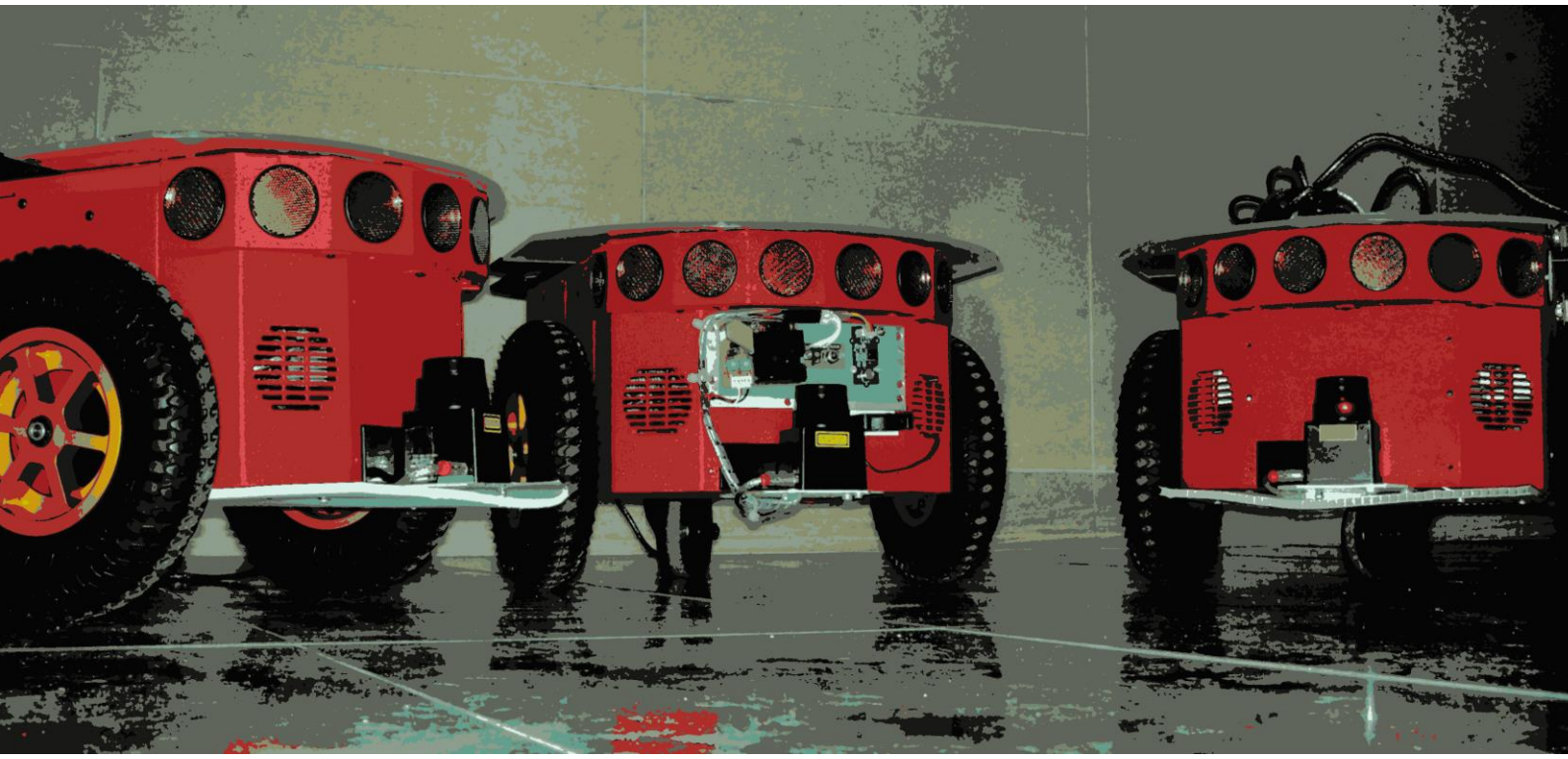


João Alexandre Simões Martins

MRSLAM – Multi-Robot Simultaneous Localization and Mapping

Setembro, 2013

· U C ·

Universidade de Coimbra

Department of Electrical and Computer Engineering
Faculty of Sciences and Technology
University of Coimbra

MRSLAM – Multi-Robot Simultaneous Localization and Mapping

João Alexandre Simões Martins

Supervisor:

Prof. Doutor Rui P. Rocha

Co-supervisor:

Eng. David Portugal

A Dissertation presented for the degree of Master of Science in Electrical and Computer Engineering

Jury:

President : Prof. Doutor Paulo José Monteiro Peixoto
Member : Prof. Doutor Paulo Jorge Carvalho Menezes
Member : Prof. Doutor Rui Paulo Pinto da Rocha

Acknowledgments

Several people have been instrumental for the construction, development and conclusion of this project. I would like to thank them for their important and gracious support. These people include not only the academics and colleagues, but also friends and family who have assisted me in different ways during this last year.

First of all, I would like to thank my supervisor, Prof. Rui Rocha, for his guidance, with whom, I have discussed this thesis at the crucial moments. I would also like to remark the great motivation he has given to me when things did not go as well as expected as well as forthrightness when something was clearly misguided. I would like to thank my co-supervisor, Eng. David Portugal for all that I have learned with him during this year, for helping me solving problems during my dissertation and especially for always believing in my capabilities, thanks David.

Special thanks to my closest family: My father Pedro Martins and my sister Catarina Martins for all the support and motivation and my grandmother Nina for all the spiritual faith that she has placed on me. These are the people, whom by their merely existence, allow me to go on, because of their unconditional love, care and encouragement throughout my life. I am very grateful to my best friend, João Cabrita and my best Lab partner João Santos for the invaluable support, for helping me all the times i was in trouble and for the many nights we stayed up working together. More that a friend, one brother for life. Thank you both. My appreciation goes, as well, to my undergraduate and closest friends from Coimbra for their support, friendship and listening skills, even when they had no idea what I was speaking about. I further would like to mention my laboratory mates Luis Santos, Ricardo Martins, Diego Faria, Micael Couceiro and José Santos for the pleasant working environment that subsisted during last year and also for their comments and suggestions. Many thanks to my colleagues Sergio Santos, Renato Batista and João Faro for fruitful discussions and appreciated company for snack breaks.

Abstract

Nowadays, a collection of two or more autonomous mobile agents working together are denoted as teams or simply societies of mobile robots. In Multi-Robot Systems (MRS) robots are allowed to coordinate with each other in order to achieve a specific goal. In these systems, robots are far less capable as an entity, but the real power lies in the cooperation of the team. The simplicity of MRS has produced a wide set of applications such as in military tasks , searching for survivors in disaster hit areas, parallel and simultaneous transportations of vehicles and delivery of payloads.

The success of single-robot Simultaneous Localization and Mapping (SLAM) in the past two decades has led to research on Multi-Robot Simultaneous Localization and Mapping (MRSLAM). A team of robots is able to map an unknown environment faster and more and reliably. However, MRSLAM raises several challenging problems, including map fusion, unknown robot poses and scalability issues. Rao-Blackwellized Particle Filters (RBPFs) have been demonstrated as an effective solution to the problem of single robot Simultaneous Localization and Mapping (SLAM), and a few extensions to teams of robots exist. However, these approaches are usually characterized by strict assumptions on both communication bandwidth and prior knowledge on relative poses between teammates.

In this dissertation, we describe in detail a distributed MRSLAM approach using RBPF in the case of possibly constrained communication and unknown relative initial poses using Robot Operating System (ROS). We consider the environment as a two dimensional space with several obstacles, which are explored by a team of cooperative mobile robots, equipped with laser sensors. In order to efficiently tackle the problem, the cooperation between agents and the memory space available for observations storage must be taken into account. Experimental results using a team of up to two robots in a large indoor area show the robustness and performance of the approach.

Key Words: Simultaneous Localization and Mapping, Multi-Robot Systems, Mobile Robotics, ROS.

Resumo

Hoje em dia, um conjunto de dois ou mais agentes móveis autônomos que trabalham em cooperação são designados de equipas de robôs ou simplesmente de sociedades de robôs móveis. Em Sistemas Multi-robô (MRS), os robôs devem coordenar uns com os outros de modo a alcançar um objetivo específico. Nestes sistemas, os robôs são individualmente menos capazes, sendo que o seu verdadeiro poder reside na cooperação em equipa. A simplicidade de MRS gerou um potencial de aplicação em várias áreas, tais como, tarefas militares, missões de busca e salvamento em situações de desastres naturais, transportes paralelos e simultâneos de veículos, assim como em distribuição de cargas.

O sucesso da tarefa de Localização e Mapeamento Simultâneos (SLAM) com um único robô nas últimas duas décadas tem levado a imensas pesquisas sobre Localização e Mapeamento Simultâneos com Múltiplos Robôs (MRSLAM). A equipa de robôs pode explorar um ambiente de forma mais eficiente e robusta. No entanto, MRSLAM levanta também muitos problemas desafiadores, incluindo fusão de mapas, robôs com pose desconhecida e problemas de escalabilidade. Filtro de partículas Rao-Blackwell(RBPF) têm demonstrado ser uma solução eficaz para o problema de SLAM com um único robô existindo algumas extensões para equipas de robôs. No entanto, estas abordagens são normalmente caracterizadas por severas restrições na largura de banda de comunicação e conhecimento prévio da pose relativa entre agentes robóticos.

Nesta dissertação, descrevemos em detalhes uma abordagem distribuída para MRSLAM utilizando RBPF considerando situações de comunicação com largura de banda limitada e pose inicial relativa desconhecida usando ROS. Consideramos o meio ambiente como um espaço bidimensional com vários obstáculos, que é explorado cooperativamente por uma equipa de robôs móveis equipados com sensores laser. De modo a enfrentar o problema de forma eficaz, a cooperação entre os agentes e o espaço de memória disponíveis para armazenamento de observações devem ser levados em conta. Os resultados experimentais, utilizando uma equipa até dois robôs num espaço vasto em ambiente indoor demonstram a robustez assim como o desempenho da abordagem apresentada.

Palavras-Chave: SLAM Multi-robô, Sistemas Multi-robô, Robótica Móvel, ROS.

Contents

List of Acronyms

C-SAM Colaborative Smoothing and Mapping

CHOPIN Cooperation between Human and rObotic teams in catastroPhic INcidents

EKF Extended Kalman Filter

GNSS Global Navigation Satellite System

IP Internet Protocol

ISR Institute of Systems and Robotics

KF Kalman Filter

LIDAR Light Detection And Ranging

LRF Laser Range Finder

MANET Mobile Ad Hoc Network

MRL Mobile Robotics Laboratory

MRS Multi-Robot Systems

MRSLAM Multi-Robot Simultaneous Localization and Mapping

OLSR Optimized Link State Routing Protocol

OLSRD OLSR deamon

RBPF Rao-Blackwellized Particle Filter

ROS Robot Operating System

SaR Search and Rescue

SEIF Sparse Extended Information Filter

SLAM Simultaneous Localization and Mapping

SURF Speeded-Up Robust Features

USAR Urban Search and Rescue

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation describes work that has been done in the context of the R & D research project Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN)[1], which is ongoing in the Mobile Robotics Laboratory (MRL) of the Institute of Systems and Robotics (ISR) at the University of Coimbra, in Portugal.

Nowadays, there has been an increase in the need to respond effectively to catastrophic and unexpected incidents, including natural and civil disasters, industrial accidents and terrorism acts and crime. The CHOPIN project aims at studying the cooperation between human teams and robotic teams, including collaborative context awareness and efficient information sharing and the project exploits the human-robot symbiosis in the development of human rescuers' support systems for small-scale Search and Rescue (SaR) missions in urban catastrophic incidents as shown in Figure 1.1. To that end, cooperative robots can be very useful on assisting humans in many distributed activities, especially hazardous scenarios, by extending human perception and actuation with the capabilities they possess [Couceiro et al., 2013].

Figure 1.1: Currently, mobile robots are increasingly seen as a real asset to situations of natural disasters and accidents.

The main goal of this master degree project is to implement a distributed SLAM algo-

[1]Project's page: `http://chopin.isr.uc.pt`

rithm for cooperative teams of robots [Howard, 2006] in order to share efficiently information within a robotic system comprised of several mobile robots, towards the stimulation of a co-operative behavior of the multi-robot system. For this purpose, Pioneer 3-DX, equipped with Hokuyo-URG-04lx-UG01 lasers were used.

This chapter starts by relating the research reported in this dissertation with the context of mobile robotics and, more specifically, in the context of robotic systems comprised of more than one robot, *i.e.*, Multi-Robot Systems (MRS). There were four main work phases. In the beginning, mobile robotic theory such as SLAM techniques and its relationship to SLAM strategies for multi-agents systems was studied in detail. The second phase consisted of the conception of the distinct modules necessary for a MRSLAM approach. Afterwards, the new MRSLAM strategy was developed, refined and tested to validate the results through simulation and also real world experiment.

In this chapter, an overview of the document is given and the context of the research is specified as well as outlines for the usage of multi-agent systems.

1.1 Context and Motivation

It has been verified in the past that multiple robots can cooperate to perform complex tasks that would otherwise be impossible for one powerful robot to accomplish [Rocha, 2006]. The fundamental idea behind multi-agent robotics suggests dispatching the problem into smaller sub-problems for each individual robot, and allowing them to interact with each other to find solutions for these problems. Simple robots can be built and made to cooperate within a team to achieve complex behaviors. It has been observed that MRS may be very cost effective when compared to building a single expensive robot with all incorporated capabilities.

For a robot to navigate within its environment, map building is a crucial task that has to be done. Concurrently building up the map of the environment and using the map to obtain the estimation of the localization of the robot is a fundamental problem in mobile robotics. This problem is usually called Simultaneous Localization and Mapping (SLAM) and it deals with estimating the position of the robot with respect to the map, while building it using the sensory input and the estimated robot's pose. The maturity of SLAM techniques with a single robot has been acknowledged recently. However, the extension of such techniques to multiple robots, in order to perform cooperative SLAM tasks in unknown and/or dynamic environments is still a great challenge.

The problem is usually denoted as Multi-Robot Simultaneous Localization and Mapping (MRSLAM). In MRS the accuracy by which robots can model their environment has a deep impact on individual and team performance. A team of robots is expected to build a consistent representation of the environment in a much quicker way than a single robot. Even though MRS have the ability to improve efficiency, precision and robustness in such missions, there are several sources of complexity in MRSLAM, which requires additional

effort, for example, to estimate the positions of different robots and to merge partial maps of each agent, which is something not necessary in the single robot case. In MRSLAM, the dynamics and unpredictability of the environment as well as noisy sensor readings must be addressed. In addition, as opposed to single robot SLAM, new challenges and difficulties are involved, such as: coordination of robots, integration of information collected by different robots into a consistent map and dealing with limited communication. The problem has been recently studied, as shown in chapter 2.

1.2 Multi-Robot Systems

There are several advantages of using MRS in exploration tasks as well as in other applications. In some cases, due to the need of combining multiple tasks and the dynamics of the environment it is only viable to achieve the mission with a multiple distributed autonomous robotic system. A fully-equipped autonomous mobile robot with sensors of different types may adequately fulfill the assignment, but it may also prove to be costly and have diminutive fault tolerance. MRS are characterized by distributed control, autonomy, greater fault tolerance, and communication. For example, without the help from other robots, a single robot may be vulnerable to an hostile environment or enemies, such as in some military actions or exploring an unstable building. In several other applications, an agent may get assistance from other nearby agents during emergencies, such as failures or malfunctions [Portugal, 2009].

Multi-Robot Systems benefit from having many robots in many places and carrying out diverse tasks at the same time, *i.e.* space distribution. When the problem denotes more complexity, sometimes it is useful to divide it in simpler subtasks and assign them to different robots of the team. The decomposition of complex problems linked with effective cooperation is a major advantage of these systems. This feature can be used, for example, in exploration of unknown environments. To increase reliability and robustness, in comparison to a single autonomous mobile robot incorporated with all kind of sensors and abilities, a team of multiple robots may be heterogeneous by having spread resources. For that reason, each unit becomes simpler and, as a consequence, its cost may be reduced. Furthermore, these systems enable redundancy and graceful degradation, remaining functional if some of the agents fail. Another main motivation for adopting robots is the possibility of reducing the risk to human operators in the face of dangerous exploration missions. MRS can ease arduous, tiring and time-consuming tasks like surveillance, infrastructure security or monitoring. In addition, one or more robots could replace humans in dangerous situations like SaR operations, which require strong effort by rescuers in a very dangerous scenario that poses many threats to human rescue teams. Replacing people with autonomous robots in these environments provides inestimable benefits [Couceiro et al., 2013]. Also, as reported before, these systems may offer the possibility to relieve and/or assist human beings from monotonous or repetitive tasks enabling them to be occupied in nobler tasks.

1.3 The approach followed in this dissertation

In this work, distributed coordination between agents is assumed, where independent agents divide the task of building a global map, providing higher tolerance to error and failures, robustness, flexibility and scalability. This kind of coordination promotes task parallelism, which contributes to larger success warranties. Centralized coordination architectures may lead to an expensive solution in terms of time and resources due to the large communication flow between local agents and the central agent. Highly complex tasks may become impractical due to the size of the search space and may require continuous centralized information about the robots' positioning.

