PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University of Amar Telidji - Laghouat

Faculty of Technology

Department of Electronics

MASTER THESIS

DOMAIN: Science & Technology

FIELD: Electronic

SPECIALTY: Embedded System

Abdelhadi Bousaid & **Betaimi youcef**

Theme

Lidar odometry and mapping for Pioneer robot

Jury members:

Djerfaf Fatima	Prof	President
Feknous Safia	MAA	Examiner
Chouireb Fatima	Prof	Supervisor
Benchrif Aissa	Docteur	Co-Supervisor

2022 / 2023

DEDICATION

I would like to express my heartfelt gratitude to Allah for His blessings and guidance throughout my life. I am truly grateful for His unwavering support and love.

I would also like to extend my deepest appreciation to my beloved late grandfather, Laid Bousaid. May God bless his soul and grant him eternal peace. His wisdom, kindness, and presence have had a profound impact on my life, and I am forever grateful for the memories we shared.

Furthermore, I would like to thank my other grandfather, Taher Bensmaine, for his constant love, care, and support. His guidance and encouragement have played an instrumental role in shaping the person I am today.

To my wonderful family, Said Bousaid, Mama Bensmaine, A. Bousaid, L. Bousaid, and T. Bousaid, I am immensely thankful for your unconditional love and unwavering belief in me. Your support has been a constant source of strength and motivation throughout my journey.

Last but not least, a special mention goes to my dear friend, A. Ghobsi. Your friendship has been a true blessing in my life. Your unwavering support, laughter, and shared memories have made this journey all the more beautiful.

To all those mentioned above, I am deeply grateful for your presence, love, and support. You have enriched my life in ways beyond measure, and I am forever indebted to each and every one of you. May Allah bless you all abundantly.

Abdelhadi Bousaid

DEDICATION

I dedicate this modest work: To My parents Fadila and ahmed symbols of courage and will, who have dedicated their lives to my well-being. To my little sister I wish her success in baccalaureate.

To my older sister and her husband and daughters Hanine and Jouairiyaa To all my friends without exception To our supervisor who facilitated the success of the project To all those who have contributed from near or far to my arrival at this moment.

To my precious grandmother,i wish her all the best

Youcef Betaimi

ACKNOWLEDGMENTS

We thank above all Almighty God who gave us strength, courage and will to do this work. We would like to thank our supervisor Ms. F. Chouireb and our co-supervisor Mr . A.Benchrif for accepting us during this year and for their follow-up, guidance and valuable advice. They were able to share their great experience with us.

We thank the jury members who agreed to review this work. We We would also like to express our thanks to all the teachers who helped us during our university degree.

A big thank you to all our families especially my mother and father for their presence, their concern and the concern that they are made for us, their encouragement and their follow-up, with patience, of the unfolding of our project.

Finally, We warmly thank all those who have contributed directly or indirectly to the realization of this work.

مـــــــلخص

واحدة من أهم المشاكل التي يجب حلها في روبوتات التنقل هي مسألة معرفة مكان الروبوت وشكل العالم من حوله. يطلق على هذه المشكلة إسم **SLAM** .

في هذه الأطروحة، نحن نهتم بنظام **SLAM** بإستخدام حساس الليدار كأداة للتصور.

أولا، نقدم نظرة عامة على **LIDAR BASED SLAM** و أهميته في أنضمة الروبوتات ذاتية التحكم ثم ، نقوم بتحليل مفصل لحساس قياس الليزر **SICK LMS 5XX** المستخدم في هذا العمل، بما في ذلك مبادئ تشغيل و قدرات قراءة البيانات والتكوين بإستخدام اداة **SOPAS ET** .

بعد ذلك، نستكشف في هذه الأطروحة تنفيذ **SLAM** بإستخدام نظام تشغيل الروبوت **ROS** .

يغطي البحث تثبييت و إستخدام جميع الحزم الضرورية، و يقدم دليلا شاملا حول كيفية إعدادها و تشغيلها. ثم يتم تقييم فعالية تنفيذ **SLAM** من خلال المحاكات و التجارب في العالم الحقيقي عل روبوت **PIONEER 3-DX**

الكـــلمات الرئـــيسية:

التوطين المتزامن ورسم الخرائط **SLAM** ، الروبوتات المتنقلة، **Pioneer 3-DX, SOPAS ET, Sick LMS5xx, Robot Operating System (ROS).**

ABSTRACT

One of the most important problems to be solved in mobile robotics is the question of where the robot is, and what the world around it, looks like. This is called Simultaneous Localization and Mapping (SLAM) problem.

In this thesis, we are interested in the SLAM system using the Lidar sensor as a perception tool. First, we start by providing an overview of Lidar-Based SLAM and its significance in autonomous robotic systems. After that, we delve into the detailed analysis of the Sick LMS5xx laser measurement sensor used in this work, including its operational principles, data reading capabilities, and configuration using the SOPAS Engineering Tool. Then, the thesis explores the implementation of SLAM using the Robot Operating System (ROS).

It covers the installation and utilization of all necessary packages, providing a comprehensive guide on how to set up and run them. The effectiveness of the SLAM implementation is then evaluated through simulations and real-world experiments on real Pioneer-3DX mobile robot.

Keywords: Simultaneous Localization And Mapping (SLAM) , Mobile Robots , Pioneer 3-DX , SOPAS ET , Sick LMS5xx , Robot Operating System (ROS).

RÉSUMÉ

L'un des problèmes les plus fondamentaux à résoudre dans la robotique mobile est de savoir où se trouve le robot et à quoi ressemble le monde qui l'entoure. C'est ce qu'on appelle problème de localisation et cartographie simultanées (SLAM). Dans ce mémoire, on s'intéresse au système SLAM utilisant le capteur Lidar comme outil de perception. On commence tout d'abords par donner un aperçu sur ce type de système et son importance dans le fonctionnement autonome des robots mobiles. On présente ensuite, une analyse détaillée du capteur laser SICK LMS5xx utilisé dans ce travail, y compris ses principes de fonctionnement et ses capacités de lecture des données, ainsi que sa configuration à l'aide de l'outil logiciel SOPAS. La mise en œuvre du système SLAM en utilisant le Robot Operating System (ROS) est ensuite détaillée. Elle couvre l'installation et l'utilisation de tous les paquets nécessaires, fournissant ainsi un guide complet sur la façon de les configurer et de les exécuter. L'efficacité de la mise en œuvre du SLAM est enfin évaluée au moyen de simulations et d'expériences réelles sur le robot mobile Pioneer-3DX.

Mots-clés: Localisation et cartographie simultanées (SLAM), Robots mobiles, Pioneer 3-DX, SOPAS ET, Sick LMS5xx, Système d'exploitation robotique (ROS).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SOPAS ET	Sick Open Portal Application Software Engineering Tool
SLAM	Simultaneous Localization and Mapping
LIDAR	Light Detection and Ranging
Gmapping	Gridmapping
P-3DX	Pioneer-3DX
ROS	Robot Operation System
RViz	ROS Visualization
EKF	Extended Kalman Filter

GENERAL INTRODUCTION

Mobile robots have gained significant attention in various fields, including manufacturing, logistics, agriculture, search and rescue operations. The ability of these robots to navigate and interact autonomously in dynamic environments is crucial for their successful deployment. To achieve such autonomy, mobile robots must be equipped with perception systems that allow them to understand and interpret their surroundings accurately [2].

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics that addresses the challenge of building a map of an unknown environment while simultaneously estimating the robot's pose within that environment. By combining sensor measurements and motion models, SLAM algorithms enable mobile robots to create maps of their surroundings and determine their own position relative to the map.

One of the key technologies used in SLAM is Light Detection and Ranging (Lidar), which is a remote sensing method that measures distances by illuminating the environment with laser beams and analyzing the reflected light. Lidar sensors provide accurate and dense 2D or 3D point cloud data, making them highly suitable for mapping and localization tasks in mobile robotics.

This thesis focuses on Lidar-based SLAM for mobile robots, with a specific emphasis on the Sick LMS5xx laser measurement sensor. The LMS5xx sensor is widely used in robotics due to its high precision, reliability, and versatility. Understanding the operation and capabilities of this sensor is essential for effectively implementing Lidar-based SLAM algorithms.

The main objectives of this thesis are to investigate the principles of Lidar-based SLAM, explore the operation of the Sick LMS5xx sensor, and develop a Robot Operating System (ROS) implementation for performing SLAM using this sensor [15]. The thesis also aims to evaluate the performance of the proposed system through simulation experiments and real-world tests .

By achieving these objectives, this research aims to contribute to the field of mobile robotics by providing insights into Lidar-based SLAM and demonstrating the capabilities of the Sick LMS5xx sensor for accurate mapping and localization. The findings of this study have the potential to enhance the autonomy and navigation capabilities of mobile robots, enabling them to operate effectively in complex and dynamic environments.

In the following chapters, we will delve into the theoretical foundations of Lidar-based SLAM, explore the operation and data reading of the Sick LMS5xx sensor, discuss the im-

plementation of SLAM algorithms in ROS, and present the results of simulation experiments and real-world tests. Through this comprehensive analysis, we aim to provide valuable insights and practical guidance for researchers and practitioners working in the field of mobile robotics and SLAM.

plementation of SLAM algorithms in ROS, and present the results of simulation experiments and real-world tests. Through this comprehensive analysis, we aim to provide valuable insights and practical guidance for researchers and practitioners working in the field of mobile robotics and SLAM.

CHAPTER 1

Lidar-Based SLAM for mobile robots

1.1 Introduction

Simultaneous Localization and Mapping (SLAM) is a well-established technique that enables mobile robots to autonomously navigate and map their environments in real-time. SLAM has gained significant attention in the field of robotics due to its ability to address the simultaneous challenges of localization and mapping, providing a fundamental capability for autonomous exploration, path planning, and environment understanding.

Lidar sensors, which utilize laser beams to measure distances to surrounding objects, have become a popular choice for perception in mobile robotics. Lidar sensors offer high- resolution and accurate 3D point cloud data, making them well-suited for mapping and localization tasks in SLAM. By leveraging Lidar-based perception, mobile robots can effectively perceive their surroundings and generate detailed representations of the environment [11].

