium* 1.1-35 (2002), p. 1 (cit. on p. 22).

[228] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press Cambridge, 2000 (cit. on pp. 24, 51).

[229] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. "Robust Monte Carlo localization for mobile robots". In: *Artificial Intelligence* 128.1 (2001), pp. 99–141 (cit. on p. 20).

[230] Sebastian Thrun and Michael Montemerlo. "The graph SLAM algorithm with applications to large-scale mapping of urban structures". In: *The International Journal of Robotics Research* 25.5-6 (2006), pp. 403–429 (cit. on p. 28).

[231] Rafael Toledo-Moreo, Carlos Colodro-Conde, and Javier Toledo-Moreo. "A multiple-model particle filter fusion algorithm for GNSS/DR slide error detection and compensation". In: *Applied Sciences* 8.3 (2018), p. 445 (cit. on p. 82).

[232] Julián Tomaštík, Martin Mokroš, Peter Surový, Alžbeta Grznárová, and Ján Merganič. "UAV RTK/PPK method—an optimal solution for mapping inaccessible forested areas?" In: *Remote sensing* 11.6 (2019), p. 721 (cit. on p. 83).

[233] Anirudh Topiwala, Pranav Inani, and Abhishek Kathpal. "Frontier based exploration for autonomous robot". In: *arXiv preprint arXiv:1806.03581* (2018) (cit. on p. 48).

[234] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. 1999, pp. 298–372 (cit. on p. 31).

[235] Darko Trivun, Edin Šalaka, Dinko Osmanković, Jasmin Velagić, and Nedim Osmić. "Active SLAM-based algorithm for autonomous exploration with mobile robot". In: *IEEE International Conference on Industrial Technology (ICIT)*. 2015, pp. 74–79 (cit. on p. 49).

[236] John W Tukey. "A survey of sampling from contaminated distributions". In: *Contributions to Probability and Statistics* (1960), pp. 448–485 (cit. on p. 31).

[237] Stephen Tully, George Kantor, and Howie Choset. "A unified bayesian framework for global localization and SLAM in hybrid metric/topological maps". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 271–288 (cit. on p. 23).

[238] Spyros G Tzafestas. "Mobile robot control and navigation: A global overview". In: *Journal of Intelligent & Robotic Systems* 91.1 (2018), pp. 35–58 (cit. on p. 38).

[239] Yuko UEKA and Seiichi ARIMA. “Development of Multi-Operation Robot for Productivity Enhancement of Intelligent Greenhouses: For Construction of Integrated Pest Management Technology for Intelligent Greenhouses”. In: *Environmental Control in Biology* 53.2 (2015), pp. 63–70 (cit. on p. 79).

[240] Shimon Ullman. “The interpretation of structure from motion”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 203.1153 (1979), pp. 405–426 (cit. on p. 51).

[241] H. Umari and S. Mukhopadhyay. “Autonomous robotic exploration based on multiple rapidly-exploring randomized trees”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1396–1402 (cit. on p. 48).

[242] Hassan Umari and Shayok Mukhopadhyay. “Autonomous robotic exploration based on multiple rapidly-exploring randomized trees”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1396–1402 (cit. on pp. 94, 96, 107, 111).

[244] Rafael Valencia and Juan Andrade-Cetto. “Active pose SLAM”. In: *Mapping, Planning and Exploration with Pose SLAM*. Springer, 2018, pp. 89–108 (cit. on p. 96).

[245] Jan Van Sickle. *GPS for land surveyors*. CRC press, 2008 (cit. on p. 83).

[246] Fernando Vanegas, Kevin J. Gaston, Jonathan Roberts, and Felipe Gonzalez. “A Framework for UAV Navigation and Exploration in GPS-Denied Environments”. In: *IEEE Aerospace Conference (AERO)*. 2019, pp. 1–6 (cit. on p. 95).

[247] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. “Volumetric next-best-view planning for 3D object reconstruction with positioning error”. In: *International Journal of Advanced Robotic Systems (IJARS)* 11.10 (2014), p. 159 (cit. on p. 97).

