João Pedro Machado dos Santos

SmokeNav – Simultaneous Localization and Mapping in Reduced Visibility Scenarios

September 2013

· U C ·

Universidade de Coimbra

FCTUC

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

SmokeNav - Simultaneous Localization and Mapping in Reduced Visibility Scenarios

João Pedro Machado dos Santos

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

Coimbra, 2013

FCTUC

SmokeNav - Simultaneous Localization and Mapping in Reduced Visibility Scenarios

Supervisor:

Prof. Doutor Rui P. Rocha

Co-Supervisor:

Eng. David Portugal

Jury:

Prof. Doutor João Pedro de Almeida Barreto

Prof. Doutor José Nuno Panelas Nunes Lau

Prof. Doutor Nuno Miguel Mendonça da Silva Gonçalves

Prof. Doutor Rui P. Rocha

Report written for the dissertation subject, included in the Electrical and Computer Engineering Course, submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, 2013

Acknowledgments

During this journey, there were several people who fully supported me. All of them, contributed in different ways, to somehow make this possible. Therefore, I would like to thank them for their important and gracious support. First of all, I would like to thank my supervisor, Prof. Rui Rocha, for his guidance and for believing in my capacities at all times, encouraging me to achieve better and better results. Hence, I am extremely grateful for all the opportunities that he presented me and I want to point out that his availability to discuss and help me in different subjects of my work were remarkably crucial. I would like to thank my co-supervisor and friend, Eng. David Portugal, who had an important role in my academic journey this past year. With his assistance, not only did I improve my skills, but I was also able to learn how to face possible obstacles. A special thanks to my parents and sister, who always help and guide me, offering me unconditional support. I would like to thank Marta, Ana and Teresa for all the embarrassing moments they made me go through and for making me laugh even when I had "dores de costas". I am very thankful to my Lab partner, João Martins, and very proud to be able to call him friend. Besides, the constant support, he made every stressful moment much funnier and bearable. I would venture to say that he is one of the best Lab partners one may have. I would like to thank Luís Santos for all the "little John" comments, without which this work would not be the same. I would also like to mention Micael Couceiro, José Pereira, Gonçalo Augusto, Pedro Machado, Diego Faria, Ricardo Martins, Pedro Trindade and Jorge Sales for all the moments we shared in the Lab. Last but not least, I would like to thanks Emanuel Marques, Renato Baptista, João Amaro, Andrea Pinto, Rui Neto, João Faro and all my other colleagues for all the support and special memories.

Abstract

Simultaneous Localization and Mapping (SLAM) is one of the most widely researched topics in Robotics. It addresses building and maintaining maps within unknown environments, while the robot keeps the information about its location. It is a basic requirement for autonomous mobile robotic navigation in many scenarios, including military applications, search and rescue, environmental monitoring, etc. Although SLAM techniques have evolved considerably in the last years, there are many situations which are not easily handled, such as the case of smoky environments where commonly used range sensors for SLAM, like Laser Range Finders (LRF) and cameras, are highly disturbed by noise induced in the measurement process by particles of smoke. There is an evident lack of solutions to this issue in the literature. This work focuses on SLAM techniques for reduced visibility scenarios.

The main objective of this work is to develop and validate a SLAM technique for those scenarios, using dissimilar range sensors and by evaluating their behavior in such conditions. To that end, a study of several laser-based 2D SLAM techniques available in Robot Operating System (ROS) is firstly conducted. All the tested approaches are evaluated and compared in 2D simulations as well as real world experiments using a mobile robot. Such analysis is fundamental to decide which technique to adopt according to the final application of the work. The developed technique uses the complementary characteristics between a LRF and an array of sonars in order to successfully map the aforementioned environments. In order to validate the developed technique , several experimental tests were conducted using a real scenario. It was verified that this approach is adequate to decrease the impact of smoke particles in the mapping task. However, due to hardware limitations, the resulting map is comprehensibly not outstanding, but much better than using a single range sensor modality. This work is part of the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN) R&D project, which intends to develop a support system for small scale SaR missions in urban catastrophic scenarios by exploiting the human-robot symbiosis.

Keywords: SLAM, mobile robot, ROS, smoke, visibility, LRF, sonar.

Resumo

Localização e mapeamento simultâneo, também conhecido por SLAM (*Simultaneous Localization and Mapping*), é uma das áreas mais investigadas em Robótica. Aborda o problema da construção e actualização de mapas em ambientes desconhecidos, enquanto o robô mantém a informação relativa à sua posição. SLAM é um requisito básico para robôs móveis autónomos nas mais variadas situações, que podem ir desde aplicações militares, busca e salvamento, exploração subaquática ou até monitorização ambiental. Apesar das técnicas de SLAM terem evoluído consideravelmente nos últimos anos, existem inúmeras situações em que os resultados ficam aquém do esperado, como no caso de ambientes com fumo, onde os sensores de distância geralmente usados, tais como *Laser Range Finders* (LRFs) ou câmeras, falham. Estes sensores são fortemente afectados pelas partículas de fumo durante o processo de aquisição de dados. Analisando a literatura, torna-se evidente que existem poucas abordagens para este problema.

Este trabalho foca-se em técnicas de SLAM para cenários de visibilidade reduzida. O grande objectivo é desenvolver e validar uma técnica de SLAM para este tipo de cenários usando sensores de distância complementares. Para tal, começa-se por apresentar um estudo de diversos algoritmos de SLAM 2D para o *Robot Operating System* (ROS). Todas estas técnicas são avaliadas e comparadas através de simulações 2D e experiências reais usando um robô móvel. Esta análise é fundamental para decidir qual a técnica que servirá de base de acordo com a aplicação pretendida. A técnica desenvolvida faz uso das características complementares do laser e dos sensores de ultrassons (*i.e.*, sonares), de modo a que seja possível fazer mapeamento em condições de visibilidade reduzida. A sua validação é feita atráves de várias experiências num cenário real. Os resultados mostram que através desta abordagem é possível diminuir o impacto do ruído provocado pelas partículas de fumo nos sensores e realizar a tarefa de mapeamento com sucesso. No entanto, a qualidade dos mapas gerados fica compreensivamente aquém do que se possa esperar devido a limitações do hardware.

Palavras-Chave: SLAM, robô móvel, ROS, fumo, visibilidade reduzida, laser, sonar.

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

Contents

List of Acronyms

API	Application Programming Interface
CHOPIN	Cooperation between Human and rObotic teams in catastroPhic INcidents
CPU	Central Processing Unit
EKF	Extended Kalman Filter
EM	Expectation Maximization
FOV	Field of View
IMU	Inertial Measurement Unit
ISR	Institute of Systems and Robotics
KF	Kalman Filter
LED	Light-Emitting Diode
LIDAR	Laser Interferometry Detection and Ranging
LRF	Laser Range Finder
MCL	Monte Carlo Localization
MRL	Mobile Robotics Laboratory
MRS	Multi-Robot System
PF	Particle Filter
POI	Point of Interest
RAM	Random-Access Memory
RBPF	Rao-Blackwellized Particle Filter

ROS Robot Operating System

SaR Search and Rescue

SIS Sequence Importance Sampling

SLAM Simultaneous Localization and Mapping

SNR Signal-to-Noise Ratio

SPA Sparse Pose Adjustment

TDoA Sonar Time Difference of Arrival

TIC Thermal Imaging Camera

TOF Time of Flight

UAV Unmanned Aerial Vehicle

UWB Ultra-Wide Band

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation describes the work that was done during the "SmokeNav: SLAM in Reduced Visibility Scenarios" dissertation project, which took place in the Mobile Robotics Laboratory (MRL) of Institute of Systems and Robotics (ISR) in Coimbra. The main goals of this master degree project are the study of several laser-based 2D Simultaneous Localization and Mapping (SLAM) techniques available in Robot Operating System (ROS), as well as the development of a new approach for 2D SLAM in low visibility scenarios due to smoke particles.

There were three main work phases. In the beginning, SLAM and some of the most important aspects of mobile robots were studied in detail. The second phase consisted in the study and evaluation of several laser-based 2D SLAM techniques available in ROS, the chosen framework in this project. This was performed through 2D simulations and real world experiments. The study conducted provides an insight on the weaknesses and strengths of each technique. Such analysis is fundamental to decide which solution to adopt according to the properties of the intended final application. Afterwards, a new approach for 2D SLAM which can cope with smoke/dust particles was developed. Several tests were carried in order to analyze the performance of the developed approach.

This chapter presents not only the context and motivation, but also the main goals of this dissertation and how its organized.

1.1 Context and Motivation

SLAM has been one of the most studied subjects in Robotics. It is a fundamental process in mobile robotics, since it is responsible for building maps and at same time estimate the robot position in the environment. SLAM is essential for autonomous mobile robots to accomplish useful tasks with no *a priori* information about the environment. Due to its

importance, the number of SLAM approaches increased significantly in the last years. There are many approaches to the SLAM problem and each of them focuses on a particular issue.

Proprioceptive sensors are subject to cumulative errors when estimating the mobile robot's motion and odometry alone is insufficient to determine accurately the robot's pose. The high dimensionality of the environment that is being mapped, the problem of determining whether sensor measurements taken at different points in time correspond to the same object in the world, and the fact that the world changes over time, represent the biggest challenges in SLAM [TBF05].

Despite of the evolution of SLAM in the last years, there are still many open challenges, such as the case of localization and mapping of smoky or foggy environments, where commonly used range sensors for SLAM, like the commonly used Laser Range Finder (LRF), are highly disturbed by noise induced in the measurement process by particles of smoke, steam or dust. The presence of smoke or dust in the environment may obscure partially (or totally) the environment.

This dissertation studies are part of the Cooperation between Human and rObotic teams in catastroPhic INcidents (CHOPIN)[1] R&D project. The focus of the CHOPIN project is the application of Multi-Robot Systems (MRSs) in urban Search and Rescue (SaR) missions [CPR13] in order to assist first responders in scenarios of eminent danger. The greatest advantage of such systems over a single robot is the efficiency, parallelism and robustness in distributed tasks [Roc05], possibly safeguarding human lives in hazardous scenarios. Additionally, MRS can provide rich and accurate environmental information, since sensors are distributed through the scenario. The project also intends to exploit the human-robot symbiosis, in order to achieve a support system for small scale SaR missions in urban catastrophic scenarios. However, these catastrophic scenarios are usually associated with low visibility environments which drastically decrease the progress of human rescuing forces and the accuracy of the robot sensorial system, thus compromising the mobile robot's localization system.

There are already several solutions to the SLAM problem and each has its own advantages and disadvantages. The suitability of a given technique is highly related to the application in hand. Moreover, mobile robotics is a trendy field and its techniques are constantly evolving, therefore it is foreseeable in the short- and mid-term future that mobile robots will assist and even replace human operators in dangerous, monotonous or undesirable tasks, which are still performed by humans nowadays [PR13]. This is the case of SaR missions, which take

[1] `http://chopin.isr.uc.pt/`

place in extreme conditions and pose very difficult challenges including sometimes reduced visibility. The applicability of SLAM methods in these situations is scarce. Hereupon, it is necessary to develop new techniques.

Robot Operating System (ROS) is the most popular robotics framework nowadays. It is free and open-source, providing a set of tools, libraries and drivers in order to help develop robot applications by taking advantage of hardware abstraction [QCG+09].

A major problem inherent to the explosive growth in robotics is the impossibility to reuse the code in different robotic platforms. However, the hardware abstraction layer in ROS along with its messages service allows the creation of new code that can be used in many different robotic platforms. For these reasons, the development of the proposed SLAM technique for reduced visibility scenarios has been conducted in ROS.

1.2 Objectives

The first objective of the present work is to conduct a comparative evaluation of several SLAM approaches available in ROS through 2D simulations and real world experiments. Secondly, the development of a new technique to solve the SLAM problem in smoky/foggy environments with reduced visibility, like those considered in the CHOPIN R&D Project. The majority of SLAM solutions only use one type of range sensors, however in this particular case, sensor fusion will be a major part of the solution. In this case, information from several sensors, such as sonars and LRFs is combined to improve upon existing SLAM techniques, in these extreme conditions.

1.3 The approach followed in this dissertation

This work addresses the problem of successfully mapping environments with reduced visibility conditions. Almost any SLAM technique requires a "clean" environment, where the range sensor, which is usually based on light propagation (e.g., LRF, RGBD camera, stereo-vision, etc.) is not strongly affected by smoke/dust particles. Usually, SLAM techniques use the sensorial data of only one type of range sensor, which in turn, increases the probability of failures in the system in a real scenario.

Clearly, the need to develop techniques based on a multi-sensor system for reduced visibility environment emerges. Using the output of several sensors with different characteristics, provides increased reliability. Thus, by studying the behavior of the sensors in such con-

ditions and using them in order to complement each other, it is possible to improve the effectiveness of the SLAM technique.

In this dissertation, the chosen approach is based on the opposite features of different range sensing technologies, aiming to take advantage of the best option according to the environment conditions. The main propose of this multi-sensor approach is to successfully map environments in the previously described conditions.

1.4 Outline of the dissertation

This document is organized in different sections, each referring to distinct phases of the work developed. The first chapter introduces the context and motivation, and the objectives of this work.

Chapter 2 introduces SLAM as well as an overview of the most important SLAM variants, such as Particle Filter (PF), Extended Kalman Filter (EKF) and graph-based approaches. Also, a short review of commonly used map structures is presented.

Chapter 3 outlines the state of the art of SLAM in scenarios where the visibility is reduced, such as in smoky environments.

An evaluation of several 2D laser-based SLAM techniques available in ROS is carried out in chapter 4. A simple error metric is developed in order to correctly compare all the approaches. Also, the results of simulation and real-world experiments are discussed. Additionally, some useful tools available in ROS are presented.

Chapter 5 describes the implementation of the proposed SLAM for reduced visibility scenarios and the results of the experiments carried out are discussed.

Finally, Chapter 6 sums up the work, providing final conclusions and directions for future research.

Chapter 2

Simultaneous Localization and Mapping

In this chapter, an overview of some of the most commonly used map representations in mobile robots is presented. Also, an overview of several 2D laser-based SLAM algorithms. This study reveals the basis of SLAM and introduces some fundamental aspects to the development of the intended SLAM technique in this dissertation.

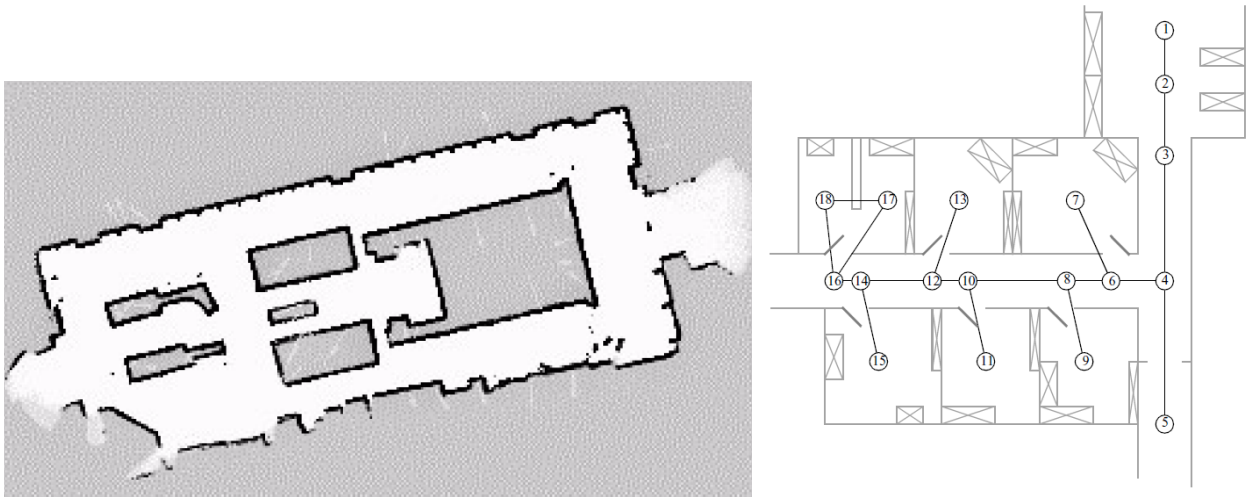
2.1 Map Representations

This section focuses on two distinct map representations used in mobile robots. Since the main goal of SLAM algorithms is to build maps of unknown environments while, at the same time, tracking the robot's position, it is important to study map structures, which techniques are used, their advantages, disadvantages and applications.

In the SLAM problem, two popular types of maps can generally be maintained by mobile robots: metric maps and topological maps [TB98]. There are also hybrid maps which are a mixture of both. Metric maps are built based on an absolute reference frame and numerical estimates of object's position in the environment. The robotic platform can build its own metric map from sensorial information or the map can be preloaded. This approach is extremely simple and the resulting map can be easily interpreted by humans [DJ10]. However, the process of building the metric map may become slow due to its detailed nature which leads to high dimensionality in larger environments.

Occupancy grids are the most dominant type of metric maps. In these maps, the environment is represented by a matrix of cells. Each cell characterizes a small area in the environment and represents the probability of that area being occupied [Roc05]. In these grids, both occupied, unknown and free space are represented. An example is show in Fig. 2.1a.

Topological maps represent environments by means of graphs. In such graphs, the nodes



(a) Occupandy Grid (b) Topological Map

Figure 2.1: Examples of Map Representations.

correspond to distinct places or landmarks [TB98]. Nodes are connected by edges if there is a direct connection between them, as shown in Fig. 2.1b. This is method is very effective in large scale environments, where the number of landmarks is large. As mentioned before, metric maps require much more memory if the environment is larger. The topological representation is much more compact than an array and this fact allows fast trajectory planning. However, there are several disadvantages as shown in Table 2.1. For instance, with this solution, it is difficult to distinguish different places that look alike. According with Thrun [TB98], topological approaches may not recognize geometrically places, since the sensor data depends strongly on the viewpoint of the robot.

Thrun [Thr02] have proposed a new algorithm for obtaining occupancy grid maps that solves inconsistencies between the generated map and sensorial data in cluttered environments. The proposed solution consists in the adoption of the Expectation Maximization (EM) algorithm in the map estimation process. Most of known algorithms decompose the high-dimensional mapping problem into many one-dimensional mapping sub-problem. This approach can lead to estimation issues, such as the case when the robot is equipped with sonars and passes by a door. According to the author, the map is obtained by gradually maximizing the likelihood of all measurements in a hill-climbing fashion. The map is more sensitive to changes in the environment and the computational cost is higher, but generated maps are fairly more consistent.

Portugal *et al.* [PR12] proposed a method of obtaining a global topological map from a preexisting metric representation of the environment. This approach is based on image processing techniques and consists in acquiring the skeleton from occupancy grids with some

Table 2.1: Characteristics of Metric and Topological Maps.

Type	Characteristics
Metric	• Metric maps • Easy to build • Easy to represent and to maintain • Higher computational cost • Memory problems (depending on the size of maps) • Recognition of spaces is non-ambiguous • Theoretically equal for any robot
Topological	• Graph-like maps • Efficient use • Simpler than Metric maps • Does not require exact position of robot • Vertices correspond to landmarks • Vertices connected by edges (represent path between them) • Harder to recognize and maintain consistency in large scale environments • Recognition of places often ambiguous • May differ according to the sensing range of the robot that builds the map

improvements. The algorithm was tested using a large number of maps with different morphologies and complexities. There are many advantages, like the simplicity, accuracy and performance, but the dependency on a correct parameterization of the Voronoi thinning process (EVG-THIN method) is a disadvantage.

In [TB98], the author present a hybrid approach to integrate these two paradigms. Focus is given to the way maps are built. Grid-based maps are learned using neural networks and adopting Bayes rule, and topological maps are generated by partitioning the grid-based map into critical regions. This method has exclusive advantages such as accuracy, consistency and efficiency.

2.2 Simultaneous Localization and Mapping

Presently, all recognized algorithms for robot mapping have a common feature: they rely on probabilities. The estimation of maps by robotic platforms implies gathering and then analyzing information from the sensorial system. The advantage of applying probabilities in that process is the robustness to measurement noise and the ability to formally represent uncertainty in the measurement and estimation process. Most of the probabilistic models used to solve the problem of mapping rely on Bayes rule [TBF05].

In mobile robotics, we need to deal with two types of data over time: sensors measurements and motion feedback. The extension of Bayes rule to the time dimension is called Bayes filter. Bayes filter is a recursive estimator, because it determines the posterior probability at time t using as input the probability calculated one step earlier at time $t-1$.

In robot mapping, there are two quantities that influence the sensors measurements z_t: (i) the map and (ii) the robot's pose in the environment at the time of measurement. Both map and pose need to be estimated at the same time, therefore the state vector x_t will be $x_t = [s_t, m_t]$, where s_t denotes the robot's pose and m_t denotes the map [TBF05]. Most common algorithms assume that the world is static and that robot motion is independent of the map. Consequently, the Bayes filter applied to robot mapping is calculated through equation 2.1.

$$p(s_t, m|z^t, u^t) = \eta p(z_t|s_t, m) \int p(s_t|u_t, s_{t-1}) p(s_{t-1}, m|z^{t-1}, u^{t-1}) \, , \mathrm{d}s_{t-1}. \tag{2.1}$$

In equation 2.1, z_t denotes the sensors measurements and u_t denotes the robot motion estimated by odometry of the mobile platform up to the instant time t.

2.2.1 Kalman filter-based SLAM

Kalman filters (KF) are one of the most popular implementations of Bayes filters [TBF05]. The KF assumes that probability density functions are Gaussian distributions. The KF has two distinct phases: Prediction and Update. The prediction phase estimates the state space (prior) from a previous iteration, while in the update phase the estimated state is combined with observations provided by sensors. The result from the update phase is called posterior. The prediction phase of the KF algorithm is described by:

$$\bar{\mu}_t = A_{t-1}\mu_{t-1} + B_t u_t. \tag{2.2}$$

$$\Sigma_t = A_t \Sigma_{t-1} A_t^T + R_t \tag{2.3}$$

while the update phase is described by:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}. \tag{2.4}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t). \tag{2.5}$$

$$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t. \tag{2.6}$$

Vector μ is the state vector, Σ is the covariance matrix and K is the Kalman gain. These equations are equivalent to the standard KF. Arising from the prior development of the

KF, the Extended Kalman Filter (EKF) solves the problem of nonlinearity in the robot pose model. Most of the computational cost stems from to matrix multiplications, which can be implemented in $O(k^2)$ time, where k is the number of features in the map [TBF05]. However, the number of features is not known *a priori*. A list of candidate features is required to implement EKF using a separate KF for the candidates. The list grows dynamically.

A set of tests on convergence properties and inconsistency issues of the EKF-based solution to the nonlinear 2D SLAM problem is conducted in [HD07]. There are several approaches to the EKF in the literature and each author deals with different issues. When the statistical models are well known, the EKF algorithm is a solid solution, however this situation does not happen most of the time. In some cases, Gaussian distributions does not allow to correctly model random variables (*e.g.*, noise), especially in the case of multi-modal variables. Chatterjee *et al.* [CM07] propose to solve this problem with the assistance of a neuro-fuzzy system that attempts to minimize the discrepancy between theoretical and practical values in EKF. The results obtained from conventional EKF-based SLAM showed unreliable performance in many situations, while the scheme proposed improved the performance of EKF.