In this chapter, we focus on SLAM system for mobile robots, exploring the techniques and algorithms that enable robots to simultaneously determine their own position (localization) and construct a map of the environment. We delve into the fundamental aspects of SLAM, including perception in robotic systems, localization, mapping, SLAM algorithms, and scan matching. By understanding the principles and methodologies of SLAM, we can enhance the capabilities of mobile robots in autonomous navigation and mapping tasks.

1.2 Perception in robotic systems

Robotic mobile systems are autonomous robots or objects that can move around on their own utilizing their own legs, wheels, or other means of propulsion. Sensors, processors, and actuators are frequently included in these systems, allowing them to perceive their environ ment make decisions, and carry out activities without the need for human intervention. Autonomous vehicles, drones, and mobile robots employed in industry or agriculture are a few examples of robotic mobile systems.

The ability of a robot to perceive and interpret data about its environment using a variety of sensors and algorithms is referred to as perception in robotic systems. In order to extract useful information from the sensor data, this must include the use of sensors like

cameras, lidar, sonar, and radar as well as cutting-edge signal processing and machine learning techniques. A mobile robot using simultaneous localization and mapping (SLAM) can accurately map its surroundings and determine its position within that map in real-time by using perception. The classification of the sensors is generally made in relation to two families : proprioceptive and exteroceptive sensors.

1.2.1 Proprioceptive sensors

Proprioceptive sensors are sensors that provide information about the internal state of a robotic system, such as its position, orientation, velocity, and acceleration. These sensors are critical for accurate perception and control of a robotic system, as they allow the system to know its own state and adjust its behavior accordingly [5]. Some common types of proprioceptive sensors include:

Joint position sensors

A robotic system can establish its overall position and orientation by using these sensors to monitor the angles of each joint.

Figure 1.1 Joint position sensor.

Accelerometers

These sensors determine the robotic system's acceleration, which can be used to determine its position and velocity over time.

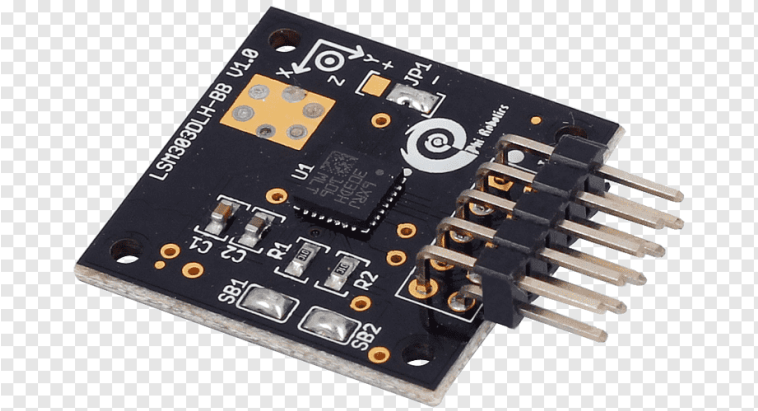


Figure 1.2 Accelerometers.

Gyroscopes

These devices measure the robotic system's angular velocity, which can be utilized to establish its orientation.

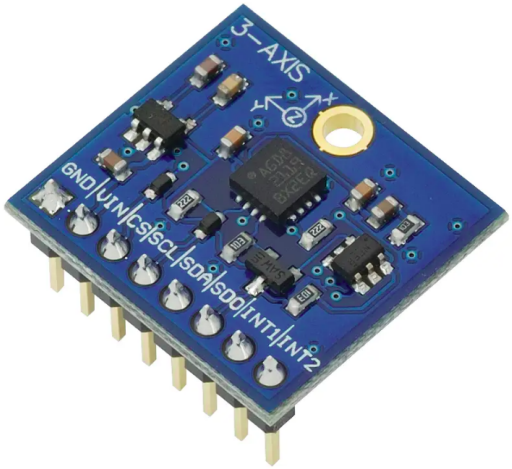


Figure 1.3 gyroscope sensor.

Odometry and IMU

Odometers measure robot's curvilinear displacements by measuring wheel rotation, calculating relative position, and providing information on wheel diameter, wheelbase, and vehicle structure.

Inertial Measurement Units (IMUs) enhance sensing capabilities, improving motion estimation, localization, and navigation in various environments.

1.2.2 Exteroceptive sensors

Exteroceptive sensors are devices that give to a robotic system , information about its surroundings . These sensors are essential for a robotic system to be able to see, comprehend, and meaningfully interact with its environment [3]. Typical exteroceptive sensor types include:

Cameras

Perhaps the most popular exteroceptive sensor type utilized in robotic systems is the camera. They record the environment's visual information, including colors, forms, textures, and movements. They can be utilized for navigation and obstacle avoidance, as well as for activities like object identification, recognition, and tracking.

Lidar

Lidar sensors create a two or three-dimensional map of the environment by measuring the distance to nearby objects using laser beams. They are frequently used in robotics and autonomous vehicle applications for localization, mapping, obstacle avoidance, and path planning.

Sonar

Sonar sensors use sound waves to locate objects and gauge their distance from one another. Together with some airborne and ground-based systems, they are frequently utilized in applications for underwater robots.

Infrared sensors

Infrared sensors locate and determine the temperature of things in the environment by detecting the heat they generate. They are frequently employed in robotics applications including search and rescue and fire detection.

A robotic system may create a thorough and detailed model of the outside environment by combining data from these various exteroceptive sensors, and it can then utilize that model to decide what to do and how to do it.

Figure 1.4 Exteroceptive sensors

1.3 Localization

Localization is one of the most fundamental steps in the navigation and mapping. Its success determines all the performances of the robot. It is necessary for other tasks such as planning, monitoring trajectory... etc.

We can distinguish three types of localization methods [6] :

- Relative localization methods, based on the use of proprioceptive sensors.
- Absolute localization methods, based on the use of exteroceptive sensors.
- Hybrid methods based on the joint use of the two types of sensors.

Relative Localization

Relative localization consists in evaluating position, orientation, and optionally the speed of the mobile robot by integrating the information obtained from proprioceptive sensors. The integration is done with respect to the starting point of the robot. This data can be the travel information (odometer), of the robot , speed (velocity) or acceleration (accelerometer).

Absolute Localization

Absolute localization is a technique that allows a robot to locate itself directly in its environment, whether in an outdoor environment (sea, space, land), or indoor environment (workshops, buildings, power plants nuclear... etc.). These localization methods are based on the use of exteroceptive sensors. To estimate the position of the robot, it is possible to build a local map representing the environment close to the current position. The comparison of this local map and the global map of the environment then allows to find position as shown in the figure 1.5.

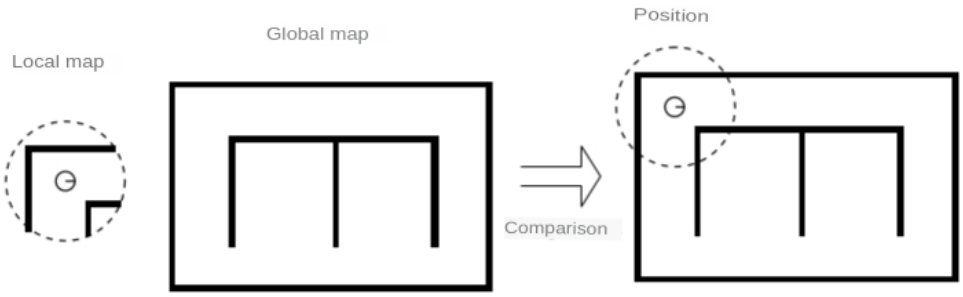


Figure 1.5 Absolute Localization Technique

[6].

Hybrid Localization

The disadvantage of relative localization methods is that they generate a cumulative error with the increasing of the travelled distance by the robot. To overcome this drift problem, an absolute location system can be used to correct the relative position estimate The combination of the relative and absolute methods is named hybrid localization.

1.4 Mapping

Several map representations are recognized as suite for SLAM purposes, they are divided into metric and topological map representations.

1.4.1 Map categories

Robot mapping is the process of creating a map of the environment that is useful for the current task of the robot. Although this process appears to be simple for humans, it is much more challenging for mobile robots. The same sensors that the robot uses to observe its environment should be used to build a suitable map for an autonomous navigation .

a)Metric maps

This kind of maps is the most accurate representation of the world. They use a global coordinate system and maintain a significant amount of information about the environment's specifics, such as distances, measures, sizes, and so forth. The processing time and storage requirements of this approach are its main drawbacks, which make it more challenging to use

in some real-time applications [23]. ”Occupancy grid maps” and ”feature-based maps” are two extensively used ways to represent metric maps of the environment in SLAM research.

Occupancy grid maps

They are discrete cell maps that could include information in 2D or 3D. The map is divided into grids, and each grid cell can either be occupied or empty. Occupancy grid maps can display different types of environments. However, the divided grids require a lot of memory, and updating them requires a lot of computation.

They determine and maintain the location of specific environmental features. Landmarks are objects that can be points, lines, or corners. SLAM can benefit from feature-based maps because of their compactness research. Due to their small size and ability to accurately represent structured environments, line-segment-based maps are frequently used in indoor environment applications because of their low memory usage and computational cost.

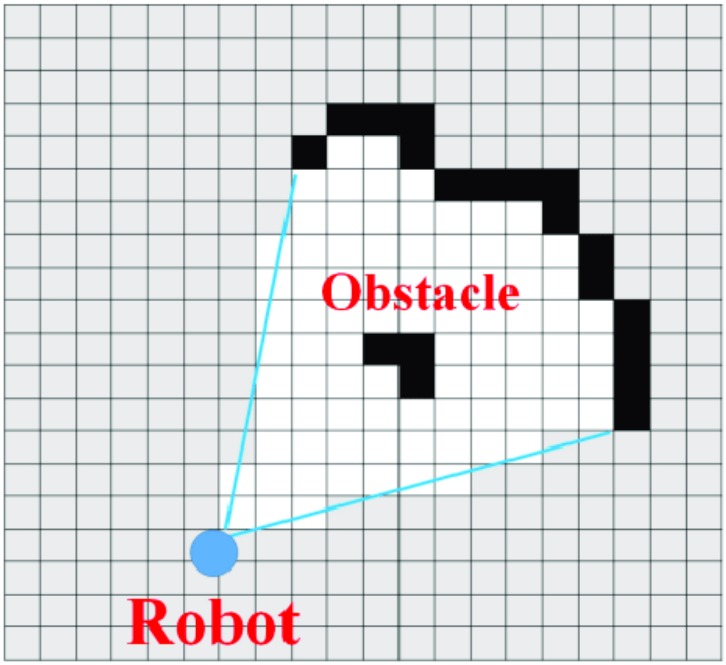


Figure 1.6 Occupancy grid map

b)Topological maps

This method creates an abstract representation of the environment, typically in the form of a graph with poses and connections between them. Links represent relationships or potential

actions that could be taken between the various locations, and poses represent environment locations with similar features. These maps take up a lot less storage space than metric maps because they are more straightforward and compact [24]. Table 1.1 resumes the advantages and disadvantages of metric and topological maps.

c) Hybrid maps

This final paradigm seeks to combine the benefits and minimize the drawbacks of each type of map in a unique mapping technique [10].

