Dissertation presented to the Instituto Tecnológico de Aeronáutica, in partial fulfillment of the requirements for the degree of Master of Science in the Graduate Program of Electronics and Computer Engineering, Field of Systems and Control.

Luiz Eugênio Santos Araújo Filho

MULTI-ROBOT AUTONOMOUS EXPLORATION AND MAP MERGING IN UNKNOWN ENVIRONMENTS

Dissertation approved in its final version by signatories below:

Prof. Dr. Cairo Lúcio Nascimento Júnior

Advisor

Prof. Dr. Pedro Teixeira Lacava

Pro-Rector of Graduate Courses

Campo Montenegro
São José dos Campos, SP - Brazil
2021

Cataloging-in Publication Data
Documentation and Information Division

Santos Araújo Filho, Luiz Eugênio
Multi-robot autonomous exploration and map merging in unknown environments / Luiz Eugênio Santos Araújo Filho.
São José dos Campos, 2021.
98f.

Dissertation of Master of Science – Course of Electronics and Computer Engineering. Area of Systems and Control – Instituto Tecnológico de Aeronáutica, 2021. Advisor: Prof. Dr. Cairo Lúcio Nascimento Júnior.

1. SLAM. 2. Autonomous exploration. 3. Multi-robot systems. 4. Map merging. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

SANTOS ARAÚJO FILHO, Luiz Eugênio. **Multi-robot autonomous exploration and map merging in unknown environments**. 2021. 98f. Dissertation of Master of Science – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

AUTHOR'S NAME: Luiz Eugênio Santos Araújo Filho
PUBLICATION TITLE: Multi-robot autonomous exploration and map merging in unknown environments.
PUBLICATION KIND/YEAR: Dissertation / 2021

Luiz Eugênio Santos Araújo Filho
Praça Marechal Eduardo Gomes, 50
12.228-900 – São José dos Campos–SP

MULTI-ROBOT AUTONOMOUS EXPLORATION AND MAP MERGING IN UNKNOWN ENVIRONMENTS

Luiz Eugênio Santos Araújo Filho

Thesis Committee Composition:

Prof. Dr.	Gabriela Werner Gabriel	President	-	ITA
Prof. Dr.	Cairo Lúcio Nascimento Júnior	Advisor	-	ITA
Prof. Dr.	Marcos R. O. de Albuquerque Maximo	Intern Member	-	ITA
Prof. Dr.	Luciano Buonocore	Extern Member	-	UFMA

ITA

To my parents and family for the support and motivation to complete this work.

Acknowledgments

I would like to thank my family at home, my parents Luiz Eugênio and Ana, my sister Ana Luiza and my brother Guilherme for the love and understanding, through all this, they kept encouraging me, even from afar.

I would like to express my sincerest gratitude to my advisor Prof. Cairo Nascimento for all the time spent providing guidance and motivation to the completion of this work. Thank you professor, your teachings were of great value.

Sincere gratitude to my colleagues at the Laboratório de Máquinas Inteligentes: Daniel Ribeiro, Walber Lima, Larissa Pinto, Pedro Gava, Marcus Ferraz, Michel Leles, Clayton Rodrigues, Pedro Henrique, Elton Sbruzzi, Sérgio Barros and Leonardo Mariga, for all the insights about the research and the countless times we got together for a cup of coffee, these were cherished moments. I thank you all.

I would like to thank Prof. Marcus Victor and Prof. Geraldo Adabo for the permission to use Lab 1224 to perform the experiments used on this work. My sincere thanks to Prof. Sérgio Barros for the permission to use and adapt the Omni robot.

To all the friends I made along my stay in São José dos Campos, thank you for all the experiences and fun we had.

My thanks to CAPES Foundation for the scholarship during my research.

"If I have seen farther than others,
it is because I stood on the shoulders of giants."
— Sir Isaac Newton

Resumo

Na robótica móvel, autonomia é obtida navegando em um ambiente usando sensores embarcados para estimar um mapa e a posição do robô em relação a este mapa. Este problema é conhecido como *Simultaneous Localization and Mapping* (SLAM) e tem sido amplamente abordado para o caso de um único robô nas últimas décadas. Porém, quando o ambiente se torna maior, uma solução multi-robô para o problema de SLAM é mais eficiente apesar do aumento da complexidade e do desenvolvimento de novos problemas, como a coordenação multi-robô.

Este trabalho tem como objetivo desenvolver e implementar uma solução para o problema de SLAM utilizando múltiplos robôs móveis terrestres. Para conseguir isso, cada robô deve explorar de forma autônoma parte do ambiente e seus mapas estimados individuais devem ser fundidos em um único mapa global. A solução proposta usa uma estrutura de SLAM baseada em *scan matching*, conhecida como GraphSLAM, para estimar a pose e o mapa de cada robô individual. Para explorar o ambiente de forma autônoma, a solução propõe o uso de segmentação de mapas por meio de grafos de Voronoi, que geram caminhos livres de colisões e possíveis pontos de destino para exploração. A fusão de mapas é realizada quando os robôs se reúnem para fazer medições inter-robôs para fornecer informações de posição relativa. Para validar a abordagem proposta, experimentos utilizando robôs simulados e reais foram realizados para explorar e mapear ambientes internos de forma autônoma.

Os experimentos do mundo real apresentaram resultados satisfatórios para um ambiente de laboratório e um ambiente de corredor não estruturado e as regras propostas para coordenação, bem como a abordagem de fusão de mapas funcionaram como esperado.

Abstract

In mobile robotics, autonomy is obtained by navigating an environment while using embedded sensors to estimate a map and the robot position with respect to this map. This problem is known as Simultaneous Localization and Mapping (SLAM) and has been extensively addressed for the single-robot case in the last few decades. However, when the environment becomes larger, a multi-robot solution for the SLAM problem is more efficient despite the increased complexity and the development of new problems, such as multi-robot coordination.

This work aims to develop and implement a solution for the SLAM problem using multiple ground mobile robots. To accomplish that, each robot must autonomously explore part of the environment and their individual estimated maps must be merged into a single global map. The proposed solution uses a scan matching-based SLAM framework known as GraphSLAM to estimate pose and map for each individual robot. To autonomously explore the environment, the solution proposes the use of map segmentation via Voronoi Graphs, which generate collision-free paths and possible target points for exploration. Map merging is performed when robots rendezvous to take inter-robot measurements to provide relative position information. In order to validate the proposed approach, experiments using simulated and real robots were carried out to autonomously explore and map indoor environments.

Real world experiments presented satisfactory results for a laboratory environment and a unstructured corridor environment and the proposed rules for coordination as well as the map merging approach worked as intended.

List of Figures

List of Tables

Contents

1 Introduction

The field of mobile robotics has greatly improved ever since the advances and early works on multiple mobile robots were introduced in the 1980s. Most researchers agree that multi-robot systems have several advantages over single-robots (SICILIANO; KHATIB, 2016). However, multi-robot systems have problems that are significantly more difficult (THRUN *et al.*, 2005). These problems vary in the scope of task requirements and sensory capabilities of the available robots (SICILIANO; KHATIB, 2016).

Problems such as limited communication, heterogeneity of the robots, global map merging and optimal coordination are just some of the well known issues when dealing with multi-robot systems (REID, 2016). While advantages fall into the overall task efficiency such as faster completion time than single-robot systems, fault-tolerant capabilities due to redundancy and global map uncertainty reduction by merging individual maps (STACHNISS, 2006).

1.1 Problem Statement

In GPS-denied environments, robots must acquire information about the environment and themselves in order to produce an estimate of its position and the spatial representations of their surroundings (THRUN *et al.*, 2005).

This work will focus on the problems regarding exploration of unknown environments and construction of a global map using multiple mobile robots, specifically ground vehicles. The scope of solution will be limited to two-dimensional (2D) representations of the robot's positions and map. The following problems will be formulated and addressed by this research:

1. Autonomous exploration of unknown environments under pose uncertainty using multi-robots. This problem is known as Active Simultaneous Localization and Mapping (Active SLAM).

2. Estimation of a global map representation of the environment via merging individual maps from different robots.

1.2 Objectives

To solve the problems listed previously, some objectives must be achieved.

1. Development of an algorithm to autonomously guide teams of mobile robots as a proposed solution for the coverage problem.
2. Execute simulated and real world experiments to validate the proposed solution.

1.3 Relevance and Motivations

Over the years, research in mobile robotics have enabled technologies that otherwise were confined into the scope of laboratories and industries to reach consumer level (REID, 2016). The impact robotics have nowadays into household products have greatly increased. An example is the popular vacuum cleaning robots that have gained such notoriety for their practicality and accessibility. Figure 1.1 shows an example the impact that SLAM have on the efficiency of these robots.

FIGURE 1.1 – Efficiency in robot vacuums cleaners. The left image shows a random navigation while the right image shows navigation aided by SLAM. Shown in green are the paths taken by two commercial robots: iRobot Roomba (left) and Neato XV-11 (right) (REID, 2016).

Technologies like that of the right image on Figure 1.1 are the result of research such as this work. Academic research is of great importance to the advance of technology around the world.

On this context, the development of this work can be applied into various segments that have the necessity of autonomous mobile robots, such as delivery drones, hostile environment search and rescue applications, robotic missions to explore other planets surface and so on.

1.4 Contributions

The contributions of this work are:

- Implementation of a cooperative, strongly centralized, weakly coordinated, aware multi-robot architecture using simulated and real world robots;
- Target selection and path planning for autonomous exploration using Voronoi graphs and simple coordination rules;

1.5 Structure of this work

Chapter 1 contains the introduction to this work, presenting the problems to be solved while achieving the listed objectives. The relevance of the research as well as the motivations that lead to the realization of this work are presented in this chapter.

Chapter 2 reviews the bibliography to established solutions for the problem. The scope of active single-robot SLAM is presented, followed by the expansion to multi-robots along with discussion on the classification and taxonomy of these systems.

Chapter 3 brings the specifications of the proposed solution, the characteristics and constraints of the algorithm and the setup used for experimentation. The simulation details and description of the real robots used are presented as well.

Chapter 4 presents the results from simulated and real world experiments, and a discussion is included to deliberate immediate aspects of the results.

Chapter 5 presents conclusions and some suggestions for future improvements and extension of this research.

Links for the source code used in this work as well as videos showing the experiments are listed below:

- ftp://labattmot.ele.ita.br/ele/lesaf/My_Software/,
- ftp://labattmot.ele.ita.br/ele/lesaf/My_Videos/,
- https://github.com/Intelligent-Machines-Lab/Active_MRSLAM.

2 Literature Review

In this chapter, a study of problems associated with the development of Multi-Robot SLAM is presented, along with an overview of the most common approaches of solutions to these problems.

2.1 Autonomous mobile robotics

Mobile robotics is a large field of research in which computer controlled devices are perceiving and moving in the physical world. The idea of intelligent machines that are capable of doing human tasks, at least with similar performance, has an enormous potential to change society (THRUN *et al.*, 2005). Robots that are capable of carrying their tasks with little to no human intervention, in other words autonomously, are challenging to develop even for the simplest of tasks.

The complexity of a task for an autonomous mobile robot has a strong relation to the information that is given *a priori* to its knowledge about the world and itself. The more a robot has to determine about the variables surrounding its task, the more difficult will be its implementation in order to reliably aid or substitute a human.

The basic tasks any autonomous robot should overcome, according to Stachniss (2006), are depicted in Figure 2.1 below. Looking at this picture, the maximum level of autonomy is the combination of the three basic tasks in which the robot has to estimate its knowledge of the environment, its own location in the environment and its next motion command.

These tasks can not be solved independently of each other. For example, before a robot can estimate a map given sensor observations, it needs to know from which location in the map the sensor observations where taken. At the center of Figure 2.1 are the integrated approaches in which the robot has to autonomously explore the environment and simultaneously build a map and localize itself (STACHNISS, 2006).

In Stachniss (2006) is stated that the problem in most robotic tasks is in the uncertainty regarding the several estimates. Common sources to these uncertainties are:

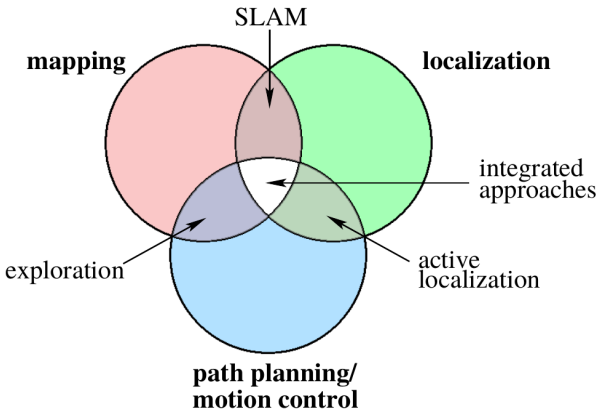

FIGURE 2.1 – Basic tasks in autonomous mobile robotics (STACHNISS, 2006).

- sensor measurements are noisy;
- mathematical models are not a complete representation of the real world;
- assumptions can be made to simplify the task, therefore becoming approximations.

Since it is not possible to escape these problems, the challenge is to accommodate most of the uncertainty in a way the estimates are good enough to finish the task. In the next section some methods to represent the state of the robot and its surrounding environment are discussed.

2.2 State representation

In order for a robot to acquire knowledge, a representation of itself and its environment is necessary, for instance, to plan and take reliable actions. Environments and the robot pose are characterized by state. A state can be described as the collection of all aspects that can impact the future of the robot and its environment. State variables can be static, like the location of walls in buildings, or dynamic, such as the location of people in the vicinity of a robot (THRUN *et al.*, 2005).

2.2.1 Robot model

For rigid mobile robots, the pose is usually described as six state variables, three for their Cartesian coordinates and three for their angular orientation (roll, pitch, yaw). In the case of rigid mobile robots that navigates in a 2D planar environment, their state can be represented by only three variables, their (x, y) location in the plane and their heading direction (yaw). Occasionally, the robot linear and angular velocities can be added to the state as well (THRUN *et al.*, 2005).

These aforementioned robot state representations are very reliable, hence the majority of research in the field has adopted. However, that is not the case for environment state representation, and different types can be found on papers and research around the world.

2.2.2 Environment model

Environment state representation are maps in which the robot can make use of for localization, navigation and planning. The most common types are feature maps, geometric maps and grid maps (STACHNISS, 2006). Figure 2.2 shows three aerial photographs of urban areas and their mapped counterparts: a 3D surface map, a 3D point cloud map and a 2D occupancy grid map.

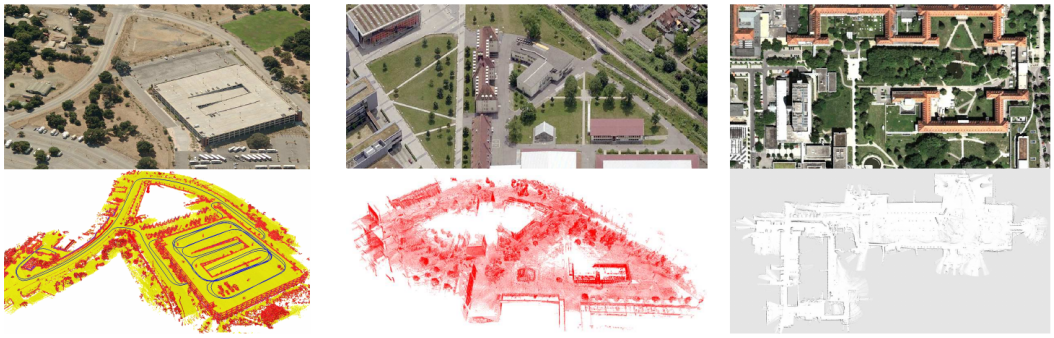

FIGURE 2.2 – (Left) A 3D map of the Stanford parking garage acquired with an instrumented car (bottom), and the corresponding satellite view (top). This map has been subsequently used to implement an autonomous parking behavior. (Center) Point cloud map acquired at the university of Freiburg (courtesy of Kai. M. Wurm) and relative satellite image. (Right) Occupancy grid map acquired at the hospital of Freiburg. Top: a bird's eye view of the area, bottom: the occupancy grid representation. The gray areas represent unobserved regions, the white part represents traversable space while the black points indicate occupied regions (GRISETTI *et al.*, 2010).

Feature maps are stored as a set of distinguishable features like points or lines and use very little memory resources. The level of detail is compromised since it is not a very complex representation of space. Geometric maps are as compact as feature maps, but they represent obstacles as geometric objects, like circles or polygons. Grid maps discretize the environment in cells, each cell stores information about the area or volume it covers. When the information contained on each cell is the probability of the cell being occupied or not, the map is called an occupancy grid (STACHNISS, 2006). Although occupancy grids make use of more memory to store information, they represent the environment with great detail and can be used very reliably to plan obstacle-free paths.

In the next section, state of the art integrated approaches for SLAM using autonomous exploration are discussed for the case of a single robot performing the task.

2.3 Single-robot SLAM using autonomous exploration

2.3.1 SLAM

Simultaneous localization and mapping (SLAM) consists of the concurrent construction of a model of the environment, and the estimation of the state of the robot moving within it (CADENA *et al.*, 2016). It is referred as one of the most fundamental problems in robotics and many methods for solving it were developed over the years. Thrun *et al.* (2005) classify solutions in either filtering or smoothing, also known as the online SLAM or full SLAM solution, respectively.

2.3.1.1 Online SLAM

Filtering approaches model the problem as a state estimation where the state consists of the current robot pose and the map. The state is augmented as new measurements are taken and incorporated in the map. The most popular techniques use a probabilistic approach such as Kalman filters, particle filters and information filters (GRISETTI *et al.*, 2010). These techniques form the basis for well established SLAM algorithms, such as EKF-SLAM, FastSLAM and DP-SLAM (GRISETTI *et al.*, 2010). Figure 2.3 shows a graphical representation of the state variable and its dependencies, where x is the robot pose, u is the control applied, z is the measurement taken, m is the map and subscripts are the time steps.

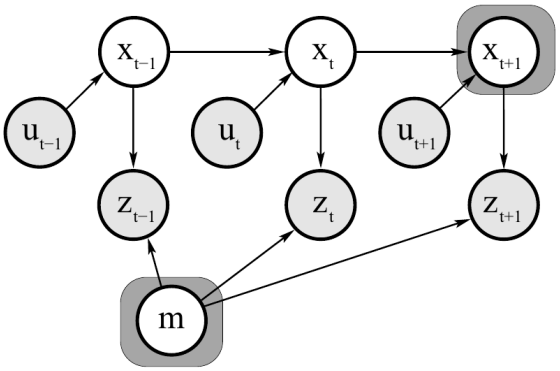

FIGURE 2.3 – Graphical visualization of the online SLAM problem. A variable pointing to another one means it has direct influence over the state of the latter. The shaded box denotes the estimation objective of the current iteration (THRUN *et al.*, 2005)

2.3.1.2 Full SLAM

The solutions characterized as smoothing can also be referred in the literature as Smoothing And Mapping (SAM). In this approach, the algorithm maintains an estimate

of the full trajectory of the robot as well as the map model. Full SLAM solutions usually rely on least-square error minimization on data structures that abstract raw sensor measurements (GRISETTI *et al.*, 2010). Solutions such as the GraphSLAM algorithm are the current standard and state of the art approaches for most applications of SLAM. Figure 2.4 follows the same style of Figure 2.3 but for the full SLAM problem.

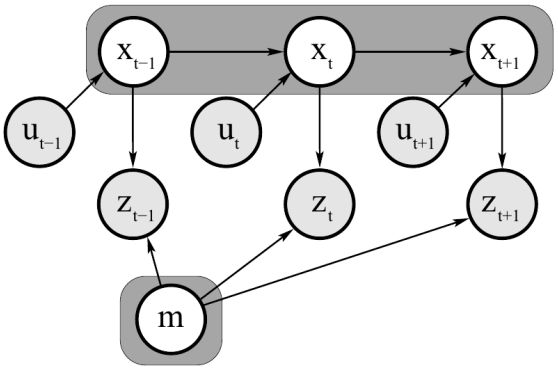

FIGURE 2.4 – Graphical visualization of the full SLAM problem. A variable pointing to another one means it has direct influence over the state of the latter. The shaded box denotes the estimation objective of the current iteration (THRUN *et al.*, 2005)

2.3.2 Autonomous exploration

2.3.2.1 Active SLAM

SLAM by itself is usually a problem that is carried out passively by the robot given the sensor data, that is, the robot is not deciding on where to collect sensor data. The problem of controlling the robot movements in order to minimize the uncertainty of the map representation and of the robot localization is usually named active SLAM. This inserts a new problem to the structure of SLAM since it falls on the exploration-exploitation dilemma, i.e., finding a balance between visiting new places (exploration) and reducing the uncertainty by revisiting known areas (exploitation) (CADENA *et al.*, 2016).

Currently, many approaches are used for solving active SLAM. Most of them formulate the problem as a decision making problem, such as Theory of Optimal Experimental Design (TOED) which selects future actions based on the predicted map uncertainty (CADENA *et al.*, 2016). Information theoretic approaches have been applied to solve the decision making problem using the notion of information gain about potential actions (CADENA *et al.*, 2016). The problem can be modeled as a Partially Observable Markov Decision Process (POMDP), that generalize such decision making processes under action and observation uncertainties (THRUN *et al.*, 2005). Generally, the problem of choosing the robot path go can be summarized in the following three steps:

- the robot identifies possible locations to explore or exploit in its current map,
- the robot computes the utility of visiting each location and selects the action with highest utility value,
- the robot carries out the selected action and decides if it is necessary to continue or terminate the task.

The active SLAM problem also leads to another decision making problem, i.e. when to stop the task. For real world applications, the task should have an intended goal to achieve, so a stopping criteria is a necessity since, eventually, it is provable that more information would lead to a diminishing return effect but also, in case of contradictory information, to an unrecoverable state. For this reason, most criteria are application-driven and difficult to compare across systems (CADENA *et al.*, 2016). Figure 2.5 shows a situation where the robot has several locations to choose and the utility associated with each one.

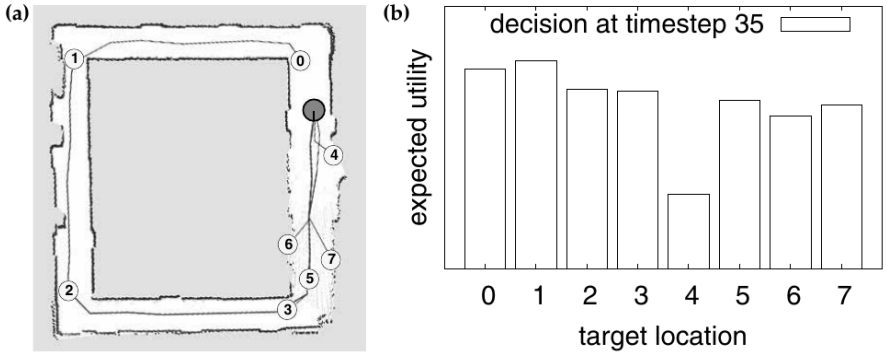

FIGURE 2.5 – (a) Exploration actions considered by the robot given the current map estimate and (b) the expected utility of each action, in this case, action 1 is selected to maximize the expected utility (THRUN *et al.*, 2005).

2.3.2.2 Exploration on static vs. dynamic environments

Another characteristic that has a substantial impact on the difficulty of the SLAM problem is the type of the environment. Environments can be static or dynamic and most real environments are dynamic, with changes occurring at a range of different speeds (THRUN *et al.*, 2005).

Static environments are environments where the only dynamic state is the robot pose, that is, only the robot moves in a static environment. All other objects in the environments remain at the same location for the duration of the task (THRUN *et al.*, 2005).

Dynamic environments possess objects other than the robot whose location or configuration changes over time during the task. Of particular interest are changes that persist over time, and that impact more than a single measurement reading (THRUN *et al.*, 2005).

Active SLAM in dynamic environments is more difficult than in static ones. In certain situations sensor data can be filtered so as to eliminate changing obstacles. In this case, the dynamic part does not incorporate the map. In some cases, these dynamic obstacles must be considered and dealt with in the planning step (THRUN *et al.*, 2005). Algorithms that consider measurement readings and avoid obstacles to reach a specific objective can be used, such as Dynamic Window Approach (DWA) and Vector Field Histogram (VFH) controller (ROBIN; LACROIX, 2016a).

2.3.3 Particular case of Active 2D SLAM

In recent years, the active 2D SLAM has attracted the interest of researchers, allowing the development of robust and efficient algorithms such as GraphSLAM, besides the formulation for this solution being proposed by Lu and Milios in 1997 (GRISETTI *et al.*, 2010).

A graph-based SLAM approach uses maximum likelihood estimation (MLE) to estimate the best configuration of a robot trajectory that aligns sensor measurements. On a graph structure, the nodes are the robot poses and the edges are constraints formed by measurements between poses (REID, 2016). In general, these constraints are spatial relationships between poses and usually are represented in the form of transformations from one pose to another. In the case of 2D SLAM this is a rigid transformation (rotation and translation) (REID, 2016). Figure 2.6 illustrates the relationship between two poses in a graph, $\mathbf{x}_i$ and $\mathbf{x}_j$.

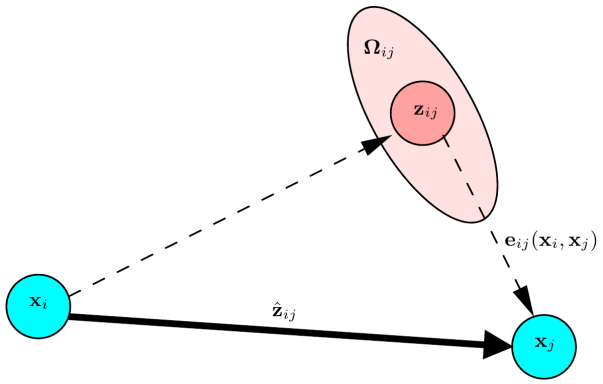

FIGURE 2.6 – Example of an edge connecting the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. An edge is fully characterized by its error function $\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ and by the information matrix $\mathbf{\Omega}_{ij}$ of the measurements that accounts for its uncertainty (GRISETTI *et al.*, 2010).

In Figure 2.6, an edge is produced from the transformation $\mathbf{z}_{ij}$, which is computed by maximizing the overlap from the real measurements taken in i with the ones acquired from j. From the relative position of the estimated poses, it is possible to compute the expected transformation $\hat{\mathbf{z}}_{ij}$ that represents $\mathbf{x}_j$ seen in the frame of $\mathbf{x}_i$ (GRISETTI *et al.*,

2010). The log-likelihood l_{ij} of a transformation $\mathbf{z}_{ij}$ is given by:

$$\mathrm{l}_{ij} \propto \left[\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right]^T \mathbf{\Omega}_{ij} \left[\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right] \tag{2.1}$$

The error function $\mathbf{e}_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right)$ will be the difference between the expected transformation and the real one.

$$\mathbf{e}_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{2.2}$$

Let $\mathcal{C}$ be the set of all pair of indices for which a constraint exists. The maximum likelihood approach will find the configuration of nodes $\mathbf{x}^*$ that minimizes the negative log likelihood $\mathbf{F}(\mathbf{x})$ of all the transformations:

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathbf{F}_{ij}} \tag{2.3}$$

Resulting on the nonlinear least-squares form:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{F}(\mathbf{x}) \tag{2.4}$$

There are two key procedures in this method. The first one is finding the maximum overlap between two sets of measurements, usually referred to as the "front-end". The second one is solving Equation (2.4), which is referred to as the "back-end" (CADENA *et al.*, 2016).