2.2.2 SLAM based on Monte Carlo Methods

Particle filters (PF) are another application of Bayes filters. They are a broad and popular class of Monte Carlo algorithms. Nevertheless, the approach to solve the problem is very different from the KF/EKF.

The posterior probability is represented by a set of weighted particles and each particle is given an importance factor. It assumes that the next state depends only on the current one and not on the sequence of events that preceded it *i.e.*, Markov assumption [TFBD00]. PFs have the advantage of covering some limitations of the EKF, representing uncertainty through multi-modal distributions and dealing with non-Gaussian noise, however they use more computational resources. This family of algorithms can build high resolution grid-based maps, but the quality of the map depends on the number of particles used.

There are two fundamental stages in the PF algorithm: Sequence Importance Sampling (SIS) and Re-sampling [Pir05]. SIS consists of assigning the importance factor to a set of particles based on a probability density function. Re-sampling eliminates particles which have a very small importance and give more importance to the particles with greater weight. In the end, only the particles with higher beliefs are propagated [Pir05]. The evolution of

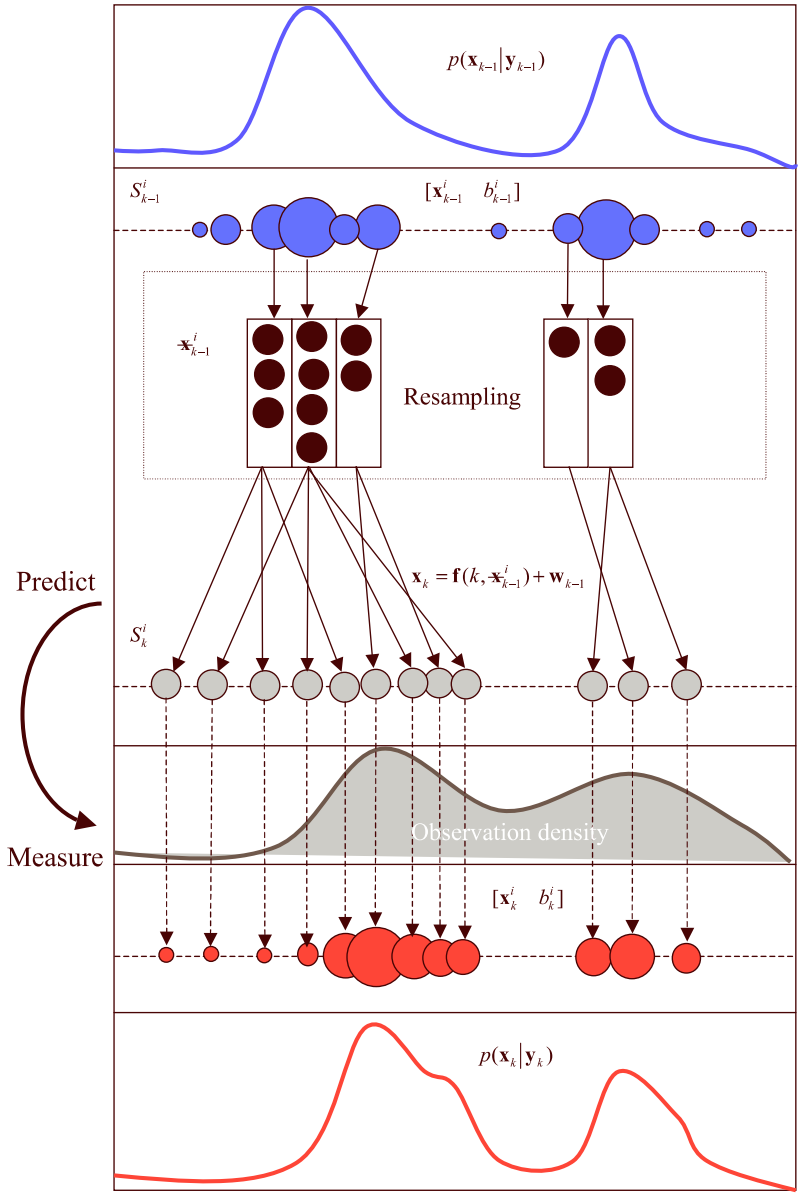


Figure 2.2: Evolution of particles in a PF [CZR05].

particles in a PF is shown in Fig. 2.2.

Montemerlo *et al.* [MTKW02] proposed a new approach called FastSLAM. It makes use of a modified PF to estimate the posterior. Afterwards, each particle possesses K Kalman filters that estimate the K landmark locations. It was shown that the computational effort to execute this algorithm is lower than EKF approaches. Also, the approach deals with large number of landmarks even with small sets of particles and the results remain appropriate. Inspired by the work of Montemerlo, an approach based on Rao-Blackwellized Particle Filter is proposed in [GSB07]. This work is discussed in more detail in Section 4.2.2.

Hähnel *et al.* [HBFT03] proposed a novel algorithm that combines a PF with scan matching. Scan matching is used to reduce the errors from the odometry system. Also, a prob-

abilistic model of the scan matching errors is used in the resampling phase of the PFs, decreasing the number of particles required.

2.2.3 Graph-based SLAM

Equally important are graph-based SLAM algorithms, as they cover some weaknesses of PFs and EKFs techniques [TM05]. Additionally, their relevance has grown considerably in the last few years. In graph-based SLAM, the data extracted is used to build a graph. The graph is composed by nodes and arcs. Each arc in the graph represents a constraint between successive poses, which can be a motion event or a measurement event. In Fig. 2.3 a pose-graph is presented. In this case, each node represents a robot pose and edges model spatial constraints between robot poses. In order to obtain a map, all the constraints are linearized and a sparse matrix is obtained, representing the sparse graph. The optimization process is fundamental in graph-based SLAM approaches.

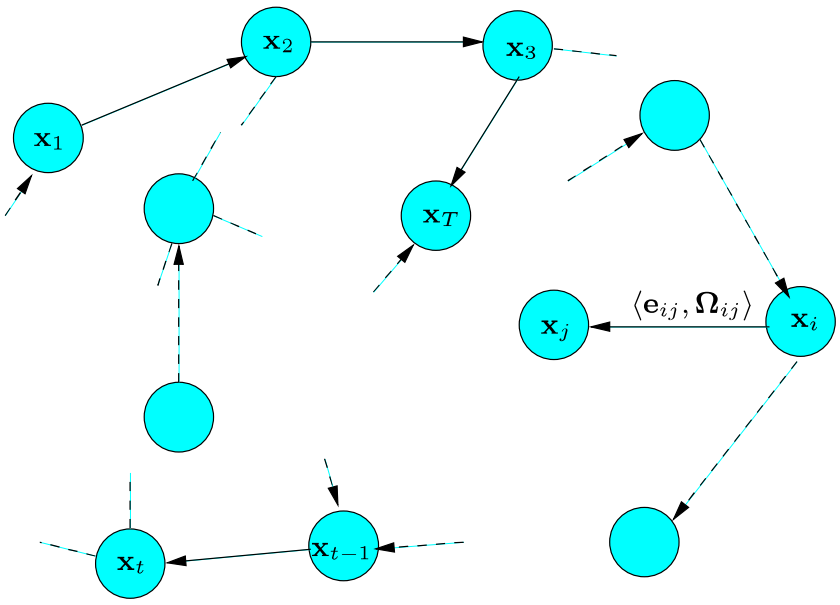


Figure 2.3: A pose-graph representation of a SLAM process [GKSB10].

This type of algorithms were first presented by Lu and Milios [LM97]. In their work, pose constraints were retrieved by the scan matching process. However, due to the high computational cost of the optimization process used, the applicability of the algorithm in large scenarios is impracticable. Thrun *et al.* [TM05] presented the GraphSLAM algorithm, which is based on [LM97], and evaluated its behavior in large-scale urban environments. This has been possible due to a reduction process, which removes map variables from the

optimization process. More recently, Carlone *et al.* [CACB11] also developed a graph-based SLAM approach, which is discussed in Section 4.2.3.

2.3 Loop closure

Most of the approaches for SLAM manage thousands of landmarks. However, the most of them only estimate the position in the environment without taking into account the trajectory. In addition, during the mapping process, the robot may visit a place where it has been before. The robot needs a spatial recognition system to recognize a place that it has seen before, a problem that is known in literature as the loop closure problem [NH05]. Finding loop closing constraints is fundamental to improve the estimate of the robot's pose in the environment, as well as the precision of the model being built. Also, no matter how robust a place recognition system might be, there always exists the possibility of getting a false positive loop closing constraint.

Latif et al. [LCN12] propose an incremental algorithm to detect whether the place recognition system has generated wrong constraints, removing them if necessary. This approach is based on the observation that correct loop closure in conjunction with odometry can help in the detection of wrong loop closures. The estimation process is used to make distinction between correct and false loop closures. This method was capable to generate the best estimate of multiple maps with all information available in the moment. This was verified through simulation experiments with the real data collected at the Intel Research Laboratory.

Valencia et al. [VMDAC12] have another work in this area. Their approach selects the appropriate actions to drive the robot to maximize coverage while, at the same time, minimizing localization and map uncertainties. To guarantee the coverage condition, an occupancy grid of the environment is maintained. In this algorithm, there are two types of actions: exploratory actions and place revisiting actions. The selection of loop closure candidates takes into account the path uncertainty. The presence of significant changes in the state estimate leads to an interruption in the execution of large loop closure trajectories and triggers the replanning process. Once again, this approach was tested within a cave-like two-dimensional environment through a simulator.

Chapter 3

SLAM in Harsh Scenarios

Chapter 2 provided an overview of SLAM techniques focusing on advantages, characteristics and types of representation. However, it is important to address sensor technologies, especially range sensors, which can be used in a mobile robotic platform in order to build a map of the environment. Since the main goal of this dissertation is to implement a solution for SLAM in smoky environments, the correct choice of sensors is crucial. Hence, this chapter reviews different ranging sensor technologies and some of the most relevant related work in low visibility.

3.1 Related Work in Scenarios with Reduced Visibility

SLAM techniques have evolved considerably in the last years, but most approaches assume clean environments. When the scenario has low visibility, the majority of SLAM algorithms fails or presents unsatisfactory results. The presence of smoke or dust in the environment represents a challenge for SLAM algorithms, since they can partially or totally obscure the environment to commonly used LRFs and Time of Flight (TOF)[1] cameras.

Brunner *et al.* [BPVC12] proposed a SLAM approach robust to smoke based on different sensing capabilities. This SLAM technique relies on the combination of the information from visual cameras (RGB cameras) and Thermal Imaging Cameras (TICs). The fundamental idea behind this approach is to counterbalance the limitations of the visual camera in the presence of smoke with the robustness of a TIC in these situations. However, the TIC is less effective than visual cameras in clean environments. Visual cameras provide richer data, images with higher resolutions and provide higher Signal-to-Noise Ratio (SNR) than TICs. On the other hand, TICs have a better performance in the presence of smoke, due to much lower attenuation of infrared waves compared to visible light waves in these conditions.

[1]TOF camera is a range imaging camera system that measures distances based on the time of flight of a light signal.

(a) Shrimp in smoke conditions. (b) Sensor Frame.

Figure 3.1: Experimental Setup [BPVC12].

The sensor modality is chosen based on a prior evaluation of the sensor data quality and the Spatial Entropy is used to evaluate the visual data obtained. There are two different tests: Absolute Metric Value and the Evolution of an Individual Metric Value. The algorithm is able to choose between three different modalities: only use the visual camera, only use the TIC and a combination of both sensors. However, the latter one can be tricky, since the rate of the data acquired from the TIC is lower than the rate from the visual camera.

Experiments were performed using a robot called Shrimp[2] equipped with a Raytheon Thermal-Eye 2000B IR camera and a Point Grey Bumblebee XB3 camera set. Smoke was generated using a JEMZR22 smoke machine with Jem Pro-Smoke Super Fluid fog/smoke juice. Data was acquired through the teleoperation of the robot in specified trajectories (straight lines, circles), a larger trajectory in clean conditions and in the presence of different levels of smoke. The experimental setup is depicted in Fig. 3.1.

Trajectories were estimated using four different combinations of sensor data for each data set acquired. The authors concluded that a reasonable accurate algorithm can be obtained, but the localization accuracy decreases in the presence of smoke even if one sensor remains unaffected by smoke. This is, in fact, confirmed by the results in this dissertation (Section 5.3). Also, the results can be improved by evaluating data quality prior to its integration and rejecting corrupted images. Fig. 3.2 shows a comparison between visual and thermal cameras in smoky and clean conditions.

Deissler et al. [DT10] presented a SLAM algorithm based on a Ultra-Wide Band (UWB) radar with a bat-type antenna array consisting of two RX antennas and one TX antenna in the middle. This algorithm was developed for catastrophic scenarios, where the environment

[2] `http://www.acfr.usyd.edu.au/research/agriculture.shtml`

Figure 3.2: Representative images in clear conditions (left column) and in smoke (right) from the visual (top row) and thermal (bottom) cameras [BPVC12].

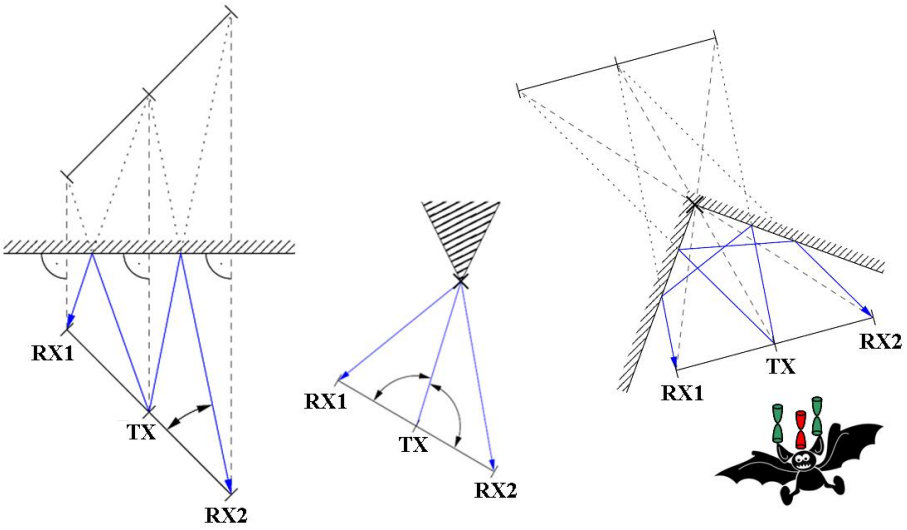


Figure 3.3: Geometric propagation path models for walls (left), edges (middle) and corners (right) [DT10].

is corrupted with smoke or dust. The UWB radar can provide additional information like life signs of humans, material characteristics and information about objects inside the walls. Since it is a radar-based approach, the smoke/dust particles in the environment do not affect this algorithm. The biggest challenge is data association, *i.e.* assigning the time of flight of a given measurement from the radar to the corresponding landmark [DT10]. This situation is solved using a Rao-Blackwellized Particle Filter (RBPF). A two-dimensional geometrical algorithm is used to determine the features in the environment. For example, walls are represented by reflecting lines, and corners are represented by double reflections at two orthogonal walls (see Fig. 3.3). The RBPF is used only for the data association process and the EKF is used to estimate the state vector. The algorithm was tested through simulations and data

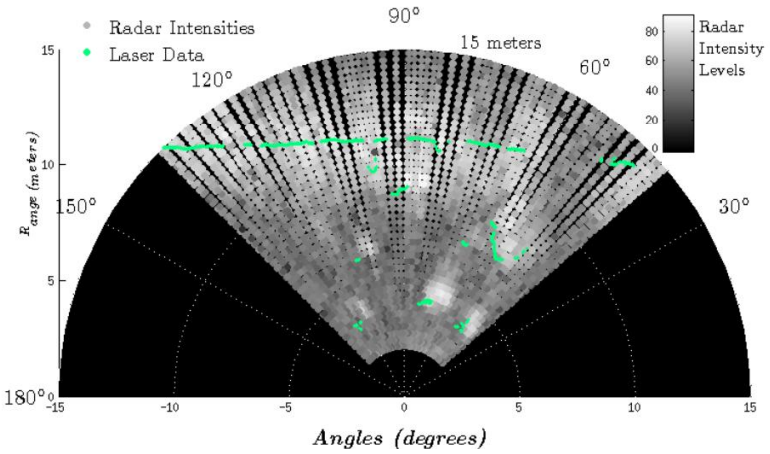


Figure 3.4: Radar spectrum, colored by intensities from black to white. The corresponding laser points are showed in green [CP12].

previously acquired. The authors concluded that the different propagation characteristics of walls, corners and other features in indoor environments can be used to distinguish those features, locate them, and use them as landmarks for navigation [DT10].

Castro et al. [CP12] proposed a reliable perception system based on sensor fusion between a millimeter-wave radar and a Laser Range Finder (LRF). Although, the LRF cannot penetrate heavy dust and smoke, the mm-wave radar can. In this type of approaches, both sensors must be perfectly aligned or calibrated previously. A relevant aspect is the fact that the scans from the radar and LRF are substantially different. The comparison between the data from both sensors needs to occur in the part of the scan, which is common to both sensors. Additionally, the range resolution of the radar is higher and the range distance is lower than the LRF sensor, as it is possible to verify in Fig 3.4. The matching between LRF scans and radar scans is done by calculating the 3D Euclidian distance between each laser point and the closest target detected by radar. The comparison between the calculated distances from both sensors is used to evaluate if they are observing the same target. In Fig. 3.5, it is shown a display of laser and radar scans.

The experiments were done using an Argo UGV, 8 wheel skid-steering platform equipped with 4 Sick LMS291/221 LRFs, a 94GHz Frequency Modulated Continuous Wave Radar, a visual camera and an infrared camera. The dataset obtained from two different areas showed that most dust points in the LRF scans were cleaned. However, some dust points (false negatives) have remained. When dust/smoke particles are too close to dense obstacles, the algorithm cannot remove them from the LRF scans. Also, when the particles are too close to radar and are detected by the LRF, they cannot be removed. This happens because these particles are seen as an obstacle by the radar, and if the LRF also detects these particles

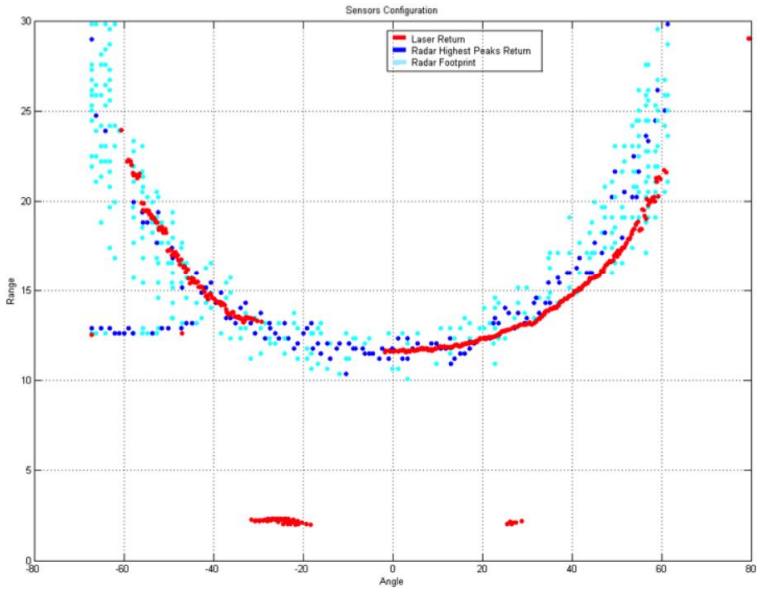


Figure 3.5: Example of laser and radar scans displayed as range vs. bearing angle: red dots are laser returns, blue points are radar peaks and dark blue points are the highest peaks [CP12].

they are wrongly confirmed to be an obstacle. This work has shown that even resorting to expensive and sophisticated hardware, the problem of SLAM in smoky conditions is extremely challenging.

Although the work done by Sales et al. [SMC+10] does not involve SLAM directly, it should be referred. A person-following algorithm in low visibility conditions, such as smoky environments, is presented. This approaches uses sensorial fusion. A vision system was adopted to determine the conditions of the environment, *i.e.*, to verify if the environment is filled with smoke. According to the authors, commercial smoke detectors only provide a signal indicating the presence of smoke when the smoke reaches the sensor and they cannot warn the presence of smoke away from the sensor. To solve this, a smoke detection system was implemented using visual cameras. The image obtained is analyzed according with two rules: the presence of grayish color of smoke in the image and the analysis of the spreading attributes of smoke. Unfortunately, this solution requires an *a priori* reference image of the environment without smoke.

Three different person-following approaches were developed: Sonar Ring following, LRF following and Sonar Time Difference of Arrival (TDoA) following. Sonar Ring following is based only on ultrasound technology and its performance is the worst of the three tested approaches, as expected due to the low resolution of sonar sensors. However, in low visibility conditions, it is an better alternative than the LRF. The LRF following is based on scans from a LRF sensor to detect the person. Finally, the TDoA uses the radio and ultrasonic

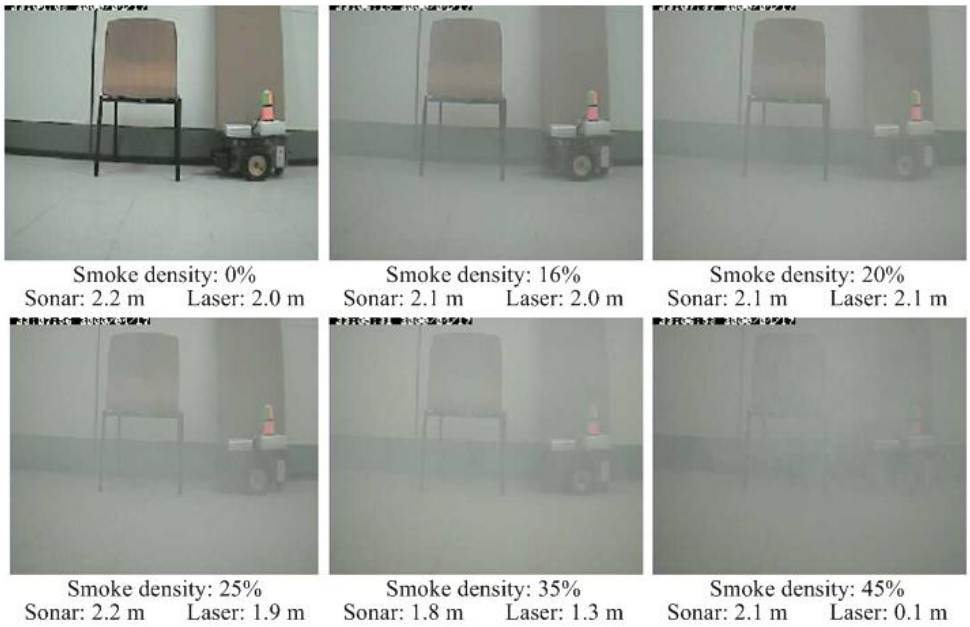


Figure 3.6: Distance obtained from LRF under different smoke densities [SMC+10].