Effective coordination can be achieved by guiding the robots into different, non-overlapping areas of the environment. However, this is only possible if the robots know their relative locations and share a common map or frame of reference. The MRSLAM algorithm's output is the iterative construction of a map of the environment through data obtained by range sensors. In this dissertation, we focus on metric maps represented by 2D occupancy grids. In order to obtain the global map, a consistent model of the environment with data collected from different robots must be built through map aligning and merging techniques. If the locations of the robots are known, map merging is a rather straightforward extension of single robot mapping. However, if the robots do not know their relative locations, it becomes much more difficult, since it is not clear how and where the robots' partial maps should be aligned. This dissertation is inspired by some aspects of the work present by [Howard, 2006].

Robots in the team may have to share large amounts of data, depending on the dimensions of the environment. Despite its suitability in scenarios where a network infrastructure is present, the strategy proposed in this work aims to also offer a solution for Mobile Ad Hoc Networks (MANETs). MANETs are wireless communication networks that do not rely on fixed, pre-installed communication devices like base stations or predefined communication infra-structures. It consists of mobile nodes which are characterized by their decentralized organization and may use multi-hop communication to deliver messages to distant peers. Therefore, MANETs are suitable for applications with MRS, such as Urban Search and Rescue (USAR) scenarios, in the aftermath of natural or man-made disasters [Couceiro et al., 2013] where a network infra-structure does not exist. Figure 1.2 illustrates how the work in this dissertation fits in the CHOPIN project. In the figure we can see several distributed teams of robots exploring different areas, communicating with each other and reporting back all the information collected to the command center who is only responsible for monitoring the mission's progress.

1.4 Outline of the dissertation

The dissertation is organized in five chapters, each referring to distinct phases of the work. After this introductory chapter, which has introduced the context and motivation, important

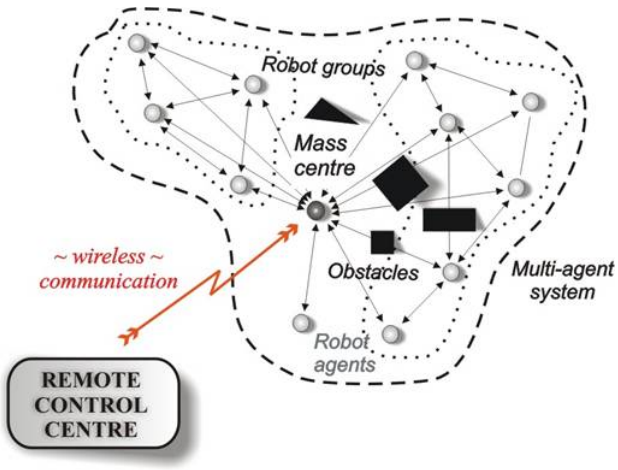


Figure 1.2: Illustration of an example of a task with multiple distributed robots, similar to the CHOPIN's project architecture.

advantages of multi-robot systems and important aspects related to the work in this project, chapter 2 reviews thoroughly the most relevant state of the art related with cooperative multi-robot systems, robotic mapping and associated problems. The key objective of this study is to point out advantages and disadvantages of existing MRSLAM approaches to be able to assist the choice of an approach for the developed algorithm.

Chapter 3 addresses the implemented MRSLAM System and several important steps that were necessary for this system to become robot-independent, distributed, functional and more importantly, the first MRSLAM solution out-of-the-box for ROS. In this chapter, we discuss with more details the basic modules of the MRSLAM System presented.

In chapter 4, the results from simulation and real experiment are revealed. We also discuss the advantages and the disadvantages of the MRSLAM approach and potential applications.

To finish, the final chapter sums up the work, provides final conclusions and prospects interesting future directions for this research. The bibliography is presented in a separate references section.

Chapter 2

Multi-Robot SLAM Overview

In this chapter, a study of important introductory issues related with MRSLAM is presented. Some preliminary concepts are established to set up the framework to the problems being discussed at a later stage. The problem definition and related work on SLAM and MRSLAM is herein addressed. In [Martins, 2013] a more extended state of the art is presented. This report is available on the project's CD release.

2.1 Classical SLAM approaches

Navigation is one of the most demanding and important skills for a mobile robot. The navigation's success depends on the results of four stages: perception, localization, cognition and motion control as stated in Figure 2.1. Moreover, these four stages consist on processing useful information from sensors, determine its position on the environment, deciding how to act to achieve its results and managing the actuator's outputs in order to achieve the intended trajectory.

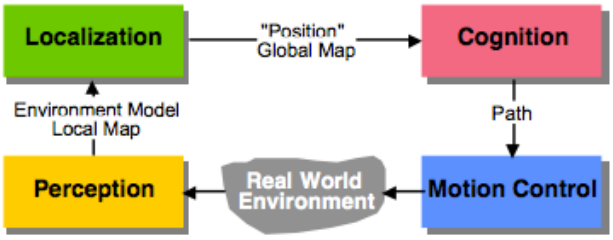


Figure 2.1: Representation of navigation process of a mobile robot.

During motion, the robot estimates its position based on the sensor's measurements, which have noise, and on movement models that may have simplifications, *e.g.*, not modeling the non-linear phenomenon referring to wheels sliding. The ability and way of interacting with the world depends on its perception. Sensors provide the robot with the ability to sense the world, therefore their use is very import when designing an autonomous system. Sensors used in Mobile Robotics can be divided in exteroceptive and proprioceptive sensors. Each kind of sensors will be described in detail next.

Exteroceptive sensors are those which acquire information through energy from the environment. Two distinct types of energy are used: electromagnetic and acoustic. The

exteroceptive sensors are grouped as active or passive, depending if they do or do not emit energy on the environment where its sensing is applied. The active sensors emit energy on the environment obtaining distance and relative speed measurements from the reaction to the energy. The most advanced techniques on distance measurement are based on flight time, phase difference, frequency modulation or triangulation. In relative speed measurements the Doppler effect is used. The main energy sensors used in Mobile Robotics are the Light Detection And Rangings (LIDARs), sonar, radars and the Global Navigation Satellite Systems (GNSSs).

The Proprioceptive sensors are sensors which measure the system's internal parameters. This kind of sensors have the advantage of not depending on the environment to make its measurements, which often makes them robust. Typical sensors of this kind used in Mobile Robotics are odometers and inertial sensors.

SLAM is defined as the problem of building a model leading to a new map, or repetitively improving an existing map, while at the same time localizing the robot within that map. There are various techniques to compensate for errors, such as feature recognition, data association or loop closure detection. Filtering is a very used method in engineering and embedded systems. A good filtering algorithm can reduce the noise from signals while retaining the useful information. Some of the statistical techniques used in SLAM include Kalman filters, particle filters and scan matching of range data.

The Kalman Filter (KF) is a recursive linear estimator which computes a minimal variance estimate for a state that evolves through time based on observations linearly related to that state. It obtains satisfactory results assuming certain linear assumptions regarding to the process noise and observations noise. The KF has many applications in aerospace navigation problems, control and general Robotics. It is acknowledged as one of the best estimators for linear systems with Gaussian noise. For non-linear systems the Kalman filter is not rigorously applicable, as the existence of linearity is important to set the Kalman filter as an optimal filter. The Extended Kalman Filter (EKF) is intended to overcome that difficulty using a linearization on the state estimate. The EKF is used to estimate the robot position through data from odometry and the observation of spacial references. This filter is normally described based on determining the robot's position by itself, which assumes the existence of the map *a priori*. When the map does not exist, as it happens in a SLAM situation, the EKF matrixes are changed and a new solution comes with the EKF-SLAM [Durrant-Whyte and Bailey, 2006].

The EKF-SLAM solution brings several benefits for the localization and navigation problem but it also has its limitations, such as computational complexity, data association and non-linearities. Under ideal EKF-SLAM conditions, the robot's location estimative covariance and the landmarks individual positions would converge to zero. However, the correction stage computational complexity grows quadratically with the number of landmarks, which is usually a problem in practical applications. Furthermore, the EKF-SLAM solution is extremely sensitive to the incorrect references and observations association. Besides, the

data association problem is very difficult when references are re-observed by very distinct observation points. Finally, the linearization of the non-linear model's movement and/or the models observation can get results with inconsistent solutions. In [Smith et al., 1990] the EKF is presented as a solution for SLAM situations, becoming over the years a classic approach to solve that problem. Many projects were developed based on this approach, like [Guivant and Nebot, 2001] and [Dissanayake et al., 2001].

Due to the EKF limitations on quadratic complexity and the sensitivity to data association flaws, the FastSLAM algorithm appeared as an alternative. This algorithm was presented in [Montemerlo and Thrun, 2003] and [Montemerlo et al., 2003] being the first to consider non-linear models and multi-modal distributions to solve SLAM situations, with the advantage of having better robustness in data association.

The FastSLAM algorithm is based on a very important SLAM problem characteristic, which is the conditional independence that exists between two different landmark sets on the map, given the robot's pose. In other words, if the robot's real trajectory is known, the estimative of all the landmarks positions in the map can be done independently between each other. This allows the implementation of a particle filter version in the SLAM situation, the Rao-Blackwellized Particle Filter (RBPF). RBPFs have been introduced as effective means to solve the SLAM's problem [León et al., 2008], [Howard, 2006], [Carlone et al., 2010], [Fenwick et al., 2002], [Zhang et al., 2009], where each particle carries an individual map of the environment. The main problem of Rao-Blackwellized approaches is their computational complexity, measured in terms of the number of particles required to build an accurate map. Table 2.1 states the advantages and disadvantages of the presented filters.

Table 2.1: Advantages and Disadvantages of the EKF-SLAM and RBPF-SLAM.

Filter Type	**Advantages**	**Disadvantages**
EKF–SLAM	• Linear update of the observations using fractioned updates; • Through sub mapping it is possible to obtain a solution in constant time;	• It assumes a Gaussian error which does not exist always; • The non-linear models linearization can cause divergences;
RBPF–SLAM	• Simultaneously keeps several hypotheses; • With a large number of samples it can efficiently represent non-linear or non-Gaussian models; • It allows data association in parallel; • It estimates the robot's entire trajectory online;	• The number of particles exponentially increases with the state's size;

2.2 Extending SLAM for multi-robot teams

Both the EKF and the RBPF were profiled for MRSLAM situations. However, other algorithms are also used in the literature. The number of projects which explore the coordination in a team of robots to complete this kind of tasks has been growing at a fast rate. This is explained by the following reasons [Rocha, 2006]:

- The total cost of a team of simple robots may be smaller than a single more complex robot;
- The ability to trade information between robots allows a decrease of uncertainty during the estimation process;
- Time distribution - several robots can do tasks, different or not, at the same time;
- Space distribution - several robots can be in different places at the same time;
- Problem decomposition - literal use of the expression "Divide and Conquer". Certain problems are easily solved if divided into smaller problems and sorted between several robots;
- Reliability and robustness towards flaws;

Consequently, in order to generalize SLAM for multiple cooperative robots, it is necessary to answer the following set of questions:

- **How many maps are used in the process?**
- **What is the SLAM algorithm used as the base for multi-robot generalization?**
- **Is it assumed that the robots know their initial positions?**
- **Which methodology is used to align the maps?**

Space and time distribution contribute to smaller sub-tasks and mission completion time while using several robots. Those distributions are related to operations occurring simultaneously in space and time respectively. These can be requested by a task such as exploring a vast area or detecting mobile targets, improving the performance on tasks intrinsically distributed such as clearing a certain area or on large scale missions in the least possible time. Although some tasks do not require a multi-robot solution, implementing a single simultaneously complex and robust robot is not advised due to the relationship between performance and reliability, as shown in Figure 2.2.

Multi-robot solutions offer better flexibility during complexity and risk distribution management. If there is an overlap on each robot's individual capabilities, the system will be

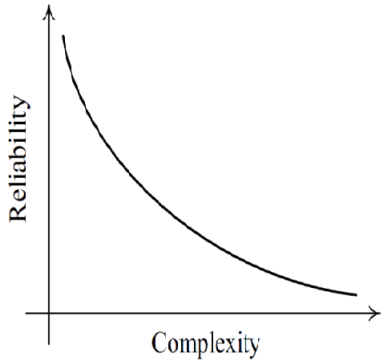


Figure 2.2: Ratio between Reliability and Complexity in robotic systems [Rocha, 2006].

more robust and in the case of a flaw, it will not necessarily mean the entire system's failure [Rocha, 2006]. Multi-robot systems are, for those reasons, adequate to a vast set of applications, such as SaR tasks and environment mapping.

The main difficulty in this challenge consists in finding an efficient strategy in order to combine the information gathered by the sensors from all the robots. The Multi-Robot Simultaneous Localization and Mapping can be implemented in two different ways: considering a single centralized global map updated by every operational robot, or each robot can build a partial map of the environment which is combined with all the team's robots partial maps [Andersson and Nygårds, 2009] in a distributed way.

Regarding the map combination process on the second option, it can be split in two phases:

- The first stage is called *Alignment*, on which the coordinates transformation is determined in order to combine a robot's pose and its respective landmarks on another robot's coordinates referential.

- On the second stage, the common landmarks estimations need to be integrated in order to generate a global map. This is called the *map merging* stage.

The answer to each of these questions and statements allows ranking different MRSLAM approaches. In a centralized approach, robots send their partial maps continuously to a central server, which is responsible for the alignment and merging of those maps, then sending the result back to each robot. This requires continuous monitoring on the robot's positions and heavy communication to/from the central server. On the other hand, robots may exchange their local maps when they meet in the environment (*rendezvous*) and align them internally, this coordination approach is called distributed or decentralized. The global map's representation built by each robot can vary, depending on the partners which they have met and exchanged maps. In this type of coordination, mutual detection methods are mandatory to relate the robots positions and assist local map alignment and merging. Properties on both of these cases in MRSLAM missions are presented next.

Table 2.2: Advantages and disadvantages of Centralized Systems and Distributed Systems.

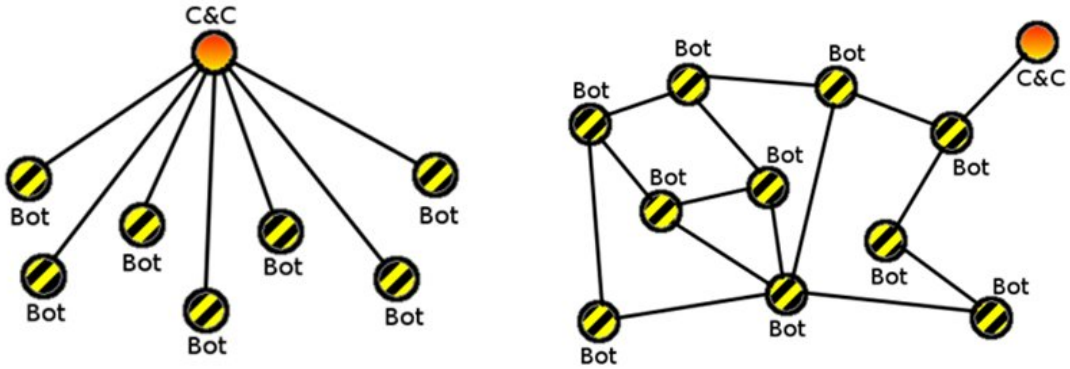
System Type	Advantages	Disadvantages
Centralized System	• intrinsically coordinated; • It allows satisfactory, coherent and comprehensive solutions; • Global information may lead to optimal results;	• Expensive solutions in terms of time and resources due to the large communication flow between local agents and the central agent; • Highly complex tasks may become impractical due to the size of the search's space; • For a large number of agents it becomes impractical to store all the environment's information in a single server; • Requires information about the robots initial positioning; • Unreliable;
Decentralized System	• Independent agents divide the task of building a global map; • Higher tolerance to errors and failures; • Robustness, flexibility and scalability; • Task parallelism; • Larger success warranties;	• High uncertainty degree, which complicates obtaining a system's coherent global behavior; • Inefficient system's coordination results in complex dynamics and creates non-linear variations and chaos situations; • Mutual detection problem;

Centralized architectures are characterized by having a single controller which is individually responsible to make decisions. It is assumed that the main process has a global model of the "World" which allows it to theoretically produce optimal solutions for multi-agent problems. Two different centralized systems classes can be considered: fully centralized or partially centralized [Rocha, 2006]. A centralized system is intrinsically coordinated and can lead to satisfactory, coherent and comprehensive solutions but it has several limitations. Depending on team size, it is very hard or even impractical to keep a global model of the "World" in a single agent, based on local and potentially inconsistent views between local agents. Besides, it is usually a costly solution in terms of resources and time, which makes use of high load and persistent communication between the local agents and the central agent, which can create severe communication jams. It also has an architecture which can be unreliable because all the information is placed in the central agent which is not allowed to fail as seen in Figure 2.3a.

Decentralized approaches are made of logic and physically independent agents' network. Each agent is capable of "thinking" about plans, choosing their own actions and perceive the

system's dynamics through interactions with other agents. Decentralized approaches can be classified by each agent's autonomy level as being Hierarchical or Distributed [Rocha, 2006]. The decentralized architectures possess several advantages when compared to centralized architectures, such as failure tolerance, reliability warranty, robustness, parallelism and task decomposition, flexibility and scalability. Distributed control based systems and distributed data have superior performances in terms of resources, communication and robustness when compared to centralized systems since there is no central controller. These can be very flexible since the role of each agent can change based on the context, as seen in Figure 2.3b.

Table 2.2 presents the advantages and disadvantages of both architectures.



(a) Centralized system.

(b) Decentralized system.

Figure 2.3: Different types of architectures for Multi-Robot Systems.