[248] Ramiro Velázquez, Edwige Pissaloux, Pedro Rodrigo, et al. “An Outdoor Navigation System for Blind Pedestrians Using GPS and Tactile-Foot Feedback”. In: *Applied Sciences* 8.4 (2018), p. 578 (cit. on p. 57).

[249] W Grey Walter. “A machine that learns”. In: *Scientific American* 185.2 (1951), pp. 60–64 (cit. on p. 4).

[250] Yuze Wang, Xin Chen, and Peilin Liu. “Statistical multipath model based on experimental GNSS data in static urban canyon environment”. In: *Sensors* 18.4 (2018), p. 1149 (cit. on p. 80).

[251] Ulrich Weiss and Peter Biber. “Plant detection and mapping for agricultural robots using a 3D LIDAR sensor”. In: *Robotics and Autonomous Systems* 59.5 (2011), pp. 265–273 (cit. on p. 56).

[252] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995) (cit. on p. 18).

[254] Brian Williams, Mark Cummins, Jose Neira, et al. “A comparison of loop closing techniques in monocular SLAM”. In: *Robotics and Autonomous Systems* 57 (2009), pp. 1188–1197 (cit. on p. 22).

[255] Stefan B Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. “An efficient approach to the simultaneous localisation and mapping problem”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. 2002, pp. 406–411 (cit. on p. 46).

[256] Ling Wu, Miguel Ángel García, Domenec Puig Valls, and Albert Solé Ribalta. “Voronoi-based space partitioning for coordinated multi-robot exploration”. In: *Journal of Physical Agents* 3.1 (2007) (cit. on p. 46).

[257] Chunlei Xia, Tae-Soo Chon, Zongming Ren, and Jang-Myung Lee. “Automatic identification and counting of small size pests in greenhouse conditions with low computational cost”. In: *Ecological Informatics* 29 (2015), pp. 139–146 (cit. on p. 79).

[258] Rui Xu, Wu Chen, Ying Xu, and Shengyue Ji. “A new indoor positioning system architecture using GPS signals”. In: *Sensors* 15.5 (2015), pp. 10074–10087 (cit. on p. 57).

[259] Zhizun Xu, Maryam Haroutunian, Alan J Murphy, Jeff Neasham, and Rose Norman. “An underwater visual navigation method based on multiple ArUco markers”. In: *Journal of Marine Science and Engineering* 9.12 (2021), p. 1432 (cit. on p. 70).

[260] Brian Yamauchi. “A frontier-based approach for autonomous exploration”. In: *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*. 1997, pp. 146–151 (cit. on pp. 48, 95).

[261] Brian Yamauchi. “Frontier-based exploration using multiple robots”. In: *Proceedings of the 2nd International Conference on Autonomous Agents (AAMAS)*. 1998, pp. 47–53 (cit. on p. 48).

[262] Brian Yamauchi, Alan Schultz, and William Adams. “Mobile robot exploration and map-building with continuous localization”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 1998, pp. 3715–3720 (cit. on pp. 46, 48).

[263] Guanci Yang, Zhanjie Chen, Yang Li, and Zhidong Su. “Rapid relocation method for mobile robot based on improved ORB-SLAM2 algorithm”. In: *Remote Sensing* 11.2 (2019), p. 149 (cit. on p. 36).

[264] Xieliu Yang and Suping Fang. “Effect of field of view on the accuracy of camera calibration”. In: *Optik* 125.2 (2014), pp. 844–849 (cit. on p. 99).

[265] Abu Sadat Mohammed Yasin, Md Majharul Haque, Md Nasim Adnan, et al. “Localization of Autonomous Robot in an Urban Area Based on SURF Feature Extraction of Images”. In: *International Journal of Technology Diffusion (IJTD)* 11.4 (2020), pp. 84–111 (cit. on p. 36).

[266] Wonpil Yu and Francesco Amigoni. “Standard for robot map data representation for navigation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on “Standardized Knowledge Representation and Ontologies for Robotics and Automation*". 2014, pp. 3–4 (cit. on p. 22).

[267]Rui Zeng, Yuhui Wen, Wang Zhao, and Yong-Jin Liu. “View planning in robot active vision: A survey of systems, algorithms, and applications”. In: *Computational Visual Media* (2020), pp. 1–21 (cit. on p. 97).