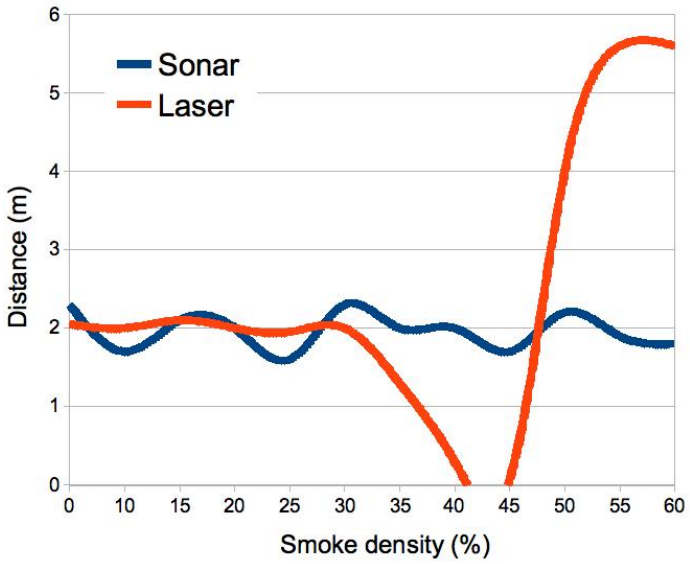


Figure 3.7: Sonar and LRF performance with increasing smoke density [SMC+10].

sensors to implement the person-following algorithm. This algorithm has achieved the best results in very different circumstances, even in smoky conditions. Furthermore, the authors have conducted a series of experiments with the visual smoke detecting algorithm and the distance measured by a LRF sensor (see Fig. 3.6). A comparison of range readings between a LRF and sonars for different smoke densities is also presented in Fig. 3.7.

Marti *et al.* [MSMS11] developed a localization method for smoky or dusty conditions. Their goal was to provide a method capable of positioning a mobile robot in front of points of Interest POIs, such as doors or fire extinguishers. When the smoke density in the en-

vironment is low, the robotic platform uses a LRF and Monte Carlo Localization (MCL) in order to navigate, implying that the map of the scenario has been preloaded. However, when smoke density increases, this process fails. Therefore, their approach makes use of fingerprinting techniques and *ZigBee* beacons around the scenario to perform localization. Additionally, *ZigBee* beacons located in Point of Interests (POIs) are equipped with a high luminosity Light-Emitting Diode (LED) panel. A visual positioning process is executed, when approaching one of these panels. Since their location is known *a priori*, these panels may be used to determine robot's pose with accuracy. This approach have been tested through multiple scenarios, such as class rooms or corridors with stairs. In order to mitigate the need of POIs, a Inertial Measurement Unit (IMU) sensor can be integrated. This approach has s severe disadvantage, which is the modification of the scenario prior to the mission. This is not possible in most real world SaR scenarios.

Pascoal *et al.* [PMDA08] carried out a set of tests in order to analyze the behavior of LRFs within low visibility scenarios. In a first approach, the warm-up time, the linearity of range values, range accuracy, and influence of surface properties were evaluated through a set of experiments in different environments. To evaluate the behavior of LRF under reduced visibility, the visibility was progressively reduced by the injection of smoke produced by a smoke machine and was homogenized by means of a ventilator. In clean environments, *i.e.*, environments with good visibility and no interfering sources, the LRF is an excellent range sensor. However, in adverse conditions, LRF provides erroneous or saturated outputs, which turns the LRF very faulty or even unusable, thus confirming Sales *et al.* results. In this case, the solution is to consider different sensing sources enabling data fusion, which in turn allows to increase measurement and estimation precision and better immunity against noise and outliers. Curiously, within all LRFs tested, the one with most unpredictable result is the Hokuyo URG-04LX, which is precisely the one used in the results of section 5 of this dissertation. This LRF is also the cheaper amongst the LRFs evaluated by Pascoal *et al.* [PMDA08].

3.1.1 Suitability of Range Sensors in Smoky/Dusty Scenarios

Implementing a SLAM solution is not possible without the correct choice of sensors, which is highly related with the environment where the robot will operate. The accuracy of most range sensors decreases in the presence of smoke. The most evident solution to this situation is fusing sensor measurements with different types of range sensors, *e.g.*, LRF with

sonars, LRF with RGBD cameras and sonars, *etc.*

It is therefore important to address sensor technologies, especially range sensors, which can be used in a mobile robotic platform. The correct choice of sensors is crucial. It should be noted that in scenarios with smoke or other disturbances the use of a single sensor modality is not viable, because the probability of misreadings is high. It is important to combine complementary types of sensors, so that faults are compensated by sensor fusion.

Diverse ranging sensors are compared in Table 3.1, taking into account their performance in the presence of smoke. Among the sensors that are immune to smoke, the UWB radar and the Thermal Imaging Camera (TIC) were not considered in this project due to their expensive nature and unavailability in our lab. Therefore, the only available sensor used that is immune to smoke was the sonar, as shown later on.

Table 3.1: Characteristics of different range sensors and their immunity to smoke.

Sensor (distance)	**Characteristics**	**Immunity to Smoke/Fog**
Radar [CP12] > 5k€	• Extremely expensive • On average, lower range and resolution when compared to common LRFs • Excellent results through heavy dust and smoke	Yes
Sonar [KK08] > 80€	• Very Cheap • Immunity to light and color • Limited angle of view • Very poor resolution • Very low power range sensor	Yes
Visual Camera [BPVC12] 40€ $<$ price $<$ 800€	• Accuracy varies with illumination conditions • Relatively cheap • Data provided can be very rich • No depth information without stereo vision	No
Thermal Imaging [BPVC12] > 4k€	• Works well in smoky environments • Hot environments decreases sensor accuracy • Extremely expensive	Yes
LRF [PMDA08] > 1k€	• Extremely reliable in clean environments • Relatively expensive • Easy to use	No
RGBD Camera [SM12] ≈ 150€	• Low cost • Image saturation in bright sunlight (outdoors) • Image noise when non-diffuse objects are in scene • Viable choice for indoor use • Muti-sensor integration (*e.g.*, Microsoft Kinect) • Cannot cover large open spaces	No

Chapter 4

Evaluation of SLAM Algorithms Available in ROS

In this chapter, we introduce the Robot Operating System (ROS) framework used in this work. Five 2D laser-based SLAM algorithms available in ROS are reviewed and evaluated, namely: HectorSLAM, Gmapping, KartoSLAM, CoreSLAM and LagoSLAM.

4.1 Robot Operating System (ROS)

ROS [QCG+09] is a very popular robotics framework. It provides several tools, libraries and drivers in order to further the development of new robotic projects. Also, it provides 2D and 3D simulation environments which increases its functionality. ROS has become one of the most used robotic frameworks, partly because of its characteristics, such as the hardware abstraction and architecture. It enables researchers to quickly and easily perform experiments through the use of its integrated drivers, libraries, visualizers, message-passing and more. This framework was originally released in 2007 and it is based on a graph architecture. All the processing takes place in *nodes* and data is exchanged using messages. Also, it provides a package system which simplifies code management and sharing.

A major problem inherent to the explosive growth in robotics is the impossibility to reuse the code in different robotic platforms. However, the hardware abstraction layer in ROS along with its messages service allows the creation of new code that can be used in many different robotic platforms. Moreover, ROS provides out-of-the-box a set of stable robotic software packages, like several SLAM algorithms, as shown in the following sections.

4.2 Evaluation of SLAM Algorithms

In the last few years, the number of SLAM approaches has increased. Therefore the need to compare different approaches grew significantly. Visually inspection of the resulting maps does not allow a correct comparison [KSD+09]. So, the need to precisely evaluate the results asks for a more accurate method — a quantitative scale. For instance, in [KSD+09], a metric for comparing SLAM algorithms was developed, wherein the result is not evaluated using a reference, but rather by considering the poses of the robot during data acquisition. This fact allows comparison between algorithms with different outputs. Also, the proposed method is independent on the sensor configuration of the mobile robot, but it requires manual editing of the dataset before being applied. The edition of the dataset is only performed one time. Furthermore, Olson *et al.* [OK09] discuss several aspects and methods for map optimization, which is one of the several topics of SLAM benchmarking.

All recognized SLAM evaluation methods rely on standard datasets available to the robotics community[1]. However, these datasets are not compatible with the ROS framework yet. Conversely, in this work, a study of the main 2D SLAM algorithms based on Laser Range Finders LRF that are available in ROS is presented. All the tested techniques use occupancy grids as the final output, which are analyzed using a performance metric for map similarities. The focus is put on the map quality instead of the pose estimation errors, since the mapping output is highly affected by localization issues. The main goal is to provide an overview of the strengths and weaknesses of all five algorithms available in ROS and also to provide a simple, yet accurate, quantitative comparison, thus defining general guidelines for ROS users to select the algorithm that best fits their requirements.

4.2.1 HectorSLAM

HectorSLAM[2] combines a 2D SLAM system based on robust scan matching and a 3D navigation technique using an inertial sensing system [KVSMK11].

Most 2D SLAM approaches require odometric information and do not leverage the high rate provided by common LRFs or more sophisticated Laser Interferometry Detection and Rangings (LIDARs). The authors have focused on the estimation of the robot movement in real-time through scan-matching, by making use of the high update rate and the low distance measurement noise from modern LIDARs. The odometric information is not used,

[1] `http://kaspar.informatik.uni-freiburg.de/~slamEvaluation/datasets.php`
[2] `http://www.ros.org/wiki/hector_slam`

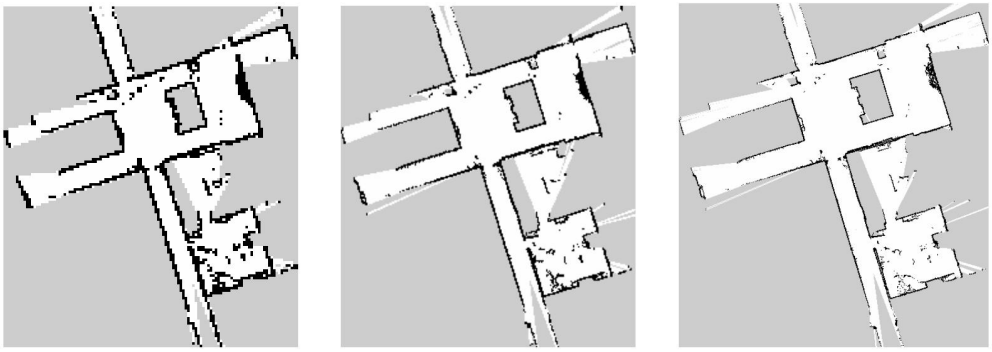


Figure 4.1: Multi-resolution representation of the map. Grid cell length (from left to right): 20cm, 10cm and 5cm [KVSMK11].

which gives the possibility to implement this approach in aerial robots like, a Quadrotor Unmanned Aerial Vehicle (UAV) or in ground robots operating in uneven terrains. On the other hand, it might have problems when only low rate scans are available and it does not leverage when odometry estimates are fairly accurate.

The 2D pose estimation is based on optimization of the alignment of beam endpoints with the map obtained so far. The endpoints are projected in the actual map and the occupancy probabilities are estimated. Scan matching is solved using a Gaussian-Newton equation, which finds the rigid transformation that best fits the laser beams with the map. According to the authors, the scan matching is done by finding the rigid transformation $\xi = (p_x, p_y, \psi)^T$ that minimizes:

$$\xi^* = \underset{\xi}{argmin} = \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2. \tag{4.1}$$

In Equation 4.1, $S_i(\xi)$ are the world coordinates of the $i-th$ scan endpoint $s_i = (s_{i,x}, s_{i,y})^T$ and $M(S_i(\xi))$ returns the map value for these coordinates. The problem is solved for the Gaussian-Newton equation:

$$\Delta\xi = H^{-1}[\nabla M(S_i(\xi))]^2. \tag{4.2}$$

In (4.2), the Hessian Matrix H is given by:

$$H = [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]^T [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]. \tag{4.3}$$

In addition, a multi-resolution map representation is used, to avoid getting stuck in local minima. The idea behind this solution is to have different maps in memory and simultaneously update them using the poses estimated previously. The computational cost remains low and the maps are always consistent, as it can be seen in Figure 4.1.

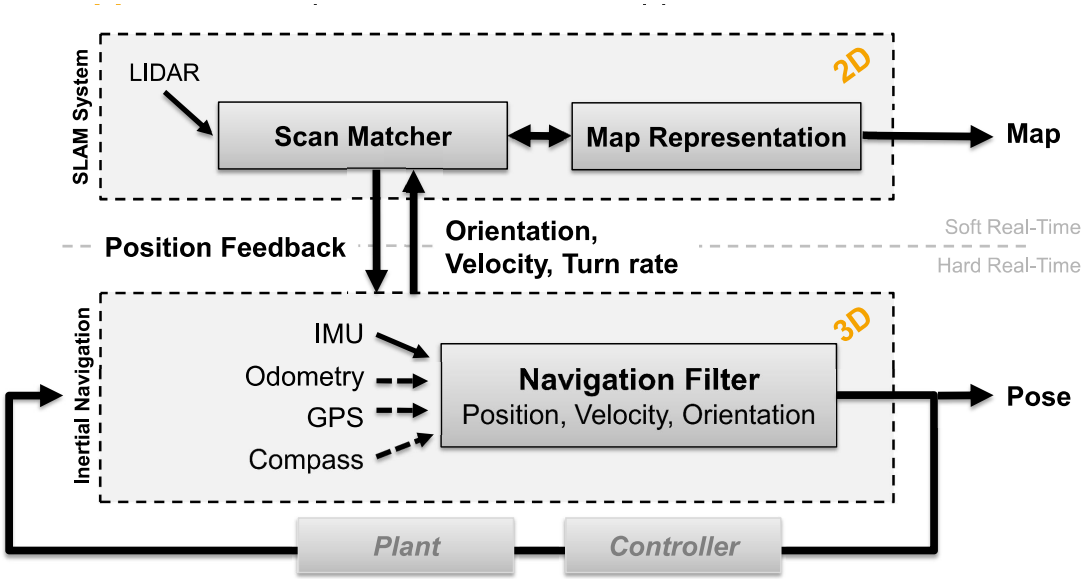


Figure 4.2: HectorSLAM overview [KVSMK11].

Finally, the 3D state estimation for the navigation filter is based on EKF. However, this is only needed when an IMU is present, such as in the case of aerial robots. Thus, it will not be used in this work. An overview of the HectorSLAM approach is shown in Figure 4.2.

4.2.2 Gmapping

Gmapping[3] is a laser-based SLAM algorithm as described by [GSB07]. Furthermore, it has been integrated in ROS and it is the most used SLAM algorithm in mobile robots worldwide. This algorithm has been proposed by Grisetti *et al.* and is a Rao-Blackwellized PF SLAM approach. This family of algorithms represents the posterior probability by a number of weighted particles and each particle is given an importance factor. However, they usually require a high number of particles to obtain good results, which increases their computational complexity. Additionally, the depletion problem associated with the PF resampling process decreases the algorithm accuracy. The particle depletion problem consists in the elimination of a large number of particles from the sample set during the resampling stage. This happens because their importance weights may become insignificant. Hence, this means that there is a small probability that correct hypothesis can be eliminated. In order to avoid the depletion problem, an adaptive resampling technique has been developed. The authors also proposed a way to compute an accurate distribution by taking into account not only the movement of the robotic platform, but also the most recent observations. In most PFs, the proposed distribution uses the odometry motion model. However, when a

[3] `http://www.ros.org/wiki/gmapping`

mobile robot is equiped 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 this, the authors integrate the most recent sensor observation z^t. Also, they compute a Gaussian approximation to efficiently obtain the next generation of particles. The Gaussian parameters are given by:

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} . \sum_{j=1}^{K} x_j . p(z_t | m_{t-1}^{(i)}, x_j) . p(x_t | x_{t-1}^{(i)}, x_j). \tag{4.4}$$

$$\Sigma_t^{(i)} = \sum_{j=1}^{K} p(z_t | m_{t-1}^{(i)}, x_j) . p(x_j | x_{t-1}^{(i)}, u_{t-1}) . (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T. \tag{4.5}$$

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 given by:

$$w_t^{(i)} = w_{t-1}^{(i)} . \eta^{(i)}. \tag{4.6}$$

Then, using (4.7), they compute the effective number N_{eff} of particles as a criterion to define when the resampling step should be performed:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\widetilde{w}^{(i)})^2}. \tag{4.7}$$

Therefore, the algorithm resamples each time N_{eff} drops bellow a given threshold of $\frac{N}{2}$, where N is the number of particles. This adaptive resampling decreases the uncertainty about the robot's pose in the prediction step of the PF. As a consequence, the number of particles required is decreased, since the uncertainty is lower, due to the scan matching process. In our experiments, the number of particles used by Gmapping was 30, which is extremely low when compared with common PF approaches.

4.2.3 LagoSLAM

The basis of graph-based SLAM algorithms is the minimization of a nonlinear non-convex cost function [CACB11]. More precisely, at each iteration, a local convex approximation of the initial problem is solved in order to update the graph configuration. The process is repeated until a local minimum of the cost function is reached. However, this optimization process is highly dependent on an initial guess to converge. Carlone *et al.* [CACB11]

developed a new approach called LagoSLAM[4] (Linear Approximation for Graph Optimization), in which the optimization process requires no initial guess. In addition, the technique can be used with any standard optimizer. In fact, the algorithm available in ROS has the possibility to choose between three different graph optimizers: Tree-based netwORk Optimizer (TORO) [GSGB07], g2o [KGS+11] and LAGO [CACB11]. In the experiments that we conducted, the LAGO optimizer proposed by Carlone *et al.* was used. Assuming that the relative position and orientation are independent for each node in the graph, the authors solve a system of equations equivalent to the non-convex cost function. To this end, a set of procedures based on graph theory were presented to obtain a first order approximation of the non-linear system, by means of a linear orientation and a linear position estimation.

4.2.4 KartoSLAM

KartoSLAM[5] is a graph-based SLAM approach developed by SRI International's Karto Robotics, which has been extended for ROS by using a highly-optimized and non-iterative Cholesky matrix decomposition for sparse linear systems as its solver [VLE10]. A graph-based SLAM algorithm represents the map by means of graphs. In this case, each node represents a pose of the robot along its trajectory and a set of sensor measurements. These are connected by arcs which represent the motion between successive poses. For each new node, the map is computed by finding the spatial configuration of the nodes which are consistent with constraints from the arcs. In the KartoSLAM version available in ROS, the Sparse Pose Adjustment (SPA) is responsible for both scan matching and loop-closure procedures [KGK+10]. For example, a robot equipped with a LRF builds a map by rendering the laser scans according to the positions of the robot in a graph, with the final output being an occupancy grid. The higher the number of landmarks, the more amount of memory is required. However, graph-based SLAM algorithms are usually more efficient than other approaches when maintaining a map of a large-scale environments. In the particular case of KartoSLAM, it is extremely efficient, since it only maintains a pose graph.

4.2.5 CoreSLAM

CoreSLAM[6] is a ROS wrapper for the original 200-lines-of-code tinySLAM algorithm, which is a laser-based approach created with the purpose of being simple and easy to under-

[4] `https://github.com/rrg-polito/rrg-polito-ros-pkg`
[5] `http://www.ros.org/wiki/karto`
[6] `http://www.ros.org/wiki/coreslam`

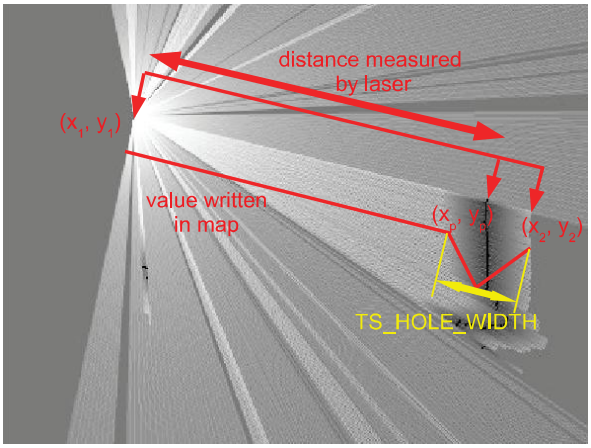


Figure 4.3: Integration of a LRF scan into the map [SEH10].

stand with minimum loss of performance [SEH10]. The algorithm is divided into two different steps: distance calculation and update of the map. In the first step, for each incoming scan, it calculates the distance based on a very simple PF algorithm. The PF matches each scan from the LRF with the map and each particle of the filter represents a possible pose of the robot and has an associated weight, which depends on previous iterations. After the selection of the best hypotheses, particles with lower weight are eliminated and new particles are generated. Since this approach is based on a PF, integrating the odometric information is a straightforward process. Additionally, the algorithm is capable of measuring very low displacements of the robot, due the use of several points for calculations. To determine the displacements of the robot, the authors introduced the latency concept. Latency is given by:

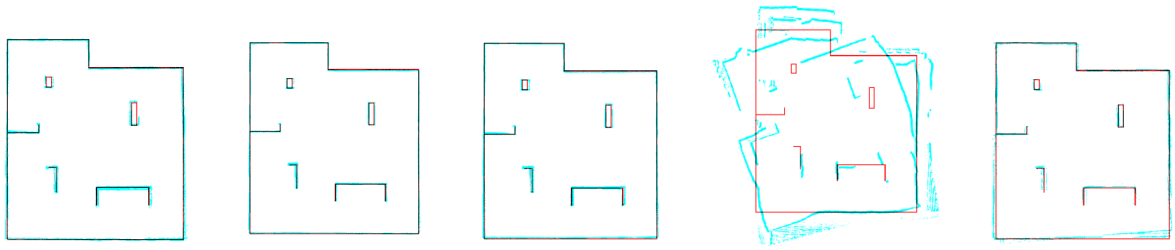
$$Latency = \frac{map\_resolution \times measurement\_frequency}{robot\_speed}. \tag{4.8}$$

In the update step, the lines corresponding to the received scans are drawn in the map using a modified version of Bresenham algorithm [Bre87]. Instead of drawing a single point when an obstacle is detected, tinySLAM draws a set of points surrounding the obstacle. The width of this set of points is an adjustable parameter of the algorithm, which may increase the convergence of the algorithm if correctly tuned. This is shown in Figure 4.3.

4.3 Experiments and Discussion

All five SLAM techniques described previously were tested using 2D simulations and real world experiments. Simulations were performed in Stage[7], which is a realistic 2D robot

[7] `http://www.ros.org/wiki/stage`



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Figure 4.4: Maps obtained through simulation in the *MRL arena* environment. Red represents the ground truth and blue represents the final map.