2.3 Related Work on MRSLAM

In [Thrun and Liu, 2003], the MRSLAM situation is solved using Sparse Extended Information Filter (SEIF) on which the maps and robots' poses are represented using Gaussian Markov Random Fields. This decentralized approach focuses on updating a subgroup of all the landmarks in order to gain computing efficiency. SEIF creates sub-maps dynamically, for this reason, the results produced are fairly precise. The algorithm avoids frequent problems in sub-maps border areas where the estimation can become unstable. The algorithm was successfully tested with eight robots using data collected in Victoria Park, Sydney.

According to [Burgard et al., 2005], target points are chosen for each robot in order to explore different areas of the map at the same time. In this centralized approach the reach cost and usefulness of target point is computed. Robots' limited communication is considered and an efficient coordination technique is described. A significant reduction on task execution time was revealed, when compared to other exploring approaches that do not explicitly coordinate the robots. Experimental tests show that the algorithm's performance is scalable with limited communication. An interesting discussion of the challenges involved in the works is conducted, by considering a situation where the robots initial position is unknown.

Another important work in this area is [Howard, 2006], in which the particle filter presented by [Hähnel et al., 2003] is profiled for MRSLAM. This work follows a new direction on particle based methods, assuming that the robots initial positions are unknown. In this decentralized approach, each robot's previously stored observation sequences are combined in a single environment map when *rendezvous* occurs. Finally, a method is presented with the purpose of integrating in the map, data recovered by the robots before the first encounter. During the exploring and mapping task execution, it is assumed that robots exchange data through a trusted wireless connection and are capable of detecting teammates based on individual particularities that allow the distinction between an object and a moving robot. This algorithm was tested in a closed environment using four homogeneous robots. It is important to emphasize that the work described above shares similarities with the work in this dissertation.

[Carlone et al., 2010] produced a new decentralized method, based on RBPF (FastSLAM), similar to [Howard, 2006]. It considers limited communication taking into account the distance between robots and the original positions of the robots are unknown. The method uses cameras for robots' mutual detection and it only allows the partial maps alignment when *rendezvous* happens. The presented solution showed efficiency and robustness, successfully building a map of a real world environment.

According to [Andersson and Nygårds, 2009], the problem of aligning and combining maps created by several robots using observations made between them is addressed. The solution uses Colaborative Smoothing and Mapping (C-SAM) to combine maps created by different robots in a distributed way. The main contribution of this work is the algorithm used to solve the data association problem and eliminate fake observations when the map alignment is made during *rendezvous*. This work shares similarities with [Zhou and Roumeliotis, 2006], however the EKF is used to solve the localization problem.

In [Leung et al., 2012], a cooperative decentralized SLAM system is examined, on which robots need to estimate the maps and the states of all the other robots assuming that the communication between them is limited and the connection is dynamic. It is mathematically proven that an estimation equivalent to a centralized system can be obtained by all robots in the network in a decentralized way. Besides, the robot only needs to consider its own information from the topological network in order to detect when the equivalent centralized system is obtained. This works presents more than 250 minutes of experimental tasks using a team of real robots. The estimations are compared with their real position for all the robots poses and landmarks. Finally, the communication restriction effects on the algorithm's performance are examined and it is showed that the memory used is limited due to the use of Markov model's properties. The results also showed that the estimated centralized equivalent system can always be recovered after a period of low connectivity on the network. In terms of precision, the general results show that the pose and landmarks estimation are similar to an estimated centralized system even with a small full connectivity time percentage.

The work by [Fox et al., 2006] present a distributed MRS for exploration and mapping

Table 2.3: Features of each presented algorithm.

Algorithm	Features
Bayesian approach [Thrun and Liu, 2003]	• Uses a SEIF filter, where the agents' maps and poses are represented through Gaussian Markov random fields; • It allows an increase in computational efficiency; • The creation of dynamic sub-maps contributes to more precise results; • It prevents sub-maps border areas estimation problems; • Tested with success in a real environment using 8 robots;
Coordination with communication restrictions [Burgard et al., 2005]	• Uses target points to explore different areas in order to maximize resources and task execution time; • It considers the range cost associated to each target point as well as its utility to the task conclusion; • Limited communication between agents; • The coordination technique revealed a significant decrease in task execution time; • Scalable considering the limited communication;
RBPF approach [Howard, 2006] and [Carlone et al., 2010]	• Uses a generalized particle filter for MRSLAM; • Solution for known and unknown initial positions situations; • Merge of robot' observations in a single map when rendezvous happens; • Communication via Wireless network; • Mutual detection capable robots; • Representation of the environment using occupation grids; • Tested with success in real environments using 4 homogeneous robots; • Latency during the information switch between agents, however the final results show efficiency and robustness;
Markov model filter approach [Leung et al., 2012]	• Uses a filter with an incorporated Markov model; • Robots with unknown initial position; • Creation of shared map after robots rendezvous; • Approach with better results regarding maps fusion; • Communication via Wireless network;
C-SAM approach [Andersson and Nygårds, 2009]	• C-SAM algorithm; • Eliminates fake observations and solves data association problem during the maps alignment; • Solves known and unknown initial position situations; • Smooth approach for fusion of maps created by different robots;
Distributed approach with shared maps [Fox et al., 2006]	• Distributed approach; • Solves known and unknown initial position situations; • Environment representation using shared metric maps; • Uses an exploration strategy in order to maximize exploration efficiency; • Efficient and robust system; • Algorithm tested in Saphira simulation environment using three robots;
Topological maps approach [Chang et al., 2007]	• Environment representation using topological maps; • Project tested in simulation and real environment using three Pioneer 3-DX robots; • Main limitation is the need to optimize the topological maps information offline;

missions. The system allows teams of robots to efficiently explore the environment starting from different and unknown positions. In order to assure consistency when the data merge process for the shared maps occurs, the robots actively check their relative positions. Using shared maps, the robots coordinate their exploration strategy in order to maximize the

efficiency of the operation. The system was tested in the Saphira[1] simulator, a robot control system for application programmers, being quite efficient and robust.

In [Chang et al., 2007], a decentralized MRSLAM algorithm which uses topological maps is presented. Vertices contain local metric information and edges describe the relative position of the adjacent local maps. In this work, the maps are naturally merged between robots through the inclusion of an edge which establishes a connection between topological maps, and the estimative of the relative poses from the robots is done by the optimization of that edge. The authors show that by this process it is possible to unbind the SLAM situation in shorter mapping and localization problem. The experimental tests were performed with Pioneer 3-DX robots which validated the algorithm's performance. The main limitation was the need to optimize the information from the topological maps offline, being that a critical point in large environments. Table 2.3 states the main features of each presented algorithm.

2.4 Map Representation

Building cooperatively maps of unknown environments is one of the application fields of multi-robot systems. In this context, it is indeed very important for autonomous mobile robots to learn and maintain models of the environment. The main problem inherent to the map's representation model is to deal with the high dimensionality of the entities being mapped [Thrun, 2002]. While for example, a very detailed 2D geometric map may require a huge amount of memory, a description based on topological entities, such as corridors, intersections, rooms and doors, may not require much more memory to model the same environment, but obviously leads to a map with less detail. According to [Thrun, 1997], the two major distinct paradigms produced for mapping indoor environments are grid-based maps and topological maps.

Grid-based methods produce accurate metric maps, they are based in detailed representations, similarly to a blueprint of the environment. These maps require a precise location for the robot which due to the lack of efficiency on odometric systems makes it a complex task. They are easy to build but may be hard to maintain, depending on the resolution and dimension of the environment. Additionally, the size of the environment, the cells configuration and the memory capability makes the trajectory planning on these kinds of maps a complex task. The most common example of a metric map is the occupation grids, a map representation wherein each cell of the grid contains a probability value which indicates whether the related location is free space or part of an obstacle [Elfes, 1990]. It is relatively simple to implement and has an iterative nature. An example of a occupation grids map is shown in Figure 2.4.

Topological maps, on the other hand, produce graph-like maps that can be used much more efficiently. They are simpler, permit efficient planning and do not require accurate determination of the robot's position. In these maps, vertices correspond to important places

[1]Saphira Technical Manual. Available at: `http://www.cs.jhu.edu/~hager/Public/ICRAtutorial/Konolige-Salphira/saphira.pdf`

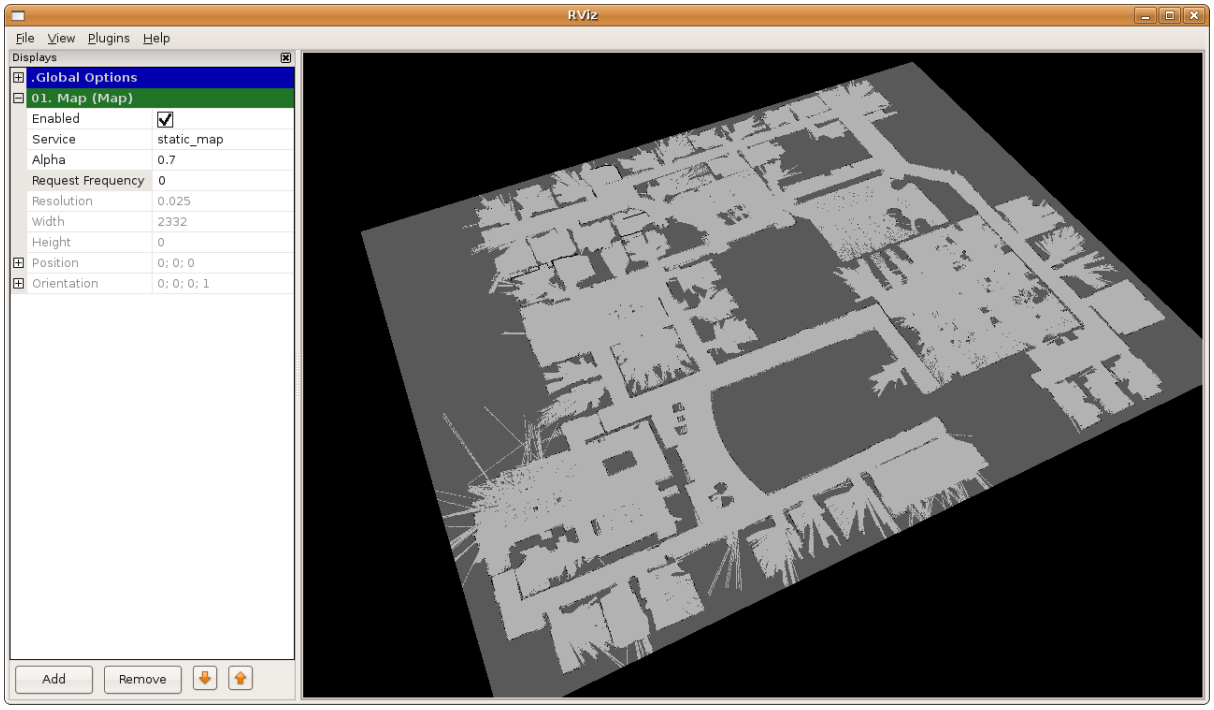


Figure 2.4: Example of a Metric Map - an occupancy grid map.

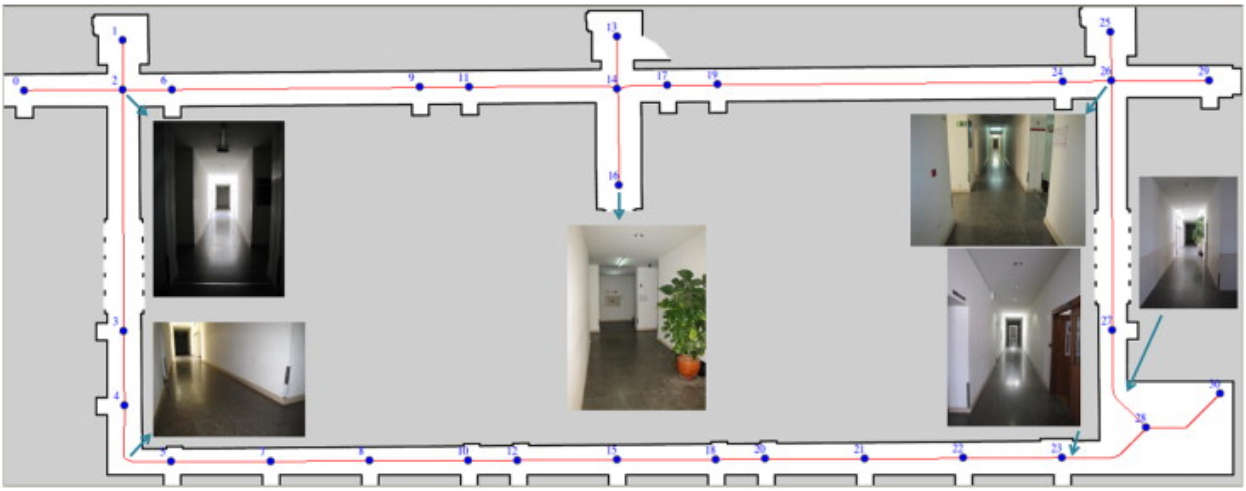


Figure 2.5: Example of a Topological Map [Portugal and Rocha, 2013].

or landmarks, which are connected by edges that represent the paths between them [Portugal and Rocha, 2012]. However, the inaccuracy of the method makes it harder for recognition and maintaining consistency in large scale environments, particularly if sensory information is ambiguous, which may result in perceptual aliasing, *i.e.*, difficulties in recognizing similar places that look alike [Rocha, 2006]. Trajectory planning is much simpler since it is possible to adapt search algorithm known for graphs. An example of a topological map is shown in Figure 2.5 and a overview of characteristic of both types of maps is presented in Table 2.4.

Moreover, map merging consists on building a consistent model of an environment with data collected from different robots. If the initial locations of the robots are known, as well as their current location, map merging is a rather straightforward extension of single robot mapping. If robots do not know their relative locations it becomes much more difficult,

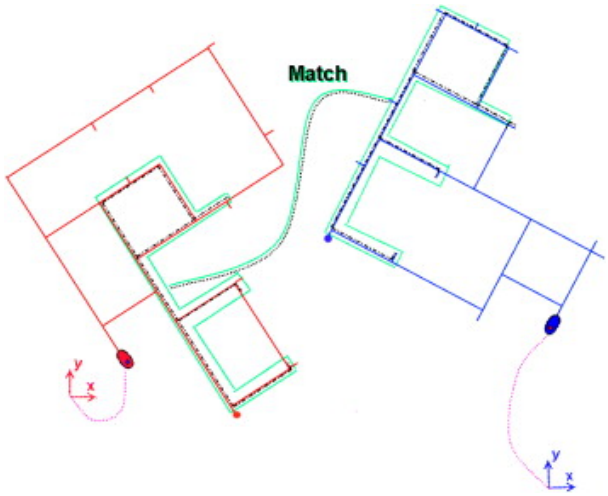


Figure 2.6: Example of a map merging task using topological maps.

Table 2.4: Characteristics of Grid-based maps and Topological Maps.

Type	Characteristics
Grid-based maps	• Metric maps • Easier to build • Easier to represent and to maintain • Higher computational cost • Memory problems depending on the size of maps • Recognition of spaces is non-ambiguous
Topological	• Graph-like maps • Efficient to use • Simpler than Grid-based maps • Does not require exact position of robot • Vertices correspond to landmarks • Vertices connected by edges witch represent path between them • Harder to recognize and maintain consistency in large scale environments • Recognition of places often ambiguous

since it is not clear how and where the robots' partial maps should be aligned together. The concept is illustrated in Figure 2.6.

2.5 Inter-Robot Communication

Another important matter is the communication among individual robots in a MRS. For example, robots in a MRS may be quite simple and have limited communication capabilities. Communication is crucial for group architectures, because it influences of inter-agent interaction, the ability of agents to model successfully other agents' states, *i.e.*, the agents' awareness, and the agent's ability to share information and successfully build a world model – a basis for reasoning and coherently acting towards a global system goal. Communication in the literature is divided into two fields: implicit communication and explicit communication.

Implicit communication also called *stigmergy* is a method of communicating through the environment, suitable in a situation where real time response is not required (for example, communications through RFID tags in the environment) and is usually predominant in social insects (for example, the pheromone in trails of ants). On the other hand, explicit communication is a direct messaging technique that can be applied when the number of mobile agents is low and when a fast reaction is expected [Rybski et al., 2004].

The communication between robots can augment their reasoning and perception and increase the efficiency of the mission. While most of the research on communication for MRS has been devoted to design axes that are mostly related with the communication structure, there are others important questions about communication that should be answered:

- What should be communicated?
- When to communicate?

The idea behind these questions is to avoid communicating redundant information so as to use efficiently communication resources. Therefore the proposed MRSLAM should support distributed control and distributed data; enabling to share efficiently sensory data in a team of cooperative mobile robots, using a measure of information utility; taking advantage from the cooperation among homogeneous robots to build a map in less time than a single robot or than a team with less robots, especially after refining the architecture with the coordination mechanism.

As communication is always limited, either in resources applied to perceive the world or in bandwidth of a communication channel, using efficiently those resources is crucial to scale up cooperative architectures for teams of many robots, without limiting them to simple reactive and loosely-cooperative systems, with very limited or no awareness. These questions about limited communication, avoiding communicating redundant information and using efficiently the communication resources will be discussed in more detail in next chapter, where the description of the approach followed is presented.