Feature	Metric maps	Topological maps
Accuracy	Yes	No
Storage needs	High	Low
Path planning	Complex	Simple
Optimal routes	Yes	No
CPU Needs	High	Low

Table 1.1: Advantages and disadvantages of metric and topological maps.

1.5 SLAM

Simultaneous Localization and Mapping, or SLAM for short, is a computational technique used in robotics and autonomous systems to map a new environment while also figuring out where the robot is in it. SLAM uses data from a variety of sensors, including cameras, lidars, and odometry sensors, to build an environment map and determine the position and orientation of the robot within it. Features extraction, data association, state estimation, and map updating are some of the steps in the procedure [22].

Robots and autonomous systems benefit from SLAM because it gives them the ability to operate in uncharted territory and navigate on their own without the aid of outside localization systems. There are numerous realworld uses for SLAM, including in autonomous vehicles and drones and robotic exploration missions.

1.5.1 Formulation of the SLAM problem

he SLAM is composed of a set of methods allowing a robot to build a map of an environment and at the same time to locate itself using this map. The trajectory of the vehicle and the position of landmarks in the environment are estimated along without the need for a priori knowledge.

Consider a robot moving in an unknown environment, observing a certain number of landmarks thanks to a sensor mounted on the robot. Figure 1.7 illustrates the problem.

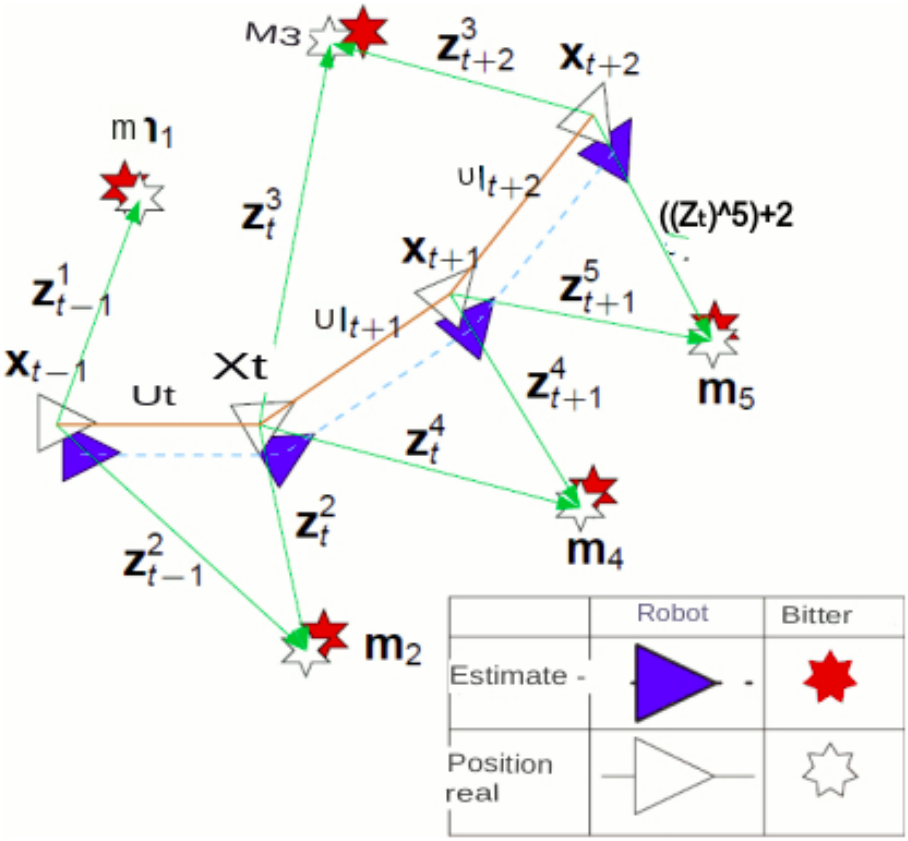


Figure 1.7 The basic idea of SLAM

[25]

The following quantities are currently being determined:

- x_k: the state vector. It contains the position of the robot.
- u_k: the control vector. Applying u_k at moment k_1 leads the robot from state x_k to state x_{k+1}.

- m_i: a vector containing the position of landmark i.
- z_k: observation measure at moment k.

The following sets are also defined:

- $X_{0:k} = \{x_0, x_1, \ldots, x_k\}$: the set of state vectors up to moment k.
- $U_{0:k} = \{u_0, u_1, \ldots, u_k\}$: the set of control vectors up to moment k.
- $Z_{0:k} = \{z_0, z_1, \ldots, z_k\}$: the set of observations up to moment k.
- $m = \{m_1, m_2, \ldots, m_n\}$: the environment map containing a list of static objects.

The probabilistic formulation of the SLAM problem requires calculating, at each moment k, of the amount of probability:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \tag{1.1}$$

In this work, to get the measures Zk, we will use a lidar sensor, which measures the amount of time it takes for laser beams to leave the sensor, bounce off nearby objects, and return. Algorithms use this data to process a 3D map of the environment and determine the robot's location within it.

Lidar-based SLAM is an excellent option for mobile robots . It has many benefits, including high accuracy, high resolution, and the ability to operate in different lighting conditions. However, it also has some drawbacks that must be taken into account when developing and putting into practice lidar-based SLAM systems, such as cost and complexity.

1.5.2 Algorithmes of SLAM

The algorithms developed in this field can be classified according to several criteria: the types of sensors used, the calculation methods adopted, the types of maps representing the environment, the types of environment... etc. Many researches try to solve the problem of simultaneous localization and mapping. The most important used algorithms are: Kalman filter, Particle filter and Pose-graph.

Kalman filter

Kalman offers a recursive solution for filtering linear data. After each meaurement acquisition,it is possible to estimate the new position of the robot. When the global environment map is built as the robot explores its environment, this filter also makes it possible to estimate the position of each geometric element introduced into the map. It intends to estimate a state vector x, containing the pose of the mobile robot and landmarks locations, as well as the corresponding covariance matrix modeling the estimation uncertainty. This is done through a recursive process based on two stages: prediction and correction [25].

- **Prediction step** which consists in the use of a motion model in order to predict the state of the robot based on the control information.

$$x_k = f(x_{k-1}, u_k) + \omega_k \tag{1.2}$$

where represents the model of the robotic vehicle and $\omega_k \sim \aleph(0, Q_k)$ is a Gaussian noise with zero mean and variance .

- **Correction step** which uses a measurement model to correct the state vector based on the acquired measurements.

$$z_k = h(x_k, m) + v_k \tag{1.3}$$

where describes the observation model and $V_k \sim \aleph(0, R_k)$ a Gaussian noise with zero mean and variance.

Particle filter

A particle filter is a recursive filter used to estimate a posteriori state using a set of particles. Unlike parametric filters like the Kalman filter, a particle filter represents a distribution by a set of samples created from this distribution. A particle filter is thus able to treat strongly non-linear systems with nonGaussian noise. The complexity of particulate filter calculations increases exponentially with the number of environmental landmarks, which is a major problem in a real-time application [27].

In order to solve these kinds of problems, some research combines particle filtering with other methods. This is the case of Montemerlo's work in FastSLAM . The use of particle filtering also suffers from the difficulties encountered when defining the number of particles. Indeed, the quality of the estimate is highly correlated to the discretization of the research

space. But it is difficult to find an optimal number of particles.

Graph based SLAM

In recent years, Graph-based SLAM is becoming the state-of-art solution for SLAM problem. The objective is to find the best robot poses X and landmarks locations M given sensor measurements Z and control U information.

This method describes the SLAM problem as a graph; a node corresponds to a robot pose or a landmark position. An edge between two nodes represents a spatial constraint between the nodes derived from some sensor data. As the robot progresses it compounds an increasing number of uncertain relative poses so that the cumulative error in the pose of the nodes will increase [27].

The solution to the SLAM problem, in this case, can be shown to be equivalent to finding the optimum of eq. (1.4):

$$X^*, M^* = \arg\min_{X,M} \left[\sum_k \Big((x_{k-1}, u_k) \Big)^k Q^{-1} \Big(x_k - f(x_{k-1}, u_k) \Big) \right.$$
$$\left. \times \sum_k \Big(z_k - h(x_k, M) \Big)^k R^{-1} \Big(z_k - h(x_k, M) \Big) \right] \tag{1.4}$$

1.6 Scan Matching

In recent years, technique surveys have been useful in resolving SLAM problems. It has been put into practice using a variety of methods, one of which is the scan-matching algorithm. Find the rigid-body transformation (translation+rotation) that best aligns scan and map, or scan and scan, or map and map.

Many SLAM algorithms make use of the idea of scanmatching, and some of them do so exclusively. Combining range measurements from one scan to the next is the idea. If the matching is done with an existing map, then new scans are implicitly matched with all previous scans, and the result can be used to estimate the transformation between them [13].