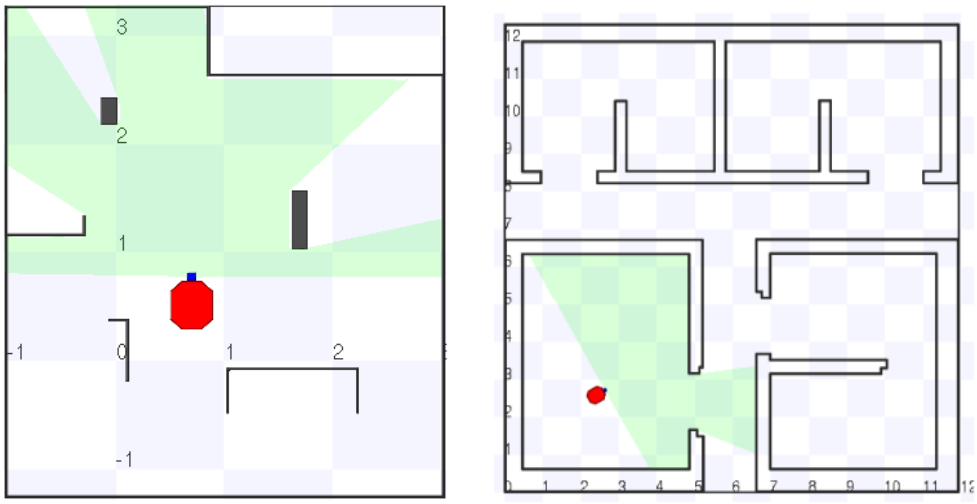
simulator integrated in the ROS framework. Additionally, tests were also conducted with a physical robot in a real world scenario, displaying the behavior of these SLAM packages in real world situations and in the absence of perfect simulated conditions.

Despite having perfect conditions in Stage simulations, like noise free odometric and range sensing information, SLAM algorithms assume that there is noise, which leads to imperfect results. In all experiments, the ROS Fuerte version was used and the robot was teleoperated. Note that the abstraction layer provided by ROS allows to use the same code for both simulation and real world experiments.

HectorSLAM requires a LRF with high update rates. The update rate of the Hokuyo URG-04LX-UG01 LRF used in the experiments is 10 Hz and Stage uses a similar maximum update rate. In order to deal with this, the robot was driven with low angular and linear speed. In the tests that were conducted, the output of each approach, described previously, was the respective generated 2D occupancy grid map.

In order to evaluate the quality of the obtained maps in the performed experiments, an analysis of the error between the generated map and the ground truth was conducted. Even though visual evaluation of the SLAM algorithm's output is possible, it does not allow to quantify the precision of each approach, therefore a performance metric based on the k-nearest neighbor concept is proposed. To that end, the best fit alignment between the ground truth and the map obtained is computed (see Figure 4.4), using intensity-based image registration tools.

The process works as follows: the resulting map of each algorithm is binarized. The binarized map only contains boundaries and obstacles of the scenario. Afterwards, the binarized map is aligned narrowly with the respective ground truth using a set of Matlab functions available in the Image Processing Toolbox. Since both ground truth map and the generated map are aligned, the distance from each occupied cell of the ground truth map



(a) *MRL Arena* (4.57×4.04m) (b) *1r5map* (12.24×11.67m)

Figure 4.5: Maps used in the simulation experiments.

to the nearest cell in the resulting map is determined using *knnsearch*, which computes the k-nearest neighbor cells (in this case $k = 1$). The sum of all distances obtained is then divided by the number of occupied cells in the ground truth map. This error metric provides a normalized measure of distance (in terms of cells), which can be applied in any generic occupancy grid, as long as the ground truth map is available.

4.3.1 Simulations in Stage

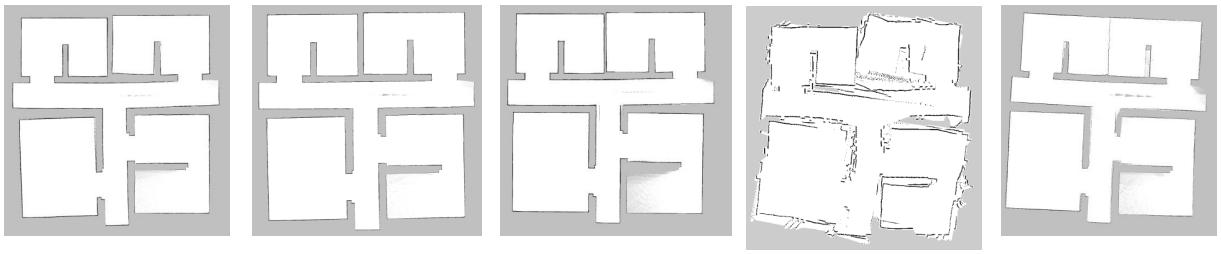
Stage simulations were performed using two different maps: the map of Mobile Robotics lab *(MRL Arena*) at the Institute of System and Robotics (ISR), University of Coimbra, and the *1r5map*, which are shown in Figure 4.5. Special focus is given to the former since it is used in both simulation and real world experiments. The *1r5map* enables the analysis of the behavior of the SLAM techniques in a larger scenario and with less features per square foot (*cf.* Figure 4.5b). This is particularly important to analyze the dependency on landmarks of each approach.

In the simulation experiments, the model of the range sensor was defined just like the sensor used in real world experiments: the Hokuyo URG-04LX-UG01, which has a maximum range of about 5.6 meters. Teleoperation was executed using the keyboard[8]. All the sensing and actuator data from the robot (LRF, odometry, velocity commands, etc.) was recorded previously and then played back for each algorithm. Thus, all SLAM packages were tested under the same exact conditions. This was only possible due to the *rosbag* tool[9] for ROS.

[8] `http://www.ros.org/wiki/teleop_twist_keyboard`
[9] `http://www.ros.org/wiki/rosbag`

Table 4.1: Error estimation for each algorithm in the *MRL Arena* (Simulation Experiments).

Simulation Experiments (*MRL Arena*)				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
0.4563	0.4200	0.5509	11.8393	1.4646



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Figure 4.6: Occupancy Grid Maps obtained through simulation in the *1r5map* environment.

Table 4.2: Error estimation for each algorithm in the *1r5map*.

Simulation Experiments (*1r5map*)				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
7.4581	5.3670	5.4380	171.5218	9.3041

For each algorithm, the resolution of the final map was set to 0.01 meters/pixel. In order to mitigate the low scanning rate, the number of sub-maps used in HectorSLAM was defined as 5. Since, each sub-map has half resolution of its precedent sub-map, the scan matching process is more accurate, *i.e.*, the scan matching performance is higher in lower resolution maps. In all experiments, the default parameters were used. For example, as mentioned before, the number of particles for the Gmapping algorithm was 30.

Analyzing the simulations results in the MRL arena, and according to Table 4.1 and Figure 4.4, Gmapping and HectorSLAM generated the map with the lowest error. On the other hand, KartoSLAM presented a slightly greater error, while results of CoreSLAM presented the highest error value. Gmapping is an extremely optimized PF algorithm with an improved resampling process, and this justifies the quality of the resulting map. Also, the scan matching process of HectorSLAM showed its efficiency. Nevertheless, it must be noted that the low speed commands given to the robot, in order to compensate the rate update from the LRF, have some influence in the results. Since both KartoSLAM and LagoSLAM are graph-based SLAM approaches, comparing the error between them is interesting. Both mapped successfully the arena. However, LagoSLAM obtained the greatest error (excluding

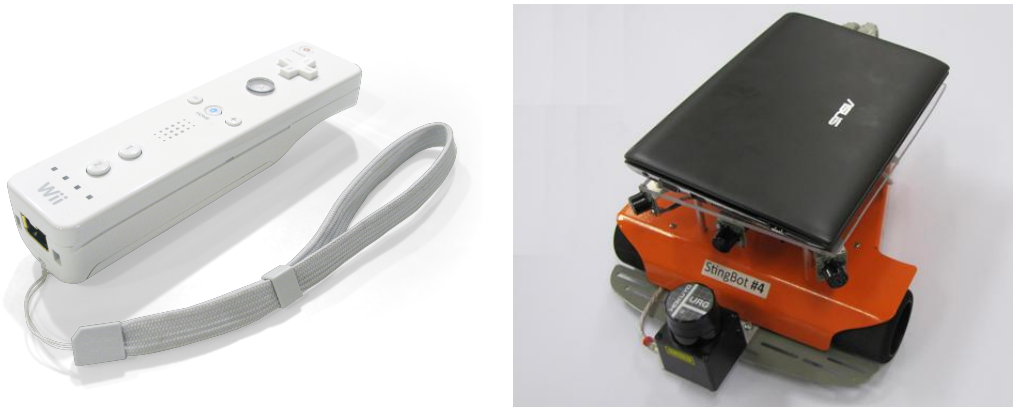
CoreSLAM). which can be explained by the impressive performance of the SPA solver method that KartoSLAM employs. Nevertheless, the quality of the resulting map obtained with LagoSLAM is still appropriate.

In order to compare all the SLAM approaches in a different scenario, a series of simulations using *1r5map* were also conducted. These results are shown in Table 4.2 and Figure 4.6. The *1r5map* is a relatively large map with a low number of distinctive landmarks. In this case, HectorSLAM obtained a higher error value than Gmapping. One of the reasons is the fact that HectorSLAM relies largely in scan matching between successive measurements. The full potential of HectorSLAM could not be observed due to the properties of the sensor used in these simulation experiments. Beyond that, due to the reduced number of landmarks, the error grows continuously, since the scan matching process is not fed with enough information. Additionally, since it is not using odometry information, a few issues arise when traversing long corridors with fixed width (see Figure 4.6a). As a consequence, the inferior result obtained with HectorSLAM in this test are not surprising. Once again, the Gmapping algorithm presents exceptional results, which reveal the accuracy of PF approaches. KartoSLAM revealed the robustness of graph-based SLAM approaches, since it obtained the second lowest error value and LagoSLAM obtained an higher error value than KartoSLAM. CoreSLAM was the worst performing algorithm. Since the error values are obtained via the Euclidean distance between points in the ground truth and the nearest point in the map, the errors obtained in the *1r5map* map are generally greater than in the other experiments due to the larger dimensions of the map. This is particularly visible in the case of CoreSLAM.

4.3.2 Experiments with a Mobile Robot

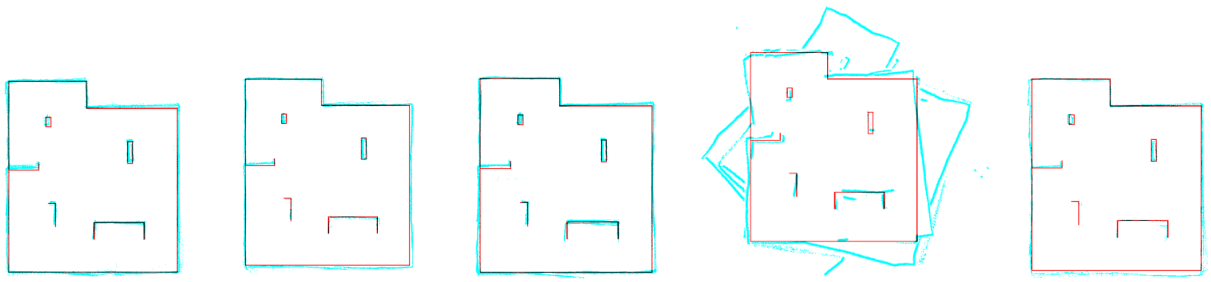
In the real world experiments, three runs with different trajectories and initial positions were performed using a Stingbot[10] robot [APCR13] that was teleoperated using a Nintendo wiimote[11] controller. The robot was equipped with an Hokuyo URG-04LX-UG01 and an Asus eeePC 1025C, running Ubuntu 11.10 and ROS Fuerte as shown in Fig. 4.7b. Once again, all the data was previously recorded and subsequently played back for each algorithm. Tests were conducted at the *MRL arena* in the ISR, in the University of Coimbra. The same algorithm parameters used in the simulation experiments were again adopted. Due to space limitations, the results of all trials are shown in Appendix A.

[10] `http://www.ros.org/wiki/mrl_robots`
[11] `http://www.ros.org/wiki/wiimote/`



(a) A Wiimote controler. (b) The Stingbot mobile robot.

Figure 4.7: The experimental setup.



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Figure 4.8: Performance Analysis in the real world *MRL Arena*. Red represents the ground truth and blue represents the final map.

Table 4.3: Error estimation for each algorithm in the *MRL Arena* (Real World Experiments).

Real World Experiments (*MRL Arena*)				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
1.1972	2.1716	1.0318	14.75333	3.0264
0.5094	0.6945	0.3742	7.9463	0.8181
1.0656	1.6354	0.9080	7.5824	2.5236

Figure 4.8 shows that all five techniques were able to map the scenario successfully. The error obtained for each algorithm is shown in Table 4.3. An analysis of the error can give a more accurate information about the performance of the algorithms. As can be seen, in general, all techniques led to worse results than in simulation. This slight performance hit is due to the existence of estimation errors in the robot position and noise on the laser scanning data, while mapping the real world MRL arena.

Despite the differences between virtual and real world environments, it is noteworthy that the results extracted from both setups follow some general trends, when one compares Table

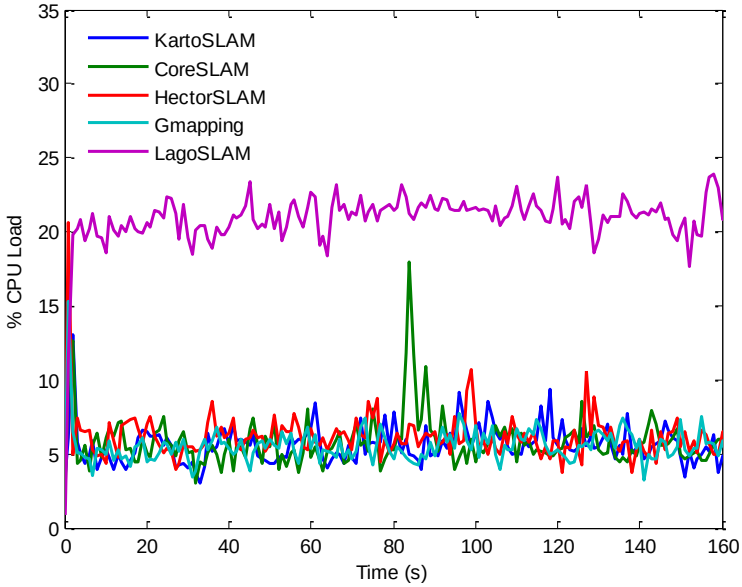


Figure 4.9: Evolution of the CPU load of each SLAM method using a real world dataset.

4.1 and Table 4.3, in particular for HectorSLAM, Gmapping and LagoSLAM. According to the authors of [SEH10], CoreSLAM can have great performance in several disparate environments. However this claim is not backed up by the results extracted from our experiments.

Being in the real world, the error increased for all approaches as expected due to noisy input. However, in the KartoSLAM algorithm, the error obtained in the real world experiments was not much higher than the error in simulations. In fact, generally KartoSLAM was the best performing technique in the real world, being less affected by noise than the other methods. This can be explained, not only due to the performance of the SPA solver used in KartoSLAM, but also because it is a full SLAM approach, *i.e.*, the map is obtained using the entire path and previous map and not only the most recent map and pose. The lower results of CoreSLAM in all experiments showed that its loop closure procedure rarely converges. This is clear in the video that shows a real world experiment and all the detailed results[12].

Beyond the error analysis conducted, an evaluation of the computational load using each technique was carried out. A representative comparison of the Central Processing Unit (CPU) load in a Laptop equipped with an Intel Core i7-3630QM and 8Gb of Random-Access Memory (RAM) running each algorithm is presented in Table 4.4 and Figure 4.9.

Looking closely at the results, LagoSLAM presented the highest percentages of CPU usage. Moreover, the values obtained are quite distant from the other four algorithms. This can be explained by the process developed to achieve the minimum cost function for the

[12]Available at: `http://www2.isr.uc.pt/~jsantos/MsC/`

Table 4.4: CPU Load (%) of the 2D SLAM approaches: mean ($\bar{x}$), median ($\tilde{x}$) and standard deviation (σ) values.

	HectorSLAM	**Gmapping**	**KartoSLAM**	**CoreSLAM**	**LagoSLAM**
$\bar{x}$	6.1107	7.0873	5.4077	5.5213	21.0839
$\tilde{x}$	5.9250	5.5800	5.3000	5.4400	21.2250
σ	1.993	4.4287	1.3018	1.6311	2.1684

given graph configuration, as referred in Section 4.2.3. The resources needed by the other four approaches during the experiments are similar, as seen in Table 4.4. This CPU analysis reveals that all five algorithms analyzed are quite efficient in terms of resources required and can be adopted online, during field experiments, to map generic 2D scenarios.

4.3.3 Discussion

In the previous section, a comparison of five representative 2D SLAM algorithms available in ROS was carried out in simulations and in real world experiments. A discussion of the weaknesses and strengths of each solution has been done. An accurate overview of each of the 2D SLAM techniques available in ROS was provided to shed light on the choice of an approach according to one's requirements.

According with our experiments, some ideas can be retained. On one hand, HectorSLAM relies only on scan matching and it does not make use of odometry, which could be an advantage or disadvantage depending on the robot and the environment's characteristics. For example, it is useful for aerial robots, but in situations where the odometry data is required to estimate the robot pose is quite disadvantageous. On the other hand, ideally it should be tested with specific hardware such as a high rate LRF, which is not available for our experiments.

Gmapping showed its robustness in every test, since the error and CPU load always remained low. It combines both scan matching and odometry in order to minimize the number of particles.

Both KartoSLAM and LagoSLAM are graph-based SLAM approaches, but their results were distinctively different. KartoSLAM provided accurate maps with lower CPU load, while LagoSLAM generated maps with higher error and CPU load. The reasons behind such discrepancies are related with the distinct processes of graph configuration and graph optimization of the two techniques.

Lastly, CoreSLAM achieved very poor results and it is possible to denote a lack of conver-

gence in its loop closure mechanism. CoreSLAM uses a simple PF which requires more particles, but has a lower computation power associated to each particle. According to [SEH10], CoreSLAM uses a very simple PF to match LRF readings with the map, which may lead to an erroneous position estimation. Additionally, the focus of the original work was to provide a simple SLAM technique with the ability to navigate within long corridors without losing its location, and not the loop closing system.

The work presented in this chapter has been submitted and accepted for publication in the Proceedings of the International Symposium on Safety Security and Rescue Robotics, which will take place in Linköping, Sweden in October, 2013. The article [SPR13] is available in Appendix B.

Chapter 5

SLAM Technique for Reduced Visibility Scenarios

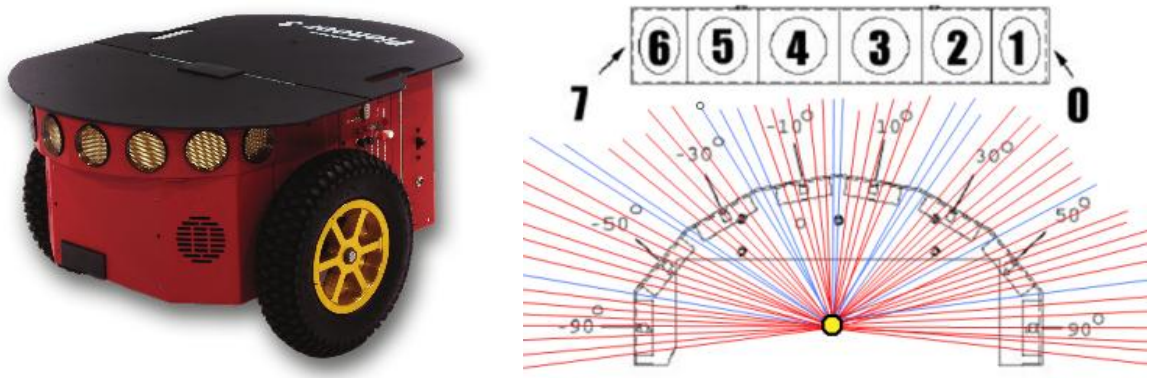
This chapter describes all the steps required to develop the SLAM approach for reduced visibility scenarios. Firstly, the selected range sensors are reviewed and tested in extreme situations. Next, the pseudo-code of the developed algorithm is presented and fundamental aspects and main challenges involved are discussed. In order to validate this approach, a set of experiments are conducted and the results are discussed.

5.1 Complementary Features of Sonars and Laser Range Finders

As mentioned before, LRFs are one the most adopted range sensors in 2D SLAM algorithms. They are extremely reliable in clean environments and easy to use. However, the main goal of this dissertation is to develop and verify a SLAM approach in environments with smoke or dust particles, which usually corrupt LRF readings. In order to decrease the impact of smoke in 2D laser-based SLAM algorithms, a multi-sensor approach such as those mentioned in Section 3.1 is required.

The mobile robot used in this work was a Pioneer 3-DX as shown in Figure 5.1a, which is equipped with an array of 8 sonars. Sonars use the propagation model of acoustic waves at higher frequency than normal hearing to extract information of the surroundings. Since they use acoustic waves, they are immune to smoke or dust/particles. Following the idea of [BPVC12] and [CP12], the sonar array from the Pioneer 3-DX can be used to overcome the behavior of the LRF in the mentioned conditions. However, the sonar sensor has some drawbacks, such as the much lower resolution and the limited angle of view.

The Pioneer 3-DX is a very versatile and robust mobile robot. It is a two-wheel, two-motor robotic platform which is commonly used for research worldwide. Its frontal sonar ring



(a) Pioneer P3-DX (b) Sonar ring displacement.