2.6 Summary

Now that the issues of localization, mapping, representations and limited communication were introduced, a MRSLAM strategy that implicitly incorporates these subjects was implemented and is described in more detail in the next chapter. As for representation, the metric approach used is occupancy grids maps, the coordination architecture is distributed, robots communicate when they are close by and when the probability of delivering messages exchanged between them is high, by monitoring the quality of the communication link between them. Further details on how the maps are acquired, when the robots exchange maps, how alignment and merging of maps is conducted and how the robots mutually detect teammates will be discussed in the next chapter.

Chapter 3

Implementation of a Multi-Robot SLAM System

This chapter provides a detailed description of each module of the MRSLAM System in order to generate a coherent global map of an environment using a team of cooperative robots.

The algorithm consists of diverse modules that are regarded as the groundwork of this MRSLAM's approach presented on Table 3.1. Firstly, some aspects about the framework used and how to support MRS are presented in section 3.1. Each robot runs a single robot SLAM approach based on RBPF-SLAM, which is described in section 3.2. In order to maintain an update list of teammates that are currently performing the cooperative task, we leverage the information made available by the communication protocol Optimized Link State Routing Protocol (OLSR). This will enable the discovery and detecting departures of teammates described in section 3.3. Furthermore, we monitor the link quality between each pair of robots so as to conduct a *rendezvous* and exchange maps between them (*cf.*, section 3.4). When a robot receives a partial map from one of its teammates, map alignment and merging is necessary to the fuse both contributions.

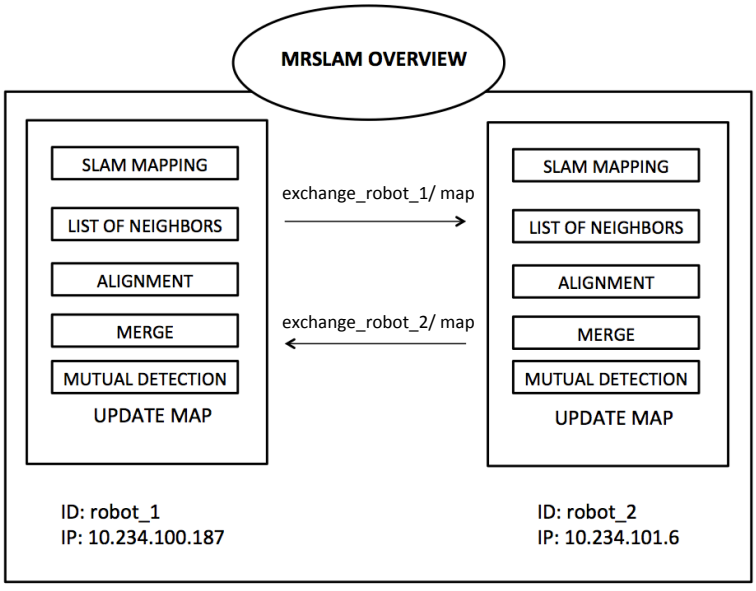


Figure 3.1: Overview of the MRSLAM approach.

MRSLAM Module	Brief Description
MRS in ROS	Organizing ROS structures for Multi-Robot Systems
SLAM approach	Using RBPF-SLAM as the individual mapping procedure of each robot
Discovery and Departure	Continuously updating the list of teammates in the mission
Rendezvous and Map exchange	Monitoring the link quality
Alignment	Finding the transformation to align partial maps
Mutual Detection	Avoiding mapping other robots as obstacles
Map Merging	Creating the global map by fusing partial contributions

Table 3.1: Modules list and brief description.

This is described in sections 3.5 and 3.7. Additionally, a method for mutual detection of teammates is described in section 3.6. The diagram in Figure 3.1 shows the basic operation of the algorithm running on each robot with the respective modules.

3.1 Multi-Robot Systems in ROS

In this work, Robot Operating System (ROS) is used as the development framework. ROS is a framework for robot software development and is maintained by Willow Garage[1]. It uses a publish/subscribe semantics architecture and a central master instance known as the *roscore*. The *roscore* provides lookup information about messages and processes for each system. Each *node* (*i.e.*, application or process) reports its register information back to the *roscore*. A *node* that subscribes to a *topic* requests a connection information through the *roscore* and connects directly (via peer-to-peer) to a publisher *node*, thus receiving messages from it (Figure 3.2). The interpretation and the execution of algorithms are controlled by such ROS *nodes*, which can be programmed in C++, Python, Java or Lisp. Therefore, custom nodes can be developed intuitively and on a high abstraction level. Moreover, ROS provides a series of tools and existing algorithms that can be easily tested in robots [Quigley et al., 2009].

In general, the multi-robot case over wireless communication in ROS presents a challenge and most of the focus for solutions is to support multiple ROS masters, *i.e.*, distributed roscores. To that end, a ROS package called *wifi_comm*[2] is used to publish and subscribe topics to/from "foreign" masters. This is done by opening a connection between the machines with known Internet Protocol (IP) which share a common topic and relaying specific information from a local topic over the given connection.

Another challenge to overcome when dealing with multiple robots are the fact that each robot should have a different namespace to avoid misinterpretation of data transferred between robots. To deal with this, when passing messages between robots with similar link

[1] Willow Garage's site: `http://www.willowgarage.com/`
[2] *wifi_comm* package: `http://www.ros.org/wiki/wifi_comm`

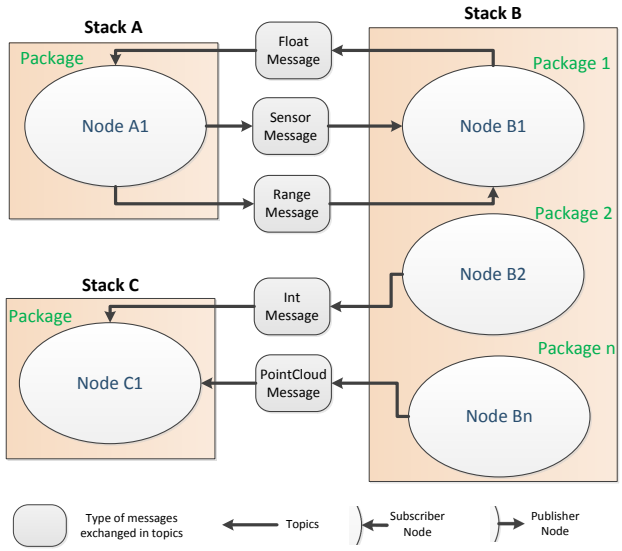


Figure 3.2: ROS architecture example diagram [Araújo, 2012].

names a *frame_id* remapping needs to be applied, both to transform data as well as to use a common naming convention. The use of the *tf_prefix* parameter is the recommended

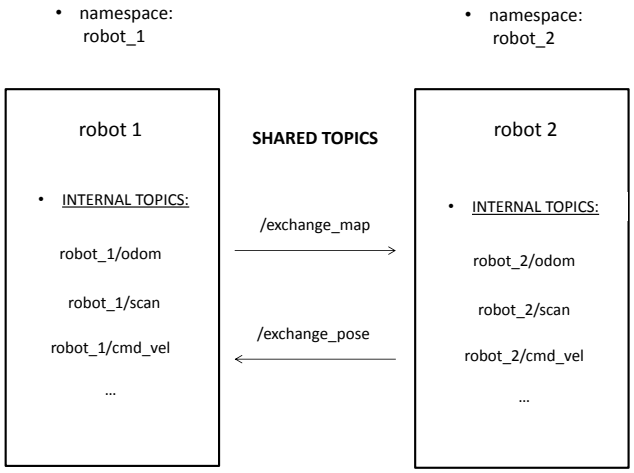


Figure 3.3: Definition of an unique namespace for each robots for all its information's.

standard way of using multiple robots, when sharing reference frame transforms. Additionally, the user may also need to remap topics of each individual robot, for example in the case where the different robots use the same node internally: the output of a Laser Range Finder (LRF) sensor from robot 1 should not be published in the same topic as the output of an LRF sensor of robot 2. Each robot should publish the scan message in a different and appropriate topic. Defining a prefix will allow each robot to use a unique namespace for all its data and transforms, this prefix will turn things like the odometry reference of the robot (typically named "/odom") into "r1/odom" and "r2/odom" automatically, depending on which robot it is. This is shown in Figure 3.3.

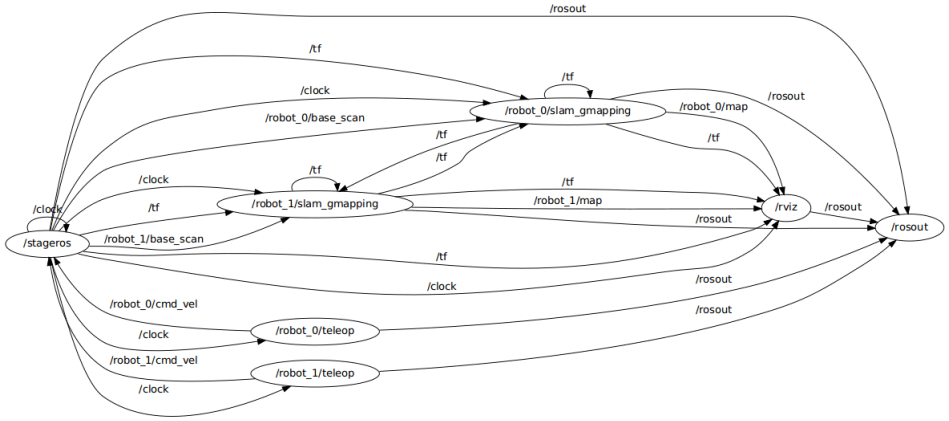


Figure 3.4: Hierarchical multi-robot control system (*rxgraph*).

ROS supports replication of functionality by allowing nodes and entire roslaunch cluster-description files to be pushed into a child namespace, thus ensuring that there can be no name collisions. Essentially, this prepends a string (the namespace) to all nodes, topics, and service names, without requiring any modification to the code of the node or cluster. The graph from Figure 3.4 is obtained running the *rxgraph* tool in ROS. Figure 3.4 shows hierarchical multi-robot nodes, where in each robot uses a SLAM package with its own namespace.

3.2 SLAM approach

In this work, we will give a first step towards developing a MRSLAM closed solution for the CHOPIN R&D Project. At this point, there is no out-of-the-box solution available in ROS for MRSLAM, which has been tested successfully and with firm credits. Accomplishing that is the main goal of this disseration. To that end, in this preliminary stage of the work, a SLAM algorithm was tested in each robot. From all the existing single-robot SLAM algorithms on ROS, *Gmapping* [3] is the most stable and commonly-used *package* with recognized results, being continuously supported and updated by the ROS community. For this reason, it has been adopted in this work to run on each robot. This package provides a laser-based SLAM as a ROS node called *slam_ gmapping*. Using *slam_ gmapping*, one can create a 2-D occupancy grid map from the data collected from a LRF. The *slam_ gmapping* node transforms each incoming scan into the odometry frame of the robot. A map is then built which can be viewed in ROS visualizator: *rviz*[4].

Gmapping is a laser-based RBPF-SLAM algorithm as described by [Grisetti et al., 2007]. These kinds of algorithms represent the posterior probability by a number of weighted particles wherein each particle is given an importance factor. On the other hand, this class of algorithms usually require a high number of particles to present adequate results. This

[3]Gmapping ROS package. Available at: `www.ros.org/wiki/gmapping`
[4]ROS's visualizator: Rviz. Available at: `www.ros.org/wiki/rviz`

requirement increases the computational complexity which is not ideal. In addition, it has to deal with the depletion problem associated with the particle filters resampling process that decreases the algorithm accuracy. This particle depletion problem consists in the filtration and consequent elimination of a large number of particles from the sample set during the resampling stage. This situation occurs because their importance weights become insignificant,*i.e.*, the particle has a small probability of representing the correct hypothesis. Therefore, an adaptive resampling technique has been developed by [Grisetti et al., 2007], which minimizes the particle depletion problem. The authors also proposed a way to compute an accurate distribution by taking into account not only robot's motion, but also its most recent observations. In mobile robotics, most of the particle filters proposed use the odometry motion model. However, when a mobile robot is equipped with a LRF, which is a very accurate sensor, the model of that sensor can be used, since it achieves extremely peaked likelihood functions. Based on these factors, the work of [Grisetti et al., 2007] integrates the most recent sensor observation z^t and computes a Gaussian approximation to efficiently obtain the next generation of particles. The Gaussian parameters are given by the following equations:

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} . \sum_{j=1}^{K} x_j . p(z_t | m_{t-1}^{(i)}, x_j) . (x_t | x_{t-1}^{(i)}, x_t) \tag{3.1}$$

$$\Sigma_t^{(i)} = \sum_{j=1}^{K} p(z_t | m_{t-1}^{(i)}, x_j) . p(x_j | x_{t-1}^{(i)}, \mu_{t-1}) . (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T \tag{3.2}$$

where K is the number of sampled points and μ is the normalized factor. Using this distribution, the weight of the i^{th} particle is:

$$w_t^{(i)} = w_{t-1}^{(i)} . \mu^{(i)} \tag{3.3}$$

Equation 3.4 computes the effective number of particles N_{eff}, which is a trigger that defines when the resampling step should be performed:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2} \tag{3.4}$$

The algorithm resamples each time N_{eff} drops bellow a given threshold of $\frac{N}{2}$, being N the number of particles. This adaptive re-sampling decreases the uncertainty about the robot's pose in the prediction step of the particle filter. Due to the scan matching process, the number of required particles decreases since the uncertainty is lower. The number of particles used on most the experimental tests made with *Gmapping* was 30, which is an extremely low value when compared with most common particle filter approaches.

In Figure 3.5, a transform tree diagram with the coordinate frames associated with each robot is presented, each frame has its own prefix that informs the ROS system to which robot it belongs. The diagram from Figure 3.5 also shows the average rate of messages of

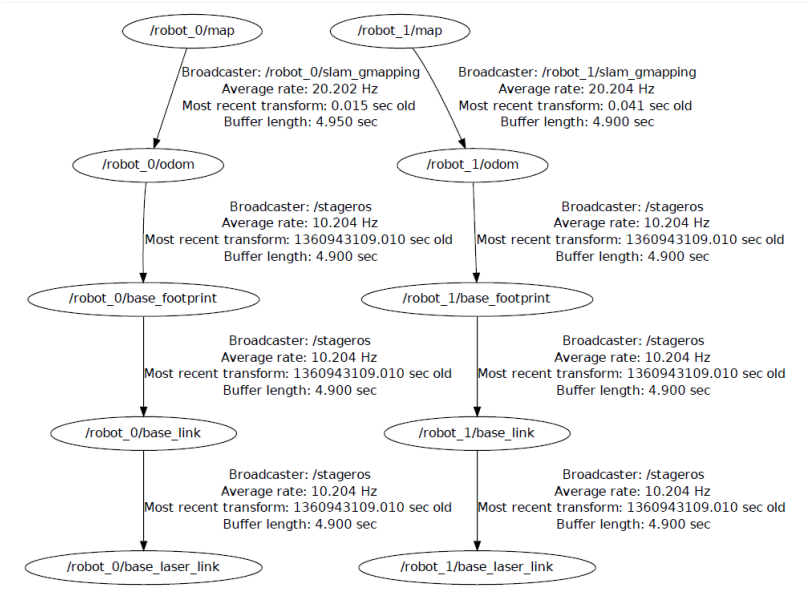


Figure 3.5: Transform tree (*tf* tree) diagram of each robot while acquiring partial maps.

each transform and the existing frames of two robots using the *Stage* simulator while they are building a partial map of the environment.

3.3 Discovery and Departure of teammates

This section describes how robots become aware of teammates that arrive and leave the network so as to cooperatively perform the mapping task. Firstly, we assume that each robot knows its own ID and IP address. In the beginning, each robot loads a table with predefined information of all its teammates, namely their IP and ID, as shown in the Table 3.2. Using the OLSR ad hoc protocol features, a list of active teammates is maintained throughout the mission, by resorting to the OLSR deamon (OLSRD)[5]. The table with predefined information of all teammates is presented next.

The OLSRD is used in this work to provide the IP addresses that are reachable to a given machine, as well as a measure of the link quality between them. The link quality is a metric that is computed, depending on the signal's strength and the rate of packet losses. This information is made available on ROS, using *wifi_comm*.

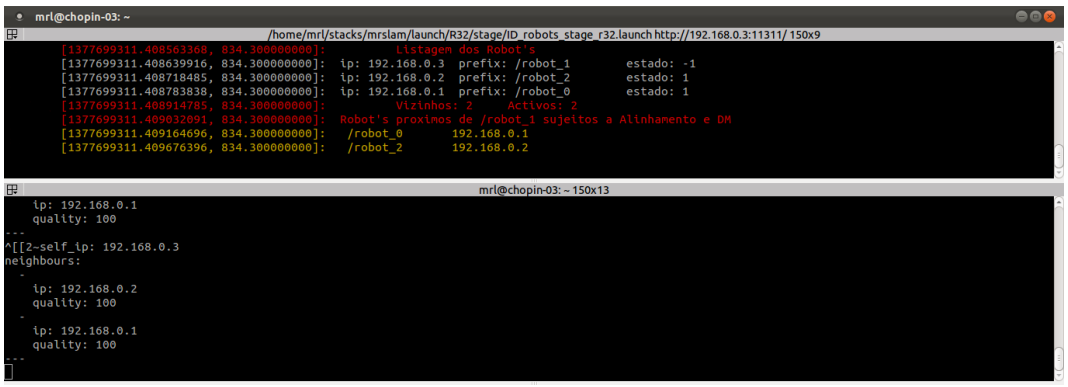
In Figure 3.6, each robot keeps track of its teammates by monitoring the link quality between them, instead of tracking the neighbors pose permanently. Therefore, robots may leave the network ,*e.g.*, to map remote areas, and return without affecting performance or jeopardizing the cooperative mission. In fact, even though the OLSRD is meant for MANETs, it also supports wifi infra-estruture networks and allows to report peer-to-peer link quality.