Several algorithms in this area are based on the most well-known scan matching, the Iterative Closest Point (ICP) algorithm, the Iterative Dual Correspondences (IDC) , the normal distribution transformat (NDT) and Likelihood-Fieldmatching LF. The problem can be stated as follows. Assume we have two sets of corresponding points a_i and b_i, i= 1... N

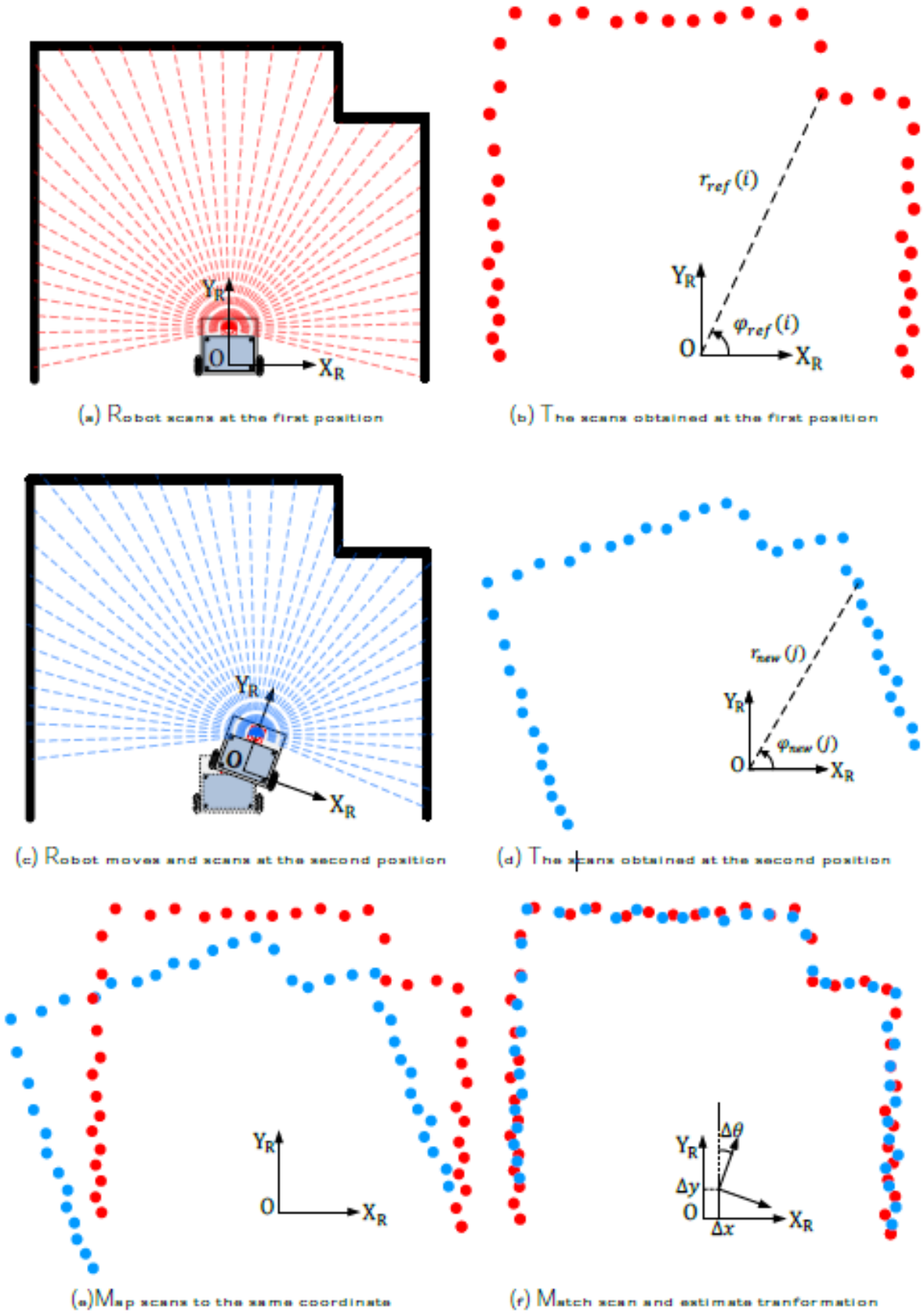
are linked by:

$$a_i = R \cdot b_i + T + V_i \tag{1.5}$$

where R is a rotation matrix, T is a translation vector and v is a noise vector. The aim is to find the optimal rotation matrix R and the translation vector T which minimizes the least squares criterion

$$\epsilon^2 = \sum_{i=1}^{N} \|a_i - Rb_i - t^2\|^2 \tag{1.1}$$

Figure 1.8 demonstrates the utilization of scan matching to estimate the transformation of a robot, enabling accurate localization and mapping through alignment of scans or maps.



(a) Robot scans at the first position (b) The scans obtained at the first position

(c) Robot moves and scans at the second position (d) The scans obtained at the second position

(e)Map scans to the same coordinate (f) Match scan and estimate tranformation

Figure 1.8 Using scan matching to estimate the transformation of the Robot

1.7 Conclusion

In this chapter, we studied the different parts of a SLAM (simultaneous localization and mapping) system. First, we discussed the perception which is a very important and indispensable step in SLAM systems. The robot must be equipped with proprioceptive sensors and exteroceptive sensors allowing it to localize and perceive its environment. We then started by defining the localization of the mobile robot and the mapping that describes the different representations of environment in SLAM systems. In a second step, the three most used for SLAM resolution were presented : is the Kalman filter, particulate filtering and GraphSlAM.

Finally, we ended our chapter with the presentation of the most common scan matching methods for aligning two successive point clouds (scans in the case of SLAM based laser rangefinder) in order to determine the motion (rotation and translation) performed by the robot between the acquisition times of two successive scans

CHAPTER 2

Sick LMS5xx laser measurement sensor operation and reading data

2.1 Introduction

Robotics and industrial automation both use the Sick LMS5xx laser measurement sensor. It produces a 2D map of the environment and measures distances using a laser beam. The sensor sends a stream of packets containing distance and angle data so that it can be read. Its uses range from obstacle detection to mapping, localization, navigation, and quality assurance in manufacturing processes. For path planning and obstacle avoidance, the LMS5xx is frequently incorporated into robotics platforms.

It is a strong and versatile sensor that can be applied in a variety of industrial automation applications. The LMS5xx offers precise and trustworthy measurements while being simple to use. A computer or microcontroller can process its data.

In this chapter, we will study the operating principle of Sick LMS5xx scanner and give its characteristics as well as how to configure it and access its measurements using the software SOAPS an Matlab.

2.2 Description of Sick LMS5xx and applications

The LMS5xx is a laser measurement technology by SICK , a german based company that specializes in industrial sensor technologies and offers advanced features and benefits in applications where speed, accuracy, and security are paramount. It is the world's first 2D laser scanner that revolutionized measurement technology and brings a whole range of pioneering features.

The LMS5xx's 5-echo technology uses high-speed sampling to provide five measurements for each emitted pulse for improved visibility and accuracy in adverse weather conditions. The laser measurement sensor offers superior performance with benefits like maximum detection reliability, superior measurement precision, high scanning speed, and flexible and powerful configuration software.

The LMS5xx laser measurement technology by SICK can be used in various applications such as anti-collision systems for long-range RTGs, RMGs, STSs and straddle carriers, navigation and geomapping, and other applications. The different models in the LMS5xx family make it easy to select the ideal product for specific requirements. More information on these applications can be found on the SICK website under the Applications Finder [17].

Figure 2.1 Sick LMS5xx

Ports

The Sick LMS5xx laser measurement sensor is used in anti-collision systems for long-range RTGs, RMGs, STSs, and straddle carriers in container terminals. These systems rely on the sensor's precise distance measurements to detect objects or personnel in the cranes' vicinity, preventing collisions and ensuring safety. Additionally, the sensor assists in trailer positioning beneath STS cranes during container loading and unloading operations.

Its accurate distance measurements enable precise alignment of trailers, enhancing efficiency and minimizing errors in container handling processes. The Sick LMS 5XX plays a crucial role in optimizing safety and productivity in these applications within container terminals [18].

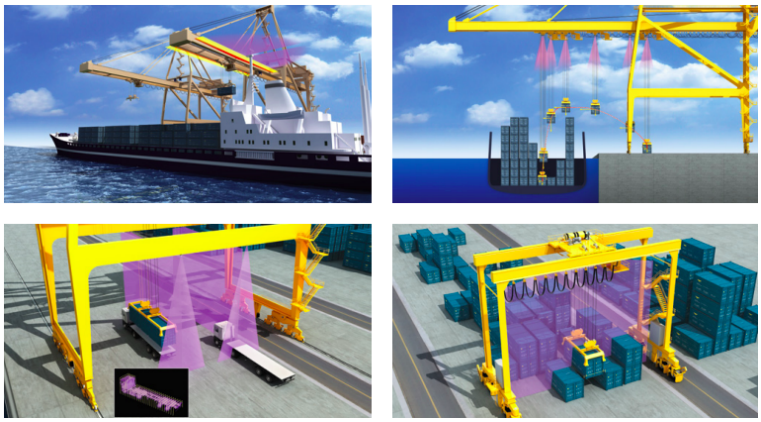


Figure 2.2 Using sick LMS5xx in saftey of container

Traffic

The Sick LMS 5XX laser measurement sensor has a wide range of applications. It can trigger cameras, classify vehicles, monitor speed, ensure vehicle separation . With its compact design and advanced capabilities the Sick LMS5xx contributes to safe traffic flow, and enhanced security.

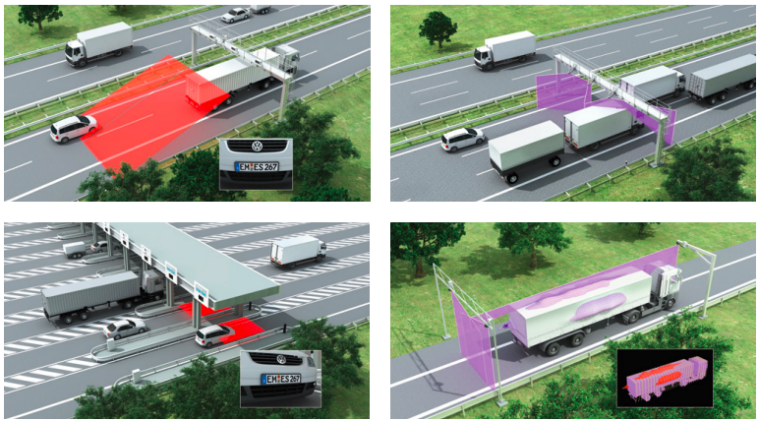


Figure 2.3 Using sick LMS5xx in saftey of traffic

Security

The Sick LMS 5XX laser measurement sensor is versatile in its monitoring and security applications. It is utilized for area monitoring in museums, galleries, and other buildings, as well as for perimeter and large façade surveillance. Additionally, the sensor offers roof

protection in high-security buildings such as prisons, banks, and art galleries. With its compact design and advanced features, the Sick LMS 5XX ensures reliable monitoring and enhanced security in various settings.