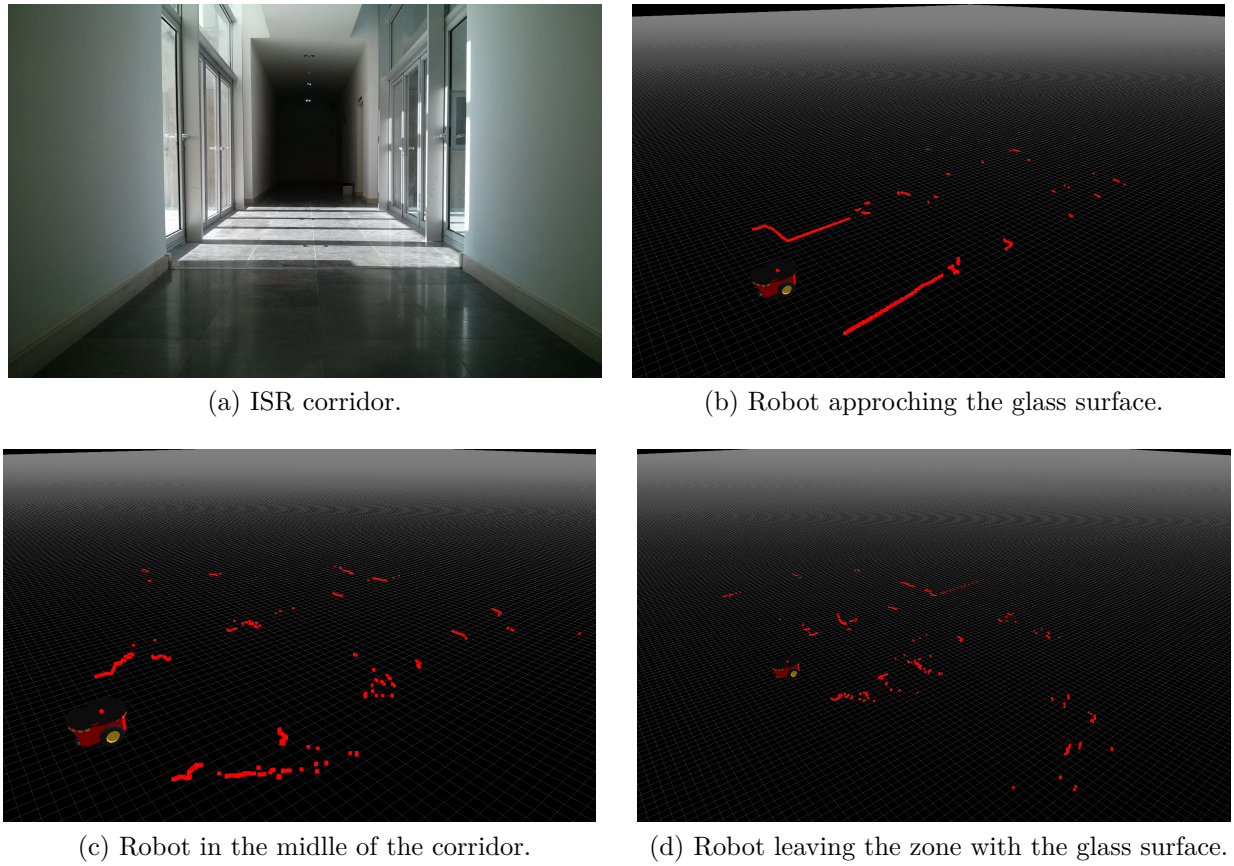
Figure 5.1: The robotic platform used in the experiments.

is composed by 8 SensComp's Series 600 Instrument grade Electrostatic Transducer [Sen08] disposed at angles: -90^o, -50^o, -30^o, -10^o, 10^o, 30^o, 50^o and 90^o degrees. Figure 5.1b illustrates how the 8 transducers are arranged in the robot. This ultrasonic sensor has a beam angle of 15^o at $-6dB$. Its maximum range is about 5 meters and the field of view FOV of the sonar ring is about 196^o degrees.

The LRF used in all our experiments was an Hokuyo URG-04LX [HAC05]. It has a maximum range of 5.6 meters, an angular resolution of about 0.36^o degrees and a FOV of 180^o degrees. Since the FOV of both technologies are similar, comparing the readings of both sensors is relatively straightforward.

Figure 5.2: The Hokuyo URG-04LX LRF.

Before proceeding to the development of the proposed SLAM technique, it is necessary to study the behavior of the selected sensors in several situations. For example, how the LRF behaves towards glazed surfaces or in environments highly corrupted by smoke. In Figure 5.3, it is showed the behavior of the LRF in a environment with glass surfaces. Since, LRFs are optical sensors using a laser beam to measure distances, glass surfaces refract a portion



(a) ISR corridor.

(b) Robot approching the glass surface.

(c) Robot in the midlle of the corridor.

(d) Robot leaving the zone with the glass surface.

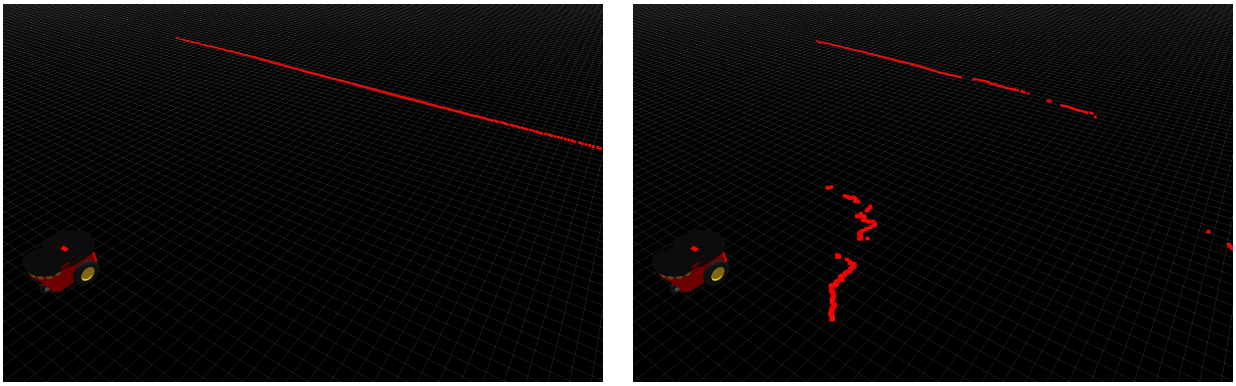
Figure 5.3: LRF readings in a corridor with glass surfaces in both sides.

of the incident laser beam and reflect the other, leading to misreadings. In Figure 5.3 it is possible to verify this phenomenon. in this experiment, the Pioneer P3-DX was equipped with an Hokuyo URG-04LX LRF and navigated in the ISR corridor which contains glass surfaces in some places.

Another special case is the behavior of the LRF in smoky environments. However, it is not possible to test the sensor with smoke from typical fire scenarios. In [PMDA08], Pascoal *et al.* tested different aspects of a given set of LRFs. The Hokuyo URG-04LX was one the tested sensors. The most interesting experience was the behavior of these sensors in smoky environments, where it was observed that all LRF sensors performed poorly in this conditions.

In order to gain some tact about this behavior and have some useful data for the development of the algorithm, some preliminary tests were carried out. The Pioneer P3-DX equipped with the Hokuyo LRF was placed in front of a flat wall and smoke was injected between the robot and the wall. The smoke machine used was a Magnum 800[1]. These machines produce a dense vapor consisting of water and glycol-based fluid, which looks similar

[1] `http://www.martin.com/product/product.asp?product=technofog&newseg=ent&mainseg=ent`



(a) LRF readings without smoke.

(b) LRF readings with smoke.

Figure 5.4: Smoke experiments with the LRF against a wall.

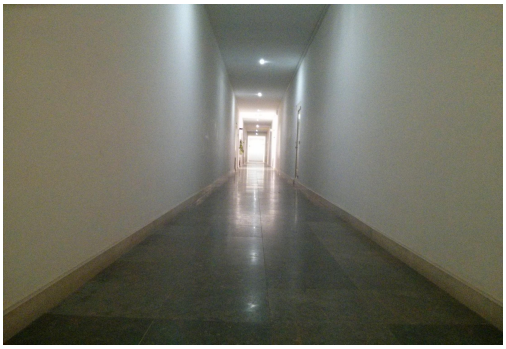
to smoke, being commonly used in theaters and night clubs. It became clear that even in small concentrations, smoke highly affects the sensor readings. Figure 5.4 shows an example of the behavior of the LRF sensor before (Figure 5.4a) and after (Figure 5.4b) the injection of smoke. It becomes clear that, in order to successfully perform a mapping task in these conditions, the sonar ring available can be useful to compensate the misreadings of the LRF, due to its immunity to smoke particles.

Following the previous tests, the sonar ring of the Pioneer P3-DX was also subjected to the same tests. However, a somehow surprising behavior in some situations occurred, which made the development of the algorithm much more challenging.

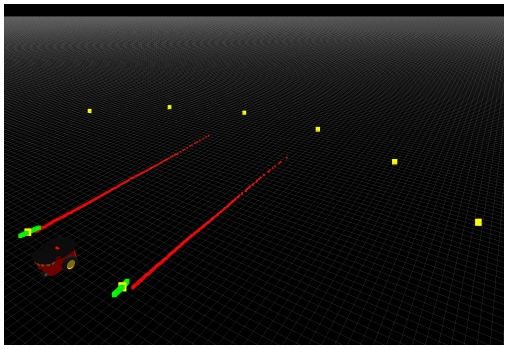
Consider a long corridor as shown in Figure 5.5a. When the robot is positioned in the middle of a clean corridor, the only sonars in the P3-DX capable of detecting walls are the ones positioned in the extremes at -90^o and 90^o degrees. Due to the small FOV of each sonar, approximately 15^o degrees, obstacles are only detected when the incident beam is normal or near normal to the obstacle.

Due to this unexpected behavior, an opportunity to test the sonar ring in a Nomad Scout arose (*cf.* Fig. 5.5c), this is also a differential robot commonly used in the research community. The behavior of both sonar rings were compared in this specific situation. Note that, the Nomad Scout has a sonar ring with 16 sonars and each sonar has a FOV of about 25^o degrees. The result of this comparison is shown in Figure 5.5. A photo of the corridor where the comparison was conducted, is shown in Figure 5.5a. Figure 5.5b and 5.5d are the sonar ring readings of the Pioneer P3-DX and of the Nomad Scout, respectively. From the analysis of the results, it is verified that the P3-DX can only retrieve correct readings from the sonars positioned on the extremes. This is not the case when using the Scout. According to Fig. 5.5d most sonars in the ring can detect the walls. This can be explained

by the higher FOV of the sonars in the Scout robot. Hereupon, it is necessary to discard this erroneous values, which correspond to saturated values, *i.e.*, the maximum range of the sonar.

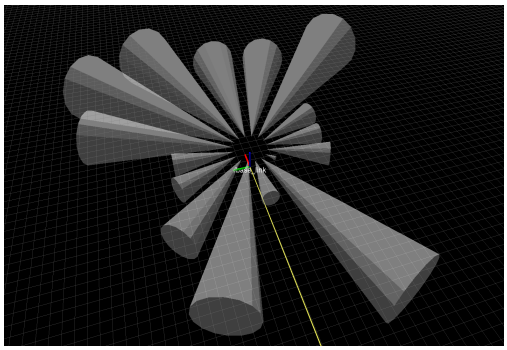


(a) ISR corridor.



(b) Sonar readings of the Pioneer P3-DX.

(c) The Nomad Scout.



(d) Sonar readings of the Nomad Scout.

Figure 5.5: Sonar ring comparison between the Pioneer P3-DX and the Nomad Scout in a plain corridor. Red dots are the LRF readings and yellow dots are the sonares readings. Green dots correspond to a conversion of the sonar data to the laser scan data type.

In Figure 5.6 the behavior of the sonar ring of the Pioneer P3-DX in the corridor with glass surfaces is shown. It is verified that valid readings detect the glass surface with high accuracy.

Finally, a test of the P3-DX sonar ring in smoky environments was conducted. Following the same approach of the LRF, the robot was placed in front of a plain wall and the smoke injected between both. As expected, smoke particles did not influence valid sonar readings, as illustrated in Figure 5.7.

5.2 Proposed SLAM Technique

In Chapter 4, several SLAM techniques were evaluated. The evaluation conducted served as a guide to choose the most suitable technique to implement as the basis of this work. According to the obtained results, Gmapping was one of the most accurate and robust

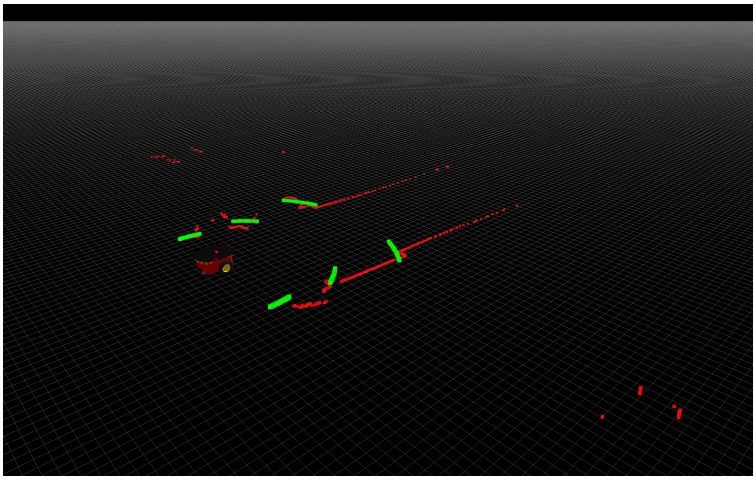
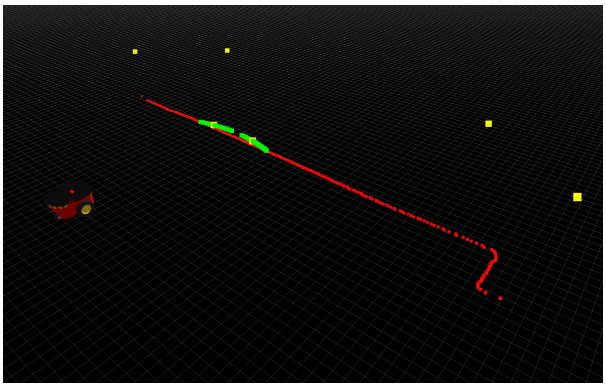
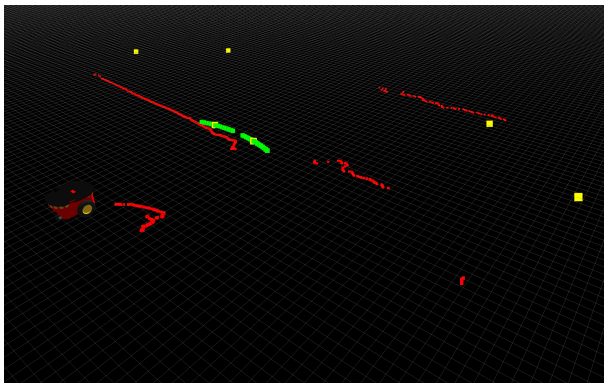


Figure 5.6: P3-DX sonar ring readings in a glazed corridor (6 valid readings and 2 filtered ones).



(a) Sonar readings without smoke.



(b) Sonar readings with smoke.

Figure 5.7: Smoke experiments with the sonar ring of the Pioneer P-3DX against a wall. Red dots are the LRF readings and green dots are the sonar readings. Note that only two of eight readings are valid in this particular situation.

approaches. It makes use of odometry to improve the efficiency of its PF and since the P3-DX provides fairly accurate pose information, it was considered the most suitable choice.

The ROS driver of the Pioneer P3-DX retrieves the sonar information in a point cloud message. However, Gmapping was developed to work with LRFs and is only capable of dealing with laser scan data. The code of the Gmapping algorithm could eventually be changed to accept both type of messages. However, this code would be limited to use with Gmapping, which is not advantageous. As consequence, the idea behind this project is to provide an "intelligent" layer, denoted hereafter as SmokeNav layer, which fuses the output of each sensor and adjusts, rectifys or ignores that information according to the situation as shown in Figure 5.8. This layer adapts to any set of range sensing data and provides the filtered data to potentially any generic 2D SLAM algorithm in smoky environments.

In this project, the set of sensors is restricted to a LRF and a sonar ring. So, it is

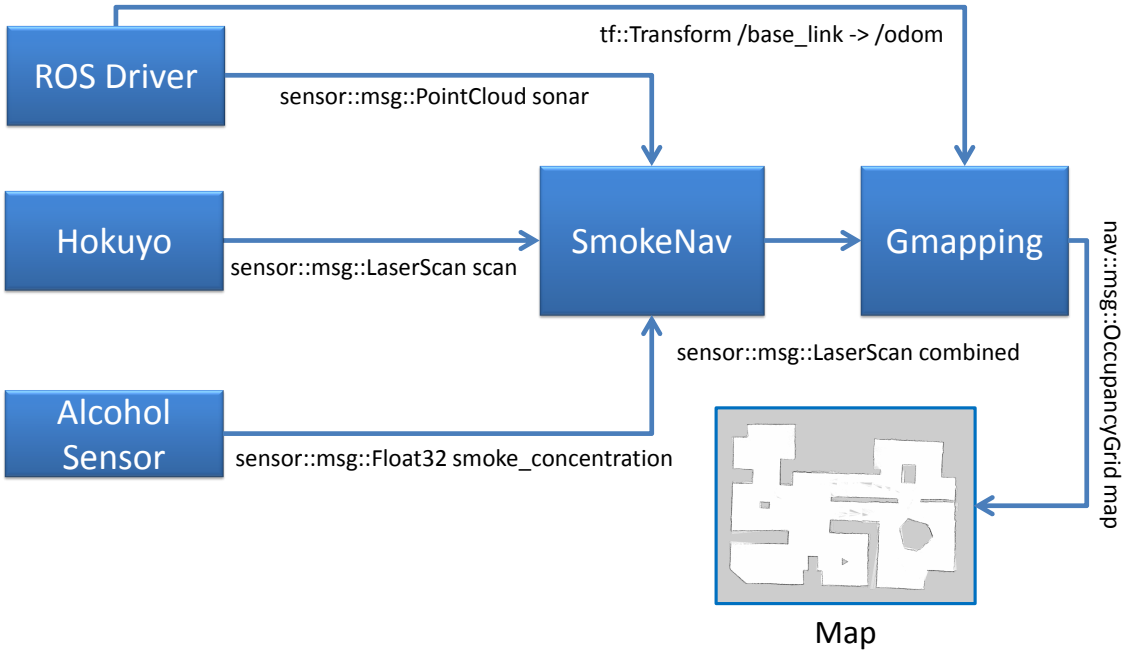


Figure 5.8: Overview of the SmokeNav layer integration.