In Figure 3.6, we can see a list of teammates available from the point of view of an agent (*robot_1*) for the exploration task. One can also observe that robots are identified by a

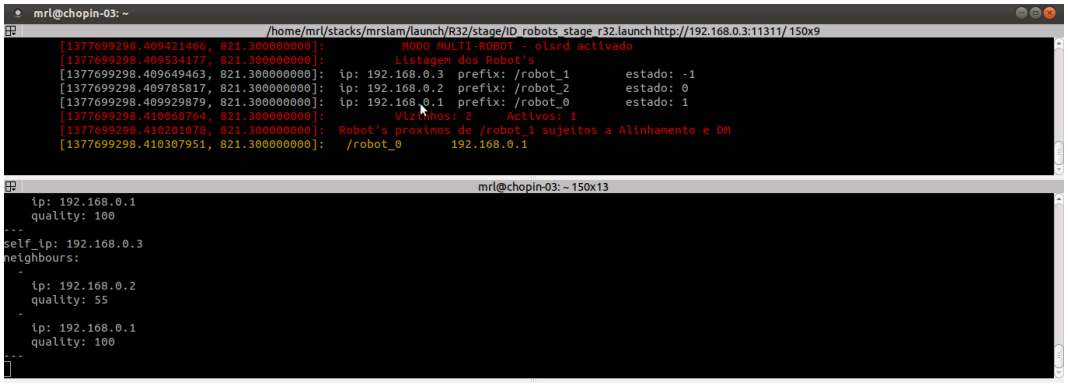
[5]OLSRD's page: `http://www.olsr.org/`

IP	ID
10.234.100.19	/robot_0
10.234.100.187	/robot_1
10.234.101.6	/robot_2
10.234.100.184	/robot_3
10.234.101.36	/robot_4
10.234.101.57	/robot_5

Table 3.2: Example of a table with IPs and IDs of possible teammates involved in the mission.



(a) Three robots near each other.



(b) One robot close by and other going away (link quality: 55%).

Figure 3.6: Tracking of teammates and monitoring link quality.

prefix associated to a fixed IP and a robot's state. This state can have three values, -1 indicating the robot where the *node* is running, 0 if the robot has low link quality and 1

otherwise. Link quality gives an estimate of robot's proximity, since the value is influenced not only by the distance between them, but also the physical obstacles around them. As seen in Figure 3.6a, with this parameter we can infer if the robots are near or far from each other. This metric enables the exchange of information between robots that are considered next to each other. Thus, we prevent in a general way the information overload of the network and continuous use of it by only exchanging maps in a pairwise fashion, between robots that are tightly connected inside the network. This is done when the state value becomes 1 (using a threshold for the minimum link quality) and a timeout has passed since the last exchanged of information. It may happen that the robots move away and the link quality decreases, getting below the threshold set. When this happens, the robot is no longer under the necessary conditions to exchange information and the list is then updated, as shown in Figure 3.6b (robot 2 state).

This module is critical because it allows to connect a prefix to each robot through the respective IP and each agent understands how many teammates are cooperating with it. Assigning a prefix to each robotic agent also allows to share the necessary information by, correctly handling the topic to create the publisher/subscriber. This information can be of different types, like occupation grids, poses, laser scans, point clouds or messages about the link quality. Another important feature about this module is that it enables the organization of all information unambiguously using the prefixes. Figure 3.7 illustrates an example of information organization in *topics*.

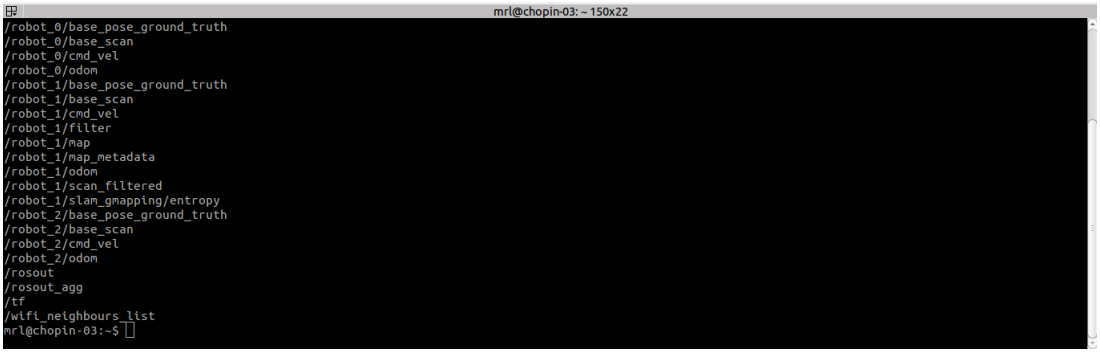
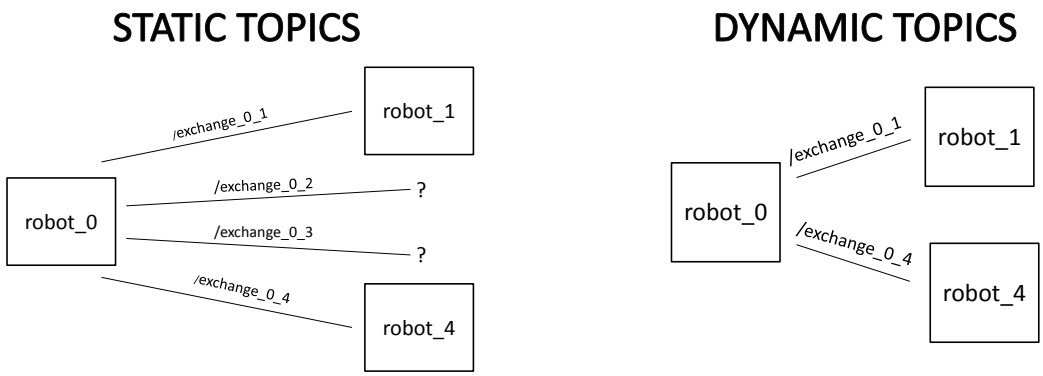


Figure 3.7: List of ROS *topics* organized by prefix of each robot.

3.4 Dynamic topics, *Rendezvous* and Map Exchange

As this is a generic and distributed MRS approach wherein the number of robots is variable, one cannot define all communication channels (*i.e.*, ROS topics) in the beginning of the mission. Therefore, dynamic topics that are created along the mission should be supported. In most cases, the user defines static topics for each node whenever the subscriber or the publish are initialized. Consider the robot running a given node, called *robot_home* and the neighbor robots running a node called *robot_away*. There will be a 1 to P relation, where P represents the number of neighbors of *robot_home*. The program must be prepared to access each neighbor robot information. If all robots taking part of the mission were

known at the beginning, this would be straightforward by using P static topics. However, this is not the case. After running the discovery module, *robot_ home* is capable of retrieving the number and IDs of its neighbors and, from that moment, it is capable of manipulating topics in order to access each neighbor's data the program must be able to dynamically create the topics whenever teammates are discovered, considering robots' prefixes. In Figure 3.8 a simple example of the previous statement is shown, for the case of static and dynamic topics with three robots.



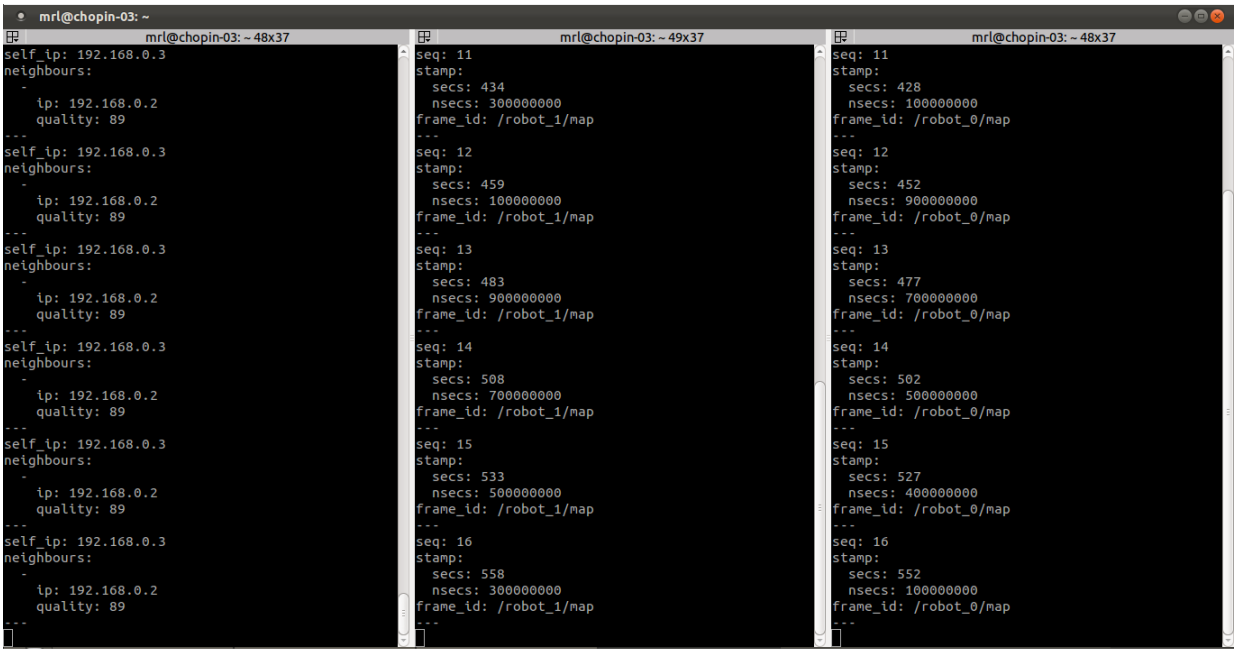
(a) Static topics created in the beginning of the mission.

(b) Dynamic topics created after teammates discovery.

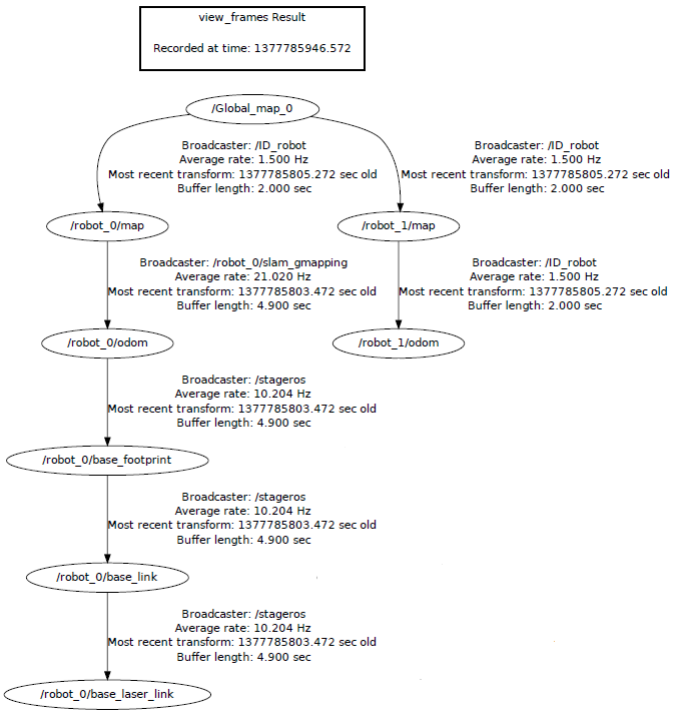
Figure 3.8: An example of a mission with 3 robots: *robot_ 0*, *robot_ 1* and *robot_ 4*. Creation of static and dynamic topics.

Dynamic topics are created similarly to static topics. However, they are only defined when robots generate the discovery table. Their naming through string manipulation is fairly straightforward once the robots' IDs are known. A *rendezvous* is a meeting between two or more agents at an appointed place and time, *e.g.*, when two people meet at a familiar restaurant. The problem of *rendezvous* is ubiquitous in nature. MRS also have an inherent need for the ability of inter-agent *rendezvous*. The ability to meet facilitates localization, allows collaborative map exploration and has a plethora of other advantages but, most importantly, enables a reliable communication as most existing hardware agents are only capable of communication over short distances. Environmental geometry, wireless transmission technology, power considerations and atmospheric conditions all contribute to fairly short communication limits. A common constraint that is rarely satisfied in the real world for a successful communication is to maintain "line-of-sight" between agents. However, MRS for the majority of real-life applications are highly rewarded only with some level of communication, when compared with single agent systems or MRS that do not communicate. In an unknown environment, however, the assumption of an absolute *rendezvous* spot is not passive to be taken. A common strategy in MRSLAM is to consider that one agent should wait to be found by other agents.

In the early stages of this project this problem did not deserve much attention because the first arena was considerably small and therefore, the maps were smaller and the information



(a) Efficient and relevant exchange of information between two robots considering a timeout and signal quality.



(b) Transform tree diagram of two robots exchanging their maps and poses into a common referential frame. Note that this is the point of view of *robot_0*.

Figure 3.9: Example of efficient exchange of information between robots and the respective Transform tree diagram.

traveled smoothly through the network.

For larger arenas, this issue appeared and required prompt solution. The best solution

was to constrain the information exchange between the robots, and to communicate all the relevant information by respecting a timeout. Doing so, specific topics with temporary subscribers were created only for the robots who needed to exchange information, thus limiting the information to the necessary robots only. The first measure was to consider exchange of information between robots based on their link quality. That way, a robot only sends information to another robot when the link quality between them is higher than a threshold, in this case, 80%. The second measure was to define a timeout (30 seconds) in order to allow the robots to explore new areas after the last exchange of maps. These measures were created in order to improve the flexibility of the MRSLAM approach and to avoid overloading the network. Both the threshold and the timeout are adjustable depending on the type of mission. These measures, combined with an extremely limited communication, foster not only an efficient and robust control, but mostly a good shareable information management without constantly flooding the network. Figure 3.9 shows an efficient information exchange by considering these measures and a transform tree diagram of two robots exchanging their maps into a common referential frame designated as */Global_map* .

Every time the link quality is within acceptable values (in this case, 80%) and the robots have not changed maps for more than the defined timeout (in this case, 25 seconds), maps are exchanged again.

Despite the proposed measures, additional measures could be implemented to further maximize communication efficiency. The implementation of a method to retrieved which map cells were updated would be a positive asset, as it would allow to send only the index as well as the value of the updated cell instead of the entire map. This contribution, combined with the herein considered measures, would significantly optimize the information exchange process, as well as to considerably minimize the network overload with unnecessary information.

3.5 Map Alignment

The problem of fusing two maps represented as occupancy grids is similar to the image registration problem studied in computer vision [Brown, 1992]. Maps can be in fact, seen as pictures, and map fusing can be seen as a particular case of image registration. Map fusion is tackled in two different stages:*map alignment* and *map merging*.

In the alignment stage, the objective is to find the transformation between the different reference systems of the local maps. In this situation, most approaches try to find the relative position of robots. In this sense, the relative position of robots is supposed to be known, since robots establish a meeting point in order to measure their relative positions using, for example, cameras and color codes to identify teammates.

In many approaches, the transformation between maps is performed with the matching of landmarks [Howard, 2006]. With relation to the previous strategies based on the *rendezvous* case, it is clear that the alignment between maps is possible and immediate if robots succeed in detecting each other. More difficult would be the approach in which robots determinate

whether any alignment exists or not, without the need of explicitly meeting just by sharing the information of their maps [León et al., 2008]. The selected aligning method is crucial, since it must be robust to sparse correspondent landmarks between the maps and offer an accurate solution to the map alignment problem. The robots start at different positions and begin their navigation tasks independently. That is to say, they have no knowledge about other robots positions nor observations. Each local map is referred to a different reference system, which is located at the initial position of the robot. The 2D aligning method computes the transformation between the two local maps, which consists of three alignment parameters: translation in x and y and the rotation θ. In order to do so, the method first obtains a list of correspondent landmarks between the maps. It is noticeable that the method obtains only a first estimate of the aligning parameters. The set of correspondences and this estimate are used as the input of a least squares minimization that eliminates outliers and obtains the final solution that is expressed in the same global reference system. Fortunately, ROS provides a *package* for alignment that does exactly that, called *mapstitch*[6].

After studying and testing the package, it was necessary to implement minor changes in some functions to adapt to the MRSLAM problem. These small changes have to do with verification of the rotation matrices produced by the alignment function. A rotation matrix is a linear transformation that when multiplied by any vector causes a rotation of that vector along an axis, maintaining its length. The properties of these matrices are:

- $\mathcal{M} \in \mathbb{R}^{N \times N}$ is a rotation matrix if and only if $\mathcal{M}$ is orthogonal. $\mathcal{M}$ is orthonormal if the scalar product between two vectors column is zero, and the scalar product with itself is a unit vector.

- $\mathcal{M}$ is anti-symmetric. Therefore, the inverse of the rotation matrix is equal to its transpose, allowing the rotation of a vector in a clockwise direction: $\mathcal{M}^{-1} = \mathcal{M}^{\top}$

- The rotation matrix determinant is equal to 1: $\det \mathcal{M} = 1$

Some of these properties were not fulfilled in the original package, as the matrix multiplication (*i.e.*, translation to the origin, rotation around Z axis of θ degrees and translation back to the point where it was) would cause one of the maps to translate to infinite. Therefore, we have used the Speeded-Up Robust Features (SURF) detector available in OpenCV[7], which is a computer vision framework that is included in ROS. SURF was presented in [Bay et al., 2006]. It is based on the similar properties of Scale-invariant feature transforms because of its remarkably good performance as compared to other detectors and descriptors.