Figure 2.4 Using sick LMS5xx in security

Navigation and geomapping

The Sick LMS 5XX laser measurement sensor is a versatile choice for AGV (Automated Guided Vehicle) applications in both indoor and outdoor environments. It offers navigation capabilities with or without reflectors, enables long-range anti-collision functionality, facilitates geographical data acquisition, and supports street profiling. With its compact design and advanced features, the Sick LMS 5XX enhances AGV efficiency and safety in various operational scenarios [18].

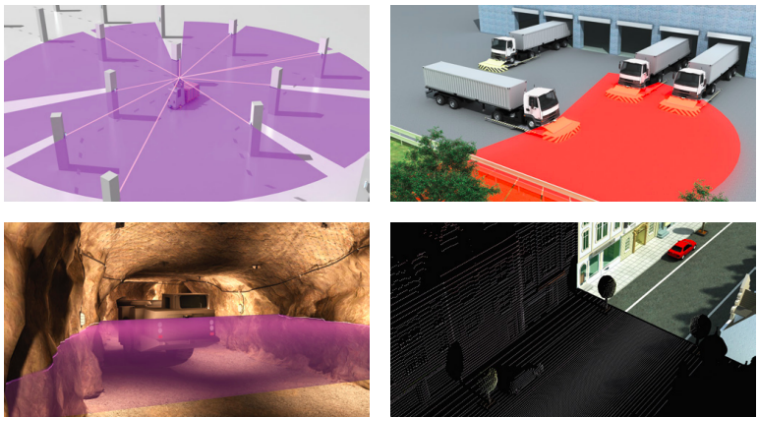


Figure 2.5 using sick LMS5xx in Navigation and geomapping

2.3 Operating principle of the LMS5xx

The Laser scanner sensors operate by shining a laser off of a rotating mirror. As the mirror spins, the laser scans 190°, effectively creating a fan of laser light as shown in the picture below. Any object that breaks this fan reflects laser light back to the sensor. The distance is

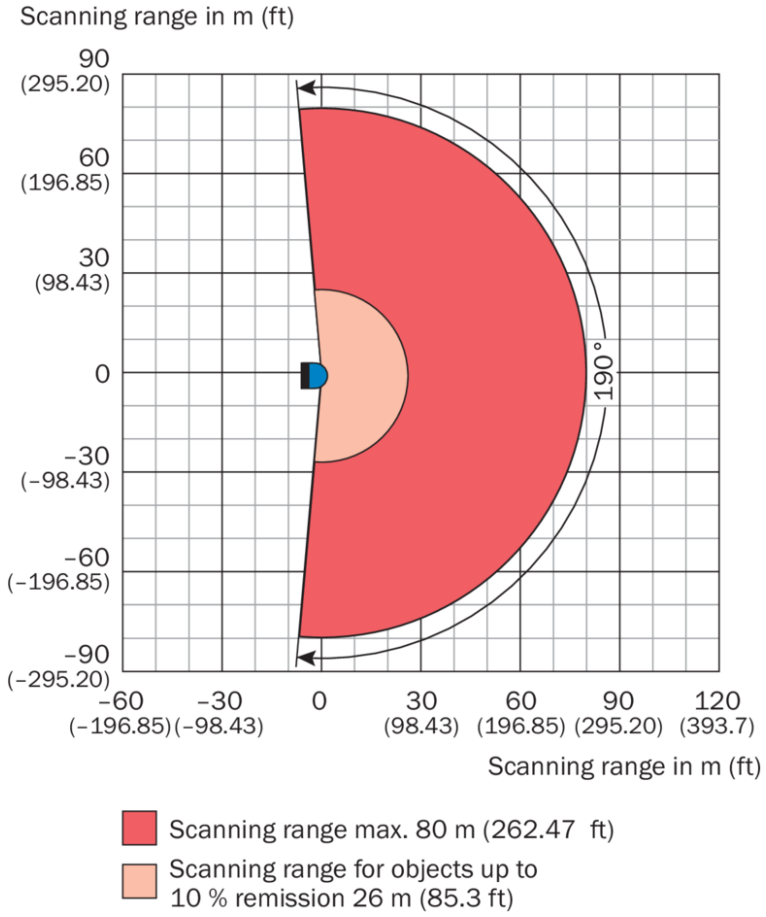


Figure 2.6 Aperture angle and range of the laser .

calculated based on how long the laser takes to bounce back to the sensor. LMS 5xx is a non-contact laser mesurement sensor that scans the surrounding perimeter radially on a single plane using pulses of light . The LMS5xx measures in two-demensional radial coordinates , if a laser beam emitted is reflected form a target object, then the position of the object is given in the from of distance and angle [4] .

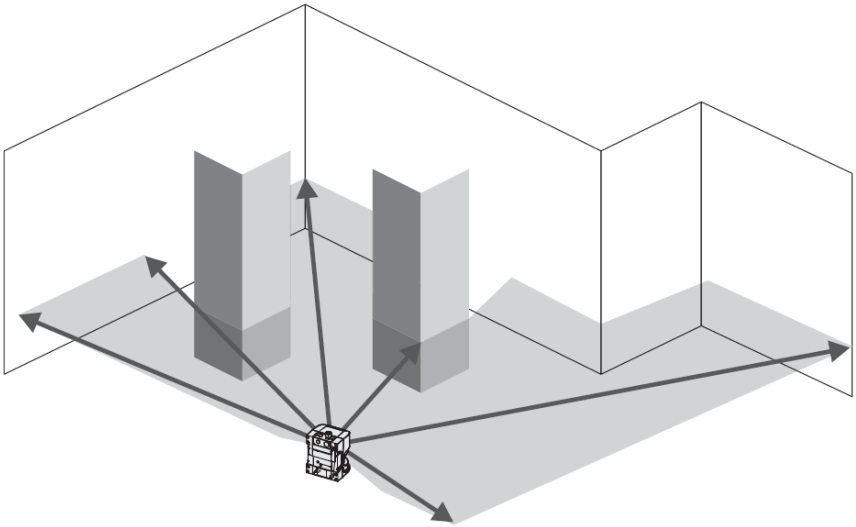


Figure 2.7 Measuring principle of the LMS5xx

2.3.1 Distance measurement

The LMS5xx emits pulsed laser beams using a laser diode. If a laser beam is reflected on a target object the reflected beam is received at the sensor.

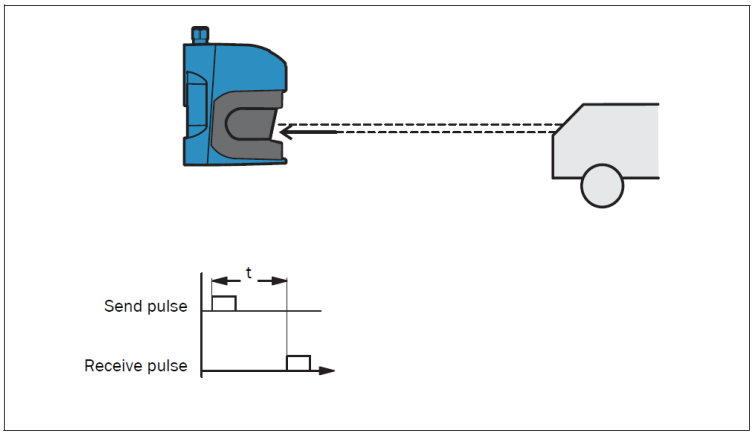


Figure 2.8 Distance measuremen of the LMS5xx

The distance to the object is calculated by the time required for the pulsed beam to be reflected and received by the sensor.

2.3.2 Direction measurement

As we said before, the emitted laser beams are deflected using an internal rotating mirror and scanning the surroundings in a circular manner. The measurements are triggered at regular steps using an angular encoder. The LMS5xx Lite scans with a scanning frequency of 25, 50 or 75 Hz and angular steps of 0.25°, 0.5° or 1°.

Special features of the LMS5xx

Model Name Part Number	**MS511-11100 lite 1054155**
Field of View	190°
Angular Resolution	0.25°, 0.5°, 1°
Response Time	> 13 ms
Resolution Power	Standard Resolution
Scanning Frequency	25 Hz, 35 Hz, 50 Hz, 75 Hz,
Scanning Range	40 m
Working Range	0.2 m ... 80 m
Data Interface	TCP/IP, USB, RS-232, RS-422
Data Transmission Rate	20 kBit/s, 500 kBit/s, 1 MBit/s, Synchronization
Supply Voltage	24 V DC, ± 20%
Power Consumption	22 W
Storage Temperature	–40 °C ... +70 °C
Poids	3.7 kg
Dimensions (L x l x H)	160 mm x 155 mm x 185 mm

Table 2.1 Specifications for the LMS5xx. [20]

2.3.3 Data interfaces

The LMS5xx has different data interfaces for the configuration and the transmission of measured values.

Ethernet interface: The Ethernet interface has a data transmission rate of 10/100 Mbit/s. It is a TCP/IP interface supporting full duplex and half duplex. The Ethernet interface allows the configuration of the LMS5xx as well as the output of measured values. The factory setting for the Ethernet interface is as follows:

- IP address: 192.168.0.1
- subnet mask: 255.255.0.0
- TCP port: 2111

If we desired to change the parameters of the Ethernet interface,we must first save the data in non-volatile memory in the LMS5xx using the software SOPAS ET and then restart the LMS5xx [17] .

Serial host interface An RS-232/RS-422 interface serves as the serial host interface. The LMS5xx can be configured via the host interface, but there is a limited measured value output.

The factory setting for the host interface is as follows: 57.6 kBd

- 8 data bits
- 1 stop bit
- no parity

USB auxiliary interface: The mini-USB auxiliary interface on the Sick LMS5xx sensors offers convenient configuration options. This interface allows users to modify parameters and settings of the sensor through a USB connection, and communicating with a host device. This flexibility and versatility provided by the mini-USB auxiliary interface enhance the usability and adaptability of the Sick LMS5xx sensors in various applications.

2.3.4 Data communication using telegrams

Data communication using telegrams refers to the process of sending messages or commands through a terminal program using either binary or ASCII telegram formats to control specific devices, such as laser scanners, and receive confirmation of commands/telegrams. The data in these telegrams can include actual commands with letters and characters, and parameters in either decimal numbers or fixed hexadecimal values with specific meanings, and are often separated by blank spaces.

The telegrams also contain headers, lengths, checksums, and framing to indicate the start and stop of each telegram and verify its accuracy. The use of telegrams allows for specific and efficient communication between devices and terminal programs [21].