2.3.3.1 Scan Matching

In recent years research in 2D SLAM took advantage of the increase in availability of 2D laser range finders, which was a result of the price reduction for Light Detection and Ranging (LIDAR) scanners (SHADE, 2011). LIDAR technology consists of actively measuring distances to objects by emitting beams of light and computing the time for the reflected beam to return. If this is done in a controlled way, the beams can produce a profile of 360 degrees around the system. A LIDAR scanner can gather accurate data for a low price and low power consumption. Furthermore, the scanning nature of this sensor can make the procedure of finding overlaps in measurements a precise method of computing transformations (GRISETTI *et al.*, 2010). Figure 2.7 shows an example of how measurements take place in a LIDAR scanner.

The idea of finding the best overlap in consecutive LIDAR scans measurements is usu-

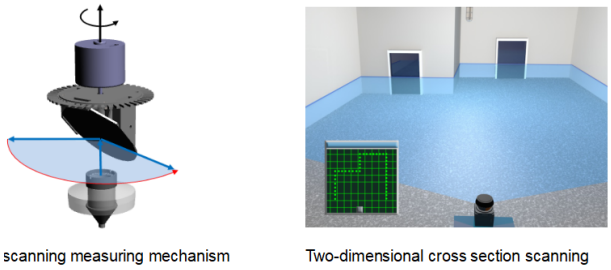

FIGURE 2.7 – Left: Model showing the mechanism to direct the beam of light. Right: Blue rays are representing the beams of light (AKUSENSE, 2021).

ally referred to as scan matching. Many techniques were developed in recent years that focus on producing a rigid transformation (translation and rotation) between two scans coordinate frames. One of the most used techniques is the Iterative Closest Point (ICP). ICP takes two scans, and an initial estimate of the transformation (typically this will come from a unreliable measurement of wheel odometry). Each point in the first scan is associated with its nearest spatial neighbour in the second scan and the estimated transformation is iteratively modified such that the sum of neighbour-to-neighbour distances is reduced. The result is a rigid transformation which results in the two scans being aligned (SHADE, 2011). This process is depicted in Figure 2.8 below, in which $\mathbf{p}_0$ and $\mathbf{p}_1$ are two consecutive poses.

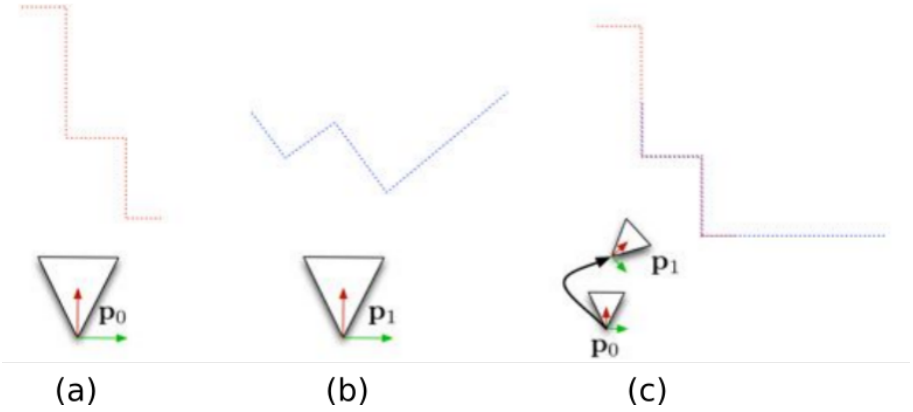

FIGURE 2.8 – Scan matching. (a) First pose in local frame (b) second pose in local frame (c) scans matched and second pose as seen from first pose's local frame (SHADE, 2011).

2.3.3.2 Solving the nonlinear least-squares equation

Reid (2016) describes two main methods of solving the least-squares optimization problem: first order gradient-based iterative methods, and second-order direct methods.

Direct methods have noted structural similarities with the bundle adjustment problem (BA) from computer vision, in which the position of a camera can be estimated by minimizing reprojection errors in 3D points. Most notably was the development of in-

cremental SAM (iSAM) by Dellaert and Kaess which exploited the sparsity in the batch least-squares optimization by using sparse QR matrix factorizations applied to online SLAM (REID, 2016).

Iteractive methods explore classical gradient-based approaches like Gauss-Newton, Gauss-Seidel relaxation, Stochastic Gradient Descent (SGD), among others. These techniques are well grounded mathematically and some are very robust to poor initial estimates (REID, 2016).

If a good initial guess of poses is provided, the popular Gauss-Newton can be used to compute the numerical solution of Equation (2.4) (GRISETTI *et al.*, 2010). The Gauss-Newton is an optimization algorithm that aims to reduce the total residual cost of the system (REID, 2016). The idea is to approximate the error function by its first order Taylor expansion around the current initial guess $\breve{\mathbf{x}}$ (GRISETTI *et al.*, 2010):

$$\mathbf{e}_{ij}\left(\breve{\mathbf{x}}_i + \boldsymbol{\Delta}\mathbf{x}_i, \breve{\mathbf{x}}_j + \boldsymbol{\Delta}\mathbf{x}_j\right) = \mathbf{e}_{ij}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) \tag{2.5}$$

$$\simeq \mathbf{e}_{ij} + \mathbf{J}_{ij}\boldsymbol{\Delta}\mathbf{x}. \tag{2.6}$$

where $\mathbf{J}_{ij}$ is the Jacobian of $\mathbf{e}_{ij}(\mathbf{x})$ and $\mathbf{e}_{ij} = \mathbf{e}_{ij}(\breve{\mathbf{x}})$. Substituting Equation (2.6) in Equation (2.3) results in:

$$\begin{aligned}
&\mathbf{F}_{ij}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) \\
&\quad = \mathbf{e}_{ij}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x})^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) && (2.7)\\
&\quad \simeq \left(\mathbf{e}_{ij} + \mathbf{J}_{ij}\boldsymbol{\Delta}\mathbf{x}\right)^T \boldsymbol{\Omega}_{ij} \left(\mathbf{e}_{ij} + \mathbf{J}_{ij}\boldsymbol{\Delta}\mathbf{x}\right) && (2.8)\\
&\quad = \underbrace{\mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathbf{c}_{ij}} + 2\underbrace{\mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{b}_{ij}} \boldsymbol{\Delta}\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}^T \underbrace{\mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{H}_{ij}} \boldsymbol{\Delta}\mathbf{x} && (2.9)\\
&\quad = \mathbf{c}_{ij} + 2\mathbf{b}_{ij}\boldsymbol{\Delta}\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}^T \mathbf{H}_{ij} \boldsymbol{\Delta}\mathbf{x} && (2.10)
\end{aligned}$$

and $\mathbf{F}(\mathbf{x})$ in Equation (2.3) can be written as:

$$\begin{aligned}
\mathbf{F}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) &= \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{F}_{ij}(\breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}) && (2.11)\\
&\simeq \sum_{\langle i,j \rangle \in \mathcal{C}} \mathrm{c}_{ij} + 2\mathbf{b}_{ij}\boldsymbol{\Delta}\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}^T \mathbf{H}_{ij} \boldsymbol{\Delta}\mathbf{x} && (2.12)\\
&= \mathrm{c} + 2\mathbf{b}\boldsymbol{\Delta}\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}^T \mathbf{H} \boldsymbol{\Delta}\mathbf{x} && (2.13)
\end{aligned}$$

The quadratic form in Equation (2.13) is obtained from Equation (2.12) by setting $c = \sum c_{ij}$, $\mathbf{b} = \sum \mathbf{b}_{ij}$ and $\mathbf{H} = \sum \mathbf{H}_{ij}$. It can be minimized in $\boldsymbol{\Delta}\mathbf{x}$ by solving the linear

system:

$$\mathbf{H}\,\boldsymbol{\Delta}\mathbf{x}^{*} = -\mathbf{b} \tag{2.14}$$

Matrix $\mathbf{H}$ is the information matrix of the system, which is sparse by construction, with non-zero elements between poses connected by a constraint. This allows one to solve Equation (2.14) using sparse Cholesky factorization (GRISETTI *et al.*, 2010).

The linearized solution is then obtained by adding to the initial guess the computed increments:

$$\mathbf{x}^{*} = \breve{\mathbf{x}} + \boldsymbol{\Delta}\mathbf{x}^{*} \tag{2.15}$$

Computing Equations (2.13), (2.14) and (2.15) in every iteration is the basic procedure to the Gauss-Newton algorithm, in this case, proposed for the special case of the optimization of pose graphs in SLAM (GRISETTI *et al.*, 2010). As the optimization occur, the pose graph is updated in the form of correcting the positions of the nodes, since the constraints act like "springs" that pushes and pulls on the nodes it connects to (CADENA *et al.*, 2016). Figure 2.9 below shows the visual effect on the graph structure after the optimization. Blue dots are depicted as nodes and red lines are the edges of the graph.

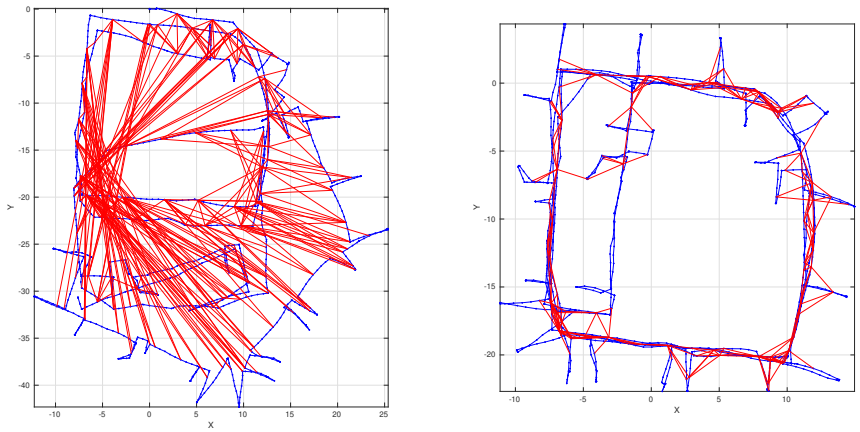

FIGURE 2.9 – Left: Pose graph structure before optimization. Right: After optimization (MATHWORKS, 2021a).

2.3.3.3 Loop Closure

The pose graph used in Figure 2.9 is from the Intel Research Lab Dataset and was generated from collecting wheel odometry and a 2D laser range finder sensor information in an indoor lab (GRISETTI *et al.*, 2010). This example shows 1228 nodes and 1483 edges, from which 256 are edges formed by the robot revisiting known locations. In other words,

they are not consecutive scans transformations, but well matched pairs found in the history of past stored scans measurements, called loop closure (GRISETTI *et al.*, 2010).

Loop closure detection is still an open problem in the context of SLAM. Matching the current measurement with all previous scans at every iteration, would be extremely inefficient and error prone (GRISETTI *et al.*, 2010). Many methods have been proposed to avoid such brute-force approach, mostly histogram-based matching, feature detection in scan data, and using machine learning (HESS *et al.*, 2016).

Grisetti *et al.* (2010) suggests that the optimization of a pose graph is performed whenever a loop closure is detected, for reasons of efficiency and reducing the computational cost of the SLAM task. Figure 2.10 shows the interaction between the main blocks of the GraphSLAM algorithm.

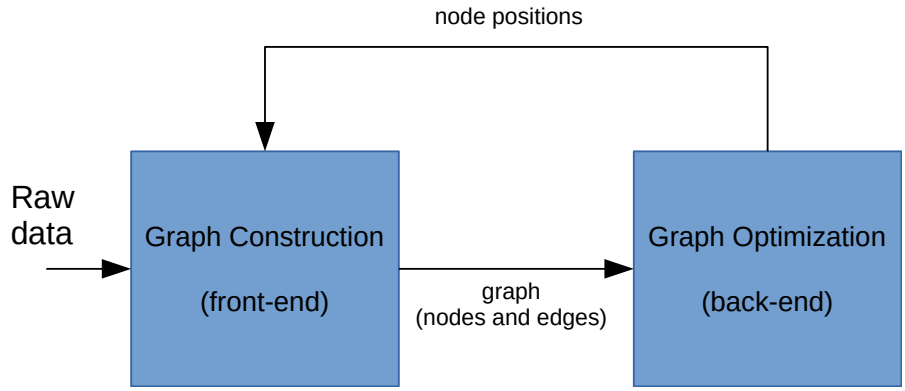

FIGURE 2.10 – GraphSLAM interaction between front-end and back-end. The state of the pose graph is incrementally built upon the front-end step and optimized on the back-end step (CADENA *et al.*, 2016)

2.3.3.4 Mapping with known poses

One common approximation made by GraphSLAM algorithms is the omission of the map in the estimation process, which optimizes only the robot poses. This approximation is that measurements are conditionally independent given the robot pose, and it holds well for relatively accurate sensors, such as LIDAR scanners, attributing most of the uncertainty to the robot pose (REID, 2016).

To compute a map, the so-called mapping with known poses is the procedure in which measurements taken at each iteration are incorporated into a map representation of the environment, given that the robot pose for that corresponding reading is known. A probabilistic algorithm used extensively in 2D laser-based SLAM is the occupancy grid mapping, that aims to represent the environment by a dense cell grid structure (THRUN *et al.*, 2005). The algorithm takes the current pose, the current scan and the previous map

state, resulting in the formulation:

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(\mathbf{m}_i|z_{1:t}, x_{1:t}) \tag{2.16}$$

where $z_{1:t}$ is the history of measurements and $x_{1:t}$ is the history of poses. The problem breaks down to estimating the probability of each grid cell $\mathbf{m}_i$, which is a binary problem with static state for the values occupied ($p(\mathbf{m}_i = 1)$) and free ($p(\mathbf{m}_i = 0)$) (THRUN *et al.*, 2005). For efficiency, the log odds representation of occupancy is preferred to avoid numerical instabilities for probabilities near zero or one:

$$l_{t,i} = \log \frac{p\left(\mathbf{m}_i \mid z_{1:t}, x_{1:t}\right)}{1 - p\left(\mathbf{m}_i \mid z_{1:t}, x_{1:t}\right)} \tag{2.17}$$

The probabilities can be recovered from the log odds ratio:

$$p\left(\mathbf{m}_i \mid z_{1:t}, x_{1:t}\right) = 1 - \frac{1}{1 + \exp\left\{l_{t,i}\right\}} \tag{2.18}$$

The implementation of the algorithm is very straightforward and efficient. Every time a scan is to be inserted into the probability grid, a set of grid points for hits and a disjoint set for misses are computed. For every hit, we insert the closest grid point into the hit set. For every miss, we insert the grid point associated with each pixel that intersects one of the rays between the scan origin and each scan point, excluding grid points which are already in the hit set (HESS *et al.*, 2016). By doing this, it is only necessary to compute a sum in order to update a cell based on a measurement, adding Equation (2.17) to the prior state (STACHNISS, 2006). Figure 2.11 shows how the cells are covered by a scan with six rays of measurement.

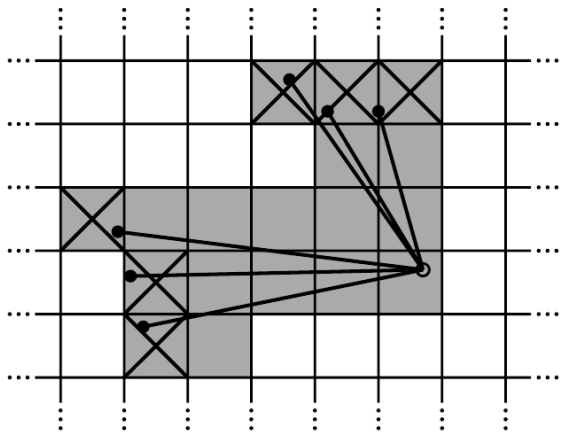

FIGURE 2.11 – Example of a scan over the cells. Hits are shaded and crossed out and misses are shaded only (HESS *et al.*, 2016).

Occupancy grid mapping can filter out dynamic obstacles like people or other robots very well (THRUN *et al.*, 2005). Because of the incremental nature of the update step, areas that captured obstacles only by a few iterations tend to quickly converge to estimates that are close to the two extremes, 1 or 0. In other words, the state that is captured more often will prevail (THRUN *et al.*, 2005). Figure 2.12 shows measurements taken in a open space with a scan of 180 degrees mapped around a robot. The darker a grid cell is, the more likely it is to be occupied.

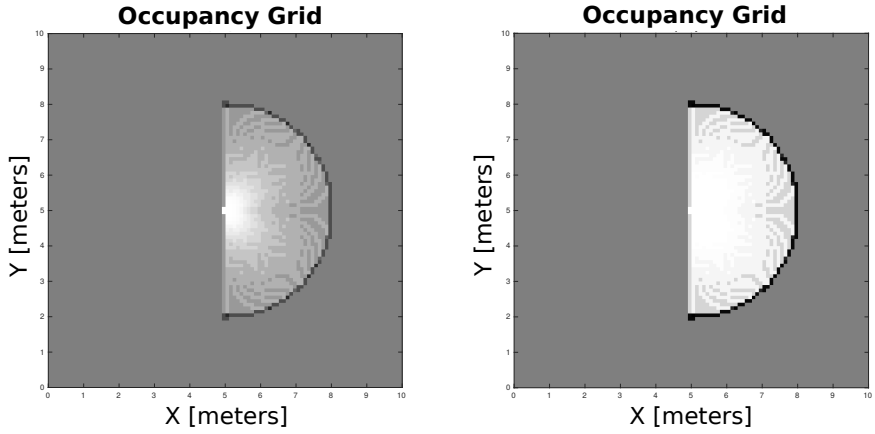

FIGURE 2.12 – Example of occupancy grid mapping with the robot stationary. Left: state of the map after one scan is inserted, Right: state of the map after four scans are inserted (MATHWORKS, 2021b)

2.3.3.5 Exploration

Exploration in 2D still have several problems to be addressed in order to become impactful in real applications (CADENA *et al.*, 2016). A very simple and popular approach for this problem is a crude approximation of the Information Gain in occupancy grid maps (THRUN *et al.*, 2005). It consists in marking cells that have been updated at least once as "explored" and all others as "unexplored", thus the gain becomes a binary function (THRUN *et al.*, 2005). This assumption pushes the robot towards unexplored terrain in the form of the so-called frontier-based exploration (THRUN *et al.*, 2005).

Frontiers are given as the borders between explored and unexplored space. In many approaches, the robot is assigned to a frontier target based on a utility function that takes into account the expected path cost or travel time (WURM *et al.*, 2008). Figure 2.13 shows a frontier border in an occupancy grid map.

The utility function that is computed for each frontier can make use of path planning algorithms. The majority of these utility functions are greedy, since it involves moving to a location along a minimum cost path, requiring simple yet effective computation (THRUN

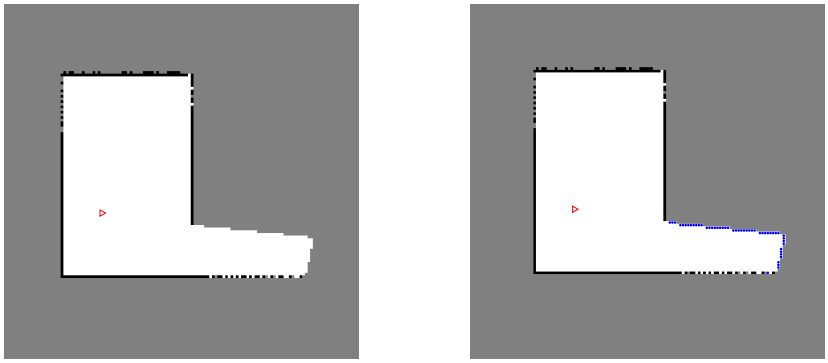

FIGURE 2.13 – Occupancy grid map, free cells (white), occupied cells (black), unknown cells (gray) and robot pose (red triangle). Left: Estimated map. Right: Map showing the frontier cells (blue).

et al., 2005). Some techniques make use of the binary information gain map, setting each location as a potential exploration via value iteration and applying a hill climbing to determine the optimal path leading directly to the nearest unexplored frontier (THRUN *et al.*, 2005). An informed search can be used as well for computing the path and utility of a target, e.g., the A* algorithm is a very popular choice as it estimates the expected travel distance between the robot and the target as well as a collision free path (THRUN *et al.*, 2005).

After defining a path, the robot should perform the necessary motion commands to follow the path or reach the target destination, a popular way to do this is using a path following controller (CADENA *et al.*, 2016). A very straightforward approach is the Pure Pursuit controller, where, given the path points and the current robot pose, a body-referenced velocity vector can be computed each time step in order to reach the last path point (COULTER, 1992). At the core of the controller is the regulator parameter to be tuned, the distance the robot should look for path nodes, the so-called lookAhead distance.

The lookAhead parameter creates a virtual circle of radius equal to its value, each point the reference path intersects with the circle is a candidate point for the robot to follow and the chosen point will be the one with the smaller angle difference with the robot heading (COULTER, 1992). Figure 2.14 shows the geometry used in the controller. The virtual arc formed from the robot to the lookAhead chosen point is shown in green.

The robot is depicted in blue with the Y axis indicating its heading direction. The lookAhead point is in red and have coordinates (x, y) from the perspective of the robot. From the smaller triangle, Equation (2.19) can be obtained and Equation (2.20) comes from the proposed constraint. Since the arc is not unique, the one selected must obey the

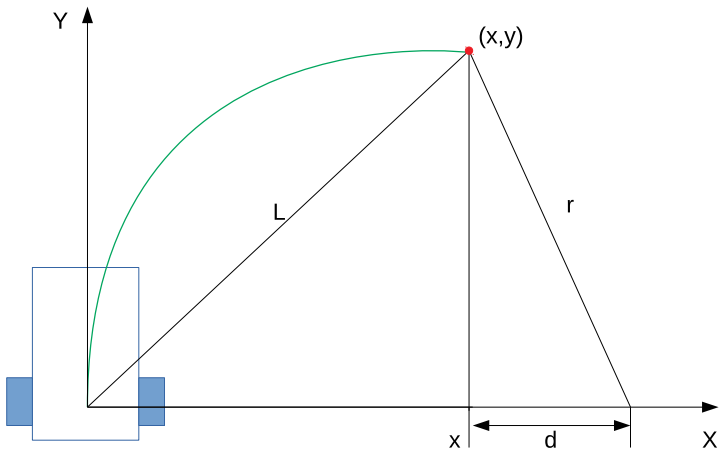

FIGURE 2.14 – Pure pursuit geometry (COULTER, 1992)

circle with center $(x + d, 0)$, image is not to scale (COULTER, 1992):

$$x^2 + y^2 = L^2 \tag{2.19}$$

$$x + d = r \tag{2.20}$$

Calculation of Equation (2.22) is straightforward from Equations (2.19) and (2.20).

$$r = \frac{L^2}{2x} \tag{2.21}$$

$$\gamma = \frac{1}{r} = \frac{2x}{L^2} \tag{2.22}$$

This final equation relates the curvature of the arc which joins the robot position and the lookAhead point. This curvature is used to define a proportional constant between linear and angular velocity (COULTER, 1992), as shown in Equation (2.23):

$$\omega = \gamma v \tag{2.23}$$

Using Equation (2.23) it is possible to fix v and define a maximum value for ω. Furthermore, the parameters of the Pure Pursuit controller are the lookAhead distance, a fixed v and a maximum ω.

Figure 2.15 shows that different values of lookAhead have an impact on the actual robot trajectory along the reference path. The robot is represented as the red circle with the black rectangle inside (heading), the blue line is the actual trajectory, the dashed gray line is the reference path and the orange line show the next path node the robot chose to follow due to the lookAhead parameter.

After describing all the previous steps to autonomously guide the robot, while per-

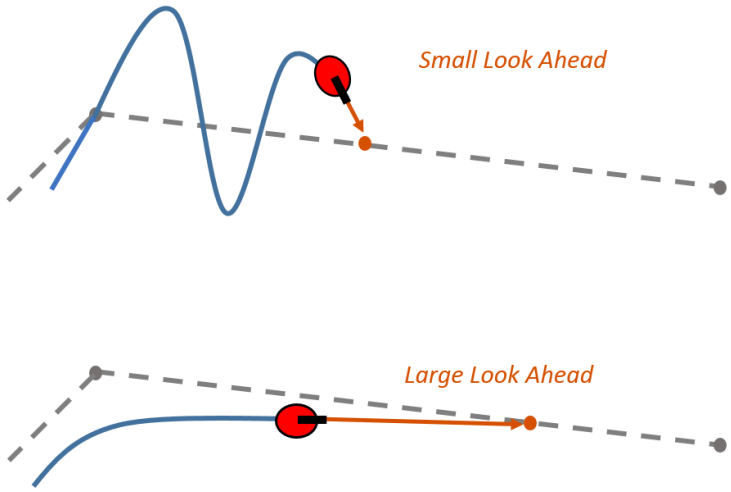

FIGURE 2.15 – Impact of the lookAhead parameter on the resulting robot trajectory. Top: Oscillatory behavior of a trajectory with a small lookAhead value. Bottom: Smooth trajectory along the expected path by using a larger lookAhead value (MATHWORKS, 2021e).

forming SLAM, the robot must have an application or purpose to achieve (CADENA *et al.*, 2016). A common criterion for terminating the task is to stop when there are no more frontier cells to explore (THRUN *et al.*, 2005). Another approach is to stop after a certain percentage of the total area is reached. In this case, the information of total area should be given *a priori* (PATEL *et al.*, 2020). The latter case is a task commonly called *coverage*, in which the objective is to maximize the mapped area and an example of this kind of application is a household vacuum cleaner robot (PATEL *et al.*, 2020).

In the next section the case of multiple robots performing autonomous exploration and SLAM is discussed along with the problems associated with it and the additional complexity of common solution approaches.

2.4 Multi-robot SLAM using autonomous exploration

Multi-robot SLAM (MR-SLAM) is a natural extension to the single-robot SLAM problem, in a way that it is not a trivial upgrade. Most documented problems associated with the nature of MR-SLAM are (REID, 2016):

1. Heterogeneous teams of robots,
2. Limited communication range and bandwidth,
3. Global map merging,
4. Relative pose computation,

5. Sensor interference (cross-talk),

6. Latency and out-of-order package communication.

However, a number of advantages is often suggested by the use of MR-SLAM (STACHNISS, 2006), to name a few:

1. Cooperation has the potential to accomplish a task faster than a single robot,

2. Redundancy introduces fault-tolerant nature to the system,

3. Merging individual maps to compensate for global map uncertainty.

Generally speaking, MR-SLAM can be characterized as a set of robots operating in the same environment. However, it is not easy to give a definition of the level of autonomy that is required for multi-robots to be simply regarded as a generalization of the single-robot case and different approaches need to be precisely characterized in terms of assumptions about the environment and in terms of the internal system organization (FARINELLI *et al.*, 2004). Researchers have proposed classification and taxonomy of MR-SLAM systems based on various aspects, the following subsections describe some definitions.

2.4.1 Taxonomy of MR-SLAM systems

2.4.1.1 Centralized vs. Decentralized

According to Reid (2016), the architectures of MR-SLAM fall somewhere between centralized and decentralized. Mostly, the classification is according to the computation of individual systems, if performed in a centralized server or distributed over all computers/robots. Figure 2.16 shows the relationships between the ones described in Reid (2016), highlighting where the scope of this work falls in.