For the process of finding frame-to-frame correspondences, called as matching procedure, we use *brute-force matcher* which looks for each descriptor in the first set and the closest descriptor in the second and attempts each one. To reject wrong correspondences, a cross-match technique is used. The idea is to match train descriptors with the query set and

[6] mapstitch ROS package. Available at: `http://ros.org/wiki/mapstitch`
[7] OpenCV's page: `http://opencv.org/`

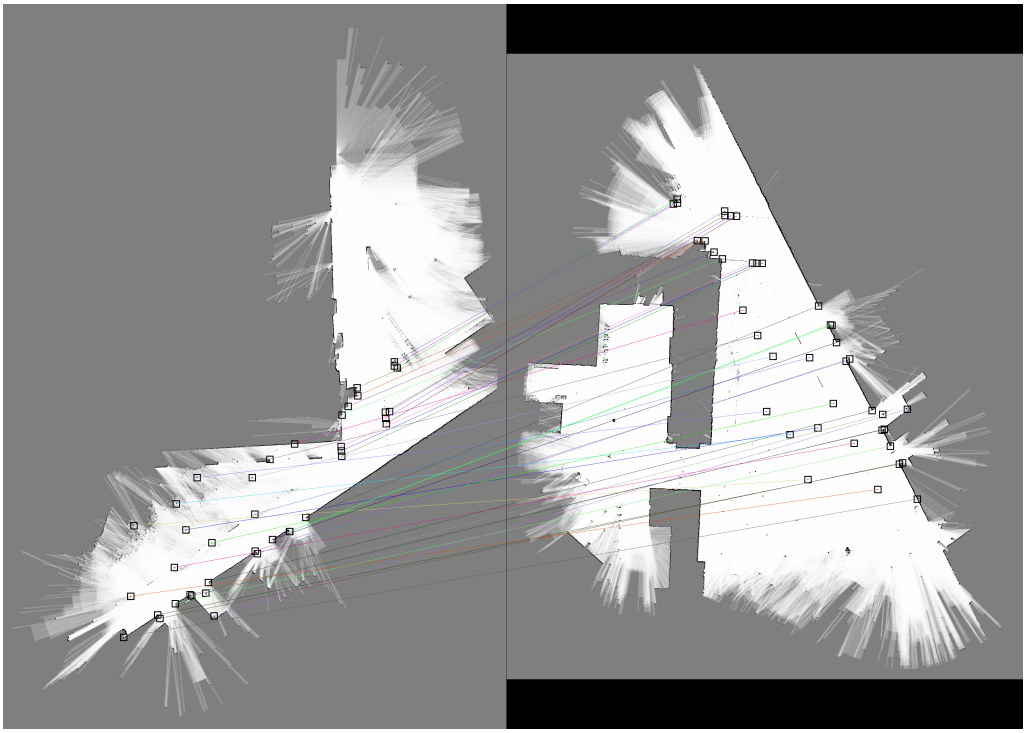


Figure 3.10: Example of Map stitching.

vice versa. Only common matches for these two matches are returned. Such techniques usually produce satisfactory results with minimal number of outliers, when there are enough matches.

Summing up, this module will generate transforms corresponding to alignment of the maps, given the features detected by the robots when they are mapping the environment. An example of this is illustrated in Figure 3.10.

3.6 Mutual Detection Model

To benefit from the multi-robot mapping approach, robots must be able to locate themselves relative to one another in a fast, accurate and reliable manner. This means that robots working for a common goal must possess the ability to sense each other. To achieve this ability, we assume that we have a team of homogeneous robots and that the physical dimensions of robots are known. In order to avoid mapping incorrectly the environment, due to obstructions caused by teammates, a process that filters the robot laser scan has been developed. This process, denoted as *mutual detection*, benefits from the received transform that results from the map alignment between two robots and relate the robots' maps, as well as their respective poses, in a common referential. That enables the use of a boundary box that will define a region whose center is located in the Cartesian coordinates of the detected teammate's pose. Each robot shape is approximated by a circle of radius that is adjustable according to the robot's model used on the mission. The definition of this technique is fundamental for the filtering process, since one can check if the robot's laser scans intersect the defined region. In these situations, the robot is detecting a second robot and the generated

map will not match the real scenario. Hence, a republication of the laser message needs to be carried out, in which all the scans that intersect that region are discarded. The result of that filtering process is a map of the environment without robot footprints.

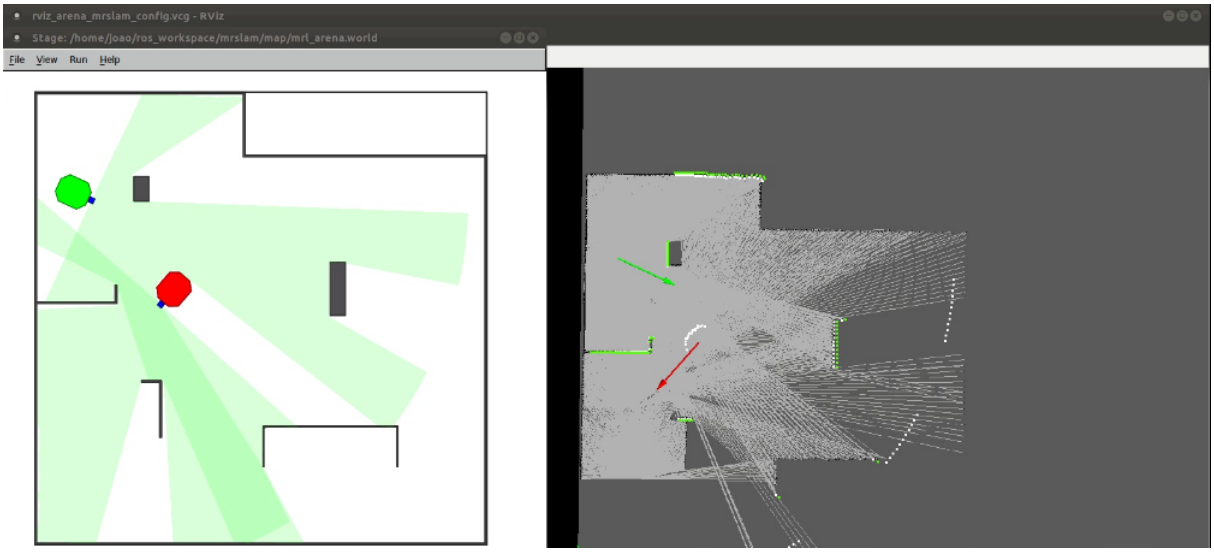


Figure 3.11: Illustrative figure of the module's operation.

A new node named "filter" was created to republish the laser scan messages and the filtered messages by the mutual detection module at a 10Hz rate. This laser message publishing node works in a fairly simple fashion. When the laser scan of a robot is detecting a teammate, a flag is activated and the mutual detection module will start filtering scans. That node, which initially only published original messages from the robot's laser to *Gmapping* will now publish filtered laser messages by the proposed mutual detection module. The node will filter messages while the robot is detecting a teammate. To switch back to the normal laser messages at the end of the mutual detection module, the flag is activated once again and the system informs the publishing node to switch back.

Figure 3.11 represents the exploration of an area using a team of two robots easily distinguished by their colors. We can observe on the right side the information relative to the green robot. The red arrow that represents the pose of the red robot was only introduced to promote an easier visualization of the map created by the green robot. The green and white dots corresponds to the filtered laser scan and the untreated data laser scan, respectively. The laser readings that detect the red robot are discarded, thus allowing the creation of the map without unwanted detection's from other teammates.

3.7 Map Merging

Map merging is a challenging task, especially for real-time multi-robot SaR operations in disaster environments. Combining the partial maps of all robots into a global map allows the robot team to avoid a repeated and undesired exploration of the regions by different robots. In this dissertation, we introduce a novel map merging methodology for occupancy

grid maps to obtain a global and consistent environment map for multi-robot exploration operations in SaR environments. A map fusion methodology that can be used in any generic environment is proposed. The presented algorithm is capable of successfully merging the partial occupancy grid map of robots with a limited degree of overlapping regions between their maps.

We assume that an occupancy grid map M is a matrix with r rows and c columns. Each cell $M(i,j)$ may contain three different values, indicating whether the cell is free, occupied, or if its status is unknown. Maps produced by contemporary SLAM algorithms usually encode occupancy beliefs for each cell, and can therefore be easily converted into the representation we require. The spatial location of cell $M(i,j)$ is indicated as $(x^{i,j}, y^{i,j})$. Given two maps, M_1 and M_2, and a rigid transformation T, as in (3.5), between the reference frames of the robots calculated in the alignment phase in section 3.5, the goal is to generate a merged map overlapping each partial map.

The second robot's environment map is transformed using (3.6), such that it aligns its own map with the first robot's map.

$$T = \begin{bmatrix} Cos(\theta) & -Sin(\theta) & \Delta_x \\ Sin(\theta) & Cos(\theta) & \Delta_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

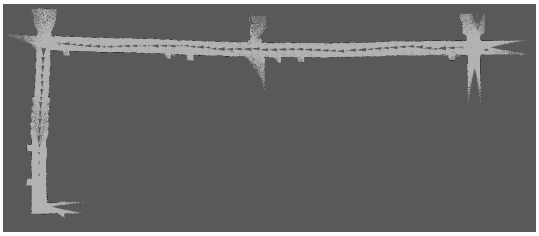
$$M_2^{'} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} .T(\Delta_x, \Delta_y, \theta).M_2 \tag{3.6}$$

The transformed occupancy grid map of the second robot $M_2^{'}$, can then be fused with the first robot's map. Given two maps M_1 and M_2, there can be multiple transformations overlapping them. Therefore, it is necessary to establish a metric to decide which one is better, and to reject possible false positives.

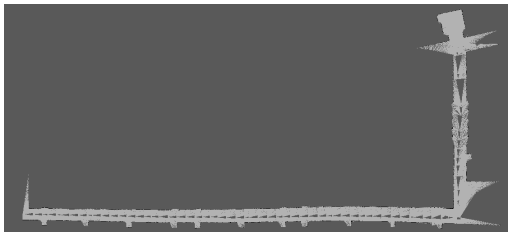
In this subsection, we present the approach to manipulate occupancy grids and merge them into a global one. In order to achieve this, a stable package available in ROS, called *occupancy_ grid_ utils*[8] was employed. This package contains a few utilities for dealing with occupancy grids, represented as nav msgs::OccupancyGrid objects. These include coordinate conversions, shortest paths, ray tracing, and union of multiple unaligned grids, which is fundamental for what we intend to achieve: a global map built from two incomplete robot maps.

When the alignment function returns a possible transform between the maps to a common referential, that transform may be considered valid or rejected if it does not meet the alignment requirements. The requirements correspond to three parameters that represent the differences between the rotation and the translation, in X and Y, of the last two transformations of the maps of the same robots. A given transform is considered valid if those differences are inferior to a certain error parameter.

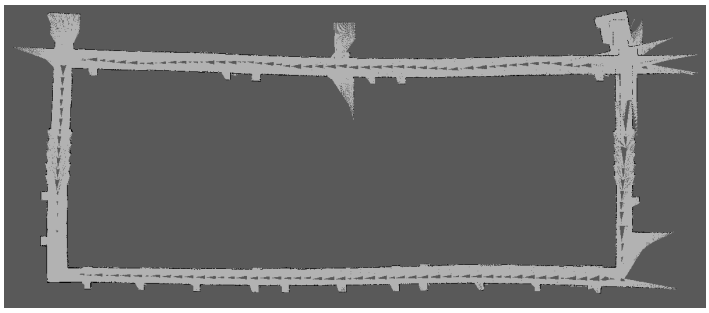
[8] *occupancy_ grid_ utils* ROS package. Available at: `http://www.ros.org/wiki/occupancy_grid_utils`



(a) Partial Map from Robot #1



(b) Partial Map from Robot #2



(c) Global Map

Figure 3.12: Simple example of the map merging process.

From the moment a valid transform exists, the conditions are set for map merging. To that end, the ROS package described in the beginning of the subsection is used to generate a merged map from the two occupancy grids and the transformation between them. Figure 3.12 presents an example of the map merging process.

3.8 Summary

In this chapter, the MRSLAM Algorithm and all its integrating modules have been fully explained. In the next chapter, simulation and real world experiments are described and discussed. The simulation tests were conducted using teams of two and three homogeneous robots and the experimental test were conducted using teams of two physical mobile robots.

Chapter 4

Results and Discussion

This chapter depicts and discusses the experimental results of the tests considered in this work. All tests were fulfilled using a distributed approach, wherein each robot is responsible for aligning and merging the maps in order to obtain the environment's global map. To represent the environment, occupancy grids maps were considered. In the next section, multi-robot simulation experiments are presented. Afterwards, we show experimental results retrieved in a real world scenario.

4.1 Simulation results

Simulation results using the Stage multi-robot simulator[1] in several arenas are presented in this section. In Figure 4.1, the ground truth of the test arena is presented. This arena is a small enclosed environment (around 4x4 m) and with only a few number of obstacles. It is located at the Mobile Robotics Laboratory in the Institute of Systems and Robotics (ISR) at the University of Coimbra. These preliminary results were taken in an initial phase of this work and only the alignment and map merging modules were functional at the time.

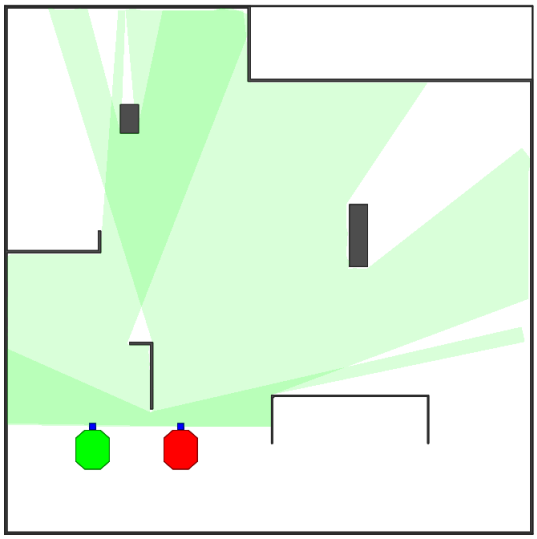
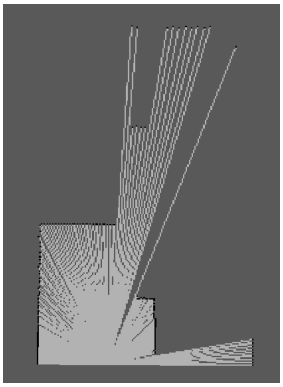


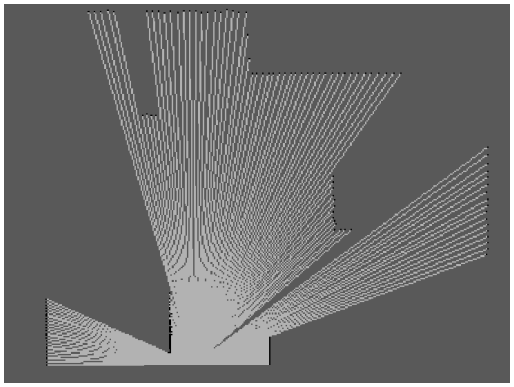
Figure 4.1: Test arena #1 used in Stage simulations with two robots at their starting point.

In order to distinguish the robots, a different color was assigned to each one: *robot_ 0*

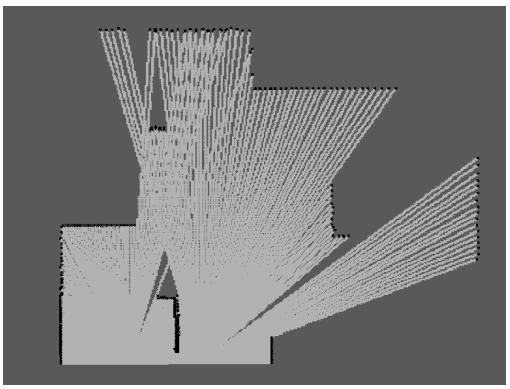
[1]Stage ROS package. Available at: `http://www.ros.org/wiki/stage`



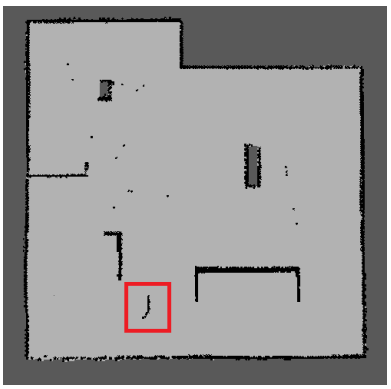
(a) Partial map built by *robot_ 0*.



(b) Partial map built by *robot_ 1*.



(c) Global map built by merging the partial maps from each robot at their starting point.



(d) Final global map at the end of the simulation. In this experiment, mutual detection was not considered.