The following functions can be run using telegrams:

- Request for measured values by the host and subsequent output of the measured values by the LMS5xx.
- Parameter setting by the host for the configuration of the LMS5xx.
- Parameters and status log querying by the host.

The telegrams each comprise a frame and the data.

Binary telegram (CoLa B)

A binary telegram (CoLa B) is the basic protocol used by scanners, where all values are in hexadecimal code and grouped into pairs of two digits. The string consists of four parts: header, data length (without CS), data, and the checksum (CS)as shown in Table 2.2.

The header indicates the start of the telegram, the size of the data length is 4 bytes (so the data part might have a maximum of 4,294,967,295 digit pairs) . The data part comprises the actual command with letters and characters converted to Hex, and the parameters are either decimal numbers converted to Hex or fixed Hex values with a specific intrinsic meaning. There is always a blank (20) between the command and the parameters but not between the different parameter values.

The checksum serves to verify that the telegram has been transferred correctly, and its length is 1 byte, CRC8.

	Frame					Telegram	Frame
Designation	STX	STX	STX	STX	Length	Data	CS
Length (byte)	1	1	1	1	4	<2.495	1
Description	start of the text character				Length of the data without(cs)	Binary encoded	Checksum (xor of all data bytes)

Table 2.2 Frame for the telegrams with binary coding

[?].

Table 2.3 illustrates an example of a binary telegram

02 02 02 02	00 00 00 17	73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 03 F4 72 47 44	B3
Header	Length	Data	CS

Table 2.3 Example of a Binary telegram

[21]

ASCII telegram (CoLa A)

An ASCII telegram is an alternative format to the binary telegram when using CoLa A protocol for laser scanners. It consists of framing, indicated with<STX>and <ETX>, which marks the start and end of the telegram, and the data part, which comprises the actual command with letters and characters (plaintext), parameter values either in decimal or hexadecimal, and fixed hexadecimal values with a specific meaning. The advantage of ASCII telegram is that commands can be written in plaint text. Table 2.4 shows the frame for Telegrams with ASCII coding [21].

	Frame	Telegram	Frame
Designation	STX	Data	ETX
Length (byte)	1	<30.000	1
Description	start of the text character	ASCII coded	End of text character

Table 2.4 Frame for the telegrams with ASCII coding (Cola A)

In CoLa A, depending on the preferences of the user, all values can be written directly in Hex. Table 2.5 show the same example but using an ASCII telegram [21].

ASCII	<STX>	sMNSPCSetAccessModeSPC03SPCF4724744	<ETX>
Hex	02	73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 30 33 20 46 34 37 32 34 37 34 34	03
	Start	Data	Stop

Table 2.5 Example of ASCII telegram

[21].

This is an example of telegram for setting the user level “Authorized Client”. As only fixed hexadecimal parameter values are needed, the option to use parameter values in decimal code with special indicator cannot be applied here:

- Framing = {STX} = telegram start = 02 (Hex)
- sMN = start of Sopas command (and blank) = 73 4D 4E 20 (Hex)
- SetAccessMode = the actual command for setting the user level (and blank) = 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 (Hex)
- 03 = fixed Hex value meaning user level "Authorized Client" (and blank) = 30 33 20 (Hex)
- F4 72 47 44 = fixed Hex value, serving as password for the selected user level "Authorized Client" = 46 34 37 32 34 37 34 34 (Hex)
- Framing = {ETX} = telegram stop = 03 (Hex)

Command basics

Description	Value ASCII	Value Hex	Value Binary
Start of text	<STX>	02	02 02 02 02 + given length
End of text	<ETX>	03	Calculated checksum
Read	sRN	73 52 4E	
Write	sWN	73 57 4E	
Method	sMN	73 4D 4E	
Event	sEN	73 45 4E	
Answer	sRA sWA sAN sEA sSN	73 52 41 73 57 41 73 41 4E 73 45 41 73 53 4E	
Space	{SPC}	20	20

Table 2.6: ASCII, Hexadecimal, and Binary Values for Communication telgram [21].

2.3.5 Measured value output

The way that data is displayed or transmitted from a sensor or device is referred to as its output format. Output formats refer to the various data formats that can be used to represent the measurements and other parameters of the Sick LMS 5XX laser measurement sensors. As well as ASCII, binary, and ROS (Robot Operating System) formats, the Sick LMS 5XX also supports other output formats.

For instance, binary format might be more effective for large data sets or high-speed data transmission while ASCII format might be simpler to read and understand. First, we request measured data by using a telegram on the interface from which we want to receive measured data. There are two possible ways to do that [?] :

- Only one measured value telegram can be requested using the **`sRN LMDscandata`** telegram. In this case, the last scanned measurement is transferred (Figure 2.9).
- Measured data can be continuously requested using the **`sEN LMDscandata`** telegram. Measured data will then be transferred until the measured value output is stopped using the **`sEN LMDscandata`** telegram (see Figure 2.10).

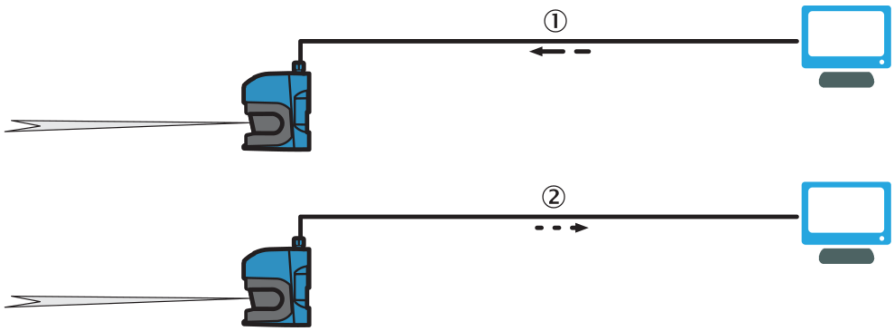


Figure 2.9 Single measured value output

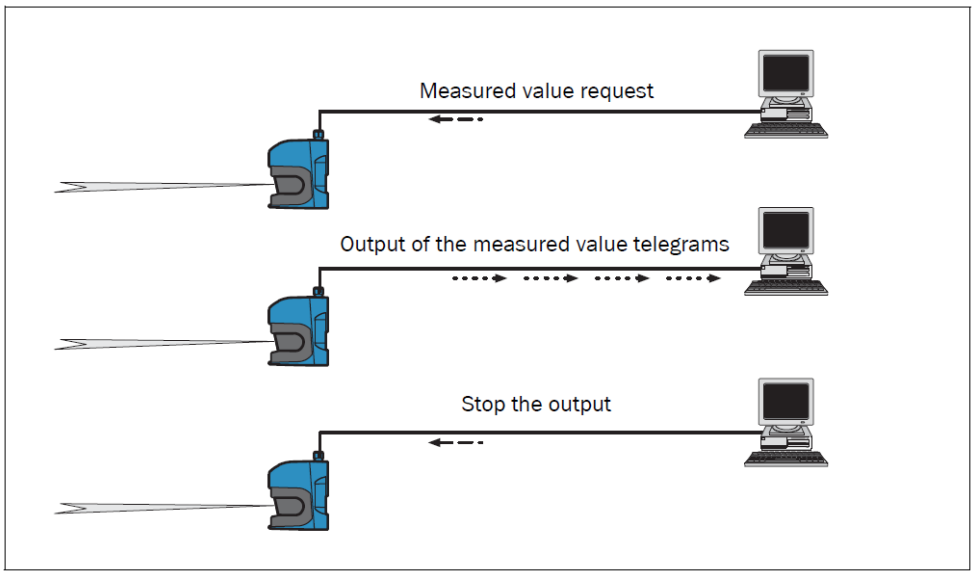


Figure 2.10 Continuous output of measured values .

2.4 Software and Reading results

In this section, we will explore the software aspects related to the operation and data processing of the Sick LMS5xx laser measurement sensor. Two important software tools that can be utilized with the LMS5xx are SOPAS Engineering Tool and MATLAB .

2.4.1 SOPAS Engineering Tool

SOPAS is a software platform for configuring and visualizing sensors made by the SICK Group. Encoder programming has been integrated into this proven software platform to increase efficiency and flexibility when managing encoders. To establish a connection between SOPAS ET and the LMS5xx device, we can either using an USB port or an RJ45 cable [19].

This a serves as a medium through which data communication was established between SOPAS ET and the LMS 5xx. By connecting the RJ45 cable to the respective ports on both SOPAS ET and the LMS 5xx, we enabled seamless communication and data exchange between the two systems. This connection allows us to efficiently manage and control the LMS 5xx device using the SOPAS ET software, ensuring optimal performance and functionality.

Configuring the Ethernet connection

When the SOPAS tool is started, a search is performed which lists all scanners available in the network (see figure 2.11). To configure the Sick LMS 5xx using SOPAS and establish a connection via an RJ45 cable, we followed a series of steps. First, we accessed the ' Parameters' section, located at the top left corner of the SOPAS interface. Next, we navigated to the 'Network/Interface' option, which allowed us to manage network settings. Within this section, we selected the 'Ethernet' option to configure the Ethernet interface.