The aspects and limitations of each architecture, according to Reid (2016) are described as follows:

1. Centralized architectures: have a single computer where all the logic processing occurs. Each robot runs a minimal software that transmits all sensor data to this computer. It requires considerable wireless communication with all the robots, which can be a problem since, even a minor communication issue can impact the MR-SLAM estimation and control reliability.

2. Decoupled centralized architectures: have a single centralized computer, however each robot maintains its own local coordinate frame that is locally smooth and

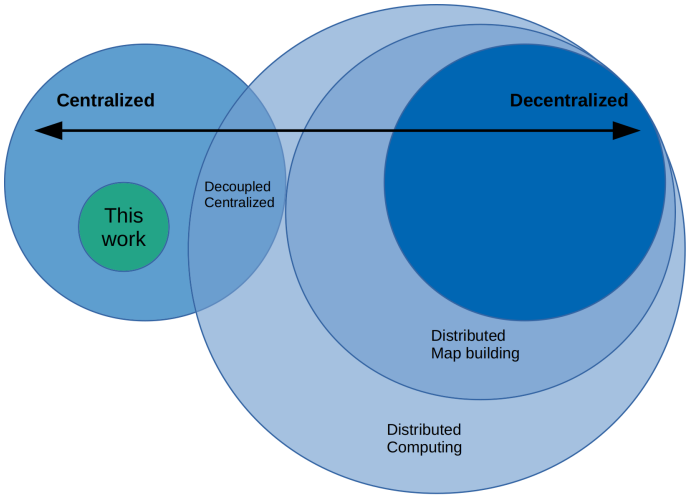

FIGURE 2.16 – MR-SLAM scope according to centralization. Adapted from Reid (2016).

central independent. Each robot software defines a decoupled coordinate system that uses a single-robot SLAM algorithm to produce pose and map estimates from sensor measurements. A significant advantage of this architecture is that individual robots can continue to perform estimation independent of a central, e.g. during brief communications outages. Joint activities such as map merging are performed at the central computer.

3. Distributed architectures: describe algorithms or computation that is divided into parts and executed across a network of multiple computers. When a MR-SLAM architecture is distributed, the various parts of the MR-SLAM algorithm and single-robot SLAM are executed on different computers/robots. There are architectures that are able to build distributed copies of the global map, i.e. distributed map building, while others require only parts of the MR-SLAM algorithm and data to be distributed. This data distribution may still be expensive for wireless networks without large broadcast capabilities.

4. Decentralized architectures: describe systems that have no centralized computer and all participant computers/robots are equal. Thus in a decentralized MR-SLAM architecture, all robots perform MR-SLAM independently by executing local instances. Each robot maintains a copy of the global map and state estimates, however they may be incomplete, since no single robot is aware of the complete network topology. This independence provides robustness to many types of failure, it comes at a cost of redundancy and risk of global map divergence, due to incomplete state information.

Regarding the use or not of a central computer to manage and process data coming from the robots, the level of cooperation between them is another aspect that separates the applications of MR-SLAM systems.

2.4.1.2 Levels of cooperation

Farinelli *et al.* (2004) brings another point of view of classification into two groups: coordination and systems. The former aims to characterize the type of coordination that is achieved in MR-SLAM systems, while the latter include the system's features that influence team development. Figure 2.17 shows a hierarchical structure with four levels regarding coordination.

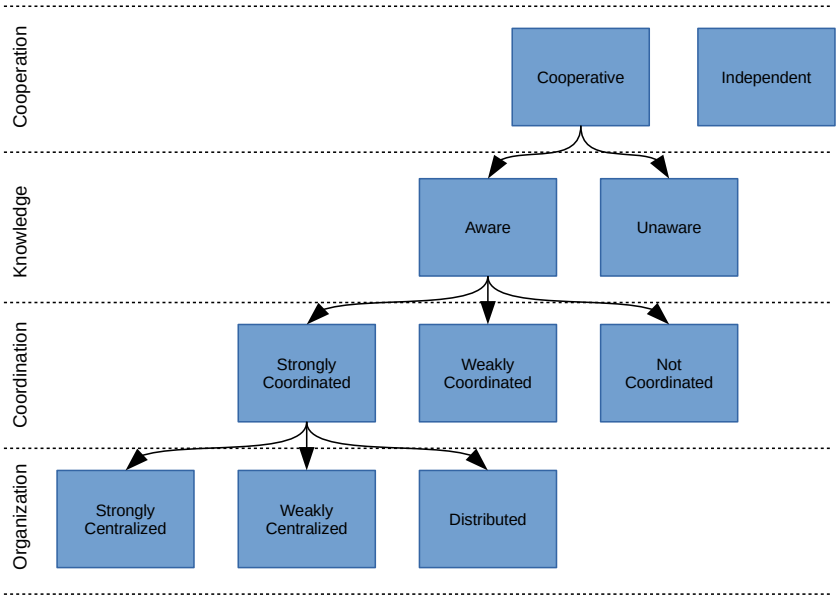

FIGURE 2.17 – Classification tree according to types of cooperation, levels of knowledge sharing, methods of coordination and types of centralization. Adapted from Farinelli *et al.* (2004).

Different levels of classification according to Farinelli *et al.* (2004) are described next:

1. Cooperation Level: The first level is concerned with the ability of the system to cooperate in order to accomplish a specific task. Cooperative systems differ from independent systems. A cooperative system is composed of "robots that operate together to perform some global task". Cooperation is often compared and merged with competition, which, up to now, has received little attention in the recent works on MR-SLAM. Independence on this level means the robots have no difference in behaviour compared to the single robot workflow, they are susceptible to repetition.

2. Knowledge Level: The second level of the proposed hierarchical structure is concerned with the knowledge that each robot has about its team mates. Aware robots have some kind of knowledge of their team mates, while unaware robots act without any knowledge of the other robots in the system. The interest in cooperating unaware MR-SLAM is motivated from an engineering point of view by the simplicity of such systems, with respect to aware ones.

3. Coordination Level: The third level is concerned with the mechanisms used for cooperation. Coordination can be seen as cooperation in which the actions performed by each robot take into account the actions executed by the other robots "in such a way that the whole ends up being a coherent and high-performance operation." However, there are different ways a robot can take into account the actions of the other members of the team. The underlying feature is the coordination protocol, that is defined as a set of rules that the robots must follow in order to interact with each other in the environment. Therefore, there are different levels inside the coordination aspect. We consider strong (weak) coordination as a form of coordination that relies (does not rely) on a coordination protocol.

The organization level is an interpretation very similar to the one described in the previous subsection, since it specifies classification with respect to centralization.

2.4.1.3 Homogeneous vs. Heterogeneous teams

According to Farinelli *et al.* (2004), the team composition aspect of MR-SLAM systems can be divided in two main categories: heterogeneous and homogeneous. Homogeneous teams are composed of robots that have exactly the same hardware and control software, while in the heterogeneous case the robots differ either in the hardware devices or in the software control procedures.

Homogeneous teams benefit from the fact that all data representation is in the same resolution/granularity, while heterogeneous teams can suffer from disparity of data resolution, such as maps (FARINELLI *et al.*, 2004), which will be shown later, can become a problem when performing a task called map merging.

With most of the taxonomy laid out on this subsection, there are still approaches that use human-in-the-loop MR-SLAM, which increases mapping accuracy and convergence by allowing a human operator to interact with the pose graph and maps coordinate frames in real-time (REID, 2016). In both cooperative and adversarial aspects, target detection and tracking encompasses a variety of decisional problems such as coverage, surveillance, search, patrolling, observing and pursuit-evasion along with others (ROBIN; LACROIX, 2016b), such classification is not on the scope of this work.

2.4.2 Map merging

Mapping, in the scope of MR-SLAM, is suggested to be the construction of a global map by merging the local maps built by individual robots (YU *et al.*, 2020). The solution to this problem depends on different aspects of each application, such as the map repre-

sentation and communication/awareness of the individual robots with each other. Most research on MR-SLAM, attempt to solve the case of merging two individual maps, making the process scalable for n robots, for this case, there are two main sources of information to perform map merging: relative positions of robots and heuristics (ANDERSONE, 2012).

If the relative position information of robots are known, the local reference frames of each robot must be transformed into a single global reference frame, making the problem of merging into computing this spatial coordinate transformation and apply to the maps involved (ANDERSONE, 2012). If no relative information is known, then the process becomes a harder case and sometimes unreliable. Most of the approaches for the latter case still attempt to compute a transformation between local reference frames, but a different method is needed in order to find the best match (ANDERSONE, 2010).

According to Andersone (2010), map merging can be put into one of the three scenarios:

1. The relative information of the robots are known at all times during the task. This case is the most simple regarding the computation of the map merging process, transformations between local frames can be computed in a very straightforward manner via geometric relations between the robots position.

2. The relative information of the robots are know during the mapping process. It is assumed the robots will eventually meet and determine their relative positions, falling into the computations of the first case. The strategy the robots need to meet to merge maps is known as the rendezvous case.

3. The relative information of the robots is unknown. In this case, the robots cannot determine their relative positions during mapping, which leads to different methods of determining the transformation between local reference frames.

This scenarios are depicted in Figure 2.18 where the colored lines and angles represent the relative information. The rendezvous case is the one used in this work and will be explained in more detail, the aforementioned methods for the third case will be briefly described in the next subsection.

The map merging procedure can be then separated in two parts, the transformation of individual maps and the association of common parts from merged maps into a single representation, called map alignment and map fusion, respectively (YU *et al.*, 2020). In the subsequent sections, an overview of popular methods for each part is described for the 2D case, most operations can be adapted for the 3D case.

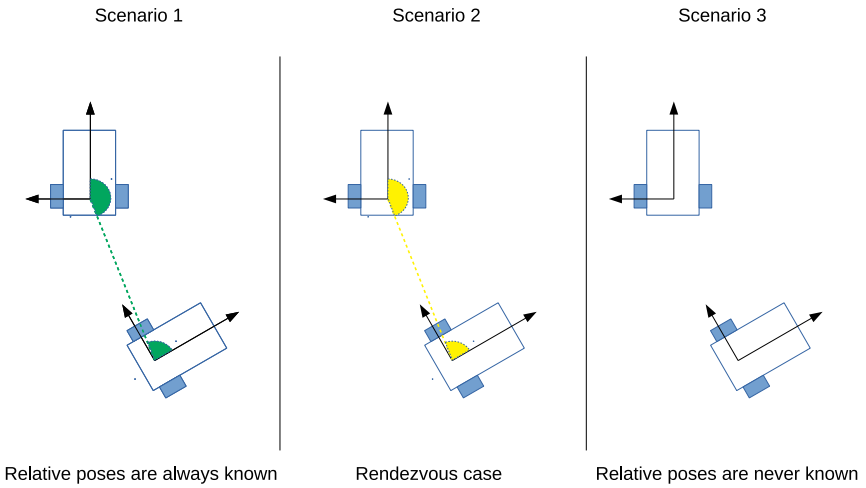

FIGURE 2.18 – Scenarios of map merging according to relative information.

2.4.2.1 Map alignment

Map alignment consists of the process of computing and applying the transformation between two individual maps. In 2D, this transformation is expressed by a roto-translation matrix shown in Equation (2.24).

$$T = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

The transformation is compressed by three parameters θ, t_x and t_y, which are the rotation angle and the translation along x and y axis, respectively. Using the matrix T, information such as history of poses and the map of a robot can be transformed by using this matrix (DINNISSEN *et al.*, 2012). Taking 2D points $[x, y]^T$ in homogeneous coordinates, they can be transformed by multiplying the matrix T as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.25}$$

where x' and y' are the new coordinates of the transformed point.

The visualization of this process can be demonstrated by Figure 2.19. The robots are fixed in different coordinate frames with respect to each other, to align these frames, a direction of merging is established so that one coordinate frame should align with a fixed one (DINNISSEN *et al.*, 2012). By knowing the relative geometric information of the two robots involved, a resultant transformation is composed by three components: the current robot pose, the relative information and the other robot pose (DINNISSEN *et al.*, 2012).

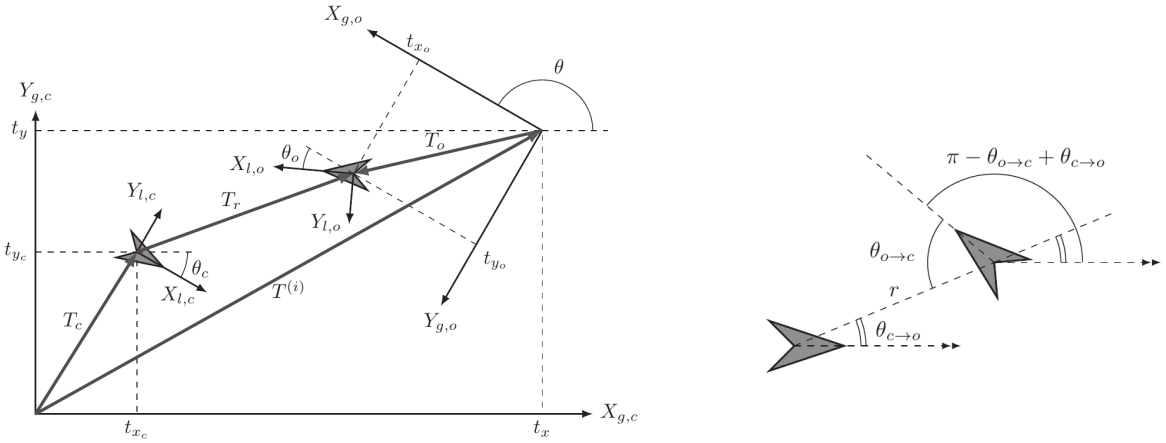

FIGURE 2.19 – Left: Two robots and the spatial relations between their positions and reference frames. Right: Local perspective where the parameters for the relative transformation are shown (DINNISSEN *et al.*, 2012).

The subscripts c, o and r refer to information on the current robot, other robot and mutual relations scope, respectively. To compute the resultant transformation T and choose the direction of merging from the other robot to the current robot, the process is summarized as shown in Equation (2.26).

$$T = T_c T_r T_o^{-1} = \begin{bmatrix} \mathrm{c}\theta_c & -\mathrm{s}\theta_c & t_{x_c} \\ \mathrm{s}\theta_c & \mathrm{c}\theta_c & t_{y_c} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta_r & -\mathrm{s}\theta_r & t_{x_r} \\ \mathrm{s}\theta_r & \mathrm{c}\theta_r & t_{y_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{c}\theta_o & -\mathrm{s}\theta_o & t_{x_o} \\ \mathrm{s}\theta_o & \mathrm{c}\theta_o & t_{y_o} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \tag{2.26}$$

The sines and cosines were substituted by the letters s and c, respectively. All the individual transformations are on the form of Equation (2.24). The current and other robot transformations are obtained by information that should be available by their individual estimates, to compute the middle component, simple geometric relations are used to define the relative transformation T_r, as shown in the right picture of Figure 2.19. The variables of interest are computed as follows:

$$t_{x_r} = r\cos\theta_{c\to o} \tag{2.27}$$

$$t_{y_r} = r\sin\theta_{c\to o} \tag{2.28}$$

$$\theta_r = \pi - \theta_{o\to c} + \theta_{c\to o} \tag{2.29}$$

The distance between the robots is denoted as r, the angles $\theta_{c\to o}$ and $\theta_{o\to c}$ can be seen as the direction in which the other robot is on the local coordinate frame of the current robot and vice versa, respectively. These parameters, alongside the distance between the

robots r, can be acquired via robot-to-robot measurements when the robots meet each other at rendezvous or by finding the correspondence between their measurements, e.g. via scan matching (LEE *et al.*, 2012).

This method of computing the transformation T is commonly referred as the direct method, and can be applied on the first and second scenarios of merging described previously (LEE *et al.*, 2012). For an indirect method, for the case of the third scenario, the transformation must be estimated by other means. Most of them attempt to solve the following equation:

$$T = \underset{\theta,t_x,t_y}{\operatorname{argmax}} \mathbf{V}(\mathbf{m}_1, \mathbf{m}_2') \tag{2.30}$$

Where $\mathbf{m}_1$ is the current robot map and $\mathbf{m}_2'$ is the other robot map transformed by T. The function $\mathbf{V}$ evaluates the merging according to the transformation applied (SAEEDI *et al.*, 2015). The complexity of this solution can differ greatly depending on the map representation (ANDERSONE, 2010). For occupancy grid maps, $\mathbf{V}$ can be treated as an objective function on an optimization problem such that the evaluation is a measure of overlapping between the maps (BIRK; CARPIN, 2006). For feature-base maps, Lee e Lee (2011) proposes extract and match spectral information, such as the Hough Spectrum, on maps to estimate the transformation. Another popular approach is the use of computer vision tools such as image registration techniques for occupancy grid maps, since they have similar properties to images, but with the disadvantage of requiring special feature descriptors for matching, making it a hybrid approach (SAEEDI *et al.*, 2015).

Such methods suffer from the lack of information regarding the existence of overlap, since most of them depend on finding common parts that overlap on both maps, the result may not be reliable (SAEEDI *et al.*, 2015). Figure 2.20 shows the merging of three occupancy grid maps, the ellipses on each map are the overlapping regions.

2.4.2.2 Map fusion

The merging process must account for the uncertainty and common areas of the maps involved. To accomplish this, different map representations require different approaches (ZHOU; ROUMELIOTIS, 2006). On feature-based maps, both mean and covariance of the corresponding features must be fused, e.g. a point feature p_c in the current robot map can be fused with p_o', the transformed feature point of the other robot map via Equation (2.25), by using the following equations (DINNISSEN *et al.*, 2012):

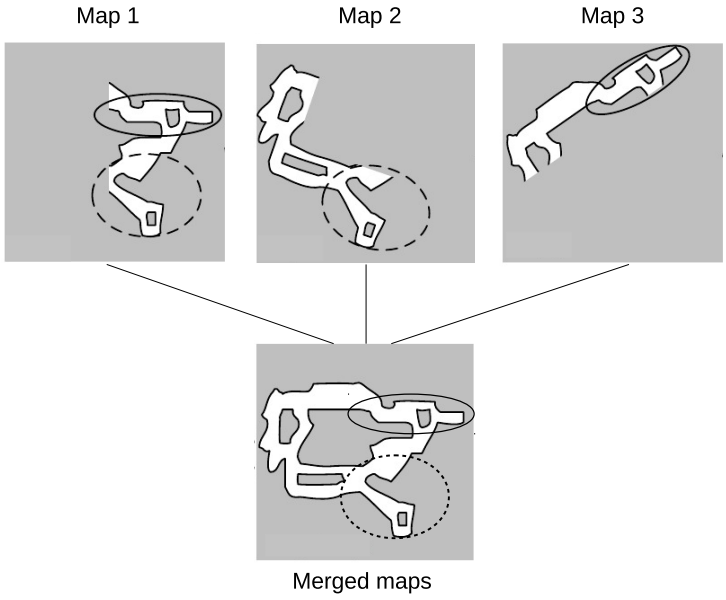

FIGURE 2.20 – Merging of three occupancy grid maps with common region. Adapted from Saeedi *et al.* (2016).

$$p_m = p_c + \Sigma_c \left[\Sigma_c + T_\theta^T \Sigma_o T_\theta\right]^{-1} (p'_o - p_c) \tag{2.31}$$

$$\Sigma_m = \Sigma_c - \Sigma_c \left[\Sigma_c + T_\theta^T \Sigma_o T_\theta\right]^{-1} \Sigma_c \tag{2.32}$$

where p_m will be the merged mean of the point feature. Σ_m, Σ_c and Σ_o are the covariance matrices of the merged feature, the current robot feature and the other robot feature, respectively. T_θ is the rotation-only transformation matrix extracted from T.

In the context of occupancy grid maps, the uncertainty is associated with each grid cell. Once the maps are aligned, the process of fusion will depend on the type of content each cell holds. For a log odds representation of occupancy the additive property holds as if the transformed map is a batch of measurements, leading to the formulation (SAEEDI *et al.*, 2015):

$$l_{t,i}^m = l_{t,i}^c + l_{t,i}^{'o} \tag{2.33}$$

for all grid cells. Where $l_{t,i}$ is the content of cell i on time step t, the superscripts are used to identify the maps, m is for the resulting fused value, c for the current robot map and $'o$ for the other robot map transformed. Since occupancy grids are discretizations of the space by nature, both in alignment and fusion there will be a loss of data.

Map merging research is still in its infancy if compared to other MR-SLAM problems, a branch of research that is not discussed in this work is the ability to make a decision when maps can be safely merged and when the merging is not possible (ANDERSONE,

2010; DINNISSEN *et al.*, 2012). Furthermore, the merging of maps of different robots is at the core of efficiency and robustness of MR-SLAM applications.

The next section elaborates on exploration strategies for MR-SLAM, as this is a key problem that falls into target assignment, the discussion will be limited to the case of 2D maps.

2.4.3 Multi-robot exploration

As described in previous sections, exploration is a fundamental task allocation problem to be solved in autonomous mobile robotics, as well as the different cooperation levels multi-robots can be classified as. Collaboration to perform exploration is one of the main advantages of MR-SLAM systems, and can be seen as a case of assignment problem (GERKEY; MATARIC, 2003). The solution can differ in complexity according to the *a priori* information of the state of robots and environment (BURGARD *et al.*, 2000). Figure 2.21 shows three robots coordinating their exploration targets to efficiently navigate the environment.

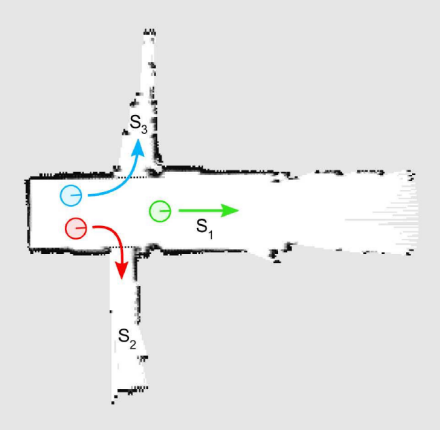

FIGURE 2.21 – Three robots navigating an environment showing the efficient choice of exploration targets (WURM *et al.*, 2008).

Gerkey e Mataric (2003) stated an analytical formulation and described important theoretical aspects of the Optimal Assignment Problem (OAP) for multi-robot task allocation, which is not the objective of this work. Methods that provide reasonably good results and may be sub-optimal will be the focus of discussion for the case of 2D maps.

2.4.3.1 Exploration strategy

The task of exploration must aim to achieve a objective, for MR-SLAM systems the exploration can become very efficient and optimized, resulting, at least, in a faster process overall (PATEL *et al.*, 2020). Objectives such as search and rescue becomes more viable for

MR-SLAM, in this case, robots must search and detect specific objects on the environment to be captured. Such objective have a distinct coordination strategy (ROBIN; LACROIX, 2016b) if compared to map coverage, in which explored area is of greater importance (PATEL *et al.*, 2020).

Many strategies assume the global map is shared with all robots at all times (KULICH *et al.*, 2015), this assumption holds for the first scenario described in Section 2.4.2. Solutions such as the Hungarian method, which solves the assignment problem via optimization, can be used since the number of workers (robots) and tasks (targets) are known (KULICH *et al.*, 2015). Regarding the second and third scenarios from Section 2.4.2, there may be two different behaviours for the exploration task, one before map merging occurs and one after.

On any strategy, the key procedures from single-robot exploration still hold for the multi-robot case with the addition of coordination on decision making and planning, the subsequent sections will describe these procedures.

2.4.3.2 Target selection

The process of choosing a exploration target can become a hard task on large environments and with a larger team of robots (KULICH *et al.*, 2015).

On a MR-SLAM system where maps are represented by occupancy grid and robots' relative position are known only at rendezvous, the coordination strategy cannot account for position and other robots' targets, since no global map is built at this point (KULICH *et al.*, 2015). To solve this, robots can behave as independent single-robot SLAM systems until knowledge of other robots comes in place.

Taking as example coverage as the exploration objective, a robot can make use of frontier-based exploration in a best-first (greedy) manner to locally maximize its individual map coverage. As soon as a map merging procedure with another robot finishes, a coordination is required to efficiently assign new exploration targets for the robots. Figure 2.22 shows the case where even coordinating different targets to robots can lead to inefficient assignment.

A simple coordination approach at rendezvous is to first assign one robot with the normal greedy rules, then assign the other the robot with the most distant target from the previously assigned target. Performing this in a aggregate manner with previous targets can fail if there are more robots than targets at the moment of rendezvous, resulting in inefficient assignment at best. Comparative studies as the ones by Kulich *et al.* (2015) and Elsefy *et al.* (2018) attempted to verify various techniques for selecting the best target, such as the Hungarian method, greedy exploration, role-based exploration and K-means

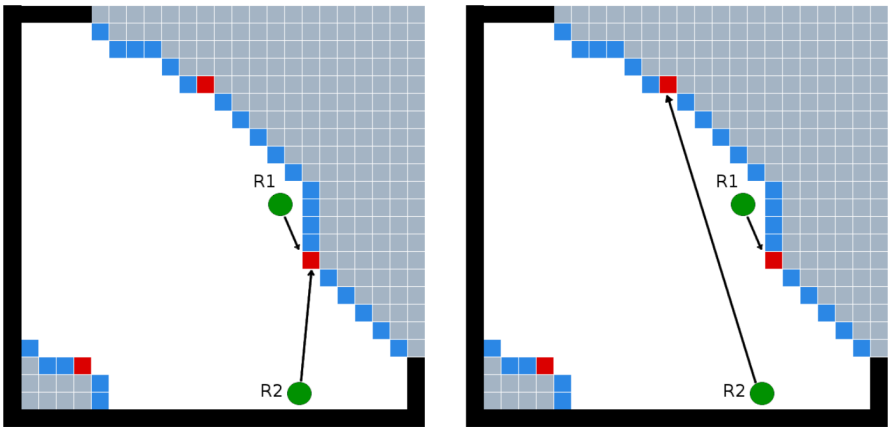

FIGURE 2.22 – Target assignment right after map merging between two robots R1 and R2. (a) Greedy assignment and (b) coordinated but inefficient assignment (KULICH *et al.*, 2015)

clustering. However, the difference in results for percentage of explored map seems to be very marginal and does not imply a definitive method.

2.4.3.3 Collision avoidance

Some assumptions made by the case of single-robot exploration are naturally violated in the case of multi-robot exploration, e.g. the environment becomes forcibly dynamic. One robot must account for the presence of other robots that will be roaming at the same environment. Multi-robot exploration requires the assurance that no robot will collide to each other during the task.

Collision avoidance controllers are an entire field of study on their own, some rely on adaptive control or predictive control theories (SICILIANO; KHATIB, 2016), and deepening the subject is not the focus of this work. Popular and easy-to-use algorithms such as the Vector Field Histogram (VFH) or the Dynamic Window Approach (DWA) are a good choice for multi-robot (ROBIN; LACROIX, 2016a).

The VFH algorithm, for instance, computes obstacle-free steering directions given a target direction for a robot based on range sensor readings, which are converted to density histograms to represent obstacle location and proximity. By applying distance thresholds on these histograms, feasible collision-free directions are obtained (BORENSTEIN; KOREN, 1991).

The first histogram obtained, called polar obstacle density histogram, is a conversion of the range and bearing readings to angular sections around the robot, where the obstacles are accounted for histograms bins. Safety distance thresholds are applied to this histogram, generating the masked polar histogram, where sections that fall greater

than an upper bound are treated as obstacles and sections less than a lower bound are considered free space. The direction is chosen based on the closest angle to the current heading of the robot and the minimum turning radius given the maximum angular velocity (BORENSTEIN; KOREN, 1991). Figure 2.23 shows an example of the controller being used on a simulated environment, where the robot uses the controller during all its trajectory.