Figure 4.2: Simulation results in arena #1 with two robots on stage.

is green, while *robot_ 1* is red. To enrich the experiments and easily interpret the results, the laser beams of each robot are presented. After merging the partial maps, it is possible to observe a global map with a definable resolution. The initial pose (x, y, θ) from *robot_ 0* (green) in the stage world is (1.0, 1.0, 90^o) and the initial pose from *robot_ 1* (red) is (2.0, 1.0, 90^o), as shown in Figure 4.1. The goal of the simulation is to test the map merging procedure and demonstrate problems that may arise if mutual detection is not dealt with, explicitly. To that end, in this test, robots should build the test arena using the two robots partial maps. The partial maps have a resolution of 0.02 m/pixel and a height/width of 1248 cells each. In this preliminary simulation experiment, robots are teleoperated and a centralized server receives their partial maps continuously at a rate of 0.2Hz. Therefore, the global map is merged every 5 seconds and publishes it to an adequate topic. Figures 4.2a and 4.2b represent each robot's partial map when they start their task. Figure 4.2c presents an instance of the global map, after merging the partial maps in Figure 4.2a and 4.2b. The global map has a resolution of 0.01 m/pixel and height/width of 2496 cells.

Figure 4.2d presents the final global map, after *robot_ 0* navigates through the test arena, while *robot_ 1* stays in the same initial place. It is noteworthy that mutual detection is still not implemented, therefore, *robot_ 1* is still detected as an obstacle by *robot_ 0*, as illustrated

by the red box in Figure 4.2d.

The results obtained in Figure 4.2 show that each robot's partial map was well aligned and merged. This validates the map alignment and merging processes since the global map obtained resembles the map in stage's virtual world.

The next results were obtained in a more mature phase of the project by considering the implementation of all modules described in chapter 3.

Figure 4.3 illustrates a snapshot of the stage window. This arena is a closed environment of large dimensions (around 13x9 m) with several obstacles. It is located at a classroom in the Department of Electrical Engineering and Computers (DEEC) at the University of Coimbra.

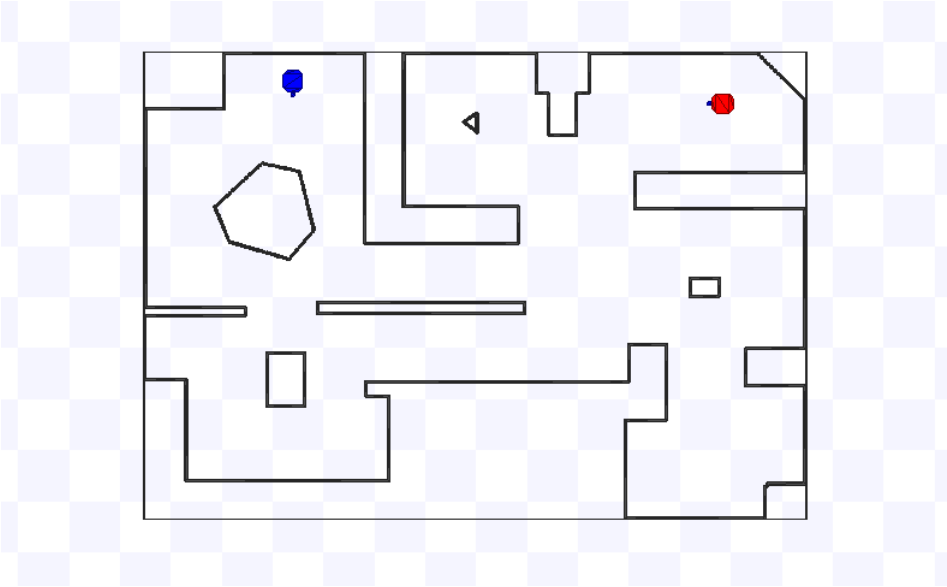


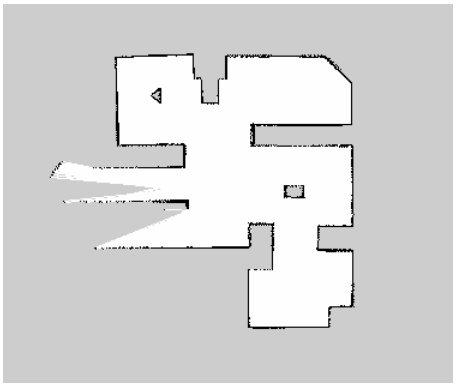
Figure 4.3: Test arena #2 used in Stage simulations with two robots at their starting point.

In order to distinguish the robots, a different color was assigned to each one: *robot_ 0* is red, while *robot_ 1* is blue as seen in Figure 4.3. The goal of the simulation is to build the entire test arena using the two robot's partial maps. The partial maps have a resolution of 0.04 m/pixel and a height/width of 544 cells each. In the simulation experiments, robots are teleoperated[2] using the keyboard and each robot sends its own partial maps to its teammates with a 30 seconds timeout, since the metric of link quality is not available in simulations. Each robot merges the map and publishes it to an adequate internal topic. Figures 4.4a and 4.4b represent each robot's partial map when they first meet. Figure 4.4 presents an instance of the global map, after merging the partial maps in Figure 4.4a and 4.4b. This map has a resolution of 0.01 m/pixel and height/width of 3329x2177 cells.

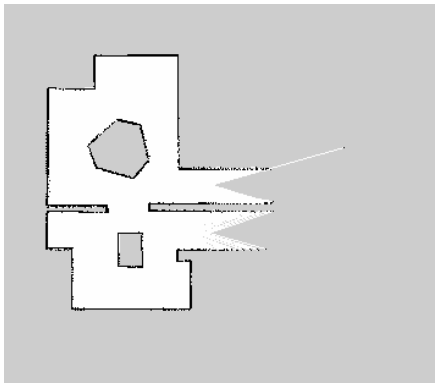
The results obtained show that each robot's partial map was well aligned and merged. To further evaluate the proposed strategies, Additional simulation experiments using the same arena with three robots on Stage were considered.

The robot with prefix *robot_ 2* was added to the same arena and identified with the green color as seen in Figure 4.5. The partial maps have the same resolution as in the previous

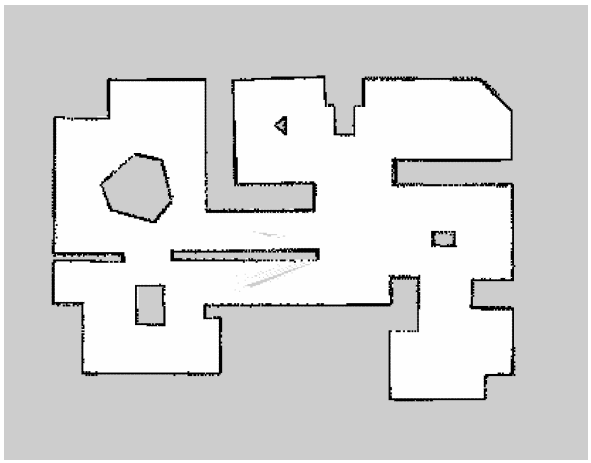
[2]teleoperation ROS package. Available at: `http://www.ros.org/wiki/teleop_twist_keyboard`



(a) Partial map built by *robot_ 0*.



(b) Partial map built by *robot_ 1*.



(c) Global map built by merging the partial maps from each robot.

Figure 4.4: Simulation results with two robots on stage.

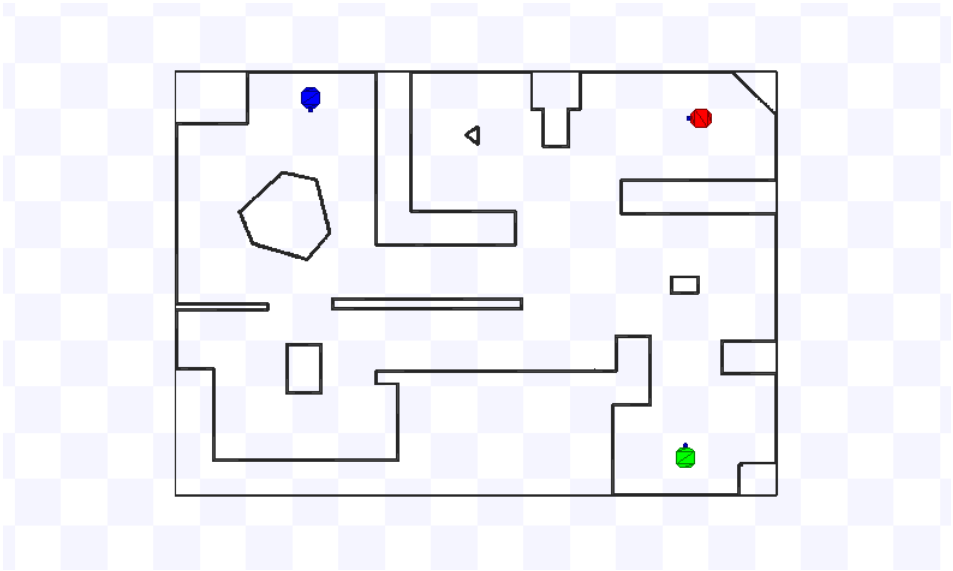
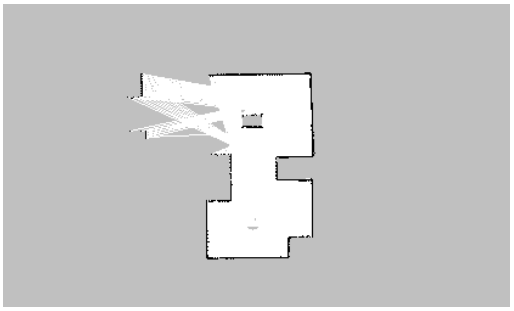
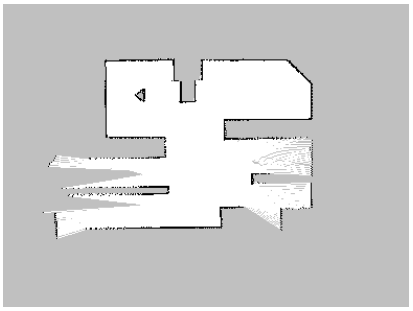


Figure 4.5: Test arena #2 used in Stage simulations with three robots at their starting point.

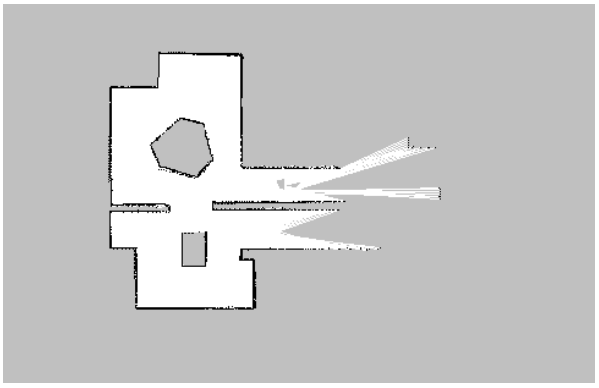
experiment. The robots send their partial maps to each other every 30 seconds, merging the maps into a global map and publishing it to an adequate internal topic. Figures 4.6a, 4.6b and 4.6c represent each robot's partial map before map exchange. When *robot_ 0* exchanges



(a) Partial map built by *robot_ 0*.



(b) Partial map built by *robot_ 1*.



(c) Partial map built by *robot_ 2*.

Figure 4.6: Partial maps from each robot using Gmapping on simulations.

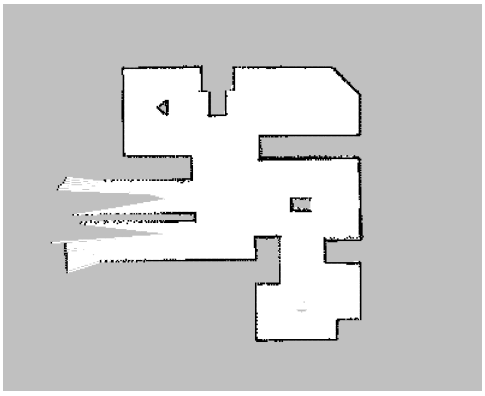
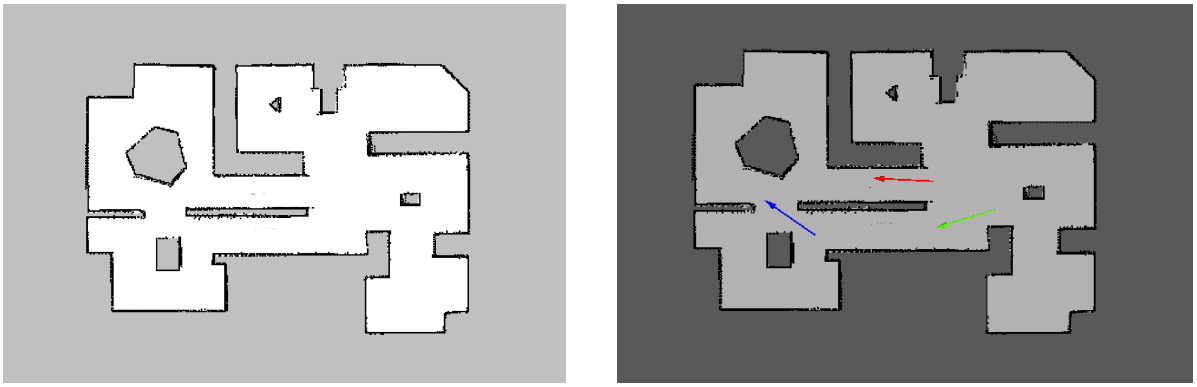


Figure 4.7: Result of merging the maps from *robot_ 0* and *robot_ 1*.

its map with *robot_ 1*, the partial maps of the robots are updated based on the result of the map fusion, as shown in Figure 4.7. Then, when *robot_ 1* exchanges maps with *robot_ 2*, the merging of the partial maps originates a global map with the contribution of all robots. This map has a resolution of 0.01 m/pixel and height/width of 3329*2177 cells. Figure 4.8 presents an instance of the global map, after merging the partial maps in Figure 4.7 and 4.6c.

Figure 4.8a presents the final global map, after the robots navigate through the test arena. The results obtained in figure 4.8b show that each robot's partial map was correctly aligned and merged and it can be concluded that the test was successful, since the global map obtained resembles the map in Stage's virtual world. It is also possible to see that, for such a large environment,*Gmapping* performance was outstanding. The reason for such



(a) Global map built by merging the contribution from each robot.

(b) Global map with the poses of each robot (*rviz*).

Figure 4.8: Simulation results with three robots on stage.

performance is mainly related with existence of several landmarks in the arena. The following section will discuss similar experiments carried out in the real world.

4.2 Real-World Results

The promising results from the simulation experiments, naturally suggested transferring our experimental setup into real robots. This section presents the results obtained from such experiments, in a real world testing environment. The proposed methods are tested with a team of two Pioneers 3-DX[3], one of which is presented in Figure 4.9.

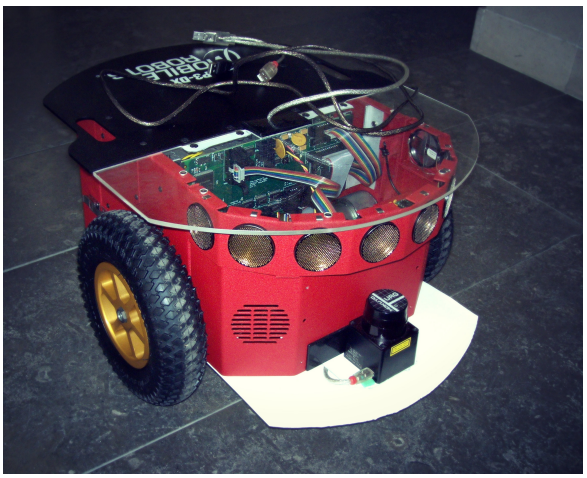
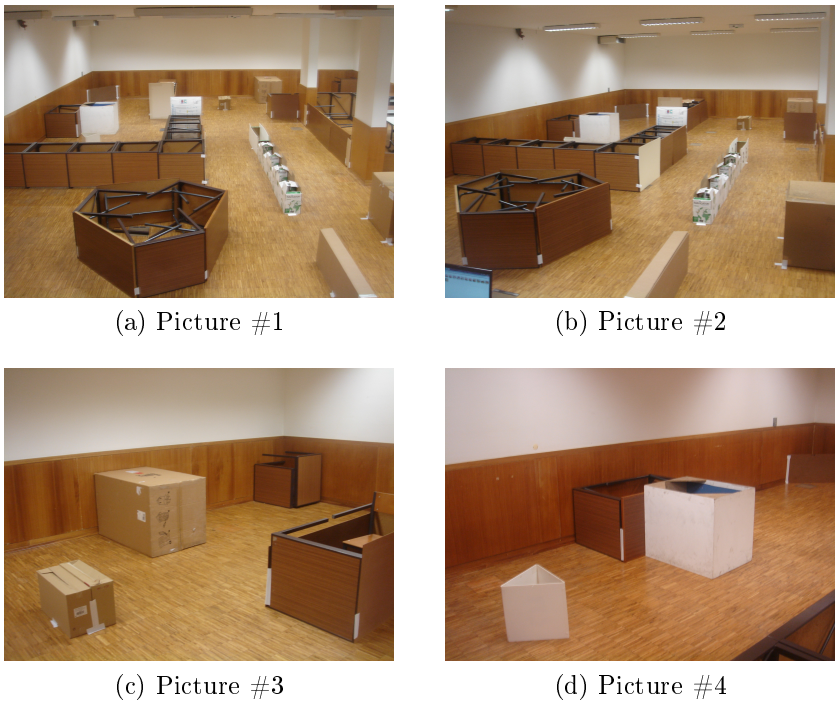


Figure 4.9: One of the robots used in the experiments.