[268]Zichao Zhang. “Active robot vision: from state estimation to motion planning”. PhD thesis. Zurich, Switzerland: Faculty of Business, Economics and Informatics, University of Zurich, 2020 (cit. on p. 50).

[269]Zichao Zhang and Davide Scaramuzza. “Perception-aware receding horizon navigation for MAVs”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2534–2541 (cit. on p. 51).

[270]Zichao Zhang and Davide Scaramuzza. “Beyond point clouds: Fisher information field for active visual localization”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5986–5992 (cit. on p. 51).

[271]Zichao Zhang and Davide Scaramuzza. “Fisher Information Field: an Efficient and Differentiable Map for Perception-aware Planning”. In: *ArXiv* abs/2008.03324 (2020) (cit. on p. 51).

[272]Tai-Xiong Zheng, Shuai Huang, Yong-Fu Li, and Ming-Chi Feng. “Key techniques for vision based 3D reconstruction: a review”. In: *Acta Automatica Sinica* 46.4 (2020), pp. 631–652 (cit. on p. 97).

[273]Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, et al. “State of the art on 3D reconstruction with RGB-D cameras”. In: *Computer Graphics Forum*. Vol. 37. 2. 2018, pp. 625–652 (cit. on p. 126).

[274]Konrad Zuse. “Der Plankalkül”. In: *Ber. Gesellsch. Math. Datenverarbeit.* 1.63 (1972), pp. 1–285 (cit. on pp. 40, 48).

Webpages

[@43]Embassy of the Czech Republic in Ottawa. *ROBOT - the most famous Czech word celebrates 100 years*. 2021. URL: `https://www.mzv.cz/ottawa/en/news_and_events/robot_the_most_famous_czech_word.html` (visited on Jan. 20, 2022) (cit. on p. 3).

[@61]United Nations - Department of Economic and Social Affairs. *Transforming our world: the 2030 Agenda for Sustainable Development*. 2015. URL: `https://sdgs.un.org/2030agenda` (visited on Jan. 24, 2022) (cit. on p. 5).

[@90]Ruoqi He and Chia-Man Hung. *PATHFINDING IN 3D SPACE-A*, THETA*, LAZY THETA* IN OCTREE STRUCTURE*. 2016. URL: `https://ascane.github.io/assets/portfolio/pathfinding3d-report.pdf` (visited on June 6, 2022) (cit. on p. 45).

[@124]Stanford Artificial Intelligence Laboratory and Open Robotics. *Robot Operating System*. 2007. URL: `https://www.ros.org/` (visited on Jan. 19, 2022) (cit. on p. 58).

[@162]Jash Mota. *Dynamic Window Approach local planner*. 2020. URL: http://wiki.ros.org/dwa_local_planner (visited on June 6, 2022) (cit. on p. 43).

[@176]ORB-SLAM3. *ORB-SLAM3: TUM-VI Stereo-Inertial, room1+magistrale1+magistrale5+slides1*. 2020. URL: https://www.youtube.com/watch?v=HyLNq-98LRo (visited on Sept. 30, 2022) (cit. on p. 34).

[@191]International Federation of Robotics. *Robot Density nearly Doubled globally*. 2021. URL: https://ifr.org/ifr-press-releases/news/robot-density-nearly-doubled-globally (visited on Jan. 20, 2022) (cit. on p. 3).

[@215]John Song. *A Comparison of Pathfinding Algorithms*. 2019. URL: https://www.youtube.com/watch?v=GC-nBgi9r0U (visited on June 3, 2022) (cit. on p. 42).

[@218]National Institute of Standards and Technology. *A Walk Through Time - Early Clocks*. 2004. URL: https://www.nist.gov/pml/time-and-frequency-division/popular-links/walk-through-time/walk-through-time-early-clocks (visited on Jan. 21, 2022) (cit. on p. 4).

[@243]TUM University. *Visual-Inertial Dataset*. 2021. URL: https://vision.in.tum.de/data/datasets/visual-inertial-dataset (visited on Sept. 30, 2022) (cit. on p. 34).

[@253]Wikipedia. *Kalman filter*. 2022. URL: https://en.wikipedia.org/wiki/Kalman_filter (visited on Mar. 11, 2022) (cit. on p. 19).