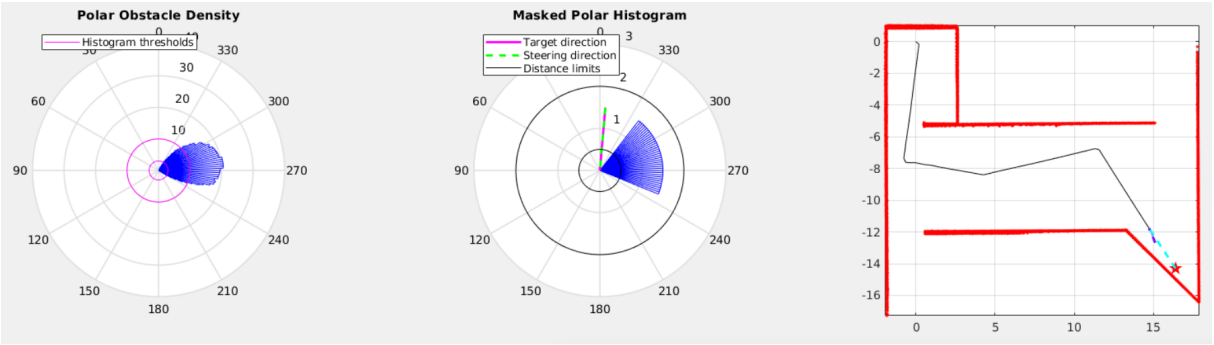

FIGURE 2.23 – Test of the VFH controller on a simulated environment. The polar obstacle density histogram is processed from the LIDAR scan (left) and converted into a masked polar histogram (middle) after the distance thresholds are applied. The controller perceived obstacles at angles starting around 250 degrees, depicted by the blue bins on the masked polar histogram. The right image shows that the robot was close to a wall, the solid black line is the robot path, red points are the point cloud map and the target objective is marked by a red star.

The following chapter will present the details of the approach chosen for implementation in this work.

3 Solution Proposal

In this chapter the details of methods used to develop the proposed solution for the problem of active MR-SLAM are presented.

3.1 Active MR-SLAM framework

This work focus on the case of 2D SLAM, in which the map and robot pose is contained in a two dimensional plane. This section will discuss all the assumptions and techniques used to achieve the task objective.

3.1.1 Task objective

The objective is to use multi-robots for autonomous exploration at an unknown environment to achieve the coverage task, i.e. explore a set amount of the environment area (at minimum), doing so, the robots stop and the task is treated as complete.

3.1.2 Assumptions

The following are requirements or approximations considered for the realization of this work:

1. The environment has a flat ground terrain throughout all available space to perform the task.
2. The environment is static.
3. The environment has *a priori* known area value.
4. The environment is globally closed and locally opened, meaning the robot can not escape the available total space and can not be trapped in a specific location such as a room.

5. The robots have full wireless communication with a central computer at all times to exchange data.

6. The robot is equipped with a range finder sensor to perceive the environment and a RGB-D (Red-Green-Blue-Depth) camera to perceive other robots.

7. The robot can be identified by other robots via visual identification of fiducial tags printed on its body.

8. Initial positions of robots are unknown, however they must be spread out at the start in such a way the robots do not encounter each other immediately.

These conditions in which the experiments were carried out were established to make the application possible. The proposed solution was not tested outside these conditions.

3.1.3 Individual active SLAM algorithm

This section will describe the key aspects of the active SLAM algorithm that each robot performs. Figure 3.1 below shows a macroscopic view of the algorithm flowchart.

The flowchart can be divided into three parts: SLAM, map merging, exploration. The start routine initializes the robot and takes a first measurement. The robot only moves at the input/output block (parallelogram), suggesting that, the whole task execution speed is determined by how fast the following blocks can be processed.

The input/output block is executed on the robot and is responsible to take raw measurements such as scans and odometry, the motion commands are given to the robot as a velocity array on the body reference frame $[v_x, \omega_z]$, which are the linear velocity on the x axis and the angular velocity on the z axis, respectively. These velocities will be denoted as v and ω from here on. A robot receiving velocity commands will translate these in motor speed commands v_l and v_r following the differential drive formula (LYNCH; PARK, 2017):

$$r = \left|\frac{v}{\omega}\right| \tag{3.1}$$

$$v_r = \omega\left(r + \frac{d}{2}\right) \tag{3.2}$$

$$v_l = \omega\left(r - \frac{d}{2}\right) \tag{3.3}$$

where d is the distance between the wheels and r is the distance from the center of the robot to the Instantaneous Center of Curvature (ICC). Figure 3.2 shows a robot moving

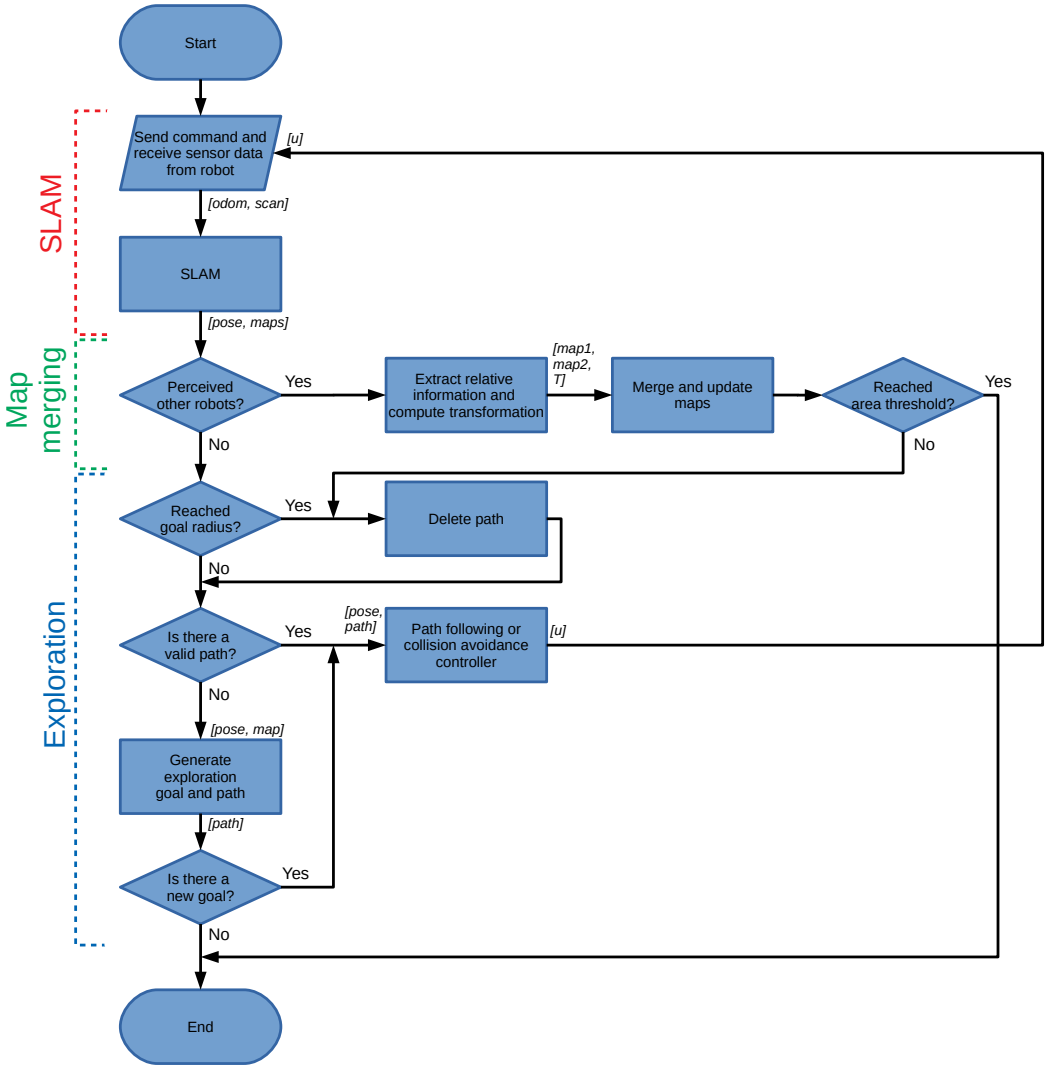

FIGURE 3.1 – Flowchart for the active SLAM algorithm.

with its ICC located at position $[x_c, y_c]$.

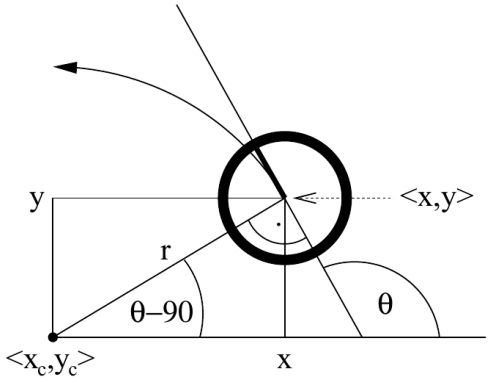

FIGURE 3.2 – Robot motion by a robot at pose $[x, y, \theta]^T$ with constant velocities $[v, \omega]$ (THRUN *et al.*, 2005).

The SLAM block receives and process the data with the robot stationary. The Graph-SLAM algorithm is performed in this stage, constructing the graph using the ICP algorithm and initializing the first pose at the origin and the first map as an occupancy grid of the first scan.

At the start it is unlikely the robots will perceive each other, so the algorithm continues to the block Generate exploration goal and path. The process of exploration uses an approach inspired by the work of Wurm *et al.* (2008). To generate an exploration target, the Voronoi Graph is first is extracted from the occupancy grid map. A Voronoi graph is a structure that aims to hold information on the map free space, creating a topological graph that comprises the cells with equidistant obstacles. This can be done efficiently from the Euclidean distance transform, that computes for each grid cell the distance to the closest obstacle, the Voronoi graph can be constructed from the skeletonization on the distance map (WURM *et al.*, 2008). The process is illustrated in Figure 3.3.

The nodes of the graph are the branching skeleton cells, with the remaining cells as edges. With this graph structure and the skeletonized map, two parts of the exploration task can be extracted, a path and candidate target points (WURM *et al.*, 2008). The skeletonization forms a collision free path on the map in a way that is efficient for coverage, the reason for that is on the centralized aspect of the path, in which the robot performs measurements in a position that gathers the most information from free space. The Voronoi graph provides candidate target points for exploration, which can be selected according to proximity with frontier cells, for the reasons discussed in Section 2.3.3.5.

The rules for selecting the exploration target are listed below, the concept was inspired by the work of Wurm *et al.* (2008) and Hu *et al.* (2020).

1. Rule 1: search the Voronoi graph nodes with degree 3, that possess two neighbors

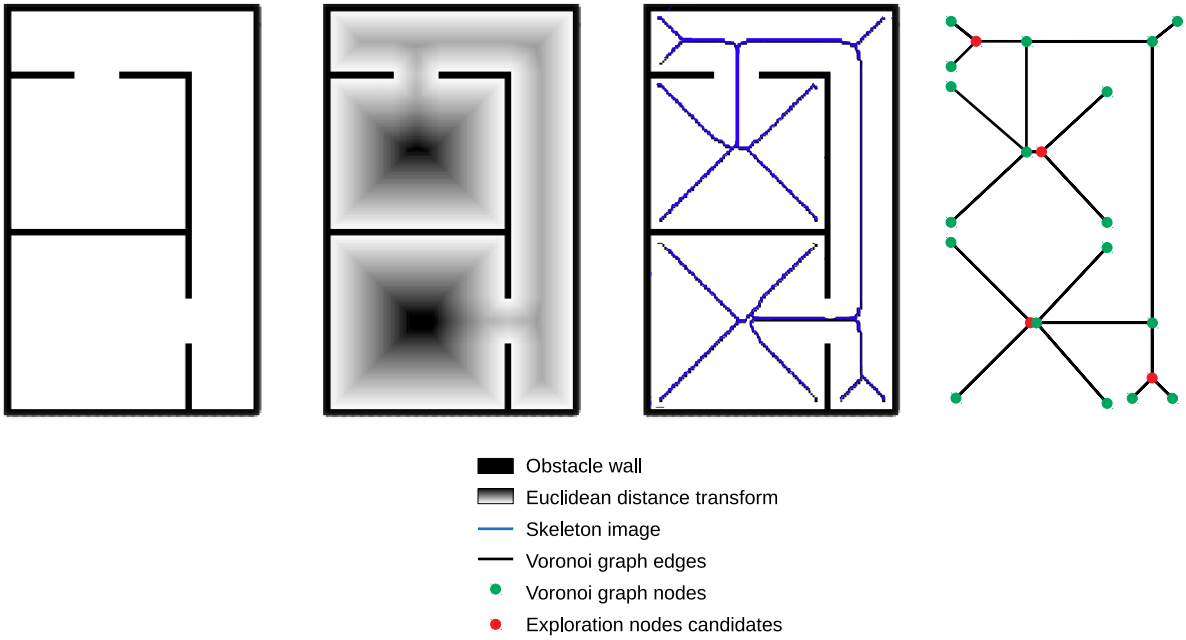

FIGURE 3.3 – Construction of Voronoi graph by skeletonization of the occupancy grid map. Adapted from Wurm *et al.* (2008).

nodes that are end nodes. These will be the target candidates.

2. Rule 2: Compute the cost to each Rule 1 node according to Equation (3.4).

3. Rule 3: The selected node is the one with minimum cost among the candidates.

$$C = \lambda d_r + (1 - \lambda) d_s \tag{3.4}$$

Equation 3.4 describes the cost to reach a graph node. The λ parameter in Equation (3.4) is a value in the interval $[0, 1]$, d_r is the distance from the node to the robot and d_s is the distance from the node to the origin. The value of λ holds the impact of the exploration-exploitation aspect to the task (HU *et al.*, 2020). Setting $\lambda = 0$ the nodes will be selected in a breadth-first fashion around the map origin, while $\lambda = 1$ performs a greedy exploration on the closest Rule 1 node (HU *et al.*, 2020).

Once the target is chosen, the path will be computed by using an A* algorithm on top of the skeleton map, with cell costs outside the skeleton cells being higher than the ones that belong to it. The algorithm should form a path that leads to the target using the skeleton map as reference.

The path is stored in memory to be used in future time steps as a reference for a path following controller. The controller used is a Pure Pursuit controller, as was stated in Section 2.3.3.5, that takes this path as set of waypoints and the current pose of the robot to compute the necessary velocity commands. These are $[v, \omega]$ on the robot reference frame, that serve as input to the input/output block, restarting the loop sequence.

A verification of reaching the target is applied before executing the controller block. If the robot is within a specific radius from the target, it is stored in memory as a visited target. The old path is deleted from memory and the algorithm should compute a new target.

After the first iteration, a verification on the image from the RGB camera is performed to detect other robots by fiducial tags after each SLAM block execution. If a robot is identified on the image, their relative information is extracted and the transformation is computed. The merged maps are updated on each robot and other information such as visited targets and scans are transformed and exchanged. The current coverage is then computed according to the following equation:

$$coverage = \frac{A_{current}}{A_{total}} \tag{3.5}$$

where $A_{current}$ is the current area, easily computed by counting all explored cells on the occupancy grid map, and A_{total} is the total area given as input to the algorithm.

This is the case because the majority of map merging procedures boost the explored area of the involved robots. If, after the merge, a robot helps another one to complete the last part of the map, this verification assures the robot finishes the task.

When a merge occurs, a collision avoidance controller is used instead of the path following. The VFH controller described in Section 2.4.3.3 is used to prevent robots that are close enough to collide and still follow the path stored in their memory.

The subsequent sections explain the simulation and experimental steps for this work.

3.2 Simulation and experimental setup

The proposed solution validation is composed by simulation and experiments with multi-robots. The environment in both cases is indoor and with known total area beforehand. A central computer is used to run the MR-SLAM framework and to establish wireless communication with the robots via a WiFi router. The MR-SLAM architecture of this work is classified as cooperative, aware, weakly coordinated and strongly centralized.

All the communication is handled by the Robot Operating System - ROS (ROS, 2021), which is capable of transmitting specific data structures in the scope of robotics. Figure 3.4 shows the interactions between elements of hardware and software.

The ROS network depicted in Figure 3.4 is a TCP/IP-based protocol that runs on all machines (computers and robots) and communicates using the existent WiFi network. ROS is also very modular, e.g. it can receive and send commands for real or simulated

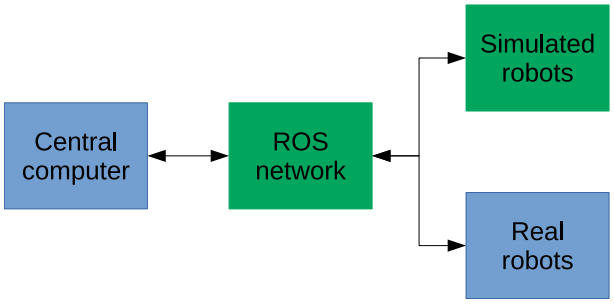

FIGURE 3.4 – Diagram showing the communication from hardware (blue) and software (green) elements.

robots by using the same code, requiring little or no modification in order to work. For this work, ROS Kinetic and Melodic versions were used, the choice of one instead of another depends on the operating system that is installed, both versions are compatible with each other.

The software used for the implementation of the proposed solution and a description of their role are listed below:

- MATLAB: active MR-SLAM algorithm;
- ROS: communication between MATLAB and the real/simulated robots;
- Gazebo: simulation of the environment and the multi-robot models;
- Python: programming language used for the low-level tasks executed by the processing boards embedded in the real robots;

The real robots used in this work are from the Laboratório de Máquinas Inteligentes (LMI) localized at the Instituto Tecnológico de Aeronáutica (ITA), they are customly built and meant to operate indoors. These robots and experiments have great influence from Medeiros *et al.* (2010), Pinto *et al.* (2021) and Buonocore *et al.* (2012), which are past works involving SLAM research located at LMI.

At the central computer, a MATLAB program is responsible to connect with the ROS network and run the MR-SLAM algorithm. MATLAB is a programming software designed to provide engineering tools to rapidly prototype as well as develop a full product (MATHWORKS, 2021d). This work makes use of MATLAB version 2021a and the following toolboxes: Navigation Toolbox, Robotics Systems Toolbox and ROS Toolbox.

The communication inside the ROS network uses the publisher/subscriber mechanism, which can send/receive messages via topics. A topic carries a unique name inside the ROS network, and can be accessed by nodes, which can be a program running on a machine connected to the network. These nodes may access low-level information such as raw sensor data or velocity commands, and enclosure them as a message that ROS

can recognize. Figure 3.5 illustrates this mechanism. The arrows show the direction data must flow, in a green rectangle the topic named "example" carrying a "String" type of message is depicted.

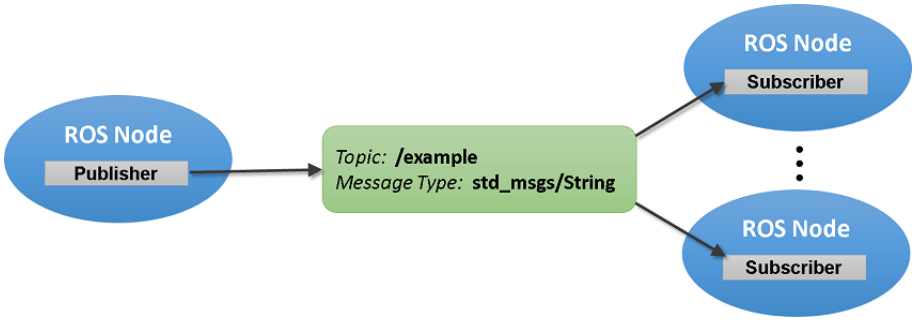

FIGURE 3.5 – Diagram showing a message being published and subscribed by nodes (MATHWORKS, 2021c).

Furthermore, the MATLAB program and each piece of software that has hardware access on the robots can communicate with each other using this protocol.

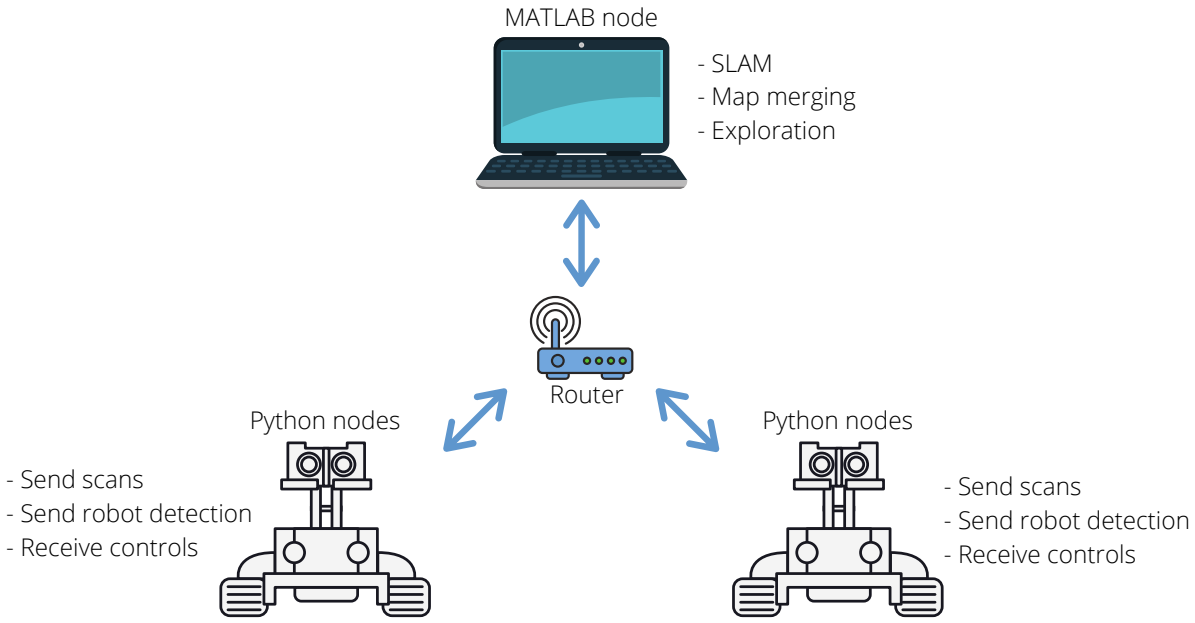

FIGURE 3.6 – Network nodes on real world experiments.

In Figure 3.6 it is shown the network nodes in the case of real world experiments, in which each node is a software that communicates using ROS. On the robots, the Python programming language is used to receive raw sensor data and send motion commands via serial communication on a Raspberry Pi board. On the central computer, the MATLAB program is responsible to maintain the current MR-SLAM estimation, process the data received and send the computed command to the robots.

3.2.1 Simulation

The simulation is performed using the software Gazebo, a 3D multi-robot capable simulation environment designed to work with ROS (GAZEBO, 2021). Gazebo allows the construction of custom environments to interact with robots via an advanced physics engine (AGUERO *et al.*, 2015). The simulator is also capable of modeling virtual sensors that perceive the environment, such as cameras, range finders, inertial measurement units (IMU), among others.

For this work, Gazebo version 9.0.0 was used and custom models were built for the simulation using Gazebo's Model Editor. A differential drive robot containing all required properties described previously was built, with the possibility of inserting multiple replicas, providing a team of homogeneous robots. Figure 3.7 shows the robot model in Gazebo.

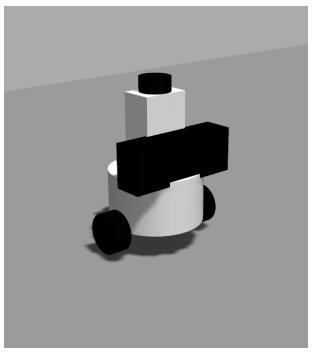

FIGURE 3.7 – Model of the custom robot made in Gazebo.

To create indoor environments, Gazebo's Building Editor allows to easily draw walls on a top-down perspective, objects and basic 3D shapes can also be added. Thereby, two environments were built containing only walls and some basic shapes. Figure 3.8 shows these environments. A small simple environment was created with dimensions 20x20 m, the larger and more complex has dimensions of 50x50 m.

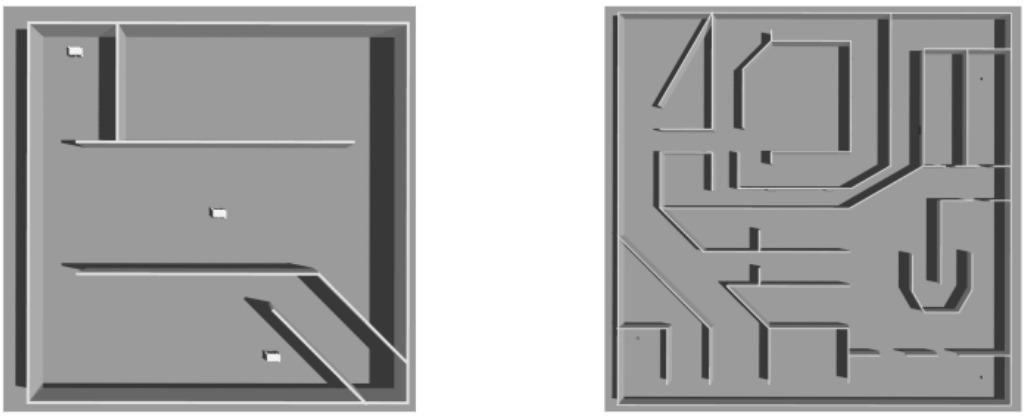

FIGURE 3.8 – Environments small (left) and large (right), built for experimentation in Gazebo.

The simulated robot is equipped with a virtual LIDAR scanner that takes 400 samples

per measurement around the robot. The virtual sensor has a maximum range of 10 m and Gaussian measurement noise with standard deviation of 0.008 around a zero mean. A virtual camera pointing forward was added but was removed later due to high computation burden, which is proportional to the number of robots in the simulation. The camera had the purpose of identifying other robots for measuring relative information, but the ground truth position was used instead when robots were close enough.

The MATLAB program process the raw data coming from Gazebo and sends velocity commands to the simulated robots one at a time.

3.2.2 Real world experiment

Real world experiments are sometimes neglected by researchers, with most being very inclined to give preference to simulation as means for validation. This work proposes the use of simulation and real world experimental results to validate the solution presented thus far. Many adaptations are required to perform experiments using real hardware, they are described in this section for the scope of this work.

3.2.2.1 Robots

The experiments in the real world were performed by two robots, Isis and Omni. Their hardware are described in Table 3.1.

TABLE 3.1 – Hardware description of robots.

Component	Isis	Omni
Processing board	Raspberry Pi 3B+ and Arduino Mega 2560	Raspberry Pi 4B and Arduino Mega 2560
Range finder sensor	SLAMTEC RPLIDAR A2M8	Intel Realsense D435i
RGB-D camera	Kinect-for-Xbox	Intel Realsense D435i
IMU	WiT BWT901CL	LSM9DS1
Drive motor	2x DC, 6V, 100 RPM	4x Stepper, 12V, NEMA 17
Wheel encoders	Yes	No

The robots used in this work are shown in Figure 3.9.

The specifications of the range finders of each robot are described on Table 3.2.