The Pioneer 3-DX is a small lightweight differential wheeled robot ideal for indoor laboratory or classroom use. The robot is equipped with front sonar array and a Laser Hokuyo-URG-04lx-UG01[4]. This kind of research robot is one of the world's most popular mobile robots for education and research because of its versatility, reliability and durability. Pioneers are pre-assembled, customized and upgradeable. The robots are placed in the arena

[3]Pioneer 3-DX page. Available at: `http://mobilerobots.com/ResearchRobots/PioneerP3DX.aspx`
[4]Hokuyo page. Available at: `http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx.html`

of approximately 120 square meters (Figure 4.3) whose virtual model was used in the previous two simulation experiments. Some photos of such space, located at the Department of Electrical and Computer Engineering of the University of Coimbra, can be seen in Figure 4.10.



(a) Picture #1 (b) Picture #2

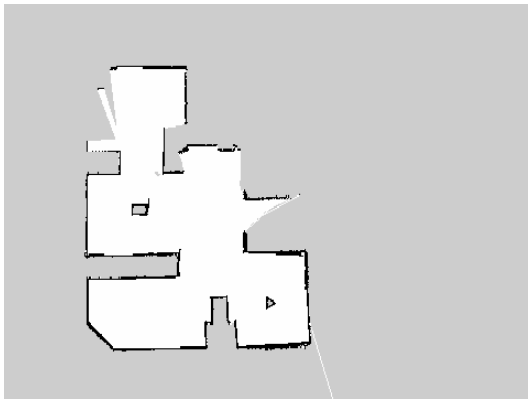
(c) Picture #3 (d) Picture #4

Figure 4.10: Photos of the real world arena used in the experimental results.

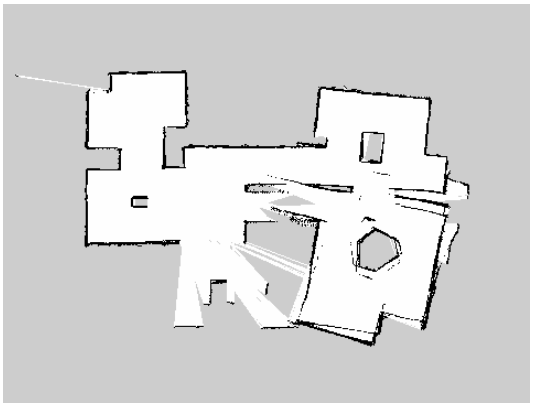
The goal of this experiment, similarly to the simulation, is to obtain a global map of the arena from two partial maps retrieved by the team of two Pioneers 3-DX. Note, however, that in this case, the laser range scans are subject to inherent noise of real world experiments. The partial maps have a resolution of 0.04 m/pixel and height/width of 544 cells each.

This experiment follows the previously described procedure, that is, the robots are teleoperated using the wii remote[5] and each sends its own map to the other, when the conditions are met, in order to generate a global representation. At a given point of a trial, robots trade partial maps, as exemplified in Figures 4.11a and 4.11b. Communication occurs when robots acknowledge each other. Upon exchanging information, a merging phase occurs generating the global map, as shown in Figure 4.11c. The global map has a resolution of 0.04 m/pixel and height/width of 572 cells each. The map from *robot_0* is acquired normally (see Figure 4.11a). As for the map acquired from *robot_1*, it is not, at least in a latter stage. We can observe that, as the robot explores, its self-location deteriorates. This is due to odometry error accumulation, a common problem in SLAM and MRSLAM, and the problem of distinguishing places that the robot has visited before, known by loop closure. Noise affecting the laser reading and/or the odometer, prevent the robot from connecting the map. This occurrence is naturally reflected in the final merged map, visible in Figure 4.11c.

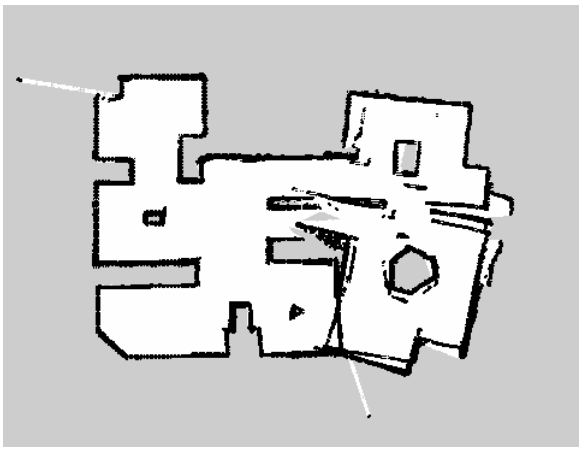
[5]Wiimote ROS package. Available at: `http://wiki.ros.org/wiimote`



(a) Partial map built by *robot_0*.



(b) Partial map built by *robot_1*.

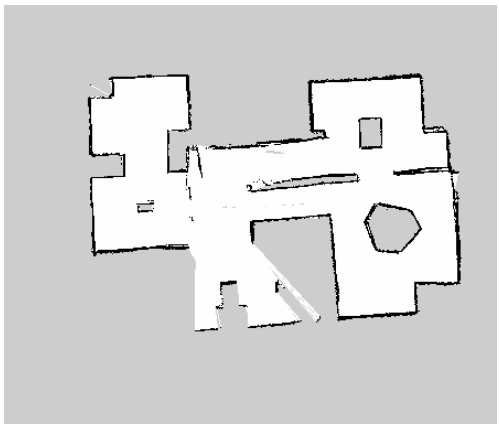


(c) Global map built by merging the partial maps from each robot.

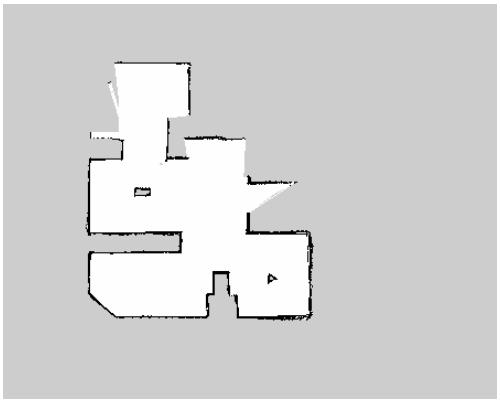
Figure 4.11: Real world test results with a team of two robots.

Apart from these issues affecting the robot's perception of the environment, experimental set-up complexity and problems also increase due to external factors: network interruptions; limited autonomy of the batteries supporting the logical unit controlling the robot and the robot itself; the task of teleoperating multiple robots simultaneously without extra human assistance; and other logistic related issues. These factors have decisively limited the realization of multiple experiments, preventing a more in depth and extended statistical result analysis, such as measuring the error on pose for mutual detection.

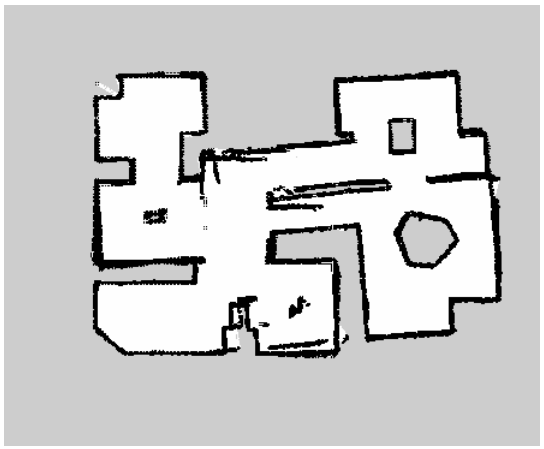
Some experiments have been successfully performed without any relevant technical issues, critically jeopardizing data acquisition. An example of those experiments follows. The robots are again teeoperated using the wii remote, and share information every 30 seconds, when link quality conditions are met, time at which merging occurs so to generate the global map. In this case, locally built, partial maps are represented by accurate data, as illustrated in Figures 4.12a and 4.12b. This local reconstruction accuracy propagates to the global map in Figure 4.12c, which has the same resolution as before, where height/width also have the same number of cells. Comparing results from Figures 4.11 and 4.12, allow concluding that, once real experiment complexity, hardware problems and issues like loop closure are dealt with, the proposed methods can present highly accurate maps and be effectively applied



(a) Partial map built by *robot_ 0*.



(b) Partial map built by *robot_ 1*.



(c) Global map built by merging the partial maps from each robot.

Figure 4.12: Real world test results with a team of two robots.

in map alignment and fusion for real applications. It is also noteworthy that a team of multiple robots exploring such scenario, is able to cooperatively map the environment in a much quicker way than a single robot, in the same conditions (robot speed, SLAM technique, etc.). As expected, technical issues have impaired our algorithm's ability when compared to flawless simulation performances. Nonetheless, these problems are recurrent in SLAM and MRSLAM approaches, for which we consider results to be extremely positive.

4.3 Summary

In this chapter, simulation and real world results to test the MRSLAM approach were presented and discussed. In the subsequent chapter, an overview of the dissertation is done. The algorithm's benefits and weaknesses are debated and finally this dissertation ends with conclusions and future work.

Chapter 5

Conclusion

Multi-Robot Simultaneous Localization and Mapping still has great potential for research and the work presented in this dissertation is one way of achieving the ultimate goal of obtaining a global map from the contributions of each robotic agent's map engaged in the task. This was done using a completely distributed approach, where each pair of robots exchange maps according to their geometric distance, communication quality and time since the last exchange of maps in order to merge locally their individual contributions.

In this last chapter, a global and self-critical overview of the proposed approach takes place in order to sum up the work. Also, final conclusions and possible related future work directions are discussed.

5.1 MRSLAM Approach overview

In recent years, several authors proposed distinct solutions for the SLAM problem. However, extensions for the multi-robot case are still lacking. MRSLAM is characterized by rigid communication limitations and the robots initial position being often known. In this dissertation, a distributed MRSLAM approach was presented, where the initial relative position are unknown and a RBPF-SLAM-based algorithm was adopted. A description of the implementation of alignment of occupancy grids, robot mutual detection and map merging in ROS was given, so as to create a global map from the respective partial maps supplied by each robot using communication based on a MANET to share relevant data.

Beyond the limited communication between the robotic agents, other challenges included the limited resources available such as data storage memory and computational effort. To clearly identify each robot and the data recovered by each one in ROS, a prefix was used to separate the namespaces. Local map alignment and merging depends on meetings between teammates in the environment when conditions to exchange maps are guaranteed to exist. During *rendezvous*, the robots exchange their maps and filter the detected teammate observations in the merged map. Each robot's partial map will be updated at *rendezvous* and, as time moves on, more information is gathered, thus we expect that several common cells will exist between partial maps, easening the alignment and map fusion. The alignment/fusion process optimization is based on the communication between robots: maps are exchanged when robots are close to each other, taking in account the link quality. In the context of the

CHOPIN project, the central command of operations (CCO) will receive global maps from the robot which is closer to it, in order to reduce communication burden.

The only assumptions of the algorithm are know IPs and robot IDs as well as teams of homogeneous robots with known physical dimension. Results in both simulation and real world experiments show that the maps obtained present high quality and are able to provide fine details of the environment, meaning that the methods adopted bring good perspectives in the future. The results show that it is possible to use a team of robots to explore and navigate in unknown environments.

There are, however, some limitations that were identified, such as maintaining consistency when robots start from very distant positions and also when there is a *rendezvous* with more than two robots. Since alignment is made in peer-to-peer fashion it becomes hard to deal with mutual detection for robots that are not included in the alignment task, which may cause a possible detection of these robots on the global map. It would be necessary to solve the Alignment problem for P robots instead of doing it in pairs, to cover all possible cases *wrt* the processing data in the mutual detection's module.

5.2 Future Work

Some issues are still left open and correspond to future guidelines that can be followed to improve current work. Concerning the context in which the work is inserted (project CHOPIN), the SLAM technique should be extended to deal with environments filled with smoke. Optimization of map exchanges using a compression method would also be important, to reduce the load of communication and time needed for information exchange between robots. Before data compression, would be important to optimize the information exchange, at the time of map merging (section 3.7). This optimization should signal the robot about which cells in the occupation grid map have new information since the last exchange, with respect to all robots in the team.

Another interesting feature would be to implement a module to align maps that did not depend on functions from OpenCV, in order to make the process faster, lighter and more importantly, compatible for P robots instead of being limited to pairs of robots. Finally, in terms of navigation and coordination of robots, it would be interesting to have an autonomous exploration technique for robots to explore the environment without depending on human teleoperation.

Bibliography

[Andersson and Nygårds, 2009] Andersson, L. and Nygårds, J. (2009). On multi-robot map fusion by inter-robot observations. In *In proceedings of 12th International Conference on Information Fusion*.

[Araújo, 2012] Araújo, A. (2012). Rosint - integration of a mobile robot in ros architecture. Master's thesis, University of Coimbra.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *In ECCV*, pages 404–417.

[Brown, 1992] Brown, L. G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24:325–376.

[Burgard et al., 2005] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.

[Carlone et al., 2010] Carlone, L., Ng, M. E. K., Du, J., Bona, B., and Indri, M. (2010). Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In *ICRA*, pages 243–249. IEEE.

[Chang et al., 2007] Chang, H. J., Lee, C. S. G., Hu, Y. C., and Lu, Y.-H. (2007). Multi-robot slam with topological/metric maps. In *IROS*, pages 1467–1472. IEEE.

[Couceiro et al., 2013] Couceiro, M. S., Portugal, D., and Rocha, R. P. (2013). A collective robotic architecture in search and rescue scenarios. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 64–69, New York, NY, USA. ACM.

[Dissanayake et al., 2001] Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-whyte, H. F., and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006.

[Elfes, 1990] Elfes, A. (1990). Autonomous robot vehicles. chapter Sonar-based real-world mapping and navigation, pages 233–249. Springer-Verlag New York, Inc., New York, NY, USA.

[Fenwick et al., 2002] Fenwick, J. W., Newman, P. M., and Leonard, J. J. (2002). Cooperative concurrent mapping and localization. In *ICRA*, pages 1810–1817. IEEE.

[Fox et al., 2006] Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Stewart, B. (2006). Distributed multi-robot exploration and mapping. In *In Proceedings of the IEEE*, page 2006.

[Grisetti et al., 2007] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23:34–46.

[Guivant and Nebot, 2001] Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257.

[Hähnel et al., 2003] Hähnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *IROS*, pages 206–211. IEEE.

[Howard, 2006] Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *I. J. Robotic Res.*, 25(12):1243–1256.

[León et al., 2008] León, A., Barea, R., Bergasa, L. M., López, E., Ocaña, M., and Schleicher, D. (2008). Multi-robot SLAM and Map Merging. In *IX Workshop of Physical Agents (WAF 08)*, pages 171–176.

[Leung et al., 2012] Leung, K. Y. K., Barfoot, T. D., and Liu, H. H. T. (2012). Decentralized cooperative slam for sparsely-communicating robot networks: A centralized-equivalent approach. *Journal of Intelligent and Robotic Systems*, 66(3):321–342.

[Martins, 2013] Martins, J. (2013). Multi-robot simultaneous localization and mapping related issues. Technical report, Faculty of Science and Technology, University of Coimbra, Portugal.

[Montemerlo and Thrun, 2003] Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastslam. In *ICRA*, pages 1985–1991. IEEE.

[Montemerlo et al., 2003] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In Gottlob, G. and Walsh, T., editors, *IJCAI*, pages 1151–1156. Morgan Kaufmann.

[Portugal, 2009] Portugal, D. (2009). Robocops: A study of coordination algorithms for autonomous mobile robots in patrolling missions. Master's thesis, University of Coimbra.

[Portugal and Rocha, 2012] Portugal, D. and Rocha, R. P. (2012). Extracting topological information from grid maps for robot navigation. In Filipe, J. and Fred, A. L. N., editors, *ICAART (1)*, pages 137–143.

[Portugal and Rocha, 2013] Portugal, D. and Rocha, R. P. (2013). Distributed multi-robot patrol: A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*. (In Press).

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

[Rocha, 2006] Rocha, R. P. (2006). *Building Volumetric Maps with Cooperative Mobile Robots and Useful Information Sharing - A Distributed Control Approach based on Entropy*. PhD thesis, Faculdade de Engenharia do Porto,.

[Rybski et al., 2004] Rybski, P. E., Larson, A., Veeraraghavan, H., Lapoint, M., and Gini, M. (2004). Communication strategies in multi-robot search and retrieval: Experiences with mindart. In *Proc. 7th Int. Symp. Distributed Autonomous Robotic Systems*, pages 301–310.

[Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). Autonomous robot vehicles. chapter Estimating uncertain spatial relationships in robotics, pages 167–193. Springer-Verlag New York, Inc., New York, NY, USA.

[Thrun, 1997] Thrun, S. (1997). Learning maps for indoor mobile robot navigation.

[Thrun, 2002] Thrun, S. (2002). Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.

[Thrun and Liu, 2003] Thrun, S. and Liu, Y. (2003). Multi-robot slam with sparse extended information filers. In Dario, P. and Chatila, R., editors, *ISRR*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 254–266. Springer.

[Zhang et al., 2009] Zhang, L., jiong Meng, X., and wu Chen, Y. (2009). A fastslam algorithm based on the auxiliary particle filter with stirling interpolation. In *Information and Automation, 2009. ICIA '09. International Conference on*, pages 167–172.

[Zhou and Roumeliotis, 2006] Zhou, X. S. and Roumeliotis, S. I. (2006). Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *IROS*, pages 1785–1792. IEEE.