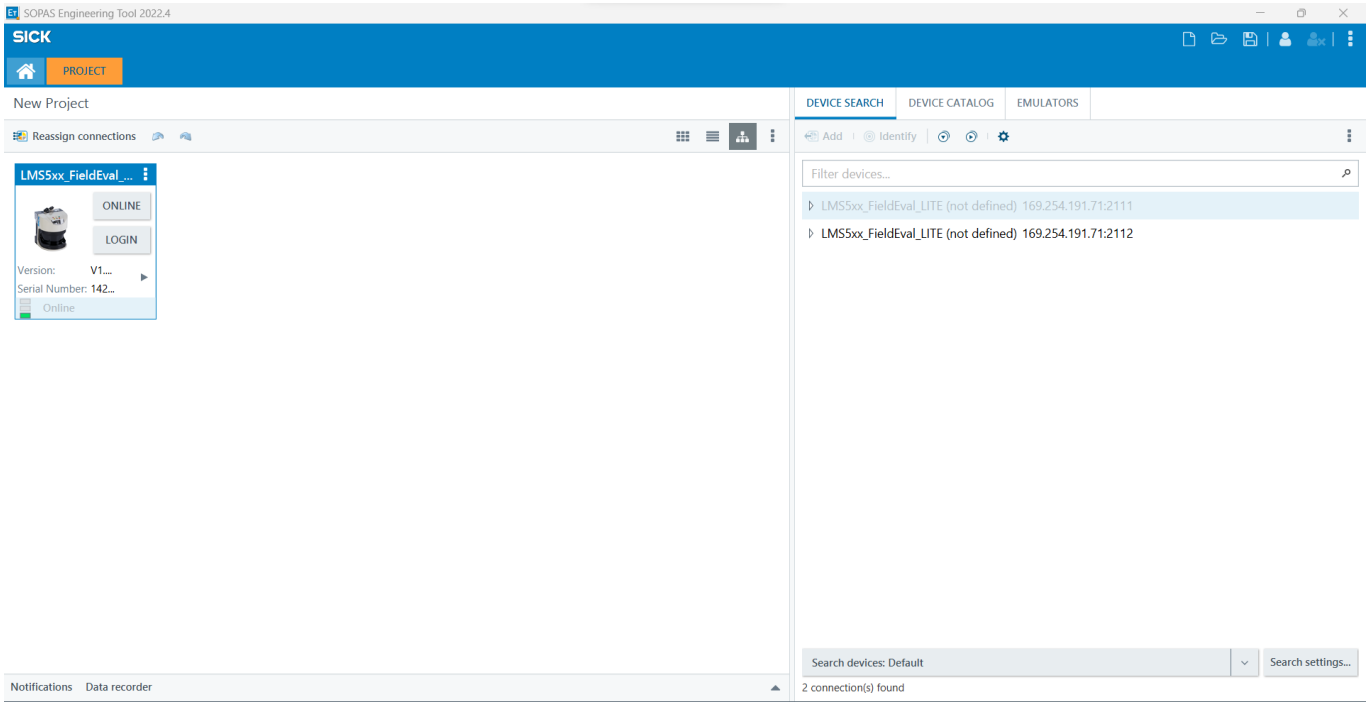


Figure 2.11 Connection between SOPAS ET and LMS5xx using RJ45 cable.

Upon selecting the Ethernet interface, a configuration table was displayed, presenting various settings. To make the necessary modifications, we clicked on the 'Write Parameters' option, numbered as 4 at the top (see Figure 2.12). Within this option, we changed the communication format from 'Cola ASCII' to 'Cola Binary,' ensuring a more efficient and streamlined data exchange process. Once the modifications were made, We saved the changes, ensuring that the new configuration was implemented successfully. This configuration adjustment allows for improved communication between the SOPAS software and the Sick LMS 5xx device, enhancing data transfer and enabling seamless control and management of the LMS 5xx.

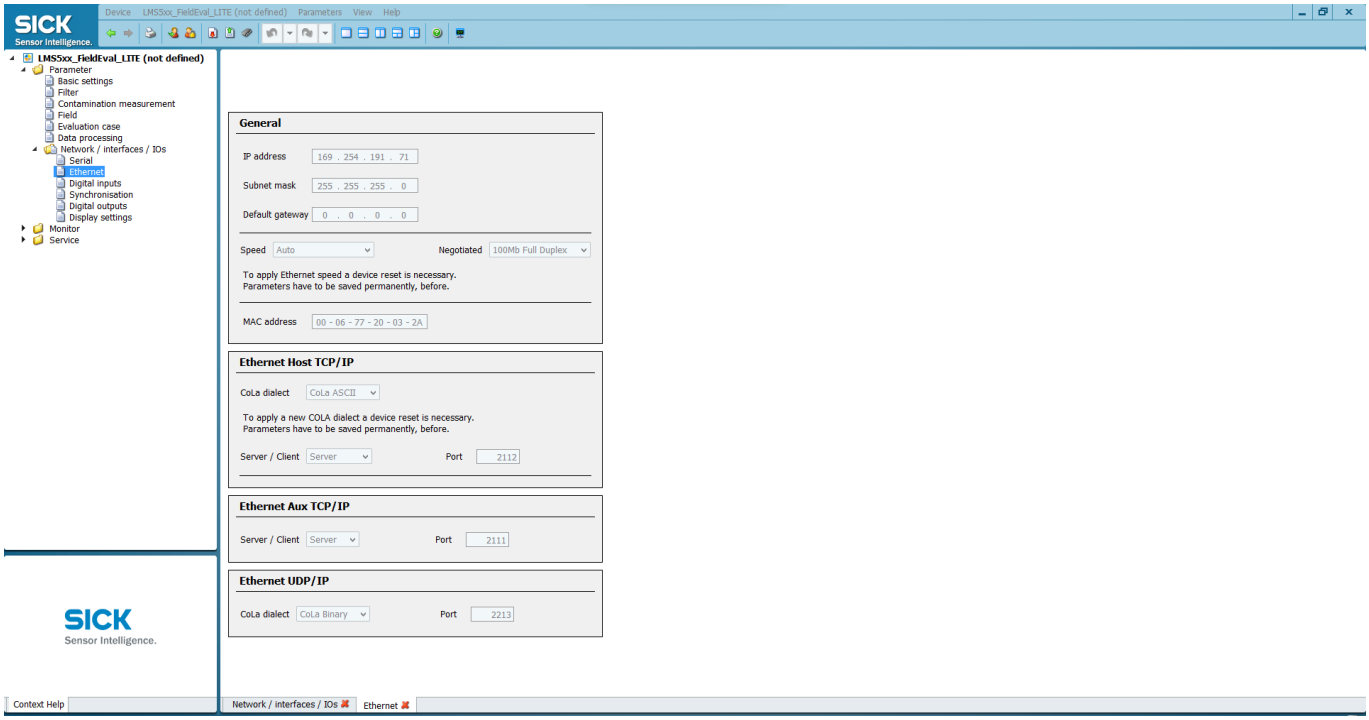


Figure 2.12 Configuration of sopas with ethernet connection

Results in SOPAS

During the configuration process of the Sick LMS 5xx using SOPAS, we proceeded to the 'Monitor' option to observe the results. Upon execution, the LMS 5xx generated a scan depicting a static environment with various obstacles as shown in Figure 2.13. The laser rangefinder accurately detected and represented these obstacles in the scan.

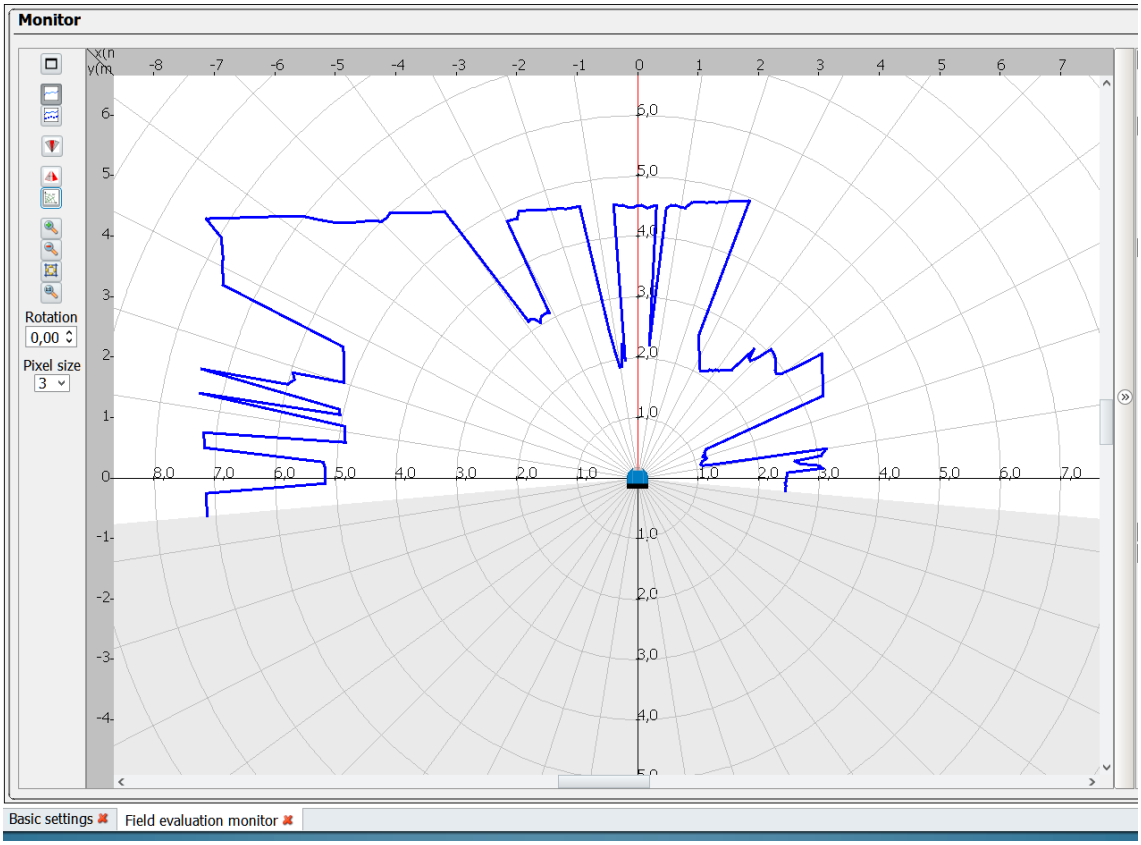


Figure 2.13 A scan of a static environment

However, the subsequent scan captured an interesting event—an individual moving in front of the laser rangefinder. The sensor promptly detected the person's presence, as indicated by a distinct red circle on the scan in Figure 2.14.

This real-time detection demonstrates the capability of the Sick LMS 5xx in accurately perceiving dynamic elements within its surroundings. In figure 2.15 we can see a scan showing the person moving in front of the laser rangefinder .