The specification values are suggested by the manufacturer (SLAMTEC, 2021) and (INTEL, 2021), and are very susceptible to changes in lighting or reflected material. Although the maximum range of the Realsense D435i camera is stated as 10 m, this value was saturated at 3 m due to tests results showing that, beyond this threshold, most depth data

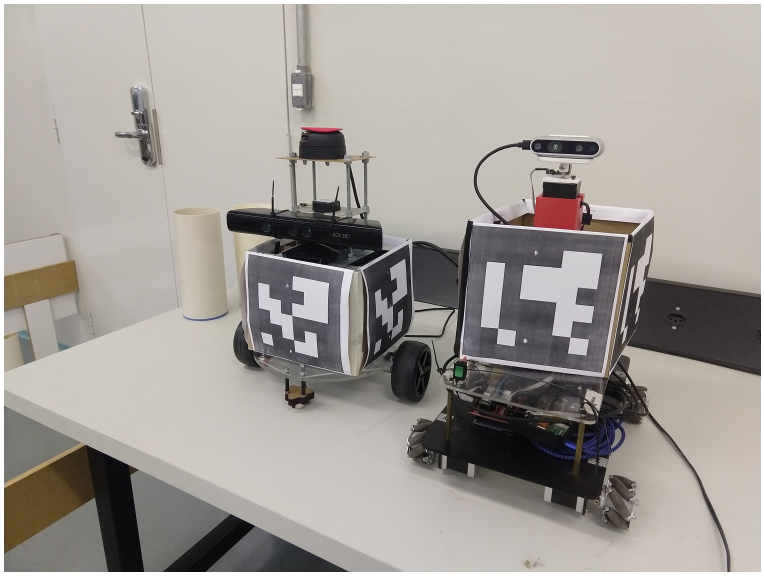

FIGURE 3.9 – Robots used in real world experiments for this work, Isis (left) and Omni (right).

TABLE 3.2 – Range finder specifications.

	RPLIDAR A2M8	**Intel Realsense D435i**
Min/Max range (m)	0.15/8	0.18/10
Distance resolution (%)	<1	≤2
Data points per scan	300	848

is unreliable and with very high noise.

The heterogeneous nature of the available robots require some adaptations to make the experiment compatible with the framework. The most noticeable distinct capability is that Omni has as its range finder sensor the RGB-D camera, while Isis possess a LIDAR scanner. The adaptation made to make data compatible was on Omni, in the form of a rotation mechanism for the RGB-D camera with a reference system.

The system was elevated via a 3D printed tower that locks a stepper motor on top, this one couples the camera on its shaft by a 3D printed custom piece. To detect a reference point for the rotation, an optical sensor was used to trigger whenever the desired point crossed the sensor. Figure 3.10 shows the mechanism.

By using this mechanism, Omni is capable of rotating the camera and capture depth data as a scan around the robot. The Realsense D435i datasheet states that the horizontal field of view (FOV) is approximately 87 degrees for the resolution used (INTEL, 2021), which makes 4 rotations of 90 degrees enough to extract a full 360 degrees scan. Figure 3.11 shows a top-down view of the robot, the camera is rotated to specific angles with respect to the robot's body reference frame. The robot is in blue, the camera in gray, the camera's FOV in red and the arrows are the robot body reference frame.

Since depth data coming from the RGB-D camera is an image, the middle row from

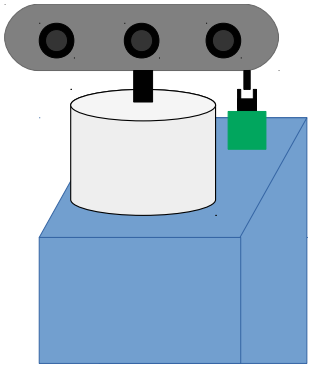

FIGURE 3.10 – Rotation mechanism for RGB-D camera (gray) showing the stepper motor (white) and reference trigger system (green).

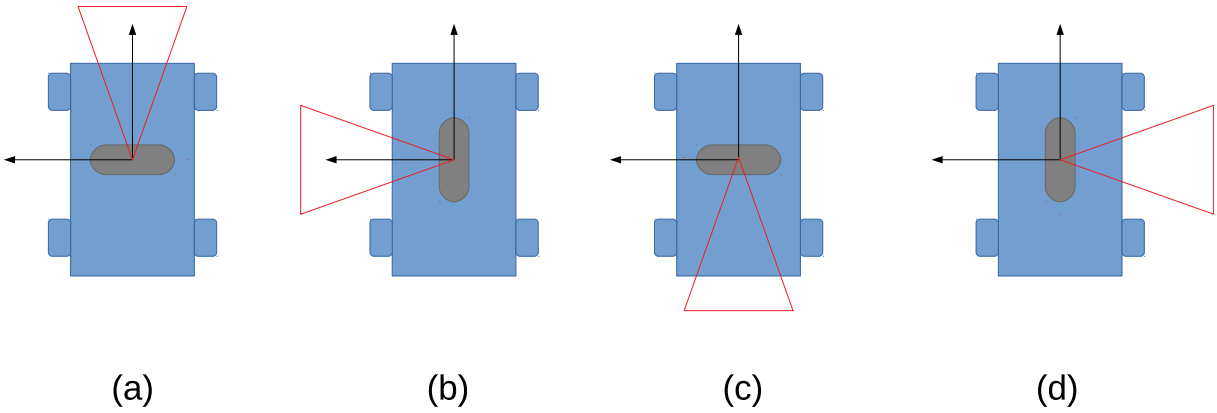

FIGURE 3.11 – Top-down view of robot showing the rotation of the camera to form a scan. (a) Angle of measurement 0°, (b) 90°, (c) 180° and (d) 270°.

the image is extracted to make data compatible with the 2D mapping. The necessary rotations are computed according to each position referent to the robot's frame. Figure 3.12 shows a montage of the 4 depth images for each rotation position, highlighting the row extracted to produce the 2D scan.

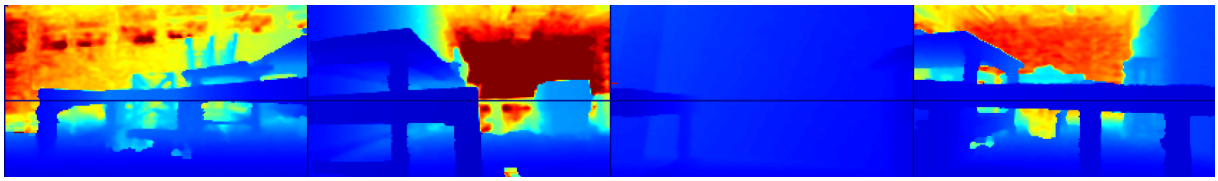

FIGURE 3.12 – The four depth images sequence showing the extracted row (black horizontal line) to produce a scan.

Fiducial tags are widely used among researchers, and many sets of markers were developed for detection and matching purposes in computer vision over the years. ArUco is a popular set that has been proved to be very robust to occlusion (GARRIDO-JURADO *et al.*, 2014). Figure 3.13 illustrates some markers from the ArUco set, showing different resolutions and sizes.

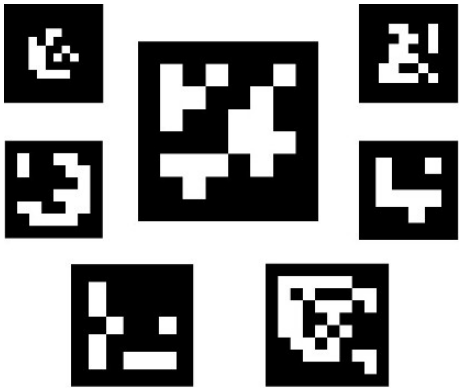

FIGURE 3.13 – Example of ArUco tags from various resolutions (OPENCV, 2021a)

This work makes use of ArUco markers printed on the side of the real robots to enable visual detection and pose estimation. By using the RGB-D cameras that both Isis and Omni have, they are able to estimate the tag pose with respect to the camera once they are in range of each other. This estimation is used for when robots are at rendezvous and need relative information for map merging. Figure 3.14 shows an estimation test.

The estimation is performed with functions from OpenCV, an open source computer vision library (OPENCV, 2021b). Using OpenCV, it is possible to detect the marker using prior knowledge of the marker, such as its size, its family set and the camera parameters, and by using a 4 point correspondence (marker corners) it calculates the camera pose using only the RGB image. The output of the estimation are the pixels that correspond with the corners of the tag, its 3D position with respect to the camera and the identification of the tag (OPENCV, 2021a). With the position of the tag in the image, the angle relative

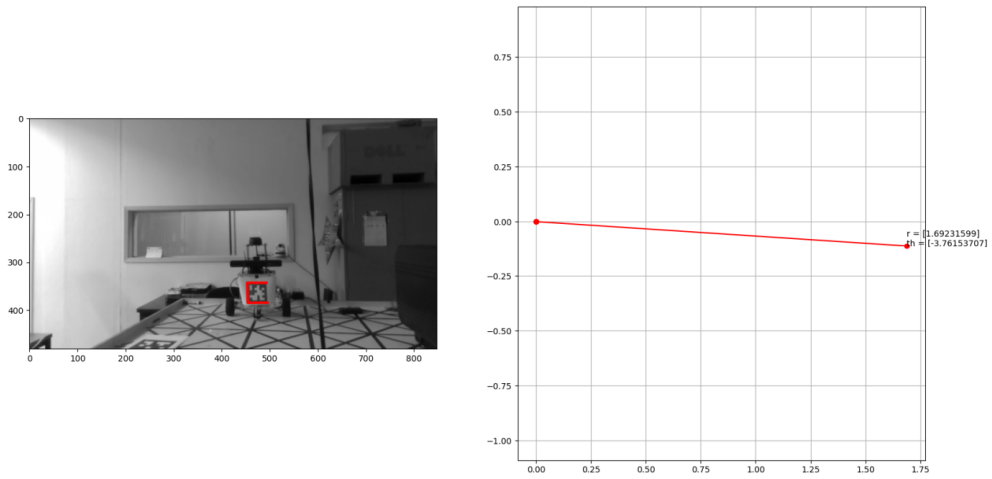

FIGURE 3.14 – Test of ArUco tag for detection and pose estimation. In the picture on the left, it is shown the perspective of Omni as it detects the tag on Isis, on the plot in the right, a vector with magnitude representing the distance to the tag and angle of detection.

to the camera is extracted by knowing the FOV and resolution of the image. Depth information from the RGB-D cameras were used to improve the measurement of distance to the marker.

3.2.2.2 Environment

The experiments were carried out on laboratory 1224 and LMI corridor located at the ITA building. Some wooden walls and boxes were placed on the ground to provide more complexity to the available area. The height in which the robot sensors are placed were specifically designed to match the wooden walls height, however, even tiny bumps on the ground or the unevenness of the floor can make a distant measurement pass through these walls, generating wrong data. To account for this, some walls were reinforced with a larger surface area. Figure 3.15 and 3.16 show photos of these environments.

MR-SLAM on the real world experiment was performed on low velocities to prevent possible collisions caused by failures in communication or software errors that could lock the code. The communication was done using a WiFi router that served as a bridge to connect the laptop running the central code with the robots on the same network.

FIGURE 3.15 – Lab 1224 environment.

FIGURE 3.16 – LMI corridor environment.

4 Experimental Results

This chapter describes the experimental results for simulated and real world tests for evaluation of the proposed solution. At first, simulation results from two different maps and different robot configurations are presented and next the real world experiment.

4.1 Simulation results

The simulated experiments were divided as small and large environments, and each environment was explored with 1, 3 or 5 robots, which are cardinally named. The results are shown for each robot as the following graphs:

1. Line plot of coverage versus iteration.
2. Line plot of estimated poses and map (point cloud and occupancy grid).
3. Line plot of error between estimated pose and ground truth.

On the graphs listed above, the coverage is in percentage of total area, the state of the robot is either exploring or idle (finished coverage), poses are in meters and radians, point cloud maps are in meters and occupancy grid maps are in grid units.

All simulated experiment cases show the initial pose of each robot according to the Gazebo coordinate system which, for the two environments, are set with origin at the center as shown in Figure 4.1.

The small environment has 2 experiments using 1 and 3 robots respectively, while the large environment has 2 cases using 1 and 5 robots respectively.

4.1.1 Small environment

4.1.1.1 Case 1: Single-robot exploration

Case 1 was performed with a single robot starting with poses described in Table 4.1. The experiment took 335 iterations to finish. Figures 4.2, 4.3 and 4.4 show the results.

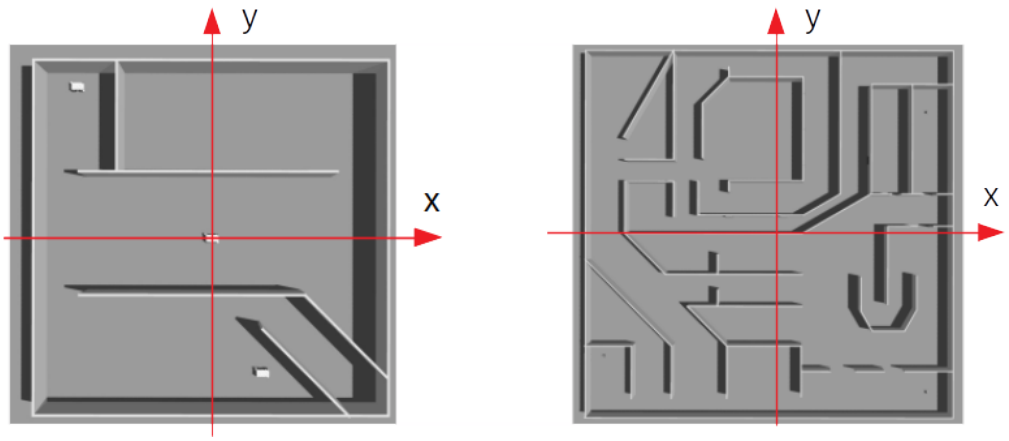

FIGURE 4.1 – Gazebo coordinate frames on top of the two environments.

TABLE 4.1 – Case 1: initial positions on Gazebo coordinate frame.

	X coordinate (m)	**Y coordinate (m)**	**Orientation (rad)**
Robot 1	-8	9	0

The parameters of the MR-SLAM algorithm were:

1. Total environment area: 400 m^2.
2. Coverage percentage threshold: 98% of total area.
3. Occupancy grid resolution: 15 cells per meter.
4. Occupancy grid size: 600x600 cells.
5. Exploration-exploitation balance parameter: $\lambda = 1$.
6. Controller fixed linear velocity: $v = 0.3\ m/s$.
7. Controller maximum angular velocity: $\omega = 1.5\ rad/s$.
8. Pure pursuit look ahead distance: 1.0 m.

Figure 4.2 shows a plot containing the robot's coverage at each iteration, represented by a continuous blue line. The horizontal dashed red line marks the coverage threshold, that, when crossed by a robot, terminates the task and stop movement from this point on.

The error graph on Figure 4.3 shows that the error in the x coordinate peaked at around 0.18 m while the y coordinate was at around 0.17 m in absolute values. The orientation error was relatively low, peaking at around 0.008 radians (0.458°).

The SLAM results on Figure 4.4 show the point cloud and occupancy grid map produced by the robot, as well as its trajectory (black line on left image).

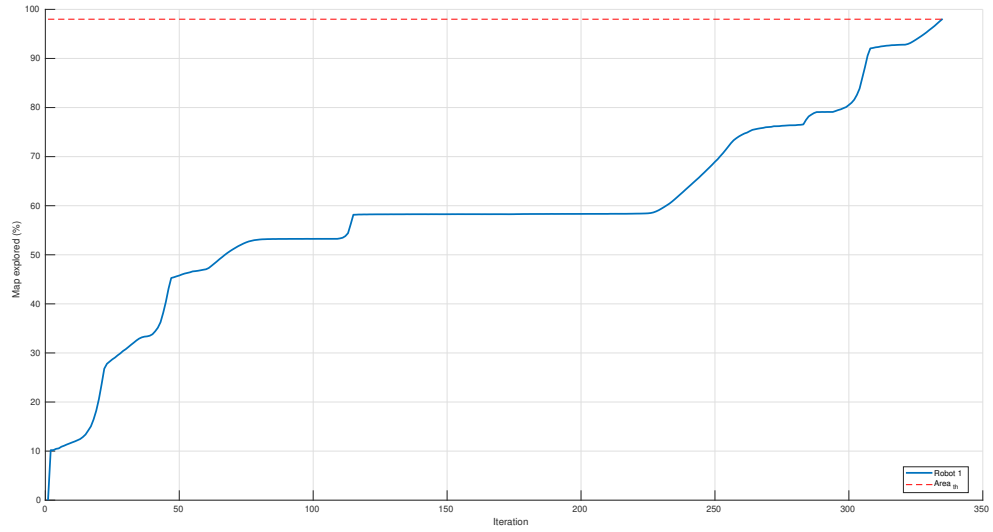

FIGURE 4.2 – Coverage history of the simulated experiment case 1.

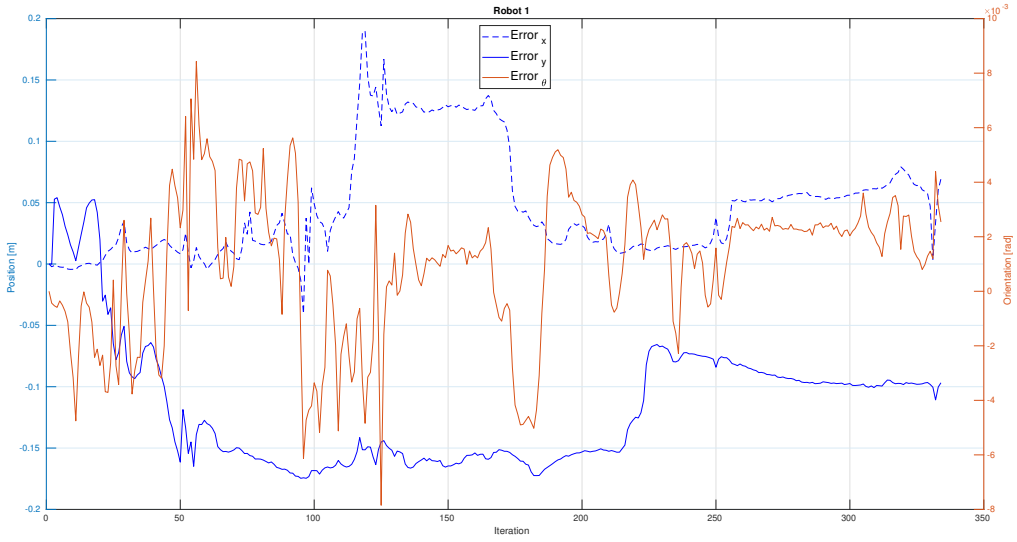

FIGURE 4.3 – Robot 1 pose error compared to the ground truth of the simulated experiment case 1.

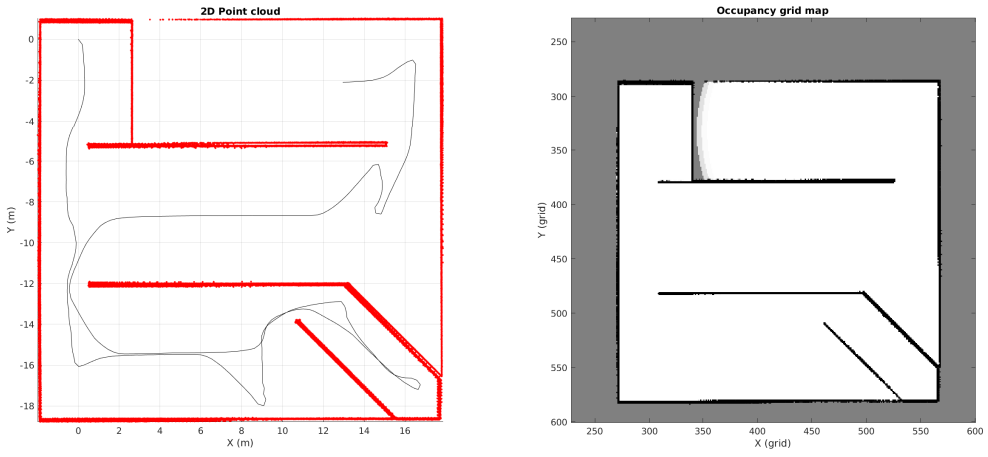

FIGURE 4.4 – Robot poses on point cloud map (left) and occupancy grid map (right) of the simulated experiment, case 1, robot 1.

4.1.1.2 Case 2: Multi-robot exploration with 3 robots

Case 2 was performed with 3 robots starting with poses described in Table 4.2. The experiment took 153 iterations to finish. Figures 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 and 4.11 show the results.

TABLE 4.2 – Case 2: initial positions on Gazebo coordinate frame.

	X coordinate (m)	**Y coordinate (m)**	**Orientation (rad)**
Robot 1	-8	9	0
Robot 2	-3	9	$3\pi/2$
Robot 3	4	-9	π

The parameters of the MR-SLAM algorithm were:

1. Total environment area: 400 m^2.
2. Coverage percentage threshold: 98% of total area.
3. Occupancy grid resolution: 15 cells per meter.
4. Occupancy grid size: 600x600 cells.
5. Exploration-exploitation balance parameter: $\lambda = 1$.
6. Controller fixed linear velocity: $v = 0.3\ m/s$.
7. Controller maximum angular velocity: $\omega = 1.5\ rad/s$.

8. Pure pursuit look ahead distance: 1.0 m.

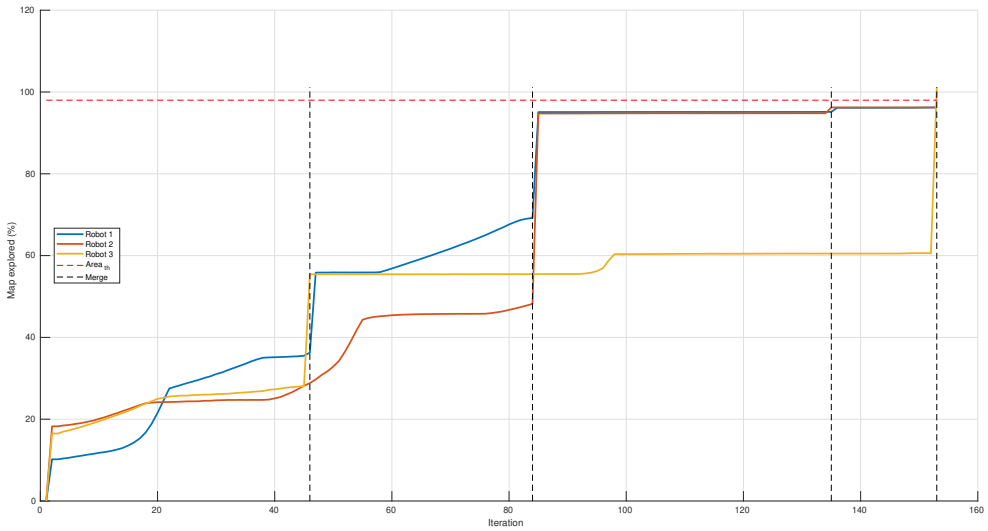

FIGURE 4.5 – Coverage history of simulated experiment case 2.

Figure 4.5 shows a plot containing the coverage of each robot at each iteration, represented by the continuous colored lines. Vertical dashed black lines mark iterations where map merging occurs, noticeable by the boost in coverage of at least two robots. The task finished after a merge between Robots 1 and 3 on iteration 153.

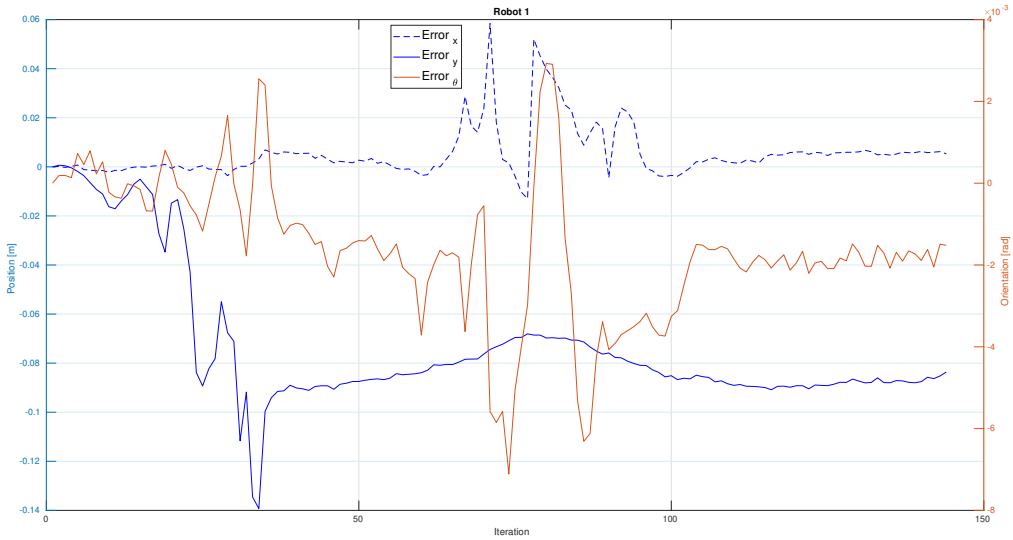

FIGURE 4.6 – Robot 1 pose error compared to the ground truth of the simulated experiment case 2.

Figures 4.6, 4.7 and 4.8 show the pose error compared to the ground truth. All robots have position error within a 0.2 meters margin and orientation error peaks for Robots 1 and 3 within 0.02 rad (1.1459°).

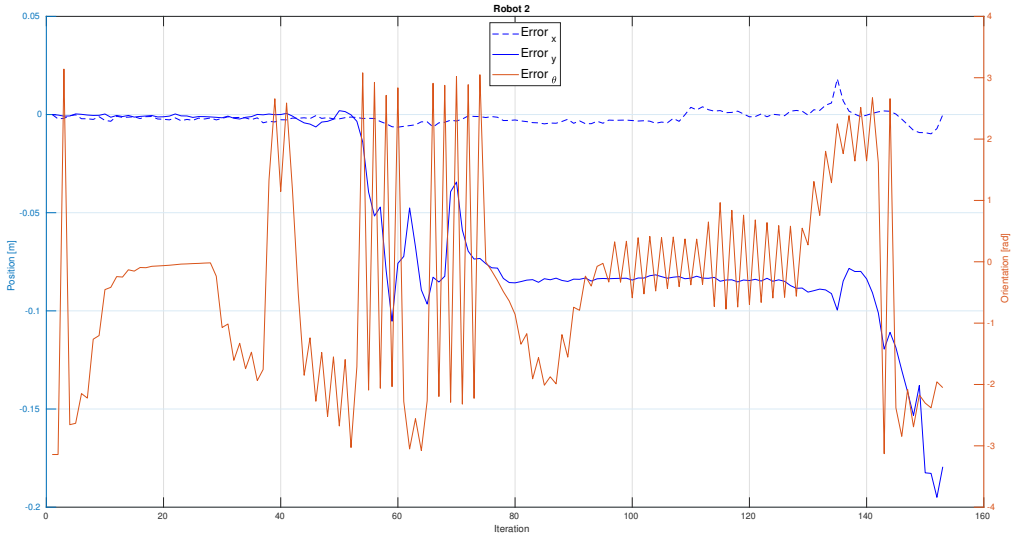

FIGURE 4.7 – Robot 2 pose error compared to the ground truth of the simulated experiment case 2.