necessary to convert sonar data, which is received as a point cloud message from ROSARIA[2], the ROS driver of the Pioneer mobile robot, to a LaserScan message type. The structure of a LaserScan message is given by:

```
std_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

Algorithm 1 presents a high-level code of the implemented conversion of sonar readings to a LaserScan message. In this code, each point retrieved by the sonar ring is transformed into a section of the LaserScan message. The conversion respects the same parameters as a message received from the Hokuyo LRF used in the experiments.

The worst scenario for the SLAM algorithm is when the smoke concentration is so high

[2]http://www.ros.org/wiki/ROSARIA

Algorithm 1: PointCloud message to LaserScan message (conversion of sonar readings).

input : sensor::msgs::PointCloud input_pcl
output: sensor::msgs::LaserScan sonar_scan

transformPointCloud(input_pcl,pcl_transformed,target_frame); //transforms sonar points to the Laser reference frame
range_min_sq = range_min × range_min;
ranges_size = ceil($\frac{angle\_max - angle\_min}{angle\_increment}$);
ranges[ranges_size] = range_max + 1.0;
for *each point in pcl_transformed* **do**
  if *x!=0* **and** *y!=0* **then**
    range_sq = x×x + y×y;
    if *range_sq* < *range_min_sq* **then**
      continue;
    end
    angle=atan2(x,y);
    for $j = \forall\ \theta \in [-\frac{angle}{2}, \frac{angle}{2}]$ **do**
      if *j* < *angle_min* **or** *j* > *angle_max* **then**
        continue;
      end
      index = $\frac{j - angle\_min}{angle\_increment}$;
      if *range_max + 1* > *range_sq* **then**
        sonar_scans.ranges[index] = $\sqrt{range\_sq}$;
      end
    end
  end
end
publish(sonar_scan);

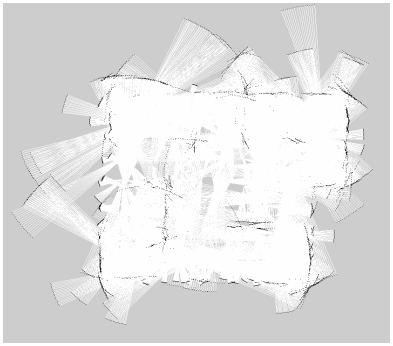
that the entire LRF readings are corrupted. In such extreme case, the mapping task must be done using only the sonar ring. Therefore, it is important to verify if this conversion allows Gmapping to successfully map the environment using only sonar data. However, the quality of the resulting map is expected to be low, due to the low resolution and nature of sonars. Several runs in the *MRL Arena* were conducted in order to refine the parameters of Gmapping so that it provides the best possible map. In order to decrease the issue of the non-normal surfaces (*cf.* Section 5.1), the range of each sonar has been limited to a maximum of 2.5 meters. The sonars return the maximum range (5.0 meters), when they do not receive the sound wave echo, thus not detecting an obstacle. Also, when the robot rotates, it typically returns erroneous measures greater than 2.5 meters.

The influence of the number of particles, map resolution and temporal thresholds has been tested too and the best results were achieved with the default parameters. Several other parameters were tested, such as the number of iterations of the scan matching and the angular resolution. The parameter that most influenced the results was the angular

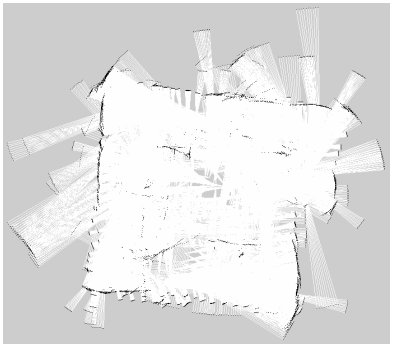
Table 5.1: Parameters used to successfully map the *MRL Arena* with Gmapping, fed with sonar data.

Gmapping Parameters	
• maxUrange = 16.0	•sigma = 0.05
• kernelSize = 1	• iterations = 5
• linearUpdate = 0.5	• angularUpdate = 0.8
• temporalUpdate = 3.0	• resampleThreshold = 0.5
• particles = 30	• delta = 0.01

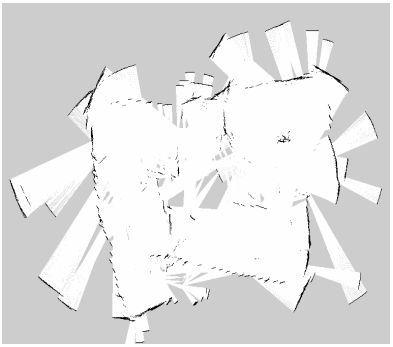
resolution of the scan. In Figure 5.9, results for different angular resolutions in different trials are shown.



(a) Angular resolution of 0.01^o degrees.



(b) Angular resolution of 0.0175^o degrees (the same resolution of the Hokuyo).



(c) Angular resolution of 0.0044^o degrees.

Figure 5.9: Resulting maps of some of the experiments conducted with Gmapping fed by sonars in the *MRL Arena*.

In the development of the SLAM algorithm for reduced visibility scenarios, it was verified that the information obtained using only two sensors is insufficient. This is the case, for example, when the comparison between both is not possible (*i.e*, sonar readings are not valid and laser scans are noisy) and the smoke density is so high that it forms an indistinguishable obstacle. Another example is when the sonar ring measures approximately the same distance as the LRF, for example, while turning, but the laser reading is erroneous due to the smoke particles. Therefore, an additional sensor to inform about the concentration of smoke in the environment was required. Typically, we would use a dust sensor like the Grove PPD42NS [3] in this context. However, this sensor is incapable of measuring the concentration of smoke emitted by smoke machines due to its distinct properties. This happens because the smoke produced is a glycol-based vapor. Therefore, an alcohol sensor may be used instead to detect different concentrations of glycol-based vapor, thus emulating a dust sensor in our

[3] `http://www.seeedstudio.com/wiki/images/4/4c/Grove_-_Dust_sensor.pdf`

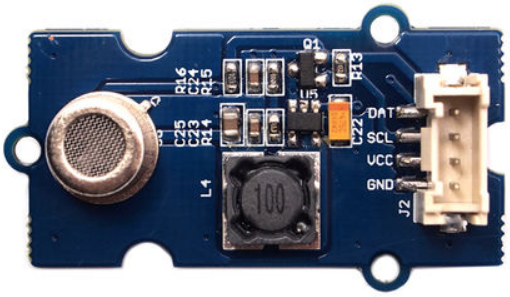


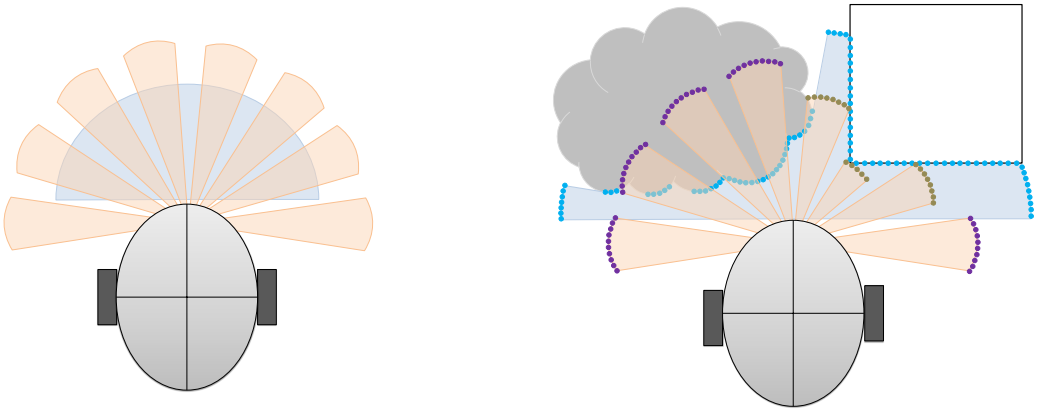
Figure 5.10: The Seeed MQ303A.

experiments. The sensor model used was the MQ303A[4], which is manufactured by Seeed Studio and the output voltage is inversely proportional to the alcohol concentration in the air. The sensor, illustrated in Fig. 5.10 has three different behaviors. Firstly, the sensor is heated and sensor values start to decrease slowly. However, when exposed to alcohol the readings denote a sharp increase. Removing the alcohol once again, the sensor value starts to decrease slowly. In a real situation, the application of the alcohol sensor to detect smoke is meaningless, but in that case the dust sensor referred before would provide a decent solution.

The previous step of incorporating the alcohol sensor contributed to the development of the proposed technique, as shown in Fig. 5.8. The algorithm receives messages from both range sensors and treats the data taking into account the time stamps of both scans. The need to synchronize messages and to minimize the delay in the processing time, dictates that the algorithm has to be simple and effective. The odometry and all sensors measurements must have the correct time stamp in order to work properly. The information arriving from the alcohol sensor is constantly monitored to infer on the visibility conditions of the environment.

Message synchronization between the sonar ring and the LRF is possible due to the message filter API[5] available in ROS. Every time a LaserScan and PointCloud message arrives, their time stamps will be compared using an approximate time policy. After that, the point cloud received will be transformed from the sonar frame to the LRF frame, to analyze and compare both readings in the same coordinates frame. Each point in the point cloud, corresponds to a section of the sonar "scan". The section is determined by transforming the point to the polar form and using the FOV of the sonar, such as in Algorithm 1. This

[4] http://www.seeedstudio.com/depot/images/product/MQ303A.pdf
[5] http://wiki.ros.org/message_filters



(a) Scheme of the LRF and sonar arrangement in the Pioneer P3-DX.

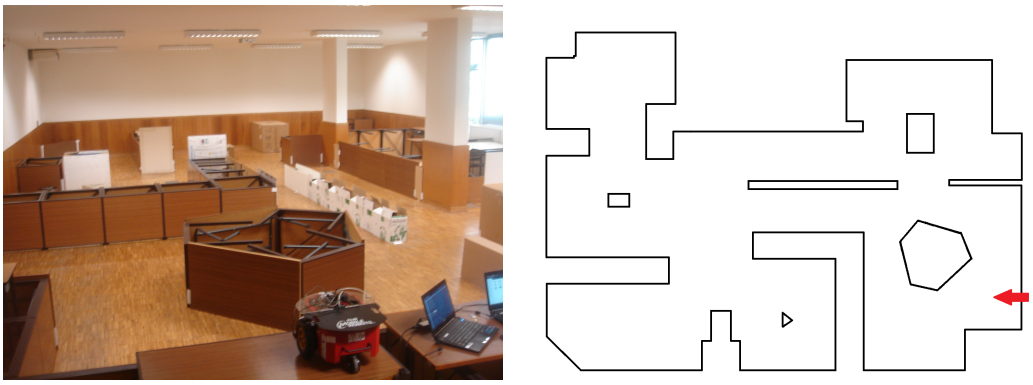
(b) LRF vs Sonar Array, when smoke and obstacles are detected.

Figure 5.11: Pioneer P3-DX sensors arrangement.

is illustrated in Figure 5.11a.

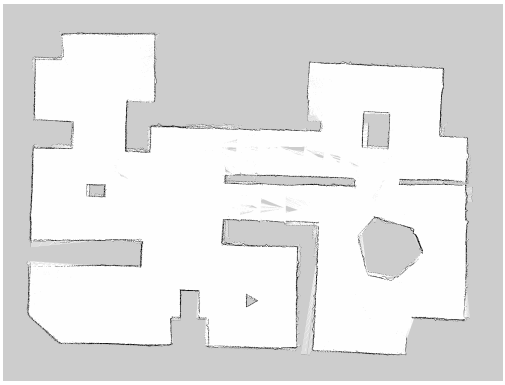
For each sonar section, the respective section in the LRF laser scan message is evaluated. A threshold is calculated taking into account the mean of the ranges of the given LRF for the given section. Next, the square differences of the ranges is determined and if they are greater than a threshold, a zero-crossing mechanism is applied. If one third of the differences crosses zero, the section will be ignored or replaced by the respective sonar reading, if it is valid. As mentioned before, this is not enough to detect smoke and the information of the alcohol sensor must be integrated. The alcohol sensor detects the artificial smoke relatively fast, but it is slow to recover when the smoke concentration is decreasing. To overcome this issue, two stages are defined: increasing smoke concentration and decreasing. According to these stages, smoke thresholds will vary. When the value of sonar readings does not match measurements taken by laser scans, and when smoke levels detected by the alcohol sensor hits values below a minimum threshold SMOKE_MIN, a greater certainty is given to readings from the LRF. In conditions where the alcohol sensor retrieves values greater than SMOKE_MIN, sonar readings will be superimposed to LRF readings. In this situation, when high differences between the laser and sonar readings in the same section occur, only the data from sonars will be used. However, due to the slow decreasing curve of the alcohol snesor, it is necessary to distinguish between increasing or decreasing values. The threshold for the increasing case is lower than for the decreasing case.

After the evaluation of all sections, all the gathered information will be used to build the new LaserScan message. If smoke is detected in a given section, the measurement of the LRF will be corrected if the sonar is available and ignored if the sonar is not available. The



(a) Photo.

(b) Ground truth map.



(c) Resulting map of a run with Gmapping.

Figure 5.12: The *R3.2 Arena* with dimensions: 13.63 × 9.87 meters. The arrow in Figure 5.12b represents the viewpoint of Figure 5.12a

sections of the LRF scan not covered by the sonars (due to its reduced FOV) will be filtered according to the state of the adjacent sections.

To further understand the developed algorithm, the high-level pseudo-code of the Smokenav layer is presented in Algorithm 2.

5.3 Results with a Mobile Robot and Discussion

In order to validate the developed algorithm, several real world experiments were performed. Since, the *MRL Arena* is a fairly small scenario, it is not possible to show the robot moving from a clean environment to a smoke-corrupted one. In order to correctly evaluate and validate the algorithm, a larger scenario was required. The solution was to build a larger arena in a class room of the Department of Electrical and Computer Engineering of the University of Coimbra. The resulting arena called *R3.2 Arena* is shown in Figure 5.12a. Additionally, a ground truth map of the arena was built, as a reference (Figure 5.12b).

A run with Gmapping on the P3-DX in this arena with clean conditions was conducted. The resulting map is shown in Figure 5.12c.

Algorithm 2: SmokeNav Layer Procedure: Fusing data from the LRF, sonars and alcohol sensor.

input : sensor::msgs:LaserScan sonar_scan, sensor::msgs::LaserScan scan, sensor::msgs::Float32 smoke_density
output: sensor::msgs::LaserScan combined