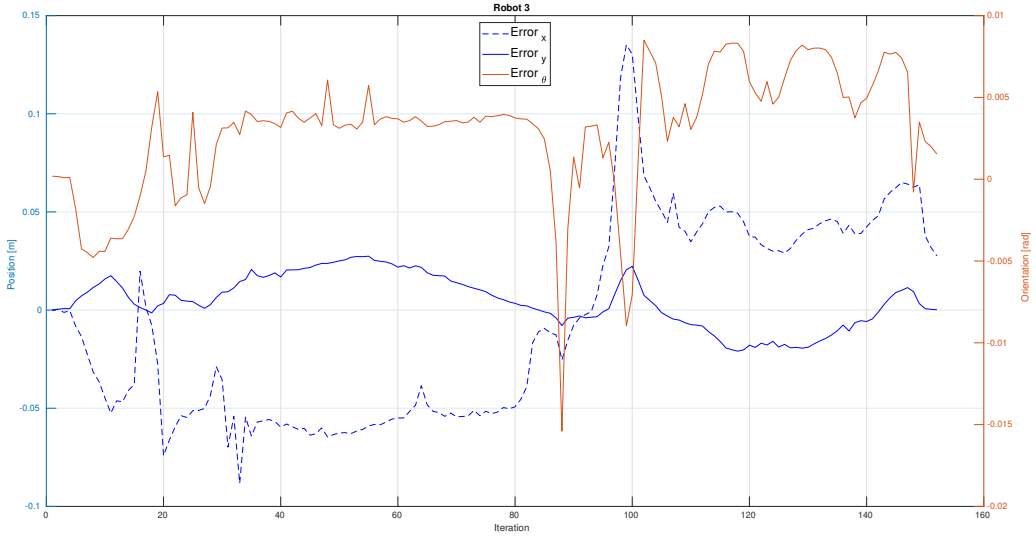

FIGURE 4.8 – Robot 3 pose error compared to the ground truth of the simulated experiment case 2.

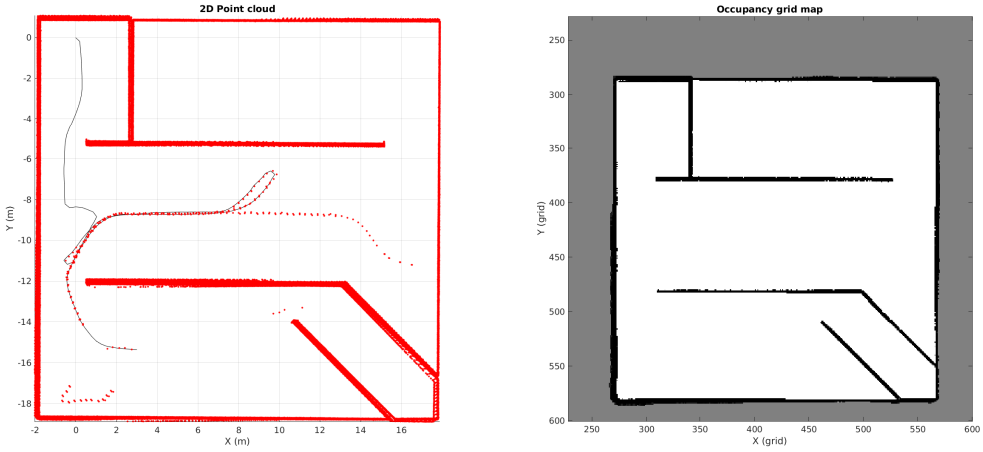

FIGURE 4.9 – Robot poses on point cloud map (left) and occupancy grid map (right) of the simulated experiment, case 2, robot 1.

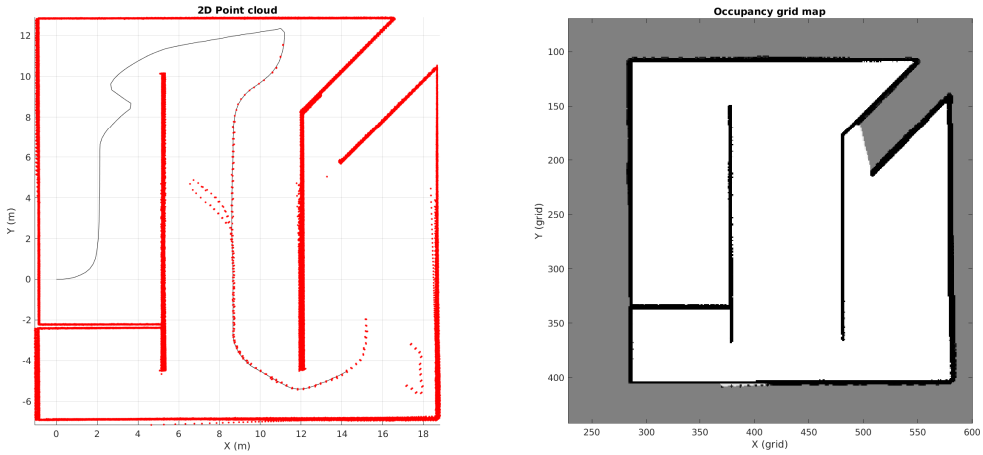

FIGURE 4.10 – Robot poses on point cloud map (left) and occupancy grid map (right) of the simulated experiment, case 2, robot 2.

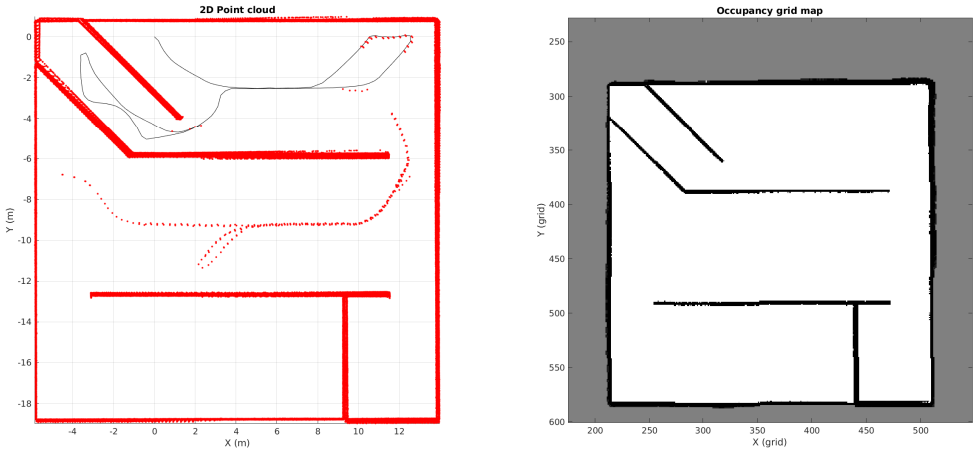

FIGURE 4.11 – Robot poses on point cloud map (left) and occupancy grid map (right) of the simulated experiment, case 2, robot 3.

The SLAM results in Figures 4.9, 4.10 and 4.11 show the quality of maps produced by the robots, as well as their trajectory (black line in the left image). The point cloud showed a trace of red dots on the middle, which are the other robots seen as obstacles. These points are filtered out in the occupancy grid maps and do not impact the final result, unless a robot captures an idle robot multiple times, which was not the case.

4.1.2 Large environment

4.1.2.1 Case 3: Single-robot exploration

Case 3 was performed with a single robot starting with the pose described in Table 4.3. The experiment took 2634 iterations to finish. Figures 4.12, 4.13 and 4.14 show the results.

TABLE 4.3 – Case 3: initial positions on Gazebo coordinate frame.

	X coordinate (m)	**Y coordinate (m)**	**Orientation (rad)**
Robot 1	-5	20	0

The parameters of the MR-SLAM algorithm were:

1. Total environment area: 2500 m^2.
2. Coverage percentage threshold: 90% of total area.
3. Occupancy grid resolution: 8 cells per meter.

4. Occupancy grid size: 900x900 cells.

5. Exploration-exploitation balance parameter: $\lambda = 0$.

6. Controller fixed linear velocity: $v = 0.3\ m/s$.

7. Controller maximum angular velocity: $\omega = 1.5\ rad/s$.

8. Pure pursuit look ahead distance: $1.0\ m$.

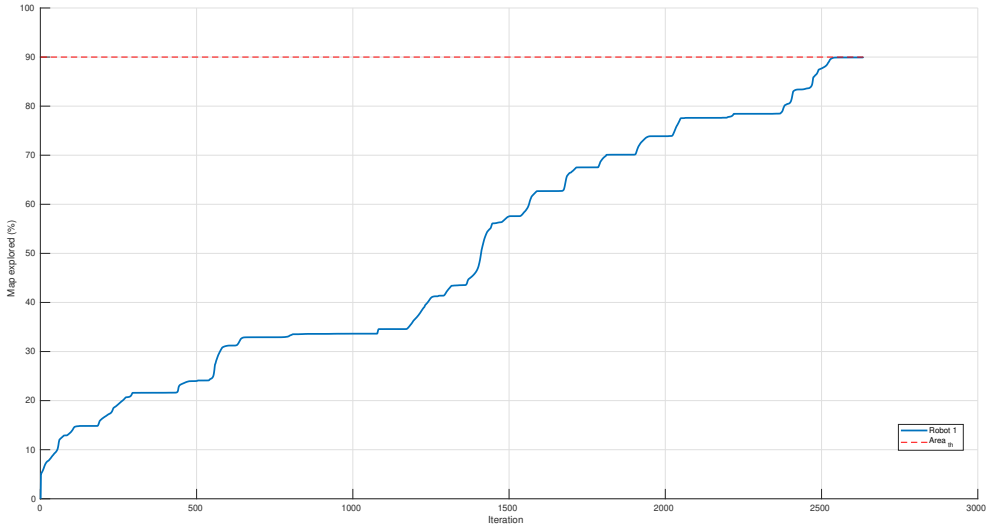

FIGURE 4.12 – Coverage history of the simulated experiment case 3.

Figure 4.12 shows a plot containing the robot's coverage at each iteration, represented by the continuous colored lines. The horizontal dashed red line marks the coverage threshold, that, when crossed by a robot, makes the robot to stop its movement from now on.

Figure 4.13 shows the pose error compared to the ground truth. The error in the x coordinate peaked at around 1.25 m while the y coordinate was at around 1.2 m in absolute values. The orientation error peaked at around 0.04 rad (2.29°).

The SLAM result on Figure 4.14 shows the map quality produced by the robot, as well as its trajectory (black line on left image). The point cloud showed a trace of red dots on the middle, which are the other robots seen as obstacles.

4.1.2.2 Case 4: Multi-robot exploration with 5 robots

Case 4 was performed with 5 robots starting with the poses described in Table 4.4. The experiment took 1978 iterations to finish, in this case, the experiment continued running until all robots reached the percentage threshold. Figures 4.15, 4.16, 4.17, 4.18, 4.21, 4.22 and 4.23 show the results.

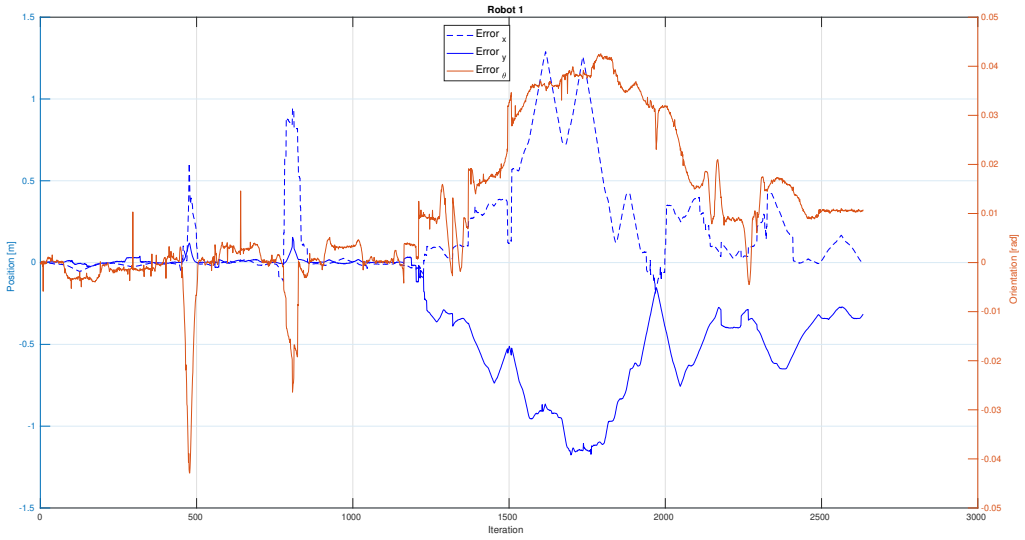

FIGURE 4.13 – Robot 1 pose error compared to ground truth of simulated experiment case 3.

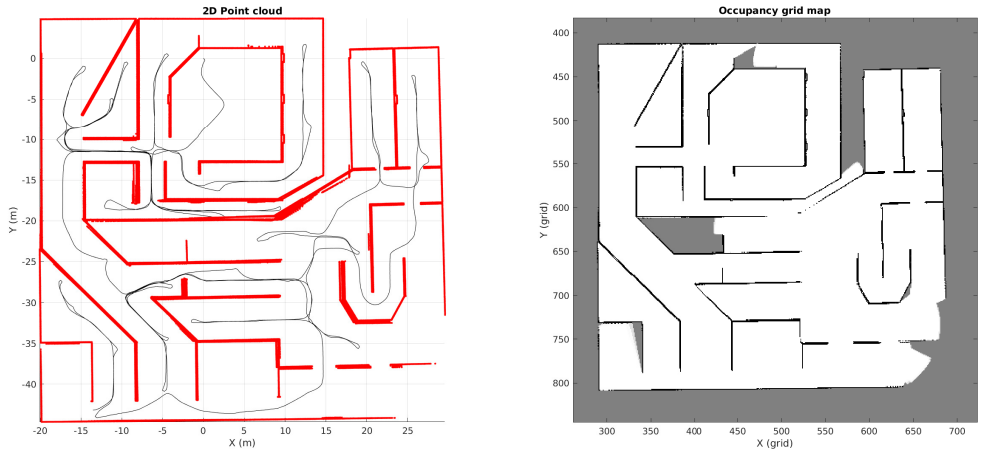

FIGURE 4.14 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 3, robot 1.

TABLE 4.4 – Case 4: initial positions in the Gazebo coordinate frame.

	X coordinate (m)	**Y coordinate (m)**	**Orientation (rad)**
Robot 1	-23	-22	0
Robot 2	22	-22	π
Robot 3	22	17	π
Robot 4	-5	20	0
Robot 5	-21	22	0

The parameters of the MR-SLAM algorithm were:

1. Total environment area: 2500 m^2.
2. Coverage percentage threshold: 90% of total area.
3. Occupancy grid resolution: 8 cells per meter.
4. Occupancy grid size: 900x900 cells.
5. Exploration-exploitation balance parameter: $\lambda = 0$.
6. Controller fixed linear velocity: $v = 0.3\ m/s$.
7. Controller maximum angular velocity: $\omega = 1.5\ rad/s$.
8. Pure pursuit look ahead distance: 1.0 m.

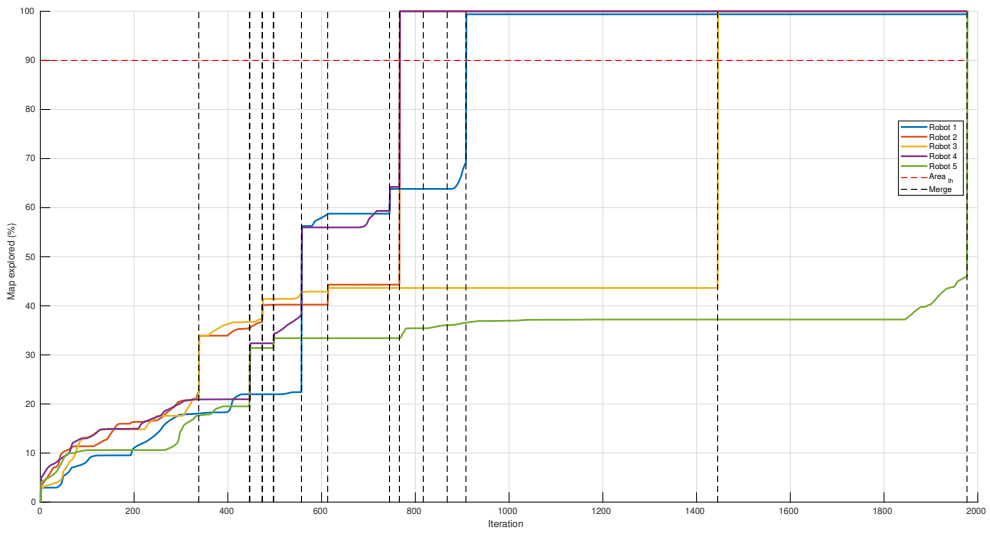

FIGURE 4.15 – Coverage history of the simulated experiment case 4.

Figure 4.15 shows a plot containing the coverage of each robot at each iteration, represented by the continuous colored lines. The horizontal dashed red line marks the coverage threshold, that, when crossed by a robot, makes the robot to stop its movement from that point on. Vertical dashed black lines mark iterations where map merging occurs, noticeable by the boost in coverage of at least two robots.

The experiment continued running until all robots reached the percentage threshold. At iteration 758, the first two robots, Robot 2 and 4, reached the percentage threshold, while the last robot took 1978 iterations to finish, still a lower number than the single-robot counterpart.

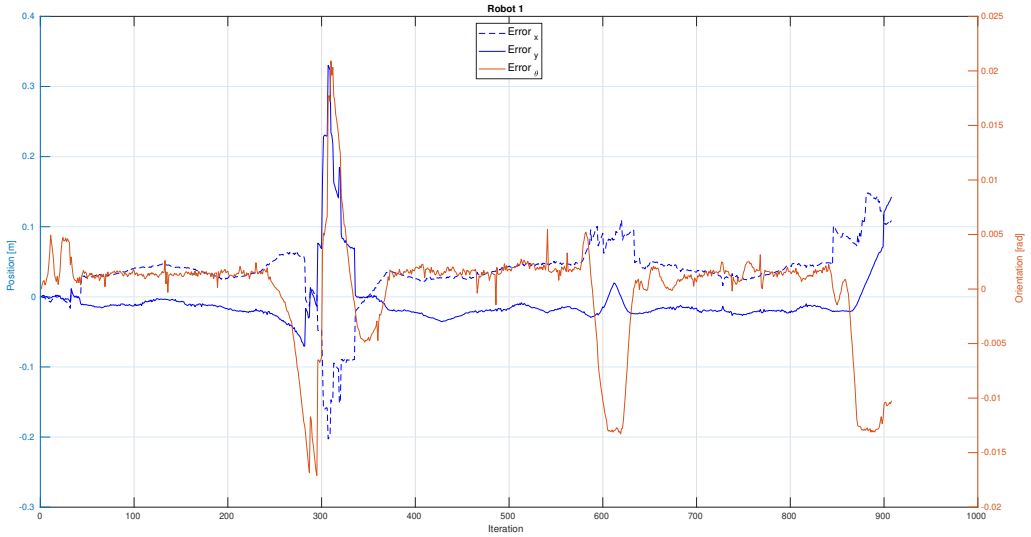

FIGURE 4.16 – Robot 1 pose error compared to the ground truth of the simulated experiment case 4.

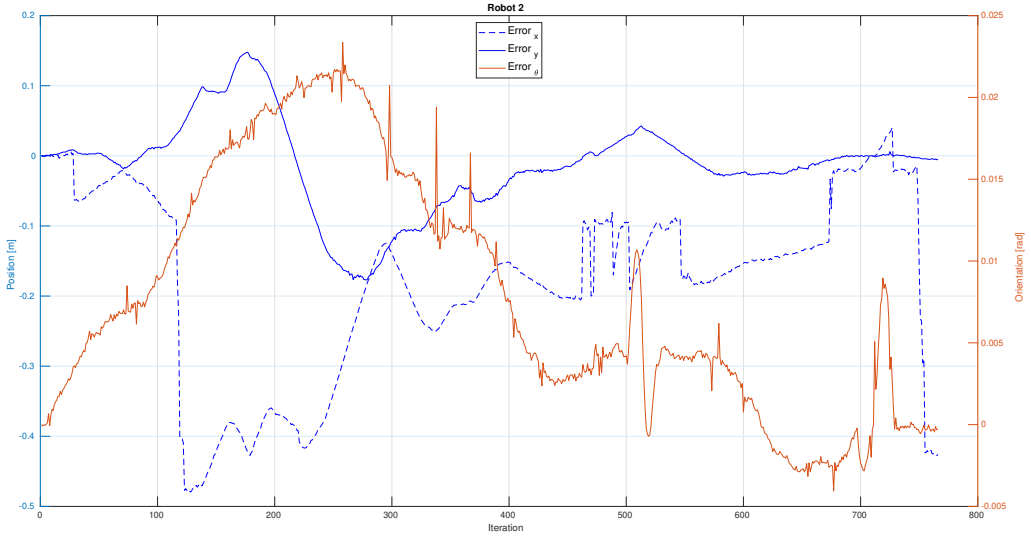

FIGURE 4.17 – Robot 2 pose error compared to the ground truth of the simulated experiment case 4.

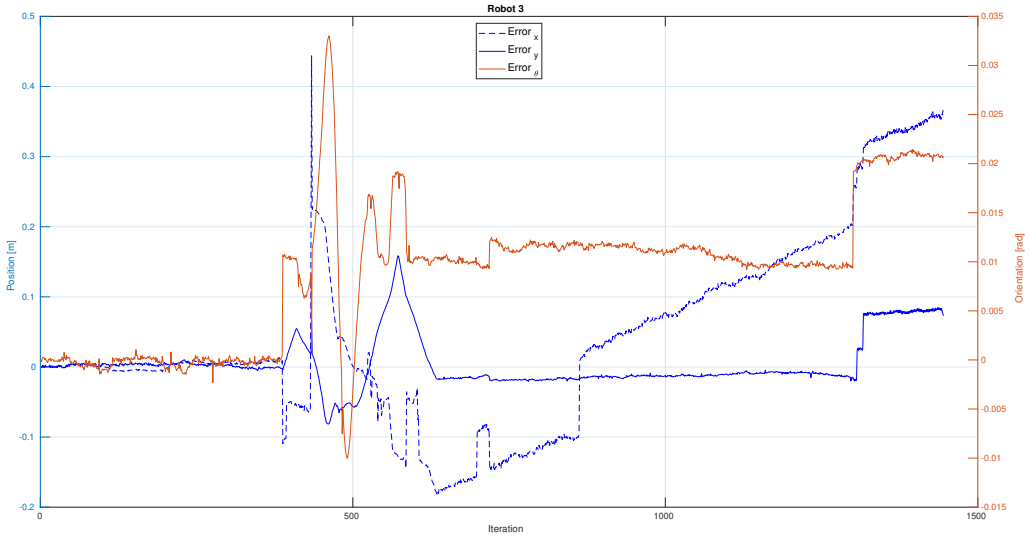

FIGURE 4.18 – Robot 3 pose error compared to the ground truth of the simulated experiment case 4.

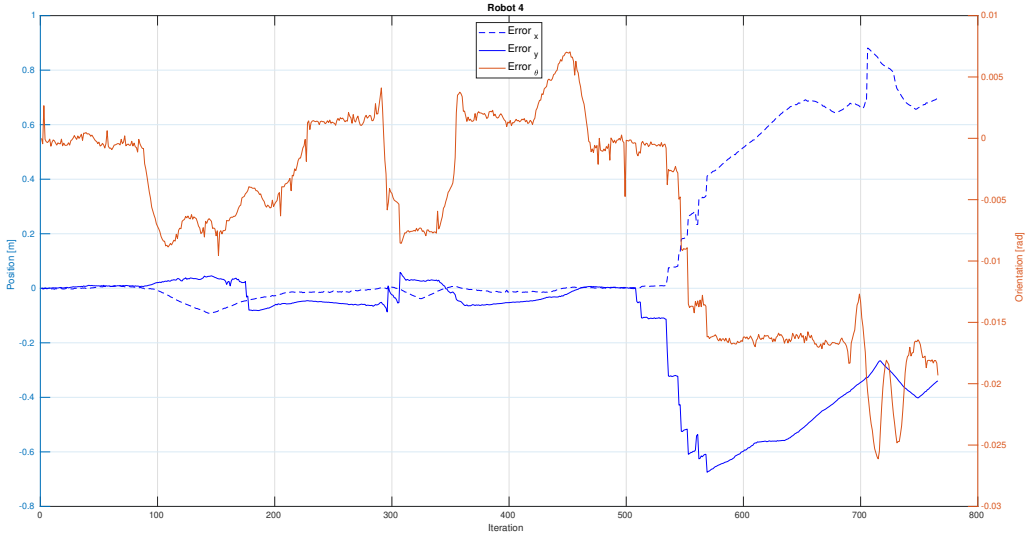

FIGURE 4.19 – Robot 4 pose error compared to the ground truth of the simulated experiment case 4.

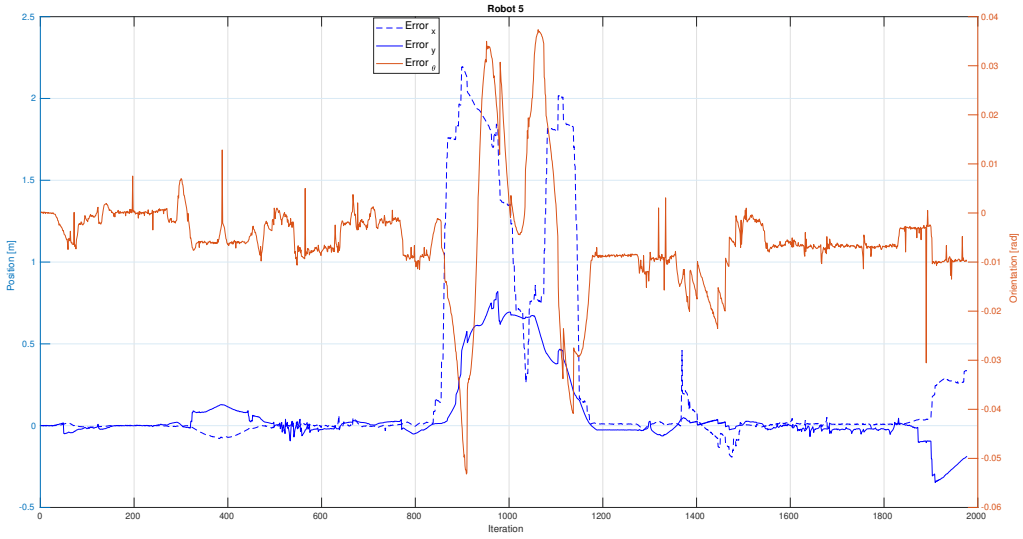

FIGURE 4.20 – Robot 5 pose error compared to the ground truth of the simulated experiment case 4.

Figures 4.16, 4.17, 4.18, 4.19 and 4.20 show the pose error compared to the ground truth. Robots 1, 2, and 3 have maximum error in position within 0.5 m in both coordinates and maximum orientation error within 0.025 rad (1.4323°). Robot 4 obtained a maximum position error of 0.9 m and maximum orientation error of 0.025 rad (1.4323°). Robot 5 had the largest absolute errors, with maximum x coordinate at 2.2 m and y coordinate at around 0.8 m, the orientation error peaked at 0.052 rad (2.9793805°).