```
1  if (smoke concentration > SMOKE_MAX_UP and is increasing) or (smoke concentration >
   SMOKE_MAX_DOWN and is decreasing) then
2  |  use only sonar data (if available);
3  else
4  |  for each sonar section do
5  |  |  if number of elements in section < 2 then
6  |  |  |  sonar is not available in this section;
7  |  |  |  smoke uncertainty set to -1 in this section;
8  |  |  else
9  |  |  |  if sonar measurement < 2.5 meters then
10 |  |  |  |  sonar available;
11 |  |  |  else
12 |  |  |  |  sonar not available;
13 |  |  |  end
14 |  |  |  if smoke concentration < SMOKE_MIN then
15 |  |  |  |  //the LRF has a greater weight than sonar
16 |  |  |  |  computes the square differences with the range values in the section;
17 |  |  |  |  dynamic_threshold = (2*PI*mean(ranges in the section)*angle_increment)/(2*PI);
18 |  |  |  |  if any difference > dynamic_treshold then
19 |  |  |  |  |  count how many times have crossed zero;
20 |  |  |  |  |  if count > 1/3 * length of the section then
21 |  |  |  |  |  |  smoke uncertainty is set to 1 in this section;
22 |  |  |  |  |  end
23 |  |  |  |  else
24 |  |  |  |  |  smoke uncertainty is set to 0 in this section;
25 |  |  |  |  end
26 |  |  |  else
27 |  |  |  |  //more weight is given to sonar
28 |  |  |  |  if (sonar measurement < section maximum range + constant) and (sonar measurement > section
   |  |  |  |  minimum range - constant) then
29 |  |  |  |  |  use the laser data;
30 |  |  |  |  |  smoke uncertainty is set to 0 in this section;
31 |  |  |  |  else
32 |  |  |  |  |  smoke uncertainty is set to 1 in this section;
33 |  |  |  |  end
34 |  |  |  end
35 |  |  end
36 |  end
37 |  if use_sonars_only == true then
38 |  |  for each sonar section do
39 |  |  |  if sonar measurement available then
40 |  |  |  |  use sonar measurement in the corresponding section of the LaserScan message;
41 |  |  |  end
42 |  |  end
43 |  else
44 |  |  for each section do
45 |  |  |  if smoke uncertainty == 0 then
46 |  |  |  |  use LRF reading;
47 |  |  |  else
48 |  |  |  |  if sonar available then
49 |  |  |  |  |  use sonar measurement in the corresponding section of the LaserScan message;
50 |  |  |  |  end
51 |  |  |  end
52 |  |  end
53 |  |  for each section between sonar sections do
54 |  |  |  if there is no smoke in adjacent sections then
55 |  |  |  |  use LRF reading;
56 |  |  |  end
57 |  |  end
58 |  end
59 end
60 publish(combined);
```

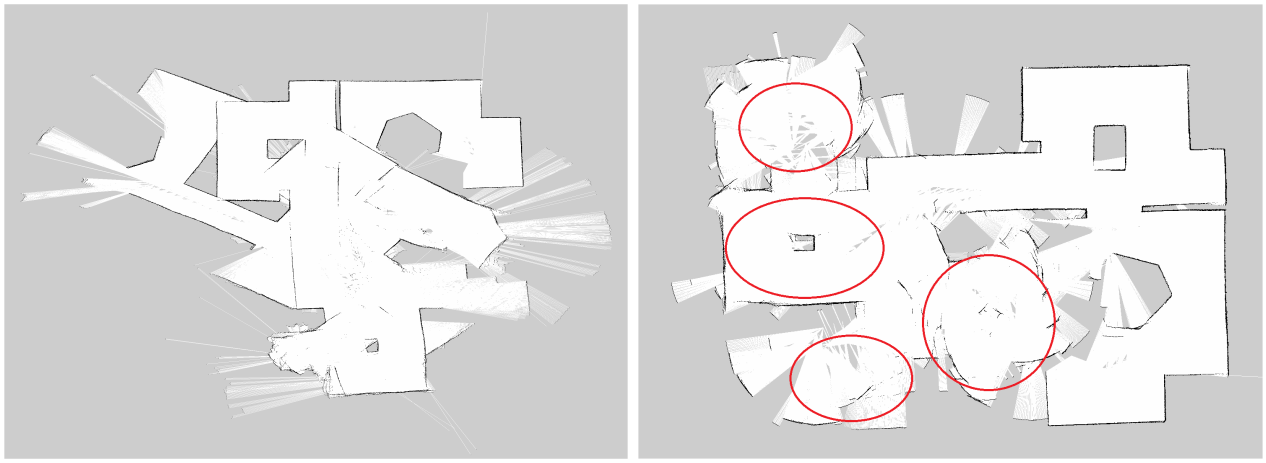
Figure 5.13: Photo of the smoky conditions in *R3.2 Arena*.

In order to extract relevant data and validate the algorithm, several runs using the robot in the new arena were recorded with the *rosbag* tool available in ROS. This allows to run the algorithm after the experiment, and also to test Gmapping in the same conditions without the SmokeNav layer and verify if there were significant improvements, *i.e.*, if the algorithm was able to successfully map with smoke.

In every conducted trial, the robot started in a clean zone. Afterwards, the robot started moving and smoke was injected in an opposite area. An example of the experimental conditions is shown in Figure 5.13. In these experiments the robot was teleoperated using Wiimote. Results of three trials are shown in Figures 5.14, 5.15 and 5.16. Red circles represent zones where the smoke was injected and where its concentration is higher.

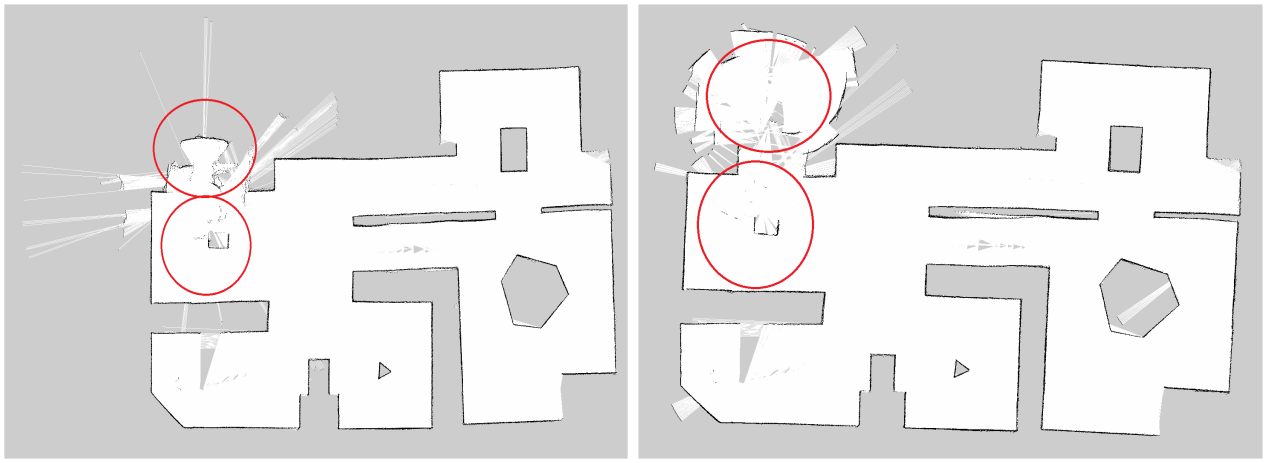
In all trials, when Gmapping is fed solely with laser data, *i.e.*, without the SmokeNav layer, it is unable to map zones corrupted with smoke. Even when the robot leaves the smoky zone, the estimation of the robot's pose becomes erroneous and the algorithm is not able to correctly recover and update the pose. This is no surprise since it is only uses the LRF and the scan matching process fails due to the false readings.

From the analysis of the results, it can be seen that the developed SLAM technique has successfully mapped the arena even when the smoke density was high. However, when the switch from the LRF to sonars occurs, the quality of the map drastically decreases, as expected. This happens not only because of the low resolution of the sonar, but also due to the previously mentioned misbehavior of the sensor. This is verified in Figure 5.14, where



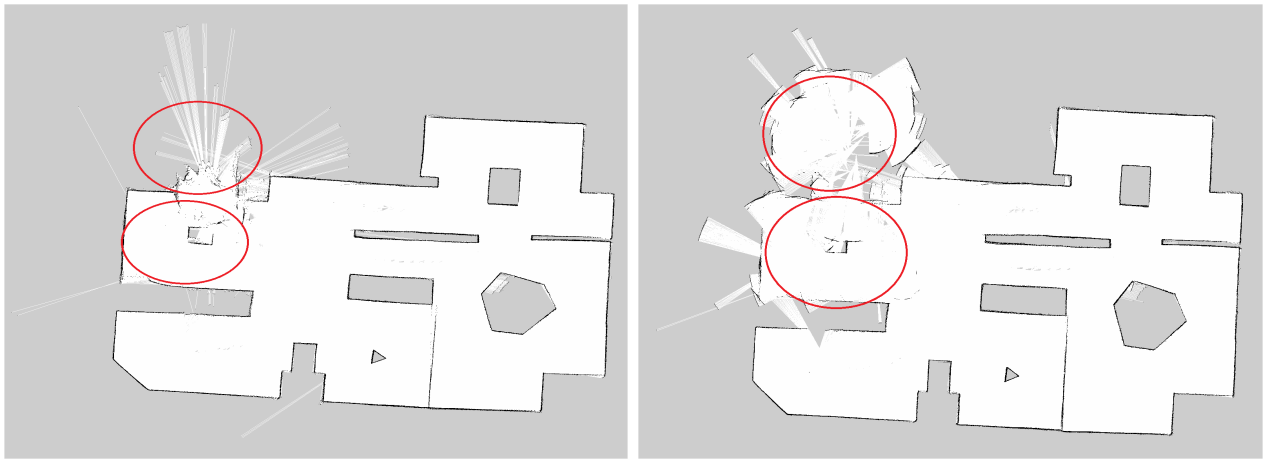
(a) Gmapping only with LRF data. (b) Gmapping with the SmokeNav layer.

Figure 5.14: Results of the first trial. Red circles denote the zones where smoke was injected.



(a) Gmapping only with LRF data. (b) Gmapping with the SmokeNav layer.

Figure 5.15: Results of the second trial. Red circles denote the zones where smoke was injected.



(a) Gmapping only with LRF data. (b) Gmapping with the SmokeNav layer.

Figure 5.16: Results of the third trial. Red circles denote the zones where smoke was injected.

the robot crossed high density smoke zones moments after the beginning of the mapping task. In the remaining trials, the smoke was injected only in the latest zone to be mapped. For this reason, Gmapping using only LRF data, did not perform as poorly as one could expect. Nevertheless, in the second and third trial, the impact of smoke in the mapping task is visible. The upper left corner was not correctly mapped. The smoke density was so high, that it formed an almost indistinguishable obstacle, and the number of false positives is very high. However, when Gmapping is fed with sonar data, that zone was successfully mapped, but the quality of the mapping is quite low. Yet, the improvements are significant and it has been proved that SLAM in such conditions is possible using a multi-sensor approach. Also, the developed layer does not introduce significant delays in the system and the solution proposed is based on a "low-cost" philosophy, using only off-the-shelf, fairly cheap sensors. Better results would be obtained using sonars with stable readings (like those in the Nomad Scout). Beyond that, it is the author's belief that a solution based on a radar sensor (instead of sonars) would be ideal to solve the problem while maintaining maps of high quality, independently of the smoke density in the environment.

The work presented in this chapter it was submitted and currently under review for oral presentation and publication on the Proceedings of the IEEE International Conference on Robotics and Automation, which will take place in Hong Kong, China in June, 2014 [SPR14].

Chapter 6

Conclusion and Future Work

The advances in mobile robotics led to an exponential growth in SLAM research. New algorithms and solutions for the most known issues are constantly being presented. However, there is a lack of research in situations where the most common sensors fail. This dissertation addresses one of these cases: SLAM in reduced visibility scenarios.

In this final chapter, an overview of the proposed approach takes place in order to sum up the work and summarize its main discussions. Also, possible future work directions are discussed.

6.1 SmokeNav Algorithm Overview

In this dissertation, a 2D SLAM technique for low visibility scenarios was proposed. This technique makes use of multiple sensors in order to successfully map environments corrupted by smoke/dust particles. It is a simple and easily adaptable approach that can potentially be applied with different 2D SLAM algorithms. Experimental results using a Particle Filter 2D SLAM approach proved its functionality. However, due to hardware limitations, the quality of the obtained maps was comprehensibly not the best. It was proved that using the complementary characteristics of multiple range sensors, it is possible to surpass this reduced visibility situation and benefit from the advantages of the distinct range sensors. Also, the addition of an accurate method to measure the smoke density around the robotic platform increases the efficiency of the algorithm. Previously, this work had provided an overview of several 2D SLAM techniques available in ROS.

6.2 Future Work

Some issues are still left open and correspond to future guidelines that can be followed to improve the current work. A probabilistic technique could be adopted to explicitly model, and assign confidence-related weights to each range sensor according with the dust sensor readings. Replacing the sonar ring by a more reliable or a range sensor with higher resolution and immune to smoke would greatly improve the results. The integration of a visual camera to provide a method to detect smoke in the FOV of the LRF, by measuring the gray intensity in the image, could also be explored. Even though the visual camera is highly disturbed by lightning conditions, it can provide richer information about the environment. It would be interesting to integrate all these changes and validate them through several scenarios with different conditions and more "realistic" smoke. Additionally, an IMU can be integrated in order to increase the accuracy of the localization technique, when the sonar is the only available sensor.

In this work, Gmapping was able to map using only sonars. However, this SLAM approach is not optimized for that. To improve the results, the development of a new SLAM approach optimized for both LRFs and sonars would also be an interesting contribution.

Bibliography

[APCR13] A. Araújo, D. Portugal, M. S. Couceiro, and R. P. Rocha. Integrating Arduino-based Educational Mobile Robots in ROS. In *Proc. of IEEE 13th Int. Conf. on Autonomous Robot Systems and Competitions (Robotica 2013)*, pages 8–13, Lisbon, Portugal, Apr. 2013.

[BPVC12] Christopher Brunner, Thierry Peynot, and Teresa Vidal-Calleja. Automatic Selection of Sensor Modality for Resilient Localisation in Low Visibility Conditions. In *Robotic: Science and Systems (RSS)*, 2012.

[Bre87] J.E. Bresenham. Ambiguities in Incremental Line Rastering. *Computer Graphics and Applications, IEEE*, 7(5):31–43, 1987.

[CACB11] Luca Carlone, Rosario Aragues, José A. Castellanos, and Basilio Bona. A Linear Approximation for Graph-based Simultaneous Localization and Mapping. In Hugh F. Durrant-Whyte, Nicholas Roy, and Pieter Abbeel, editors, *Robotics: Science and Systems*, 2011.

[CM07] A. Chatterjee and F. Matsuno. A Neuro-Fuzzy Assisted Extended Kalman Filter-Based Approach for Simultaneous Localization and Mapping (SLAM) Problems. *Fuzzy Systems, IEEE Transactions on*, 15(5):984–997, 2007.

[CP12] Marcos P Gerardo Castro and Thierry Peynot. Laser-to-radar sensing redundancy for resilient perception in adverse environmental conditions. *ARAA Australasian Conf. on Robotics and Automation, Sydney, Australia*, 2012.

[CPR13] M. S. Couceiro, D. Portugal, and R. P. Rocha. A Collective Robotic Architecture in Search and Rescue Scenarios. In *Proc. of 28th ACM Symposium on Applied Computing (SAC 2013), Special Track on Cooperative Multi-Agent Systems and Applications (CMASA)*, pages 64–69, Coimbra, Portugal, Mar. 2013.

[CZR05] E.V. Cuevas, D. Zaldivar, and R. Rojas. *Particle Filter in Vision Tracking.* Freie Universität Berlin, Fachbereich Mathematik und Informatik / B: Fachbereich Mathematik und Informatik. Freie Univ., Fachbereich Mathematik und Informatik, 2005.

[DJ10] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, 2nd edition, 2010.

[DT10] T. Deissler and J. Thielecke. UWB SLAM with Rao-Blackwellized Monte Carlo data association. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–5, 2010.

[GKSB10] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A Tutorial on Graph-Based SLAM. *IEEE Intell. Transport. Syst. Mag.*, pages 31–43, 2010.

[GSB07] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *Robotics, IEEE Transactions on*, 23(1):34–46, 2007.

[GSGB07] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[HAC05] LTD. Hokuyo Automatic Co. Scannning Laser Range Finder URG-04LX. Data sheet, Hokuyo Automatic Co., LTD., 2005. `http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/urg-04lx/data/URG-04LX_spec_en.pdf`.

[HBFT03] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 206–211 vol.1, 2003.

[HD07] Shoudong Huang and G. Dissanayake. Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM. *Robotics, IEEE Transactions on*, 23(5):1036–1049, 2007.

[KGK+10] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient Sparse Pose Adjustment for 2D mapping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 22–29, 2010.

[KGS+11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, 2011.

[KK08] Lindsay Kleeman and Roman Kuc. Sonar Sensing. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 491–519. Springer Berlin Heidelberg, 2008.

[KSD+09] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[KVSMK11] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 155–160, 2011.

[LCN12] Y. Latif, C. Cadena, and J. Neira. Realizing, reversing, recovering: Incremental robust loop closing over time using the iRRR algorithm. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4211–4217, 2012.

[LM97] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *AUTONOMOUS ROBOTS*, 4:333–349, 1997.

[MSMS11] Jose Marti, Jorge Sales, Raúl Marín, and Pedro Sans. Multi-Sensor Localization and Navigation for Remote Manipulation in Smoky Areas. *International journal of Advanced Robotic Systems*, 10(12):211–2013, 2011.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[NH05] P. Newman and K. Ho. SLAM - Loop Closing with Visually Salient Features. *IEEE International Conference on Robotics and Automation*, 18-22 April 2005.

[OK09] Edwin Olson and Michael Kaess. Evaluating the Performance of Map Optimization Algorithms. In *RSS Workshop on Good Experimental Methodology in Robotics*, June 2009.

[Pir05] G. Pires. Filtro de Partículas - Localização de um robô móvel. Technical report, Institute for Systems and Robotics, DEEC, University of Coimbra, October 2005.

[PMDA08] J. Pascoal, L. Marques, and A.T. De Almeida. Assessment of Laser Range Finders in risky environments. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3533–3538, 2008.

[PR12] D. Portugal and R. P. Rocha. Extracting Topological Information from Grid Maps for Robot Navigation. In *Proc. of 4th Int. Conf. on Agents and Artificial Intelligence (ICAART'2012)*, pages 137–143, Vilamoura, Algarve, Feb. 2012.

[PR13] David Portugal and Rui P. Rocha. Distributed multi-robot patrol: A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*, 2013.

[QCG+09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[Roc05] Rui P. Rocha. *Building Volumetric Maps with Cooperative Mobile Robots and Useful Information Sharing: a Distributed Control Approach based on Entropy.* PhD thesis, Faculty of Engineering of University of Porto, Portugal, Oct. 2005.

[SEH10] B. Steux and O. El Hamzaoui. tinySLAM: A SLAM algorithm in less than 200 lines C-language program. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 1975–1979, 2010.

[Sen08] Inc. SensComp. Series 600 Instrument grade Transducer, 2008. `http://www.senscomp.com/pdfs/series-600-instrument-grade-sensor.pdf`.

[SM12] J. Suarez and R.R. Murphy. Using the Kinect for search and rescue robotics. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–2, 2012.

[SMC+10] Jorge Sales, Raúl Marín, Enric Cervera, Sergio Rodríguez, and Javier Pérez. Multi-Sensor Person Following in Low-Visibility Scenarios. *Sensors*, 10(12):10953–10966, 2010.

[SPR13] J. M. Santos, D. Portugal, and Rui P. Rocha. An Evaluation of 2D SLAM Techniques Availabile in Robot Operating System. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR*, Linköping, Sweden, Oct. 2013.

[SPR14] J. M. Santos, D. Portugal, and Rui P. Rocha. Simultaneous localization and mapping in reduced visibility scenarios. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014.

[TB98] Sebastian Thrun and Arno Bücken. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, 1998.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[TFBD00] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[Thr02] Sebastian Thrun. Learning Occupancy Grids With Forward Sensor Models. *Autonomous Robots*, 15:111–127, 2002.

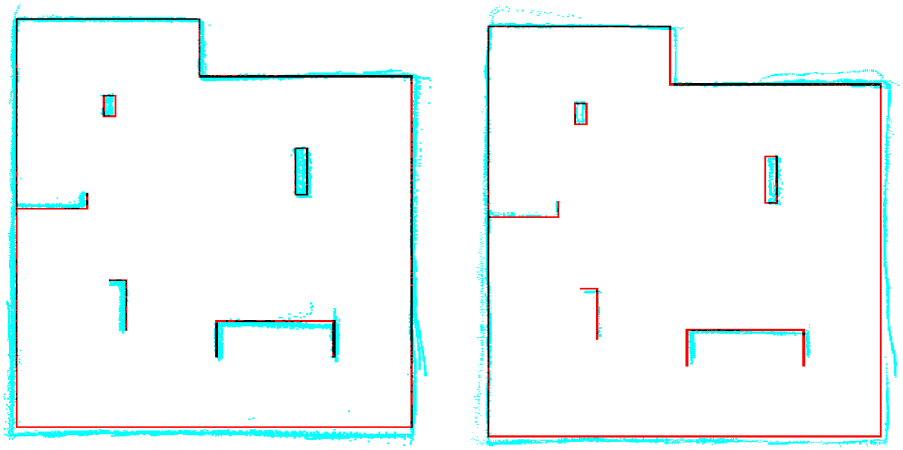
[TM05] S. Thrun and M. Montemerlo. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

[VLE10] Regis Vincent, Benson Limketkai, and Michael Eriksen. Comparison of indoor robot localization techniques in the absence of GPS. *Proc. SPIE*, 7664:76641Z–76641Z–5, 2010.

[VMDAC12] Rafael Valencia, Jaime Valls Miró, Gamini Dissanayake, and Juan Andrade-Cetto. Active Pose SLAM. In *IROS*, pages 1885–1891. IEEE, 2012.

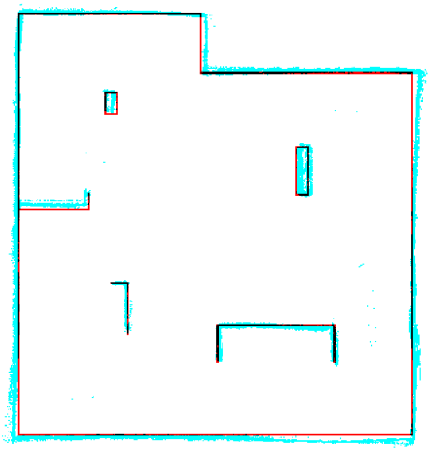
Appendix A

Results of the SLAM Evaluation



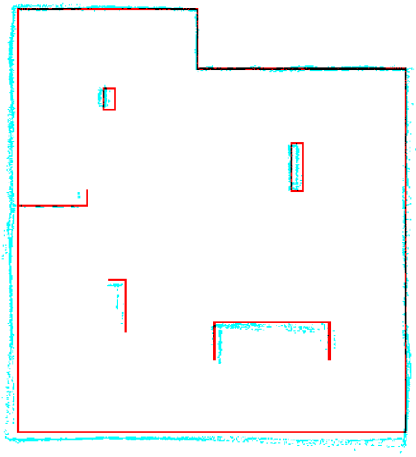
(a) HectorSLAM

(b) Gmapping.



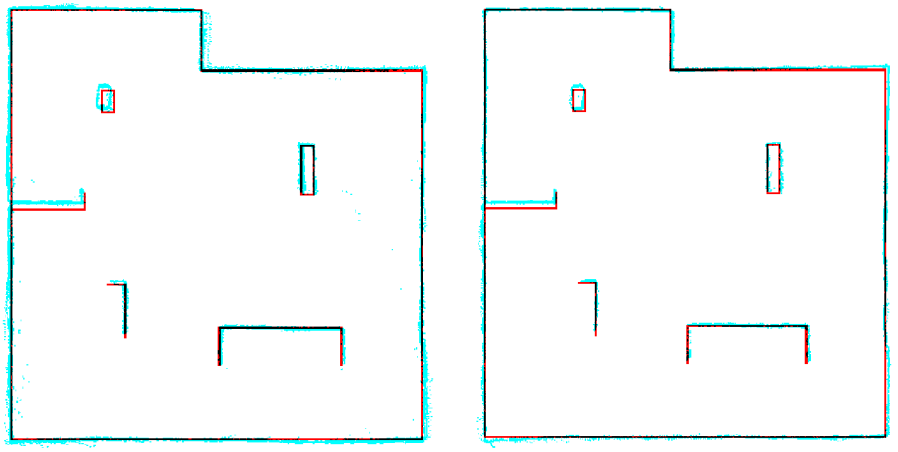
(c) KartoSLAM

(d) CoreSLAM



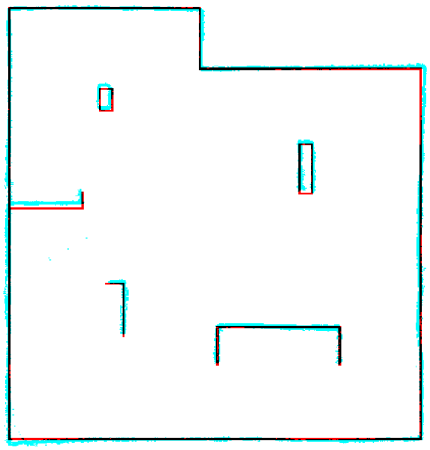
(e) LagoSLAM

Figure A.1: Performance analysis in the real world *MRL Arena* (1st trial). Red represents the ground thuth and blue represents the final map.

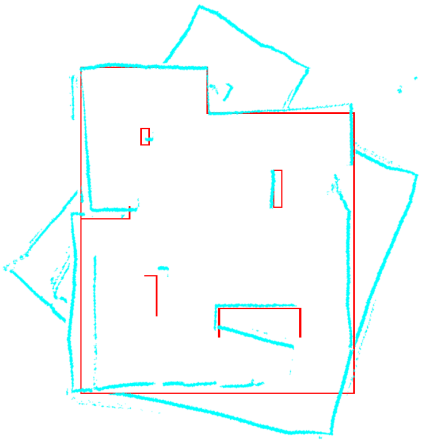


(a) HectorSLAM

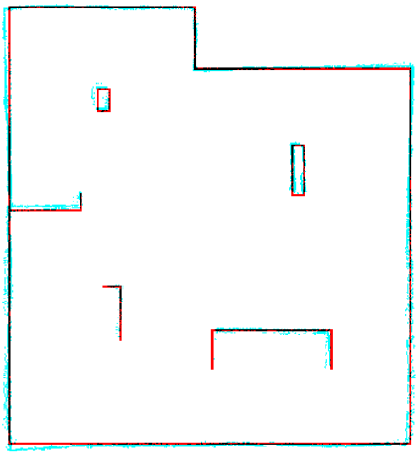
(b) Gmapping.



(c) KartoSLAM

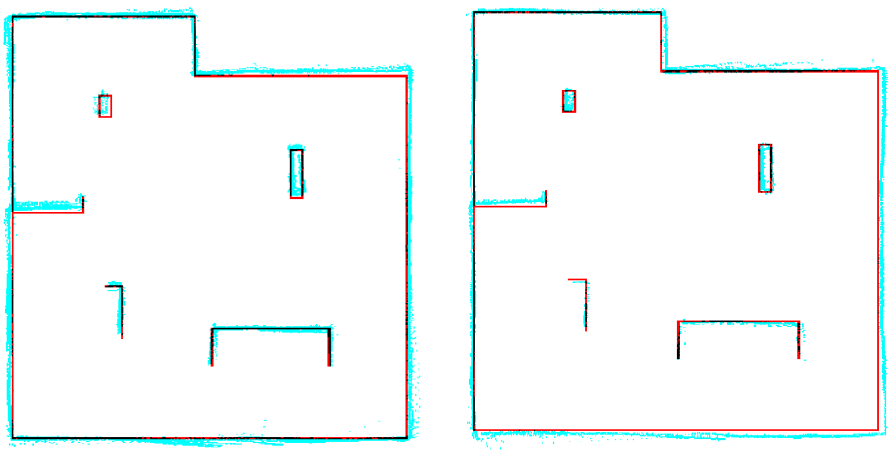


(d) CoreSLAM



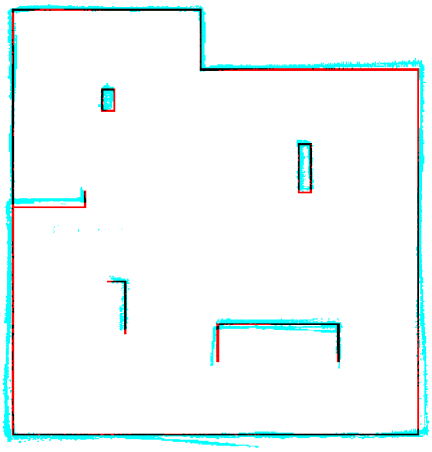
(e) LagoSLAM

Figure A.2: Performance analysis in the real world *MRL Arena* (2nd trial). Red represents the ground thuth and blue represents the final map.

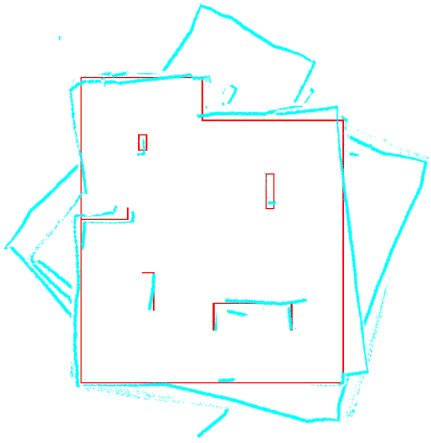


(a) HectorSLAM

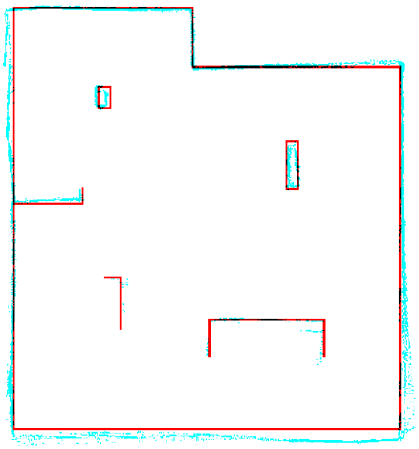
(b) Gmapping.



(c) KartoSLAM



(d) CoreSLAM



(e) LagoSLAM

Figure A.3: Performance analysis in the real world *MRL Arena* (3rd trial). Red represents the ground thuth and blue represents the final map.

Appendix B

Accepted Article in SSRR 2013

An Evaluation of 2D SLAM Techniques Available in Robot Operating System

João Machado Santos, David Portugal and Rui P. Rocha

Abstract— **In this work, a study of several laser-based 2D Simultaneous Localization and Mapping (SLAM) techniques available in Robot Operating System (ROS) is conducted. All the approaches have been evaluated and compared in 2D simulations and real world experiments. In order to draw conclusions on the performance of the tested techniques, the experimental results were collected under the same conditions and a generalized performance metric based on the k-nearest neighbors concept was applied. Moreover, the CPU load of each technique is examined.**

This work provides insight on the weaknesses and strengths of each solution. Such analysis is fundamental to decide which solution to adopt according to the properties of the intended final application.

I. Introduction

Simultaneous Localization and Mapping (SLAM) is one of the most widely researched topics in Robotics. It is useful for building and updating maps within unknown environments, while the robot keeps the information about its location.

Proprioceptive sensors are subject to cumulative errors when estimating the mobile robot's motion, the high dimensionality of the environment that is being mapped, the problem of determining whether sensor measurements taken at different points in time correspond to the same object in the world, and the fact that the world changes over time, represent the biggest challenges in SLAM [1].

The Robot Operating System (ROS) is the most popular robotics framework nowadays. It provides a set of tools, libraries and drivers in order to help develop robot applications with hardware abstraction [2]. ROS enables researchers to quickly and easily perform simulations and real world experiments.

All five SLAM techniques analyzed in this work are available in ROS and have been tested in 2D simulations through Stage and on a custom Arduino-based Robot [3]. The research presented in this article is a first step for our ultimate goal, which is to propose a SLAM technique for Urban Search and Rescue (USAR) scenarios, whose environment often contain smoke and dust particles. Therefore, it is necessary to study the most popular and commonly used approaches and this work will serve as guidance to our later technique, as well as to researchers interested in SLAM and in ROS, in general.

This work has been supported by the CHOPIN research project (PTDC/EEA-CRO/119000/2010), by a PhD grant (SFRH/BD/64426/2009) and by the Institute of Systems and Robotics (project Est-C/EEI/UI0048/2011), all of them funded by the Portuguese science agency "Fundação para a Ciência e a Tecnologia". J.M. Santos, D. Portugal and R.P. Rocha are with the Institute of Systems and Robotics, Univ. of Coimbra, Pólo II, 3030-290 Coimbra, Portugal, email: {jsantos, davidbsp, rprocha}@isr.uc.pt.

II. Related Work

Presently, all recognized algorithms for robot mapping have a common feature: they rely in probabilities. The advantage of applying probabilities is the robustness to measurement noise and the ability to formally represent uncertainty in the measurement and estimation process. Most of the probabilistic models used to solve the problem of mapping rely on Bayes rule [1].

Kalman filters (KF) are one of the most popular implementations of Bayes filters [1]. The KF has two distinct phases: Prediction and Update. The prediction phase estimates the state space (prior) from a previous iteration, while in the update phase the estimated state is combined with observations provided by sensors. The result from the update phase is called posterior. Arising from the prior development of the KF, the Extended Kalman Filter (EKF) solves the problem of nonlinearity in the robot pose model. A set of tests on convergence properties and inconsistency issues of the EKF-based solution to the nonlinear 2D SLAM problem is conducted in [4].

Particle filters (PF) are another application of Bayes filters. The posterior probability is represented by a set of weighted particles and each particle is given an importance factor. It assumes that the next state depends only on the current one, *i.e.*, Markov assumption [5]. PFs have the advantage of representing uncertainty through multi-modal distributions and dealing with non-Gaussian noise. Montemerlo *et al.* [6] proposed a new approach called FastSLAM. It makes use of a modified PF to estimate the posterior. Afterwards, each particle possesses K Kalman filters that estimate the K landmark locations. It was shown that the computational effort to execute this algorithm is lower than EFK approaches. Also, the approach deals with large number of landmarks even with small sets of particles and the results remain appropriate. Also, an approach based on PF is proposed in [7]. This work is discussed in more detail in Section III-B.

Equally important are graph-based SLAM algorithms, as they cover some weaknesses of PFs and EKFs techniques [9]. In these SLAM algorithms, the data extracted is used to build a graph. The graph is composed by nodes and arcs. Each arc in the graph represents a constraint between successive poses, which can be a motion event or a measurement event. In order to obtain a map, all the constraints are linearized and a sparse matrix is obtained, representing the sparse graph. This type of algorithms were first presented by Lu and Milios [8]. In their work, pose constraints were retrieved by the scan matching process. However, due to the optimization process

used, the applicability of the algorithm in large scenarios is impracticable. Thrun *et al.* [9] presented the GraphSLAM algorithm, which is based on [8], and evaluated its behavior in large-scale urban environments. This has been possible due to a reduction process, which removes map variables from the optimization process. In addition, Carlone *et al.* [10] also developed a graph-based SLAM approach, which is discussed in Section III-E.

In the last few years, the number of SLAM approaches has increased and the need to compare different approaches grew significantly. Visually inspection of the resulting maps does not allow a correct comparison. So, the need to precisely evaluate the results asks for a more accurate method - a quantitative scale. For instance, in [12], a metric for comparing SLAM algorithms was developed, wherein the result is not evaluated using a reference, but rather by considering the poses of the robot during data acquisition. This fact allows comparison between algorithms with different outputs. Also, the proposed method is independent on the sensor configuration of the mobile robot, but it requires manual editing of the dataset before being applied.