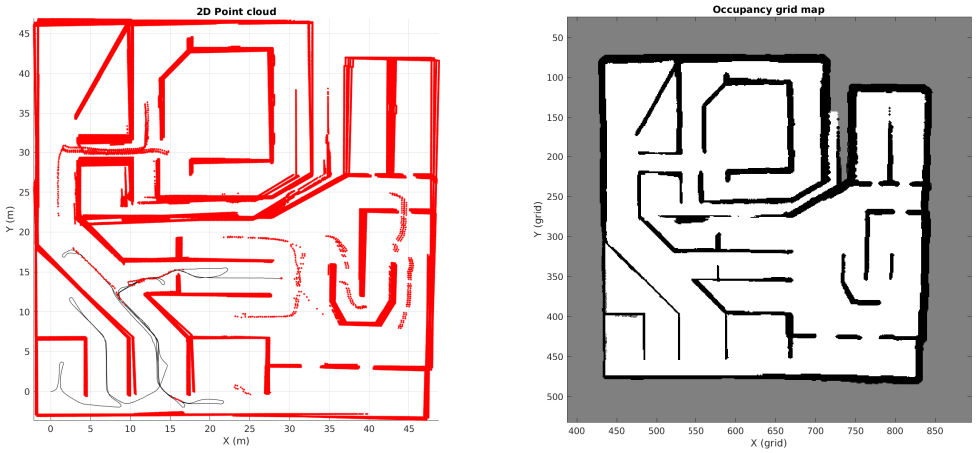

FIGURE 4.21 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 4, robot 1.

The SLAM results in Figures 4.21, 4.22, 4.23, 4.24 and 4.25 show the quality of maps produced by the robots, as well as their trajectory (black line on left image). The point

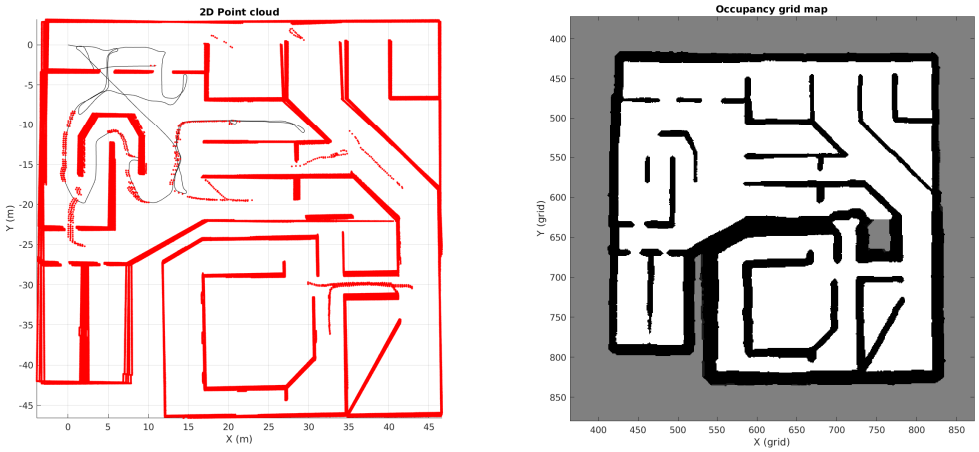

FIGURE 4.22 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 4, robot 2.

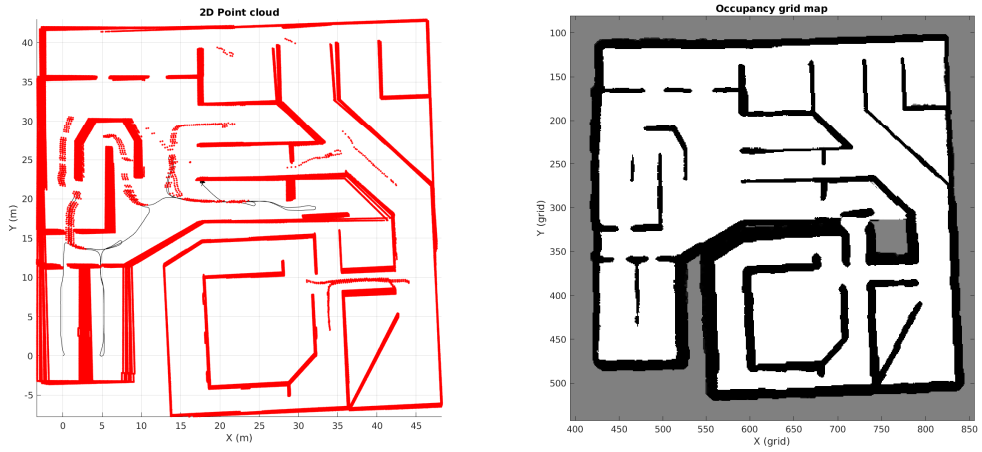

FIGURE 4.23 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 4, robot 3.

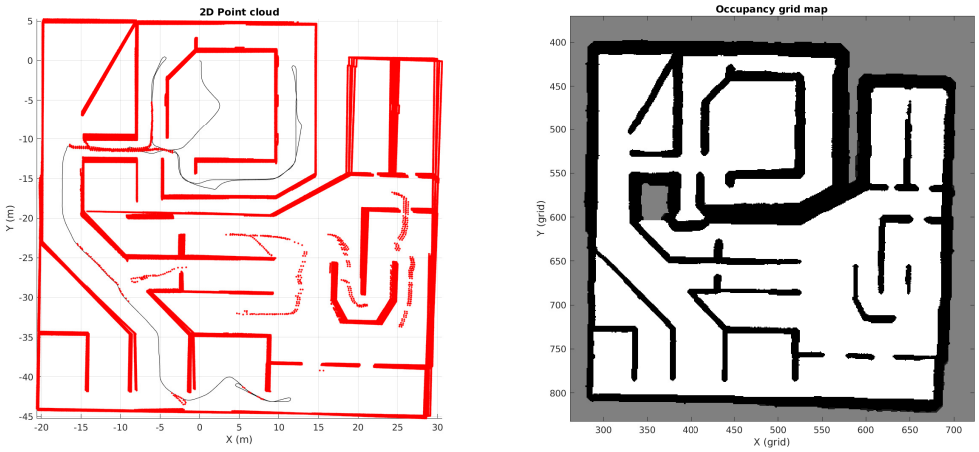

FIGURE 4.24 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 4, robot 4.

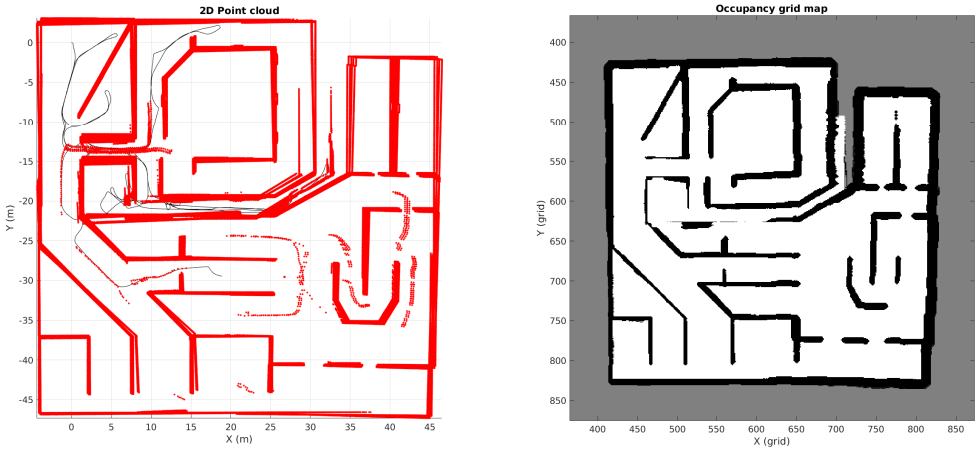

FIGURE 4.25 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the simulated experiment, case 4, robot 5.

cloud showed a trace of red dots on the middle, which are the other robots seen as obstacles.

The results showed an inconsistency around the center of the environment, where a corridor wall disappeared on the map estimation. This was a case of the so called featureless corridor, in which the robot must overcome the difficulty of estimating its position between very similar measurements from a corridor with no distinct features.

4.2 Real world results

The experiments were carried out using the two robots previously described, Omni and Isis, in a laboratory and corridor environments. The results and discussion are presented next.

4.2.1 Lab 1224

This section describes experiments performed in Lab 1224, the available space was segmented using wooden walls and cardboard boxes. The structure and dimensions are depicted in Figure 4.26.

FIGURE 4.26 – On the left, real dimensions in meters of the built environment. Right image is a photo of the environment.

There are some desks at the side and back of the room that are inside the built environment. The legs of the desks are detected by the robot sensors, however, the low metal piece that supports these legs are too close to the floor to be detected. To prevent the robots to reach such places, some wooden walls were placed specifically adjacent to these parts.

Another particular issue with the setup used is that the robots can still be detected through the wooden walls. Figure 4.27 shows a detection that happened while the multi-robot case experiment was being performed, the tags could be seen on the hollow space below the wall.

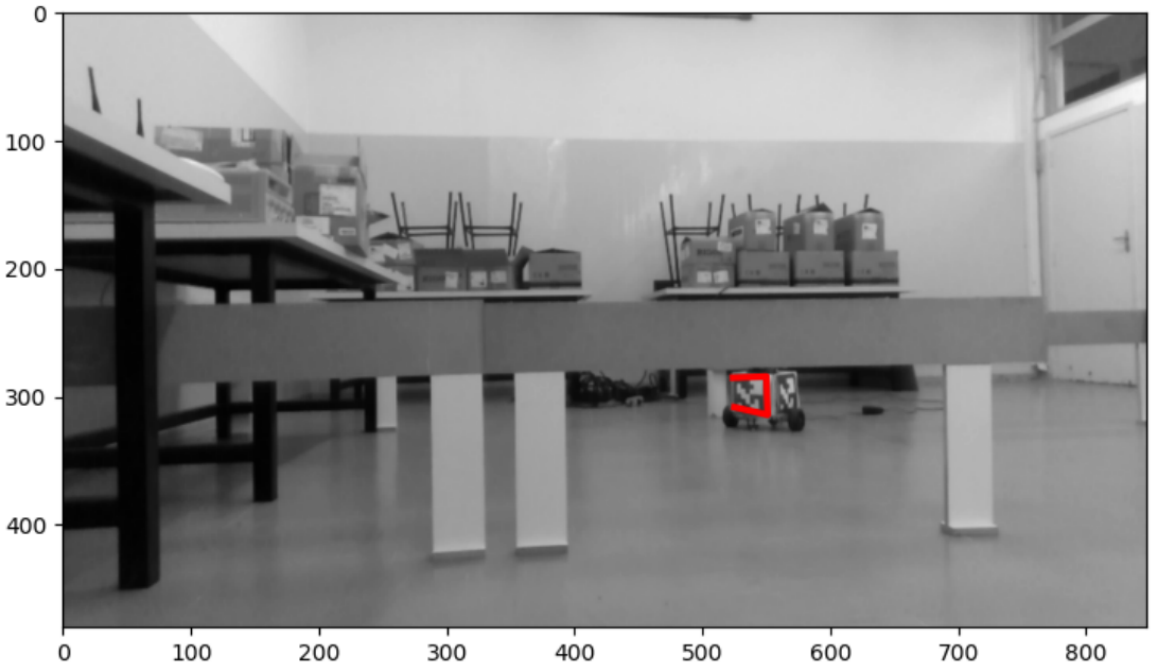

FIGURE 4.27 – Image captured by Omni showing a tag detection through the wooden wall (red marker).

This was treated as an unrealistic case, since the wooden walls are representing real walls on the environment. To solve this issue, a line of sight test was performed by simply checking if there was an obstacle on a small angle interval within the tag direction, and discarding the detection if positive.

4.2.1.1 Case 1: Single-robot exploration

Initially, the single-robot exploration was performed by the Isis robot. This experiment case took 77 iterations to be finished, the results are displayed in Figures 4.28 and 4.29.

The parameters of the SLAM algorithm were:

1. Total environment area: 53.86 m^2.
2. Coverage percentage threshold: 100% of total area.
3. Occupancy grid resolution: 15 cells per meter.
4. Occupancy grid size: 400x400 cells.

5. Exploration-exploitation balance parameter: $\lambda = 1$.

6. Controller fixed linear velocity: $v = 0.2\ m/s$.

7. Controller maximum angular velocity: $\omega = 1.0\ rad/s$.

8. Pure pursuit look ahead distance: $0.5\ m$.

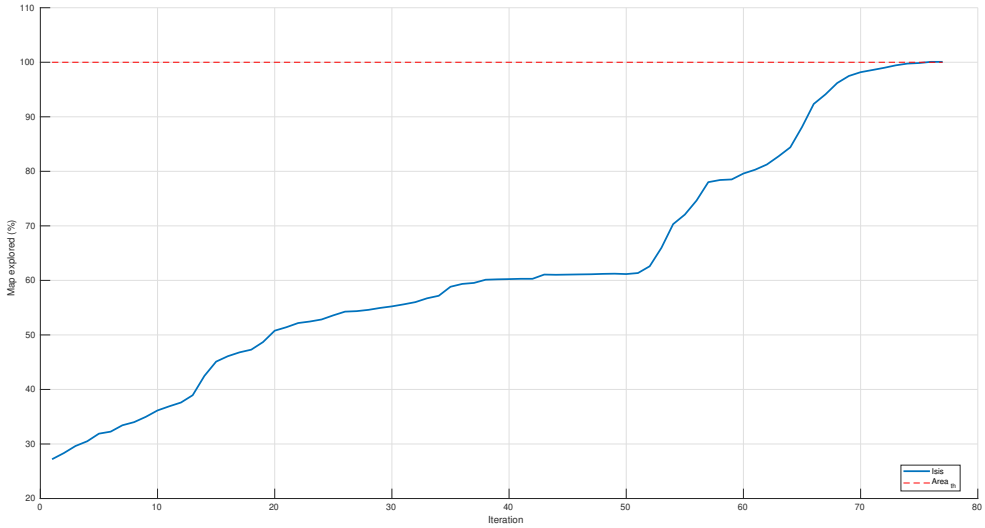

FIGURE 4.28 – Coverage history of the real experiment case 1.

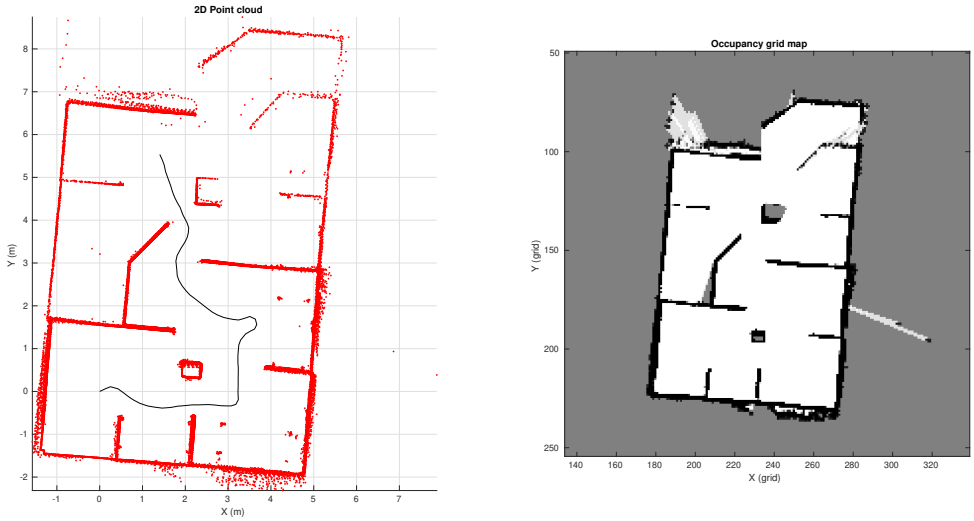

FIGURE 4.29 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the real experiment, case 1, Isis robot.

Figure 4.28 shows the evolution of the single robot case. The range of the LIDAR sensor was very significant compared to the dimensions of the environment, making the

explored percentage starting at around 30%, which contributed to a faster coverage per iteration rate if compared to the Omni robot.

Figure 4.29 shows the estimated map and positions for the real experiment case 1. The quality of the maps are comparable with the simulated ones, showing that the active SLAM algorithm performed reasonably good on real robots.

4.2.1.2 Case 2: Multi-robot exploration with 2 robots

The multi-robot exploration using Isis and Omni took 69 iterations to finish and only one merge occurred. Figures 4.30, 4.31, 4.32, 4.33 and 4.34 show the results.

The parameters of the MR-SLAM algorithm were:

1. Total environment area: 53.86 m^2.
2. Coverage percentage threshold: 100% of total area.
3. Occupancy grid resolution: 15 cells per meter.
4. Occupancy grid size: 400x400 cells.
5. Exploration-exploitation balance parameter: $\lambda = 1$.
6. Controller fixed linear velocity: $v = 0.2\ m/s$.
7. Controller maximum angular velocity: $\omega = 1.0\ rad/s$.
8. Pure pursuit look ahead distance: $0.5\ m$.

Figure 4.30 shows the evolution of coverage by the two robots. The experiment ended with the merging occurring on the last iteration, showing that the boost in explored percentage caused the maps to complement each other.

Figures 4.31 and 4.32 show the individual maps built by the robots shortly before the merge process. The merge occurred with Omni looking at Isis from the side, in a place the robots had clear line of sight to each other.

Compared to Isis, the Omni robot possess shorter range, which was an imposed limitation due to reasons described on the previous chapter. The quality of its map is relatively low, due to the sensor adaptation needed to produce a 2D map, but still consistent enough to enable collision-free navigation for the robot.

Figures 4.33 and 4.34 show the maps after the merge process. The environment was not fully explored, the algorithm terminated by the erroneous counting of occupancy grid cells that extrapolated the map limits, preventing an unknown part of being explored. These errors are caused by some aspects of the real world scenario, such as:

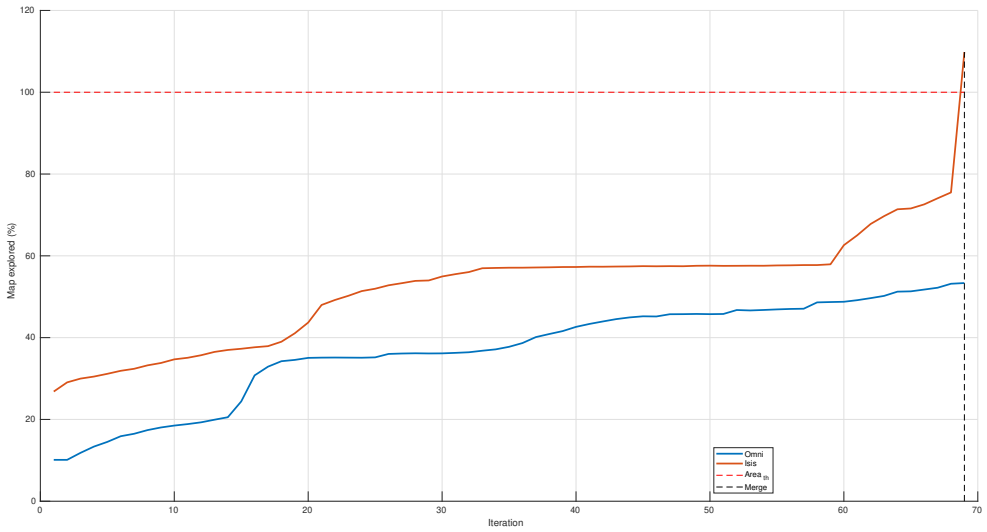

FIGURE 4.30 – Coverage history of the real experiment case 2.

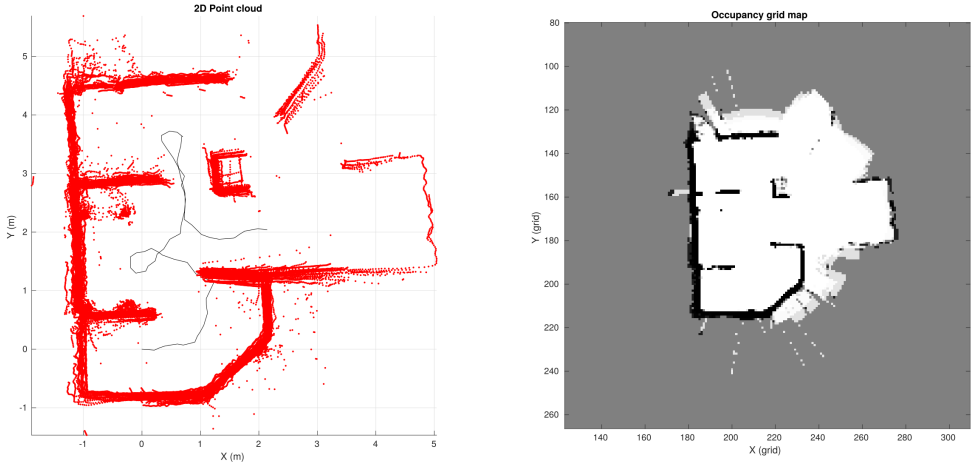

FIGURE 4.31 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the real experiment, case 2, Omni robot before map merge.

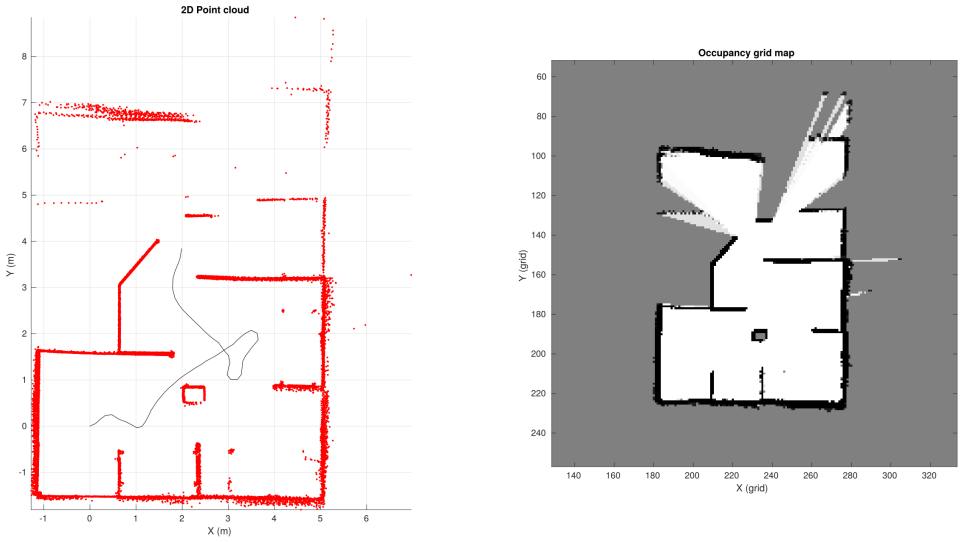

FIGURE 4.32 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the real experiment, case 2, Isis robot before map merge.

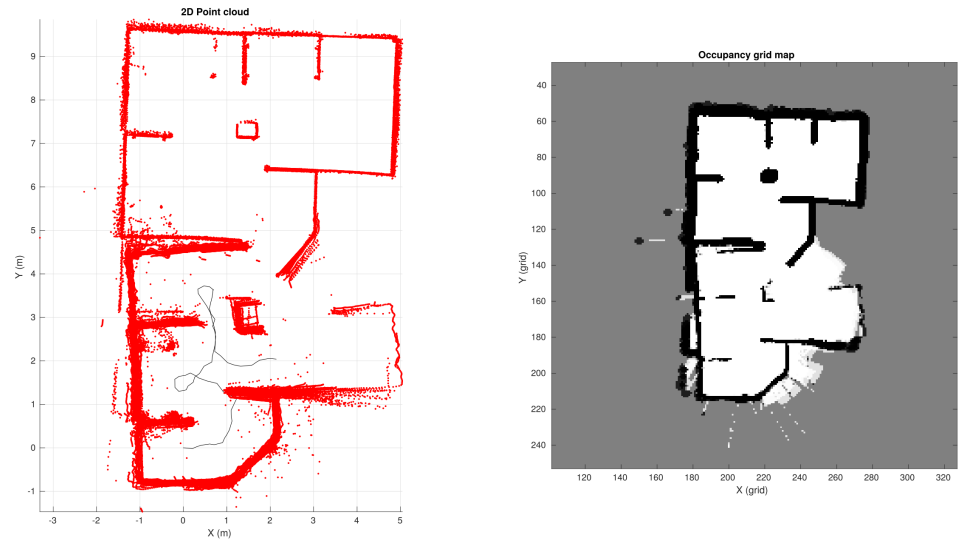

FIGURE 4.33 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the real experiment, case 2, Omni robot after map merge.

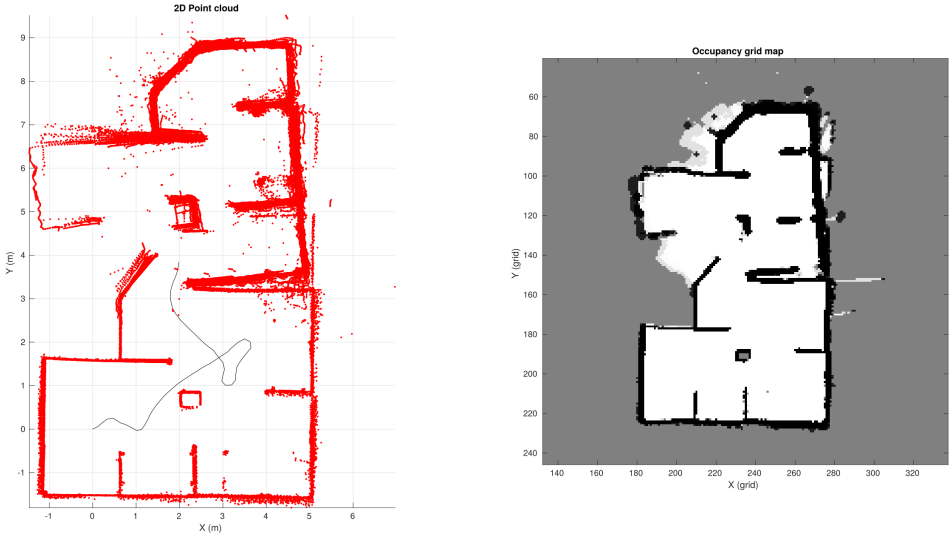

FIGURE 4.34 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the real experiment, case 2, Isis robot after map merge.

1. High uncertainty of map estimation, leading to models larger than the actual environment.

2. The computation of the transformation matrix was done using sensors measurements, which are corrupted by noise and leads to an imperfect map alignment.

Figure 4.35 shows the locations of the errors and part of the environment that was not explored.

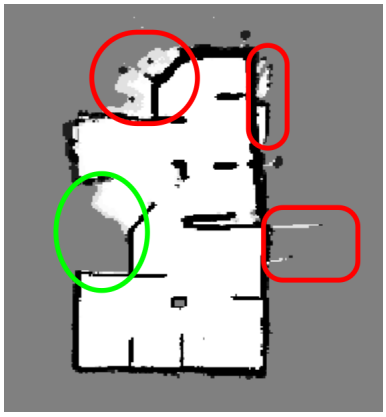

FIGURE 4.35 – Merged map showing errors that contribute to explored percentage counting. The red regions are not supposed to be mapped, while the green region is the missing part of the map.

A comparison between the ground truth and maps generated by single-robot and multi-robot SLAM shows that the overall format is still preserved. Figure 4.36 shows a side-by-side comparison between the estimated maps and the ground truth.

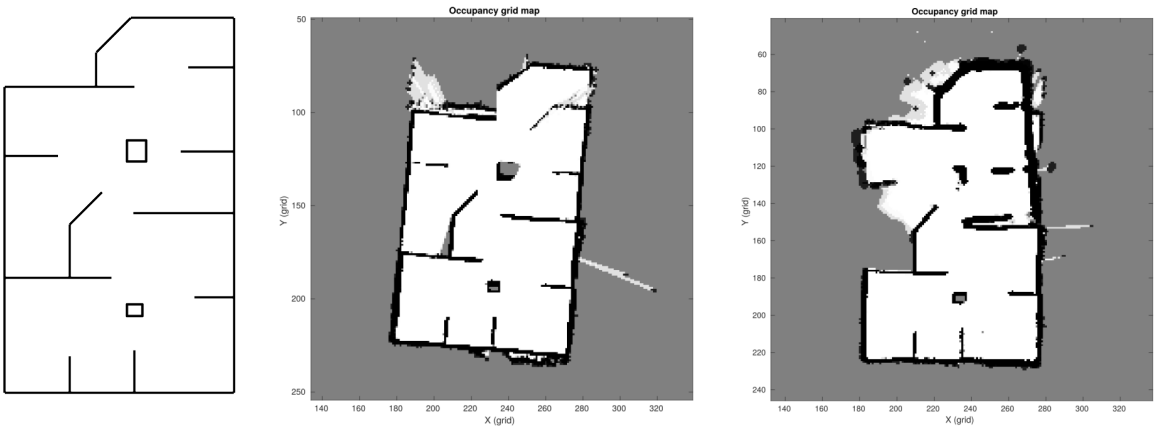

FIGURE 4.36 – Comparison between ground truth (left) and estimated maps from real world experiments using single (middle) and multi-robots (right).

4.2.2 LMI corridor

This section describes the experiment performed in LMI corridor, the available space was used unmodified to better verify the ability to navigate in unstructured environments. The top view of the corridor is shown in Figure 4.37.

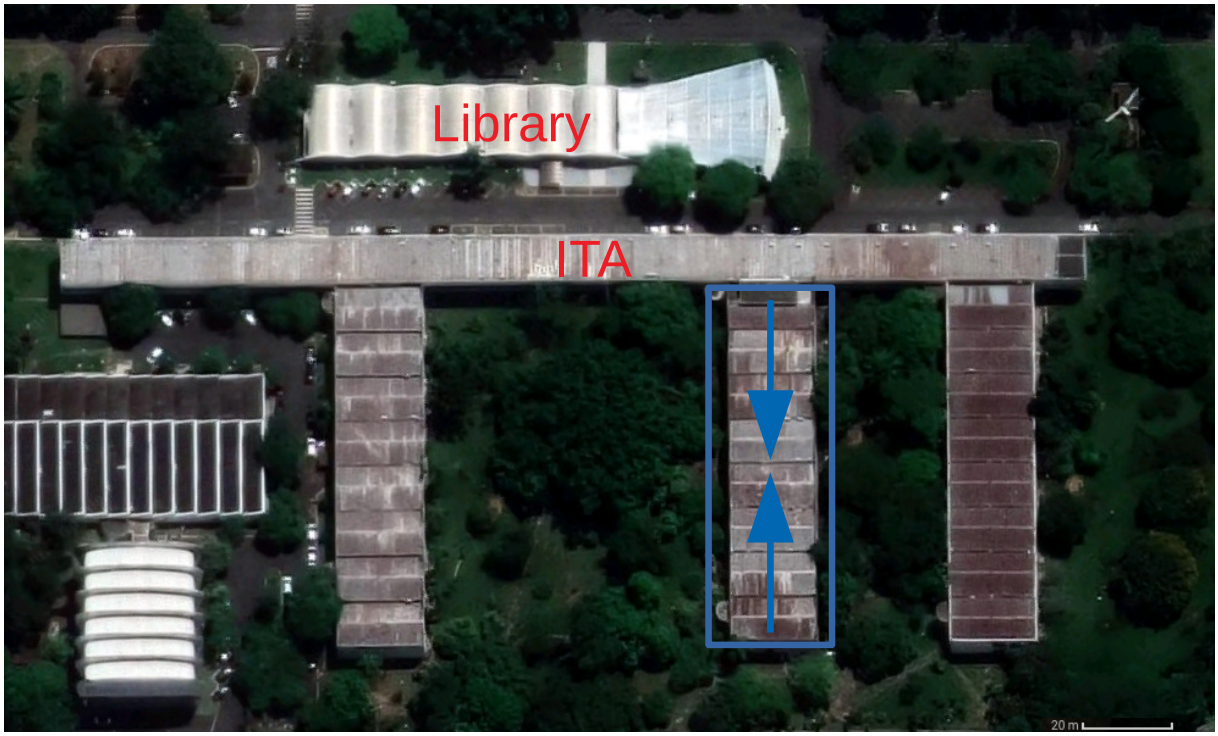

FIGURE 4.37 – Satellite view of the ITA and library buildings. The blue box shows the extent of the corridor and the blue arrows show the direction of motion of the robots. Adapted from Google (2021).

The experiment was conducted using only Isis, since it showed to perform better than Omni. To emulate a multi-robot operation, the map merging was computed in an offline manner using data from two parts of a single-robot active SLAM and forcing a rendezvous

at a predefined point. In the LMI corridor, the two parts correspond to each half of the corridor and the rendezvous happens at middle point. The procedure to emulate the rendezvous is shown in Figure 4.38. From numbers 1 to 4: the robot performs the first half of the corridor and manually stops at middle point; a marker is placed at the location the robot stopped and its orientation is traced on the floor; the second half is explored until the robot detects the marker on the floor; the same process of placing the marker is done for the second half and the robot returns to the last pose in the first half to detect the new marker.

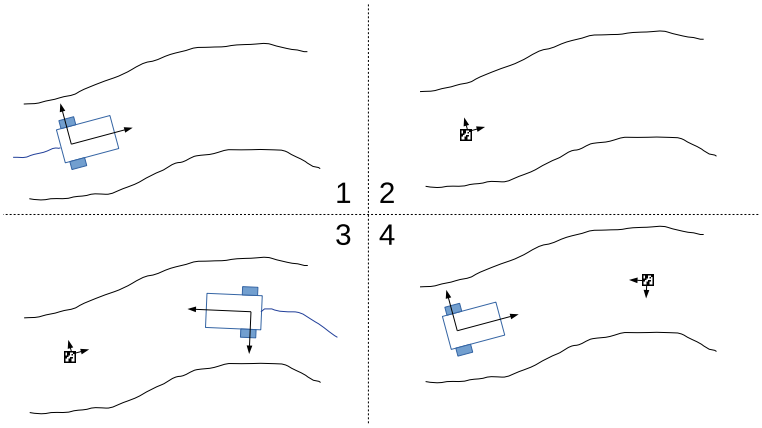

FIGURE 4.38 – Offline rendezvous procedure.

4.2.2.1 Case 3: Multi-robot exploration with 2 robots and offline map merging

The multi-robot exploration using Isis in the took 295 and 213 iterations in the first and second half respectively to finish and only one merge occurred. This experiment finished manually, so the coverage graphs were not generated. Figures 4.39, 4.40 and 4.41 show the results.

The parameters of the MR-SLAM algorithm were:

1. Occupancy grid resolution: 10 cells per meter.
2. Occupancy grid size: 2000x2000 cells.
3. Exploration-exploitation balance parameter: $\lambda = 1$.
4. Controller fixed linear velocity: $v = 0.2\ m/s$.
5. Controller maximum angular velocity: $\omega = 1.0\ rad/s$.
6. Pure pursuit look ahead distance: $1.0\ m$.

Figures 4.39 and 4.40 show the individual maps built by the robots before the merge process. Green dots marks the start and finishing poses along with the letters “S” and “F”, respectively.

The robots did not visit a known area along its trajectory, which reflected in the skewness that occurred on the map estimation and is not representative of the real environment. This result suggests that, for large scale environments, the absence of loop closure detection can impact the accuracy of the maps produced.

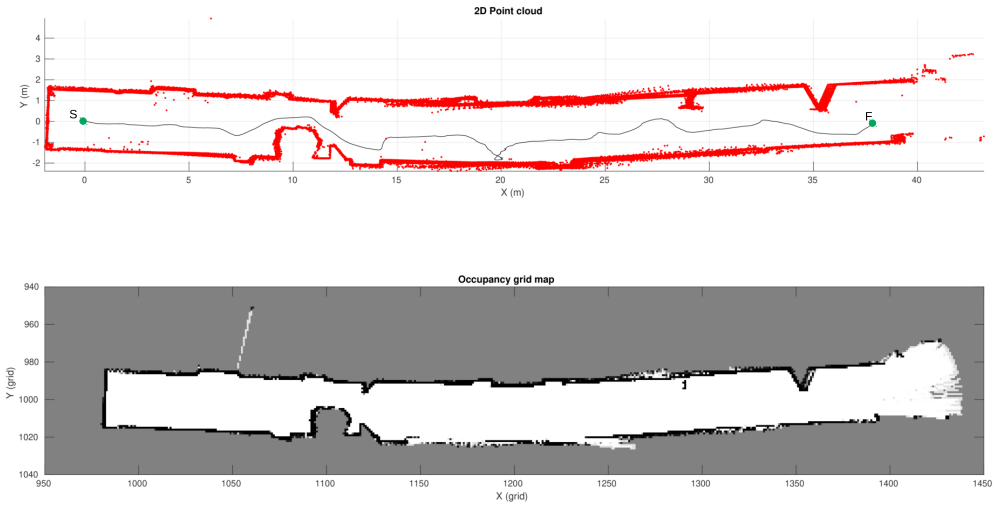

FIGURE 4.39 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the LMI corridor, case 3, first half, Isis robot before map merge.

Figure 4.41 shows the map merged relative to the second half map coordinate system. The

Figure 4.42 shows a zoom of the section where the rendezvous occurred in the point cloud map. The misalignment between the corridor walls is caused by the use of the direct computation of the transformation using only the inter-robot measurements, such alignment problem could be solved by a process to improve the transformation estimate by matching the line features of the walls.

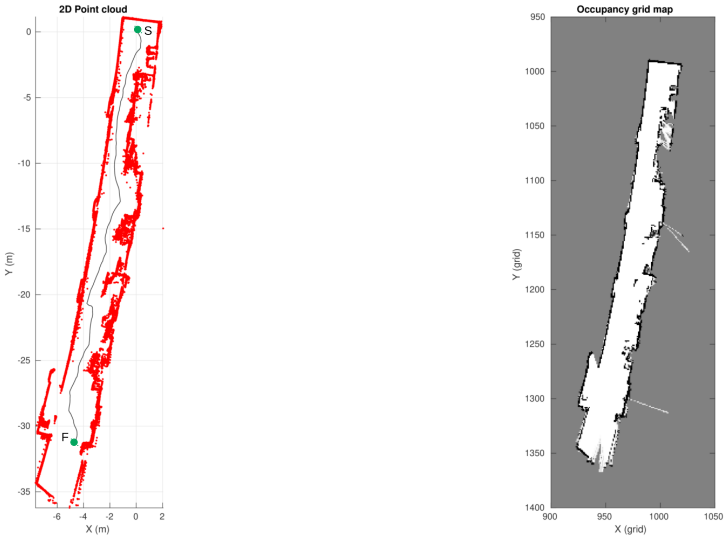

FIGURE 4.40 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the LMI corridor, case 3, second half, Isis robot before map merge.

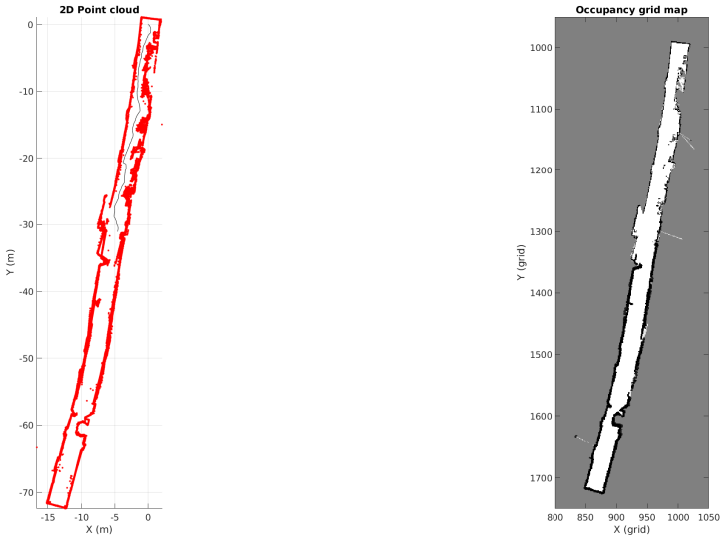

FIGURE 4.41 – Robot poses on the point cloud map (left) and the occupancy grid map (right) of the LMI corridor, case 3, Isis robot after map merge.

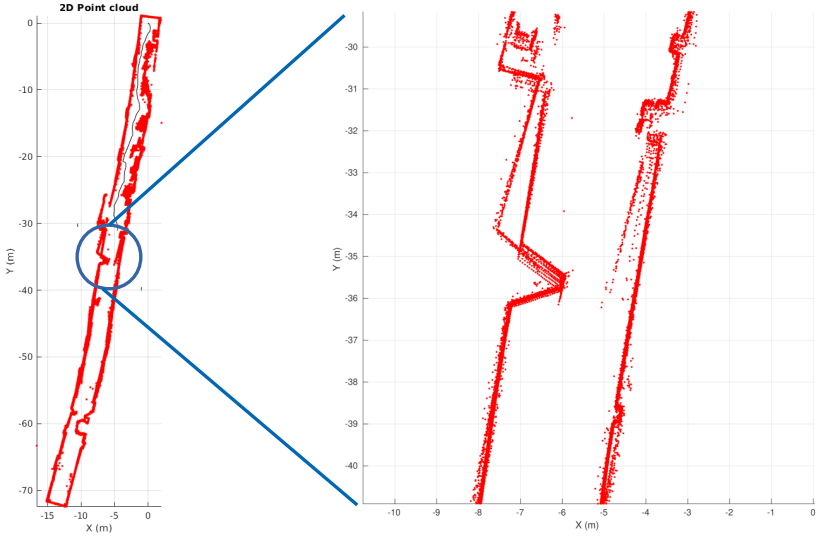

FIGURE 4.42 – Rendezvous section zoom.

5 Conclusion

This dissertation proposed the solution for the active MR-SLAM problem for the task of coverage of unknown environments, where multiple mobile robots must autonomously explore an environment using their sensors to estimate a map and their location on it. Experiments were performed in both simulated and real world environments to validate the solution.

Results from the simulated experiments showed that, using the proposal solution, the addition of multiple robots to perform active SLAM directly implies faster task completion by all robots, as foreseen by the literature. Although the coordination and exploration target selection was not optimal, the used approach resulted in good precision for the SLAM estimates and coverage over time increased in a satisfactory manner.

The use of simplified coordination rules was shown to be sufficient to solve the problem of repetitive target assignment for the scope of the experiments. With increased number of robots, this strategy may not hold as expected.

Real world experiment results showed that the solution provided good results. Although there were some adaptations and limitations imposed by the hardware and environment available, the proposed solution had the expected overall performance.

For future works, the following alternatives can be investigated:

1. Validation of the proposed solution in outdoor environments, as a mean to generalize the approach. In this context, the objective will not be complete coverage, but it will be a search and rescue application.

2. The extension for 3D estimation for both maps and poses, RGB-D cameras can be used as a sensor for this kind of estimation.

3. Make the architecture a decentralized one, where the robots do not depend on a central computer; this approach assumes limited communication between robots.

4. Validation with a larger team of robots using a more sophisticated set of rules for coordination.

Bibliography

AGUERO, C.; KOENIG, N.; CHEN, I.; BOYER, H.; PETERS, S.; HSU, J.; GERKEY, B.; PAEPCKE, S.; RIVERO, J.; MANZO, J.; KROTKOV, E.; PRATT, G. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. **Automation Science and Engineering, IEEE Transactions on**, v. 12, n. 2, p. 494–506, April 2015. ISSN 1545-5955.

AKUSENSE. **Akusense Technology Co., Ltd.** 2021. Available at `http://akusense.com/`. Accessed: 17-05-2021.

ANDERSONE, I. The characteristics of the map merging methods: A survey. **J. Riga Technical University**, v. 41, p. 113–121, 01 2010.

ANDERSONE, I. The influence of the map merging order on the resulting global map in multi-robot mapping. **Applied Computer Systems**, v. 13, n. 1, p. 22–28, 2012. Available at: https://doi.org/10.2478/v10312-012-0003-5.

BIRK, A.; CARPIN, S. Merging occupancy grid maps from multiple robots. **Proceedings of the IEEE**, v. 94, n. 7, p. 1384–1397, 2006.

BORENSTEIN, J.; KOREN, Y. The vector field histogram-fast obstacle avoidance for mobile robots. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 278–288, 1991.

BUONOCORE, L.; JR, C. N.; NETO, A. Solving the indoor slam problem for a low-cost robot using sensor data fusion and autonomous feature-based exploration. *In*: . **Proceedings** [...]. [*S.l.: s.n.*], 2012.

BURGARD, W.; MOORS, M.; FOX, D.; SIMMONS, R.; THRUN, S. Collaborative multi-robot exploration. *In*: **Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)**. **Proceedings** [...]. [*S.l.: s.n.*], 2000. v. 1, p. 476–481 vol.1.

CADENA, C.; CARLONE, L.; CARRILLO, H.; LATIF, Y.; SCARAMUZZA, D.; NEIRA, J.; REID, I.; LEONARD, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. **IEEE Transactions on Robotics**, v. 32, n. 6, p. 1309–1332, 2016.

COULTER, R. C. **Implementation of the Pure Pursuit Path Tracking Algorithm**. Pittsburgh, PA, January 1992.

DINNISSEN, P.; GIVIGI, S. N.; SCHWARTZ, H. M. Map merging of multi-robot slam using reinforcement learning. *In*: **2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. **Proceedings** [...]. [*S.l.: s.n.*], 2012. p. 53–60.

ELSEFY, A. E.; MOHAMED, K.; HARB, H. Exploration strategies of coordinated multi-robot system: A comparative study. p. 48–58, 03 2018.

FARINELLI, A.; IOCCHI, L.; NARDI, D. Multirobot systems: a classification focused on coordination. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 34, n. 5, p. 2015–2028, 2004.

GARRIDO-JURADO, S.; MUñOZ-SALINAS, R.; MADRID-CUEVAS, F.; MARíN-JIMéNEZ, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. **Pattern Recognition**, v. 47, n. 6, p. 2280–2292, 2014. ISSN 0031-3203. Available at: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

GAZEBO. **Gazebo - Robot simulation made easy**. 2021. Available at `http://gazebosim.org/`. Accessed: 17-05-2021.

GERKEY, B.; MATARIC, M. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. *In*: **2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)**. **Proceedings** [...]. [*S.l.: s.n.*], 2003. v. 3, p. 3862–3868 vol.3.

GOOGLE. **Google Maps**. 2021. Available at `https://www.google.com/maps`. Accessed: 07-07-2021.

GRISETTI, G.; KüMMERLE, R.; STACHNISS, C.; BURGARD, W. A tutorial on graph-based slam. **IEEE Intelligent Transportation Systems Magazine**, v. 2, n. 4, p. 31–43, 2010.

HESS, W.; KOHLER, D.; RAPP, H.; ANDOR, D. Real-time loop closure in 2d lidar slam. *In*: **2016 IEEE International Conference on Robotics and Automation (ICRA)**. **Proceedings** [...]. [*S.l.: s.n.*], 2016. p. 1271–1278.

HU, J.; NIU, H.; CARRASCO, J.; LENNOX, B.; ARVIN, F. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. **IEEE Transactions on Vehicular Technology**, v. 69, n. 12, p. 14413–14423, 2020.

INTEL. **Intel RealSense Depth Camera D435i**. 2021. Available at `https://www.intelrealsense.com/depth-camera-d435i/`. Accessed: 17-05-2021.

KULICH, M.; JUCHELKA, T.; PREUCIL, L. Comparison of exploration strategies for multi-robot search. **Acta Polytechnica**, v. 55, p. 162, 06 2015.

LEE, H.-C.; LEE, B. Improved feature map merging using virtual supporting lines for multi-robot systems. **Advanced Robotics**, v. 25, p. 1675 – 1696, 2011.

LEE, H.-C.; LEE, S.-H.; LEE, T.-S.; KIM, D.-J.; LEE, B.-H. A survey of map merging techniques for cooperative-slam. *In*: **2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)**. **Proceedings** [...]. [*S.l.: s.n.*], 2012. p. 285–287.

LYNCH, K. M.; PARK, F. C. **Modern Robotics: Mechanics, Planning, and Control**. 1st. ed. USA: Cambridge University Press, 2017. ISBN 1107156300.

MATHWORKS. **Create a 2-D pose graph - MATLAB**. 2021. Available at `https://www.mathworks.com/help/nav/ref/posegraph.html`. Accessed: 17-05-2021.

MATHWORKS. **Create occupancy map with probabilistic values - MATLAB**. 2021. Available at `https://www.mathworks.com/help/nav/ref/occupancymap.html`. Accessed: 17-05-2021.

MATHWORKS. **Exchange Data with ROS Publishers and Subscribers - MATLAB**. 2021. Available at `https://www.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html`. Accessed: 17-05-2021.

MATHWORKS. **MATLAB and Simulink**. 2021. Available at `https://www.mathworks.com/products/matlab.html`. Accessed: 17-05-2021.

MATHWORKS. **Pure Pursuit controller - MATLAB**. 2021. Available at `https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html`. Accessed: 17-05-2021.

MEDEIROS, I. de; BUONOCORE, L.; JR, C. N. LocalizaÇÃo de robÔ mÓvel baseada em filtro de kalman estendido usando um sonar. *In*: . **Proceedings** [...]. [*S.l.: s.n.*], 2010.

OPENCV. **Detection of ArUco Markers**. 2021. Available at `https://docs.opencv.org/3.4/d5/dae/tutorial_aruco_detection.html`. Accessed: 17-05-2021.

OPENCV. **OpenCV - Open Source Computer Vision**. 2021. Available at `https://docs.opencv.org/3.4.14/`. Accessed: 17-05-2021.

PATEL, S.; ARUL, S. H.; DHULIPALA, P.; LIN, M. C.; MANOCHA, D.; XU, H.; OTTE, M. W. Multi-agent coverage in urban environments. **ArXiv**, abs/2008.07436, 2020.

PINTO, L. d. S.; FILHO, L. E. S. A.; MARIGA, L.; JúNIOR, C. L. N.; CUNHA, W. C. Ekf-slam with autonomous exploration using a low cost robot. *In*: **2021 IEEE International Systems Conference (SysCon)**. **Proceedings** [...]. [*S.l.: s.n.*], 2021. p. 1–7.

REID, R. G. **Large-Scale Simultaneous Localization and Mapping for Teams of Mobile Robots**. Thesis (Doutorado) — The University of Western Australia, 2016.

ROBIN, C.; LACROIX, S. Multi-robot Target Detection and Tracking: Taxonomy and Survey. **Autonomous Robots**, Springer Verlag, v. 40, n. 4, p. pp.729–760, abr. 2016. Available at: https://hal.archives-ouvertes.fr/hal-01183372.

ROBIN, C.; LACROIX, S. Multi-robot Target Detection and Tracking: Taxonomy and Survey. **Autonomous Robots**, Springer Verlag, v. 40, n. 4, p. pp.729–760, abr. 2016. Available at: https://hal.archives-ouvertes.fr/hal-01183372.

ROS. **Robot Operating System**. 2021. Available at `https://www.ros.org/`. Accessed: 17-05-2021.

SAEEDI, S.; PAULL, L.; TRENTINI, M.; LI, H. Occupancy grid map merging for multiple robot simultaneous localization and mapping. **International Journal of Robotics and Automation**, v. 30, 01 2015.

SAEEDI, S.; TRENTINI, M.; SETO, M.; LI, H. Multiple-robot simultaneous localization and mapping: A review. **Journal of Field Robotics**, v. 33, n. 1, p. 3–46, 2016. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21620.

SHADE, R. **Choosing Where To Go: Mobile Robot Exploration**. Thesis (Doutorado) — University of Oxford, 2011.

SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. Springer International Publishing, 2016. Available at: https://doi.org/10.1007%2F978-3-319-32552-1.

SLAMTEC. **RPLIDAR A2 Laser Scanner**. 2021. Available at `https://www.slamtec.com/en/Lidar/A2`. Accessed: 17-05-2021.

STACHNISS, C. **Exploration and Mapping with Mobile Robots**. Thesis (Doutorado) — University of Freiburg, Department of Computer Science, April 2006.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. [*S.l.*]: The MIT Press, 2005. ISBN 0262201623.

WURM, K. M.; STACHNISS, C.; BURGARD, W. Coordinated multi-robot exploration using a segmentation of the environment. *In*: **2008 IEEE/RSJ International Conference on Intelligent Robots and Systems**. **Proceedings** [...]. [*S.l.: s.n.*], 2008. p. 1160–1165.

YU, S.; FU, C.; GOSTAR, A. K.; HU, M. A review on map-merging methods for typical map types in multiple-ground-robot slam solutions. **Sensors**, v. 20, n. 23, 2020. ISSN 1424-8220. Available at: https://www.mdpi.com/1424-8220/20/23/6988.

ZHOU, X. S.; ROUMELIOTIS, S. I. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. *In*: **2006 IEEE/RSJ International Conference on Intelligent Robots and Systems**. **Proceedings** [...]. [*S.l.: s.n.*], 2006. p. 1785–1792.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. REGISTRO N°	4. N° DE PÁGINAS
DM	29 de julho de 2021	DCTA/ITA/DM-056/2021	98

5. TÍTULO E SUBTÍTULO:

Multi-Robot Autonomous Exploration and Map Merging in Unknown Environments.

6. AUTOR(ES):

Luiz Eugênio Santos Araújo Filho

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica - ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

SLAM; autonomous exploration; multi-robot systems; map merging

9.PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Dinâmica de robôs; Navegação autônoma; Localização e mapeamento simultâneos; Robótica; Engenharia eletrônica.

10. APRESENTAÇÃO: **(X) Nacional ( ) Internacional**

ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Sistemas e Controle. Orientador: Cairo Lúcio Nascimento Júnior. Defesa em 09/07/2021. Publicada em 2021

11. RESUMO:

In mobile robotics, autonomy is obtained by navigating an environment while using embedded sensors to estimate a map and the robot position with respect to this map. This problem is known as Simultaneous Localization and Mapping (SLAM) and has been extensively addressed for the single-robot case in the last few decades. However, when the environment becomes larger, a multi-robot solution for the SLAM problem is more efficient despite the increased complexity and the development of new problems, such as multi-robot coordination. This work aims to develop and implement a solution for the SLAM problem using multiple ground mobile robots. To accomplish that, each robot must autonomously explore part of the environment and their individual estimated maps must be merged into a single global map. The proposed solution uses a scan matching-based SLAM framework known as GraphSLAM to estimate pose and map for each individual robot. To autonomously explore the environment, the solution proposes the use of map segmentation via Voronoi Graphs, which generate collision-free paths and possible target points for exploration. Map merging is performed when robots rendezvous to take inter-robot measurements to provide relative position information. In order to validate the proposed approach, experiments using simulated and real robots were carried out to autonomously explore and map indoor environments. Real world experiments presented satisfactory results for a laboratory environment and a unstructured corridor environment and the proposed rules for coordination as well as the map merging approach worked as intended.

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) SECRETO**