All recognized SLAM evaluation methods rely on standard datasets available to the community. However, these are not compatible with ROS framework yet. Conversely, in this work, a study of the main laser-based 2D SLAM algorithms that are available in ROS is presented. All the tested techniques use occupancy grids as the final output, which are analyzed using a metric for map similarities. The focus is put on the map quality instead of the pose estimation errors, since the mapping output is highly affected by localization issues. The main goal is to provide an overview of the strengths and weaknesses of all five algorithms implemented in ROS and also to provide a simple, yet accurate quantitative comparison, thus defining general guidelines for ROS users to select the algorithm that best fits their requirements.

III. 2D SLAM Algorithms

In this section, a brief description of five SLAM techniques is conducted, namely: HectorSLAM, Gmapping, KartoSLAM, CoreSLAM and LagoSLAM.

A. HectorSLAM

HectorSLAM[1] combines a 2D SLAM system based on robust scan matching and 3D navigation technique using an inertial sensing system [11].

The authors have focused on the estimation of the robot movement in real-time, making use of the high update rate and the low distance measurement noise from modern LIDARs. The odometric information is not used, which gives the possibility to implement this approach in aerial robots like, a Quadrotor UAV or in ground robots operating in uneven terrains. On the other hand, it might have problems when only low rate scans are available and it does not leverage when odometry estimates are fairly accurate. The 2D pose estimation is based on optimization of the alignment of beam endpoints with the map obtained so far. The endpoints are projected in the actual map and the occupancy probabilities are estimated. Scan matching is solved using a Gaussian-Newton equation, which finds the rigid transformation that best fits the laser beams with the map. In addition, a multi-resolution map representation is used, to avoid getting stuck in local minima. Finally, the 3D state estimation for the navigation filter is based on EKF. However, this is only needed when an Inertial Measurement Unit (IMU) is present, such as in the case of aerial robots. Thus, it will not be used in this work.

[1] http://www.ros.org/wiki/hector_slam

B. Gmapping

Gmapping[2] is a laser-based SLAM algorithm as described by [7]. Furthermore, it is the most widely used SLAM package in robots worldwide. This algorithm has been proposed by Grisetti *et al.* and is a Rao-Blackwellized PF SLAM approach. The PF family of algorithms usually requires a high number of particles to obtain good results, which increases its computational complexity. Also, the depletion problem[3] associated with the PF resampling process decreases the algorithm accuracy. This happens because the importance weights of particles may become insignificant. Hence, this means that there is a small probability that correct hypothesis can be eliminated.

An adaptive resampling technique has been developed in [7], which minimizes the particle depletion problem, since this process is only performed when is needed. The authors also proposed a way to compute an accurate distribution by taking into account not only the movement of the robotic platform, but also the most recent observations. This decreases the uncertainty about the robot's pose in the prediction step of the PF. As a consequence, the number of particles required is decreased since the uncertainty is lower, due to the scan matching process. In our experiments, the number of particles used by Gmapping was 30.

C. KartoSLAM

KartoSLAM[4] is a graph-based SLAM approach developed by SRI International's Karto Robotics, which has been extended for ROS by using a highly-optimized and non-iterative Cholesky matrix decomposition for sparse linear systems as its solver [13]. A graph-based SLAM algorithm represents the map by means of graphs. In this case, each node represents a pose of the robot along its trajectory and a set of sensor measurements. These are connected by arcs which represent the motion between successive poses. For each new node, the map is computed by finding the spatial configuration of the nodes which are consistent with constraints from the arcs. In the KartoSLAM version available for ROS, the Sparse Pose Adjustment (SPA) is responsible for both scan matching and loop-closure procedures [14]. The higher the number of landmarks, the more

[2] http://www.ros.org/wiki/gmapping

[3] The particle depletion problem consists in the elimination of a large number of particles from the sample set during the resampling stage.

[4] http://www.ros.org/wiki/karto

amount of memory is required. However, graph-based SLAM algorithms are usually more efficient than other approaches when maintaining a map of a large-scale environments. In the particular case of KartoSLAM, it is extremely efficient, since it only maintains a pose graph.

D. CoreSLAM

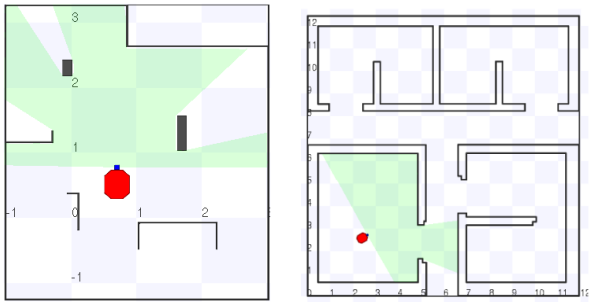
CoreSLAM[5] is a ROS wrapper for the original 200-lines-of-code tinySLAM algorithm, which was created with the purpose of being simple and easy to understand with minimum loss of performance [15]. The algorithm is divided in two different steps: distance calculation and update of the map. In the first step, for each incoming scan, it calculates the distance based on a very simple PF algorithm. The PF matches each scan from the LRF with the map and each particle of the filter represents a possible pose of the robot and has an associated weight, which depends on previous iterations. After the selection of the best hypotheses, the particles with lower weight are eliminated and new particles are generated. In the update step, the lines corresponding to the received scans are drawn in the map. However, of drawing a single point when an obstacle is detected, tinySLAM draws an adjustable set of points surrounding the obstacle.

E. LagoSLAM

The basis of graph-based SLAM algorithms is the minimization of a nonlinear non-convex cost function [10]. More precisely, at each iteration, a local convex approximation of the initial problem is solved in order to update the graph configuration. The process is repeated until a local minimum of the cost function is reached. However, this optimization process is highly dependent on an initial guess to converge. Carlone *et al.* [10] developed a new approach called LagoSLAM[6] (Linear Approximation for Graph Optimization), in which the optimization process requires no initial guess. In addition, the technique can be used with any standard optimizer. In fact, the algorithm available in ROS has the possibility to choose between three different optimizers: Tree-based netwORk Optimizer (TORO)[7], g2o [16] and LAGO [10]. In the experiments conducted, the LAGO optimizer was used. Assuming that the relative position and orientation are independent for each node in the graph, the authors solve a system of equations equivalent to the non-convex cost function. To this end, a set of procedures based on graph theory were presented to obtain a first order approximation of the non-linear system, by means of a linear orientation and a linear position estimation.

IV. Results & Discussion

All five SLAM techniques described were tested using 2D simulations and real world experiments. Simulations were performed in Stage[8], which is a realistic 2D robot simulator integrated in ROS. Additionally, tests were also conducted with a physical robot in a real world scenario, so as to study the behavior of these SLAM packages in the absence of perfect simulated conditions. Despite having perfect conditions in Stage simulations, like noise free odometric and range sensing information, SLAM algorithms assume measurement uncertainty, which may not lead to perfect results. In all experiments, the robot was teleoperated. Note that the abstraction layer provided by ROS allows to use the same code for both simulation and real experiments. HectorSLAM requires a LRF with high update rates. The update rate of the Hokuyo URG-04LX-UG01 LRF used in the experiments is 10 Hz and Stage uses a similar maximum update rate. In order to deal with this, the robot was driven with low angular and linear speed. In the tests that were conducted, the output of each approach, described previously, was the respective generated 2D occupancy grid map.



(a) MRL Arena (4.57×4.04m) (b) 1r5map (12.24×11.67m)

Fig. 1: Maps used in the simulation experiments.

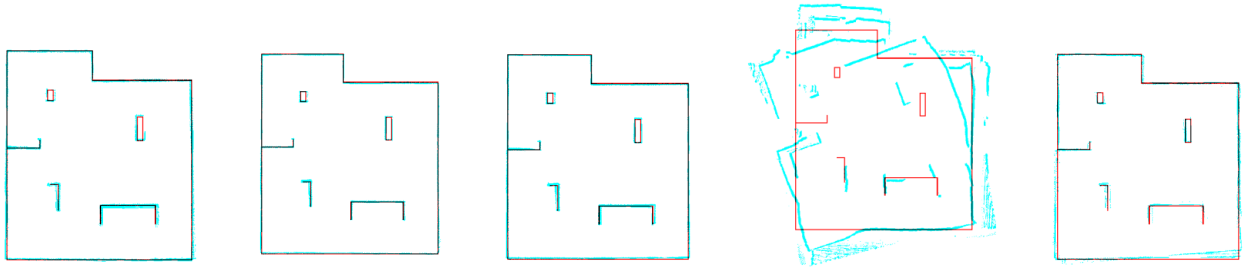
TABLE I: Error estimation for each algorithm in the MRL Arena (Simulation Experiments).

Simulation Experiments				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
0.4563	0.4200	0.5509	11.8393	1.4646

To evaluate the quality of the maps obtained, an analysis of the error between the generated map and the ground truth was conducted. A performance metric based on the k-nearest neighbor concept was used. To that end, the best fit alignment between the ground truth and the map obtained is computed (see Fig. 2), using intensity-based image registration tools.

The process works as follows: the resulting map of each algorithm is binarized. The binarized map only contains boundaries and obstacles of the scenario. Afterwards, the binarized map is aligned narrowly with the respective ground truth using a set of Matlab functions available in the Image Processing Toolbox. Since both ground truth map and the generated map are aligned, the distance from each occupied cell of the ground truth map to the nearest cell in the resulting map is determined using *knnsearch*, which computes the k-nearest neighbor cells (in this case $k = 1$). The sum of all distances obtained is then divided by the number of occupied cells in the ground truth map. This error metric provides a normalized measure of distance (in terms of cells), which can be applied in any generic occupancy grid, as long as the ground truth map is available.

[5] http://www.ros.org/wiki/coreslam
[6] https://github.com/rrg-polito/rrg-polito-ros-pkg
[7] http://www.openslam.org/toro.html
[8] http://www.ros.org/wiki/stage



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Fig. 2: Maps obtained through simulation in the MRL arena environment. Red represents the ground truth and blue represents the final map.

A. Simulation Tests

Stage simulations were performed using two different maps: the *MRL Arena* and the *1r5map*, which are shown in Fig. 1. Special focus is given to the former since the MRL arena is used in both simulation and real world experiments. The 1r5map enables the analysis of the behavior of the SLAM techniques in a larger scenario and with less features per square foot (*cf.* Fig. 1b). This is particularly important to analyze the dependency on landmarks of each approach.

In the simulation experiments, the model of the range sensor was defined just like the sensor used in real world experiments: the Hokuyo URG-04LX-UG01, which has a maximum range of about 5.6 meters. Teleoperation was executed using the keyboard[9]. All the sensing and actuator data from the robot (LFR, odometry, velocity commands, etc.) was recorded previously and then played back for each algorithm. Thus, all SLAM packages were tested under the same exact conditions. This was only possible due to the *rosbag* tool[10].

For each algorithm, the resolution of the final map was set to 0.01 meters/pixel. In order to mitigate the low scanning rate, the number of sub-maps used in HectorSLAM was defined as 5. Since, each sub-map has half resolution of its precedent sub-map, the scan matching process is more accurate, *i.e.*, the scan matching performance is higher in lower resolution maps. In all experiments, the default parameters were used. For example, as mentioned before, the number of particle for the Gmapping algorithm was 30.

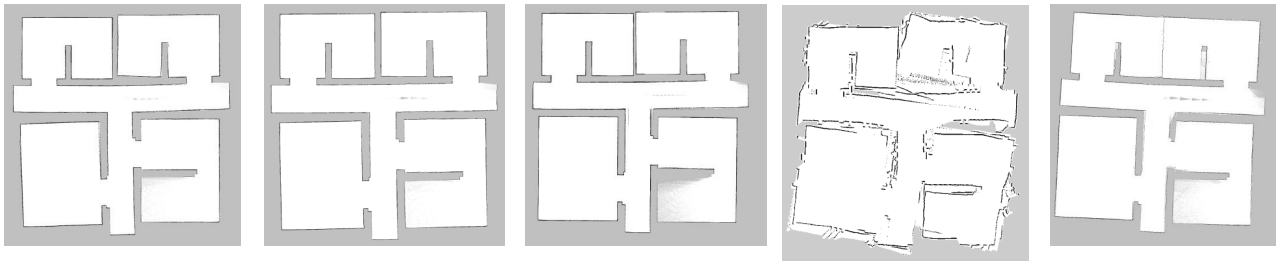
Analyzing the simulations results in the MRL arena, and according to Table I and Fig. 2, Gmapping and HectorSLAM generated the map with lowest and similar error. On the other hand, KartoSLAM presented a slightly greater error, while the results of CoreSLAM presented the highest error value. Gmapping is an extremely optimized PF algorithm with an improved resampling process, and this justifies the quality of the resulting map. Also, the scan matching process of HectorSLAM showed its efficiency. Nevertheless, it must be noted that the low speed commands given to the robot, in order to compensate the rate update from the LFR, have some influence in the results. Since both KartoSLAM and LagoSLAM are graph-based SLAM approaches, comparing the error between them is interesting. Both mapped successfully the arena. However, LagoSLAM obtained the greatest error (excluding CoreSLAM). which can be explained by the impressive performance of the SPA solver method that KartoSLAM employs. Nevertheless, the quality of the resulting map obtained with LagoSLAM is still appropriate.

[9]http://www.ros.org/wiki/teleop_twist_keyboard
[10]http://www.ros.org/wiki/rosbag

TABLE II: Error estimation for each algorithm in the 1r5map.

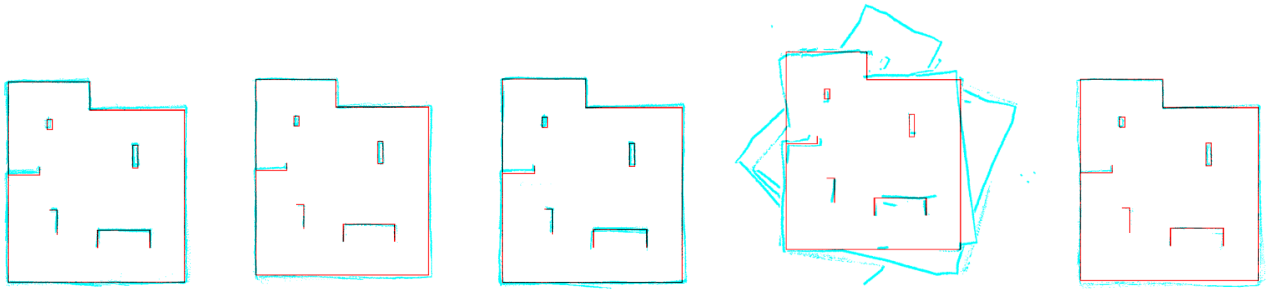
Simulation Experiments				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
7.4581	5.3670	5.4380	171.5218	9.3041

In order to compare all the SLAM approaches in a different scenario, a series of simulations using 1r5map were also conducted. These results are shown in Table II Fig. 3. The 1r5map is a relatively large map with a low number of distinctive landmarks. In this case, HectorSLAM obtained a higher error value than Gmapping. One of the reasons is the fact that HectorSLAM relies largely in scan matching between successive measurements. The full potential of HectorSLAM could not be observed due to the properties of the sensor used in these simulation experiments. Beyond that, due to the reduced number of landmarks, the error grows continuously, since the scan matching process is not fed with enough information. Additionally, since it is not using odometry information, a few issues arise when traversing long corridors with fixed width. As a consequence, the inferior result obtained with HectorSLAM in this test are not surprising. Once again, the Gmapping algorithm presents exceptional results, which reveal the accuracy of PF approaches. KartoSLAM revealed the robustness of graph-based SLAM approaches, since it obtained the second lowest error value. Once again, LagoSLAM obtained an higher error value than KartoSLAM and CoreSLAM was the worst performing algorithm. Since the error values are obtained via the euclidean distance between points in the ground truth and the nearest point in the map, the errors obtained in the 1r5map map are greater than in the other experiments due to the larger dimensions of the map, this is particularly visible in the case of CoreSLAM.



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Fig. 3: Occupancy Grid Maps obtained through simulation in the 1r5map environment.



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Fig. 4: Performance Analysis in the real world. Red represents the ground truth and blue represents the final map.

TABLE III: Error estimation for each algorithm in the MRL Arena (Real World Experiments).

Real World Experiments				
HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
1.1972	2.1716	1.0318	14.75333	3.0264
0.5094	0.6945	0.3742	7.9463	0.8181
1.0656	1.6354	0.9080	7.5824	2.5236

B. Real World Tests

In the real world experiments, three runs with different trajectories and initial positions were performed using a Stingbot[11] robot [3], equipped with an Hokuyo URG-04LX-UG01 and an Asus eeePC 1025C, running Ubuntu 11.10 and ROS Fuerte. Once again, all the data was previously recorded and subsequently played back for each algorithm. Tests were conducted at the real-world MRL arena. The algorithm parameters used in the simulation experiments were again adopted.

Fig. 4 shows that all five techniques were able to map the scenario successfully. The error obtained for each algorithm is shown in Table III. As can be seen, in general all techniques led to worse results than in simulation. This slight performance hit is due to the existence of estimation errors in the robot position and noise on the laser scanning data, while mapping the real world MRL arena. An analysis of the error can give a more accurate information about the performance of the algorithms.

Despite the differences between virtual and real world environments, the results extracted from both setups follow some general trends, in particular for HectorSLAM, Gmapping and LagoSLAM. According to the authors of [15], CoreSLAM can have great performance in several disparate environments; however this claim is not backed up by the results extracted from our experiments.

In the KartoSLAM algorithm, the error obtained in the real world experiments was not much larger than the error in simulations. In fact, generally KartoSLAM was the best performing technique in the real world, being less affected by noise than the other methods. This can be explained, not only due to the performance of the SPA solver used in KartoSLAM, but also because it is a full SLAM approach, *i.e.* the map is obtained using the entire path and map and not only the most recent map and pose. The lower results of CoreSLAM in all experiments showed that its loop closure procedure rarely converges. This is clear in the video that shows a real world experiment and all the detailled results[12].

Beyond the error analysis conducted, an evaluation of the computational load using each technique was carried out. A comparison of the CPU load in a Laptop equipped with an Intel Core i7-3630QM and 8Gb of RAM running each algorithm is presented in Fig. 5 and Table IV.

Looking closely at the results, LagoSLAM presented the highest percentages of CPU usage. Moreover, the values obtained are quite distant from the other four algorithms. This can be explained by the process developed to achieve the minimum cost function for the given graph configuration, as referred in Section III-E. The resources needed by the

[11] `http://www.ros.org/wiki/mrl_robots`

[12] Available at: `http://goo.gl/IMTKmt`

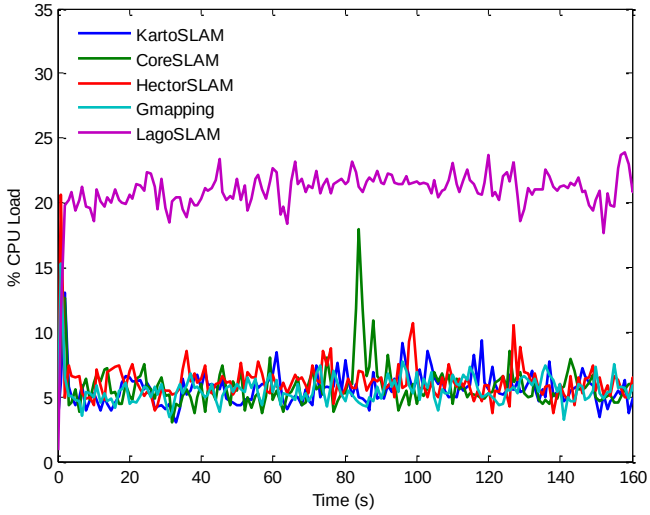


Fig. 5: Evolution of the CPU load of each SLAM method using a real world dataset.

TABLE IV: CPU Load (%) of the 2D SLAM approaches: mean ($\bar{x}$), median ($\tilde{x}$) and standard deviation (σ) values.

	HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
$\bar{x}$	6.1107	7.0873	5.4077	5.5213	21.0839
$\tilde{x}$	5.9250	5.5800	5.3000	5.4400	21.2250
σ	1.993	4.4287	1.3018	1.6311	2.1684

other four approaches during the experiments are similar, as seen in Table IV. This CPU analysis reveals that all five algorithms analyzed are quite efficient in terms of resources required and can be adopted online, during field experiments, to map generic 2D scenarios.

C. Discussion

According with our experiments, some ideas can be retained. On one hand HectorSLAM relies only in scan matching and it does not make use of odometry, which could be an advantage or disadvantage depending on the robot and the environment's characteristics. On the other hand, ideally it should be tested with specific hardware such as a high rate LFR.

Gmapping showed its robustness in all experiments, since in every experiment the error and CPU load always remained low. It combines both scan matching and odometry in order to minimize the number of particles.

Both KartoSLAM and LagoSLAM are graph-based SLAM approaches, but their results were distinctively different. KartoSLAM provided accurate maps with lower CPU load, while LagoSLAM generated maps with higher error and CPU load. The reasons behind such discrepancies are related with the distinct processes of graph configuration and graph optimization of the two techniques.

Lastly, CoreSLAM achieved the less impressive results and it is possible to denote a lack of convergence in its loop closure mechanism. CoreSLAM uses a simple PF which requires more particles, but has a lower computation power associated to each particle. According to [15], CoreSLAM uses a very simple PF to match LFR readings with the map, which could lead to an erroneous position estimation. Additionally, the focus of the original work was to provide a simple SLAM technique with the ability to navigate within long corridors without losing its location, and not the loop closing system.

V. Conclusions

In this work, five representative 2D SLAM algorithms available in ROS were tested through simulations and in real world experiments. A discussion of the weaknesses and strengths of each solution has been done. An accurate overview of each of the 2D SLAM techniques available for ROS was provided to shed light on the choice of an approach according to one's requirements.

In future work, we intend to develop a 2D SLAM technique in ROS for low visibility indoor scenarios, *e.g.*, due to smoke. This new technique will possibly adapt Gmapping or KartoSLAM, due to the observed performance in this article, and extend them with more sensing input information beyond LRFs; *e.g.*, sonars, IMUs and/or a dust sensor.

References

[1] S. Thrun., W. Burgard, D. Fox., *Probabilistic Robotics*, MIT Press, 2005.

[2] M. Quigley et al., ROS: an open-source Robot Operating System. *In IEEE International Conference on Robotics and Automation(ICRA)*, Workshop on Open Source Software, 2009.

[3] A. Araújo, D. Portugal, M. Couceiro and R. P. Rocha. Integrating Arduino-based Educational Mobile Robots in ROS, In *Int. Conf. on Autonomous Robot Systems*, Lisbon, Portugal, April 25-29, 2013.

[4] S. Huang, G. Dissanayake. Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM, In *IEEE Trans. on Robotics*, 2(5), Oct. 2007.

[5] S. Thrun, D. Fox, W. Bugard, F. Dellaert, Robust Monte Carlo Localization for Mobile Robots, In *Artificial Inteligence*, 128, 2001.

[6] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, In *AAAI National Conference on Artificial Intelligence*, 2002.

[7] G. Grisetti, C. Stachniss, W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters, In *Trans. on Robotics* , 23(1), Feb. 2007.

[8] F. Lu, E. Milios, Globally Consistent Range Scan Alignment for Environment Mapping, In *Autonomous Robots*, 1997.

[9] S. Thrun, M. Montemerlo. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures, In Proc. of the *Int. Journal on Robotics Research*, 2005.

[10] L. Carlone, R. Aragues, J.A. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping, In Proc. of the *Int. Conf. Robotics: Science and Systems*, 2011.

[11] S. Kohlbrecher, J. Meyer, O. Von Stryk, U. Klingauf. A Flexible and Scalable SLAM System with Full 3D Motion Estimation, In the *Int. Symp. on Safety, Security and Rescue Robotics (SSRR)*, Nov. 2011.

[12] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, A. Kleiner. On Measuring the Accuracy of SLAM Algorithms, *Autonomous Robots*, 27(4), Nov. 2009.

[13] R. Vincent, B. Limketkai, M. Eriksen. Comparison of indoor robot localization techniques in the absence of GPS, In Proc. of *SPIE*: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV of Defense, Security, and Sensing Symposium, April 2010.

[14] K. Konolige, G. Grisetti, R. Kümmerle, B. Limketkai, R. Vincent, Efficient Sparse Pose Adjustment for 2D Mapping, In Proc. of *Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2010.

[15] B. Steux, O. El Hamzaoui. tinySLAM: A SLAM algorithm in less than 200 lines C-language program, In Proc. of the *Int. Conf. on Control Automation Robotics & Vision (ICARCV)*, Dec. 2010.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard: g2o: A General Framework for Graph Optimization, *IEEE International Conference on Robotics and Automation* (ICRA), 2011.