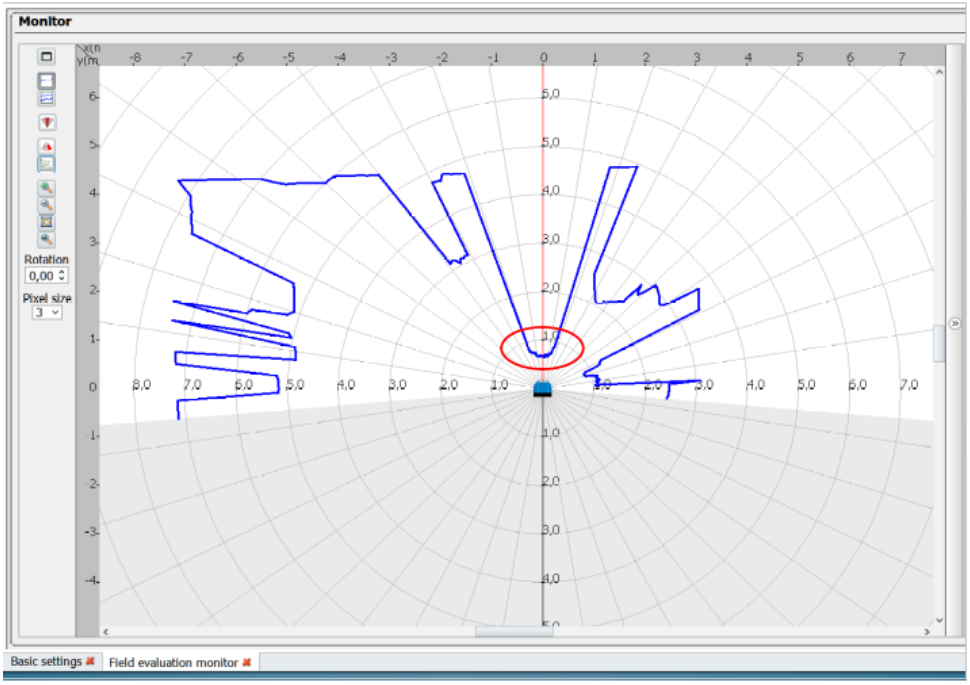


Figure 2.14 A scan showing a person moving in the environment of figure 2.14

Continuing the observation, the third scan depicted the same person moving to the left of the sensor. The red circle in the scan indicated the person's position, allowing for precise tracking of his movement. This demonstrates the ability of the Sick LMS 5xx to capture and represent changes in the environment, enabling accurate monitoring of dynamic objects. Figure 2.14 shows a scan with the person moving left] . These scans showcase the robust sensing capabilities of the Sick LMS 5xx, enabling the detection and tracking of both static obstacles and dynamic objects.

Such real-time perception is valuable for various applications, including robotics, surveillance, and safety systems. The captured images of the scans provide visual evidence of the sensor's performance and the accuracy of the detected objects' representation . Figure 2.17 shows the telegram "sRN LMDscandata" sent by the host PC (via SOPAS) to SICK LMS5xx asking to send him a single scan.

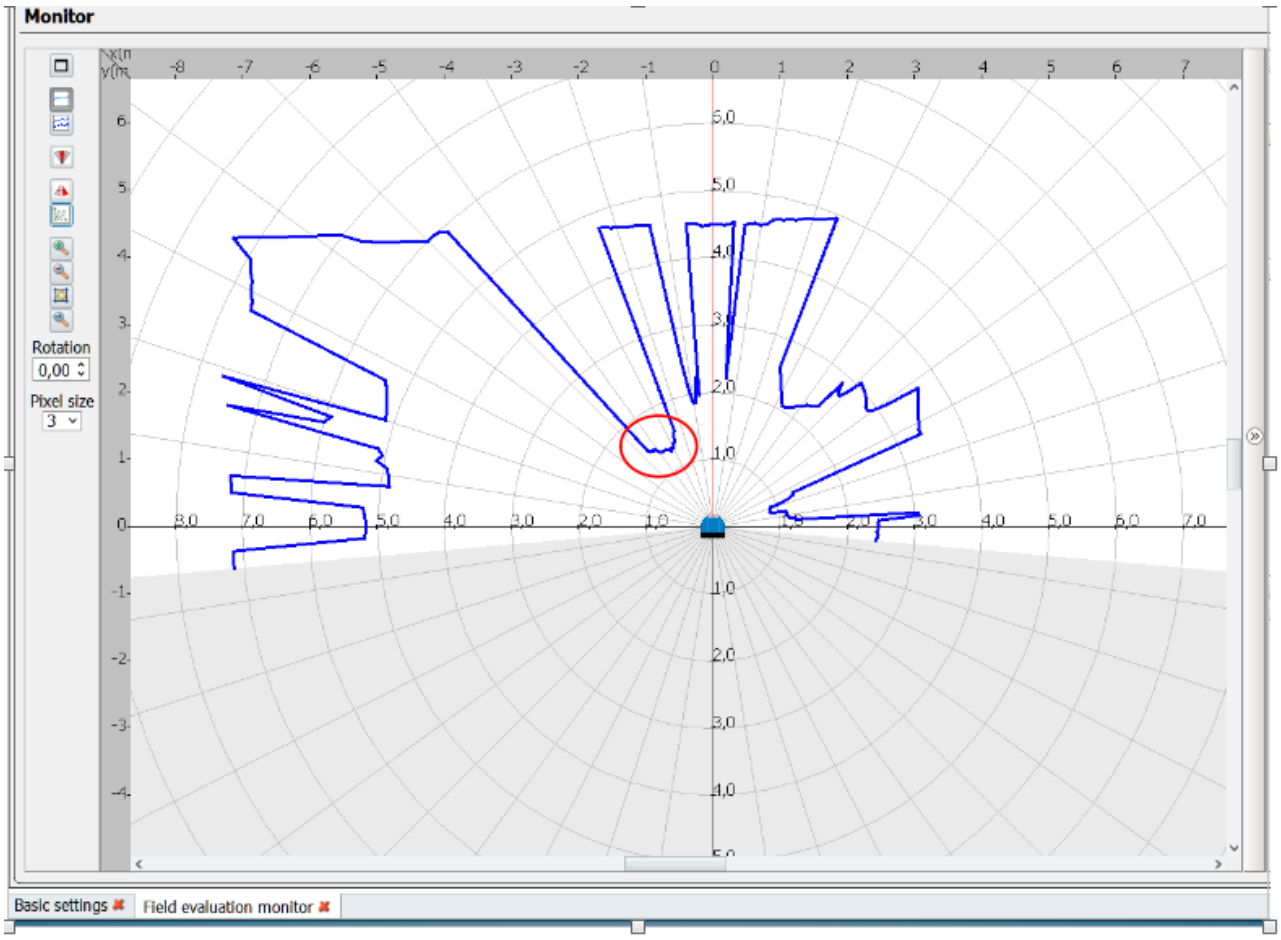


Figure 2.15 Scan with the person moving left.

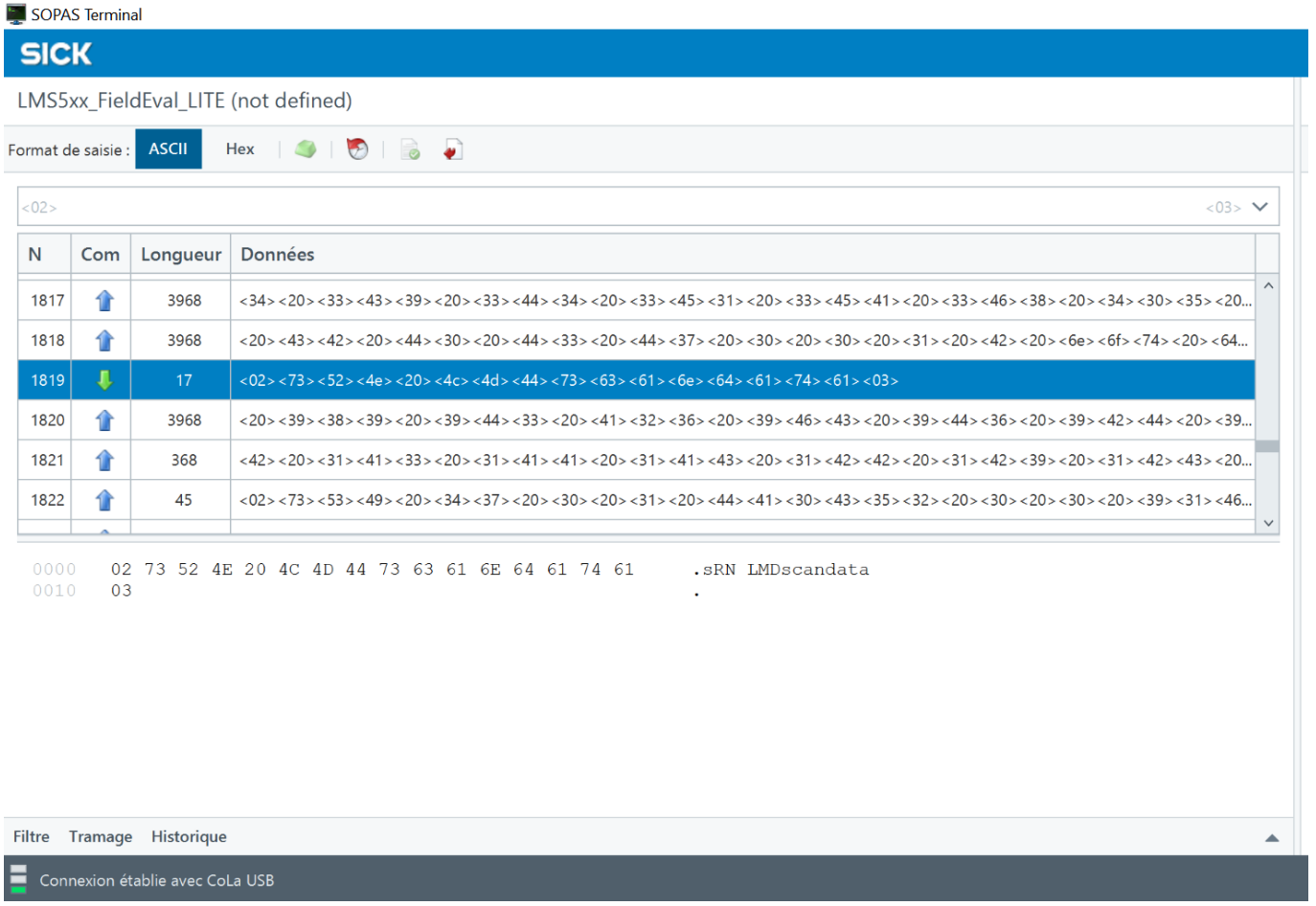


Figure 2.16 “sRN LMDscandata” telegram sent by the host PC (via SOPAS) to SICK LMS5xx

The data sent by the LMS5xx to the host PC is shown in figure 2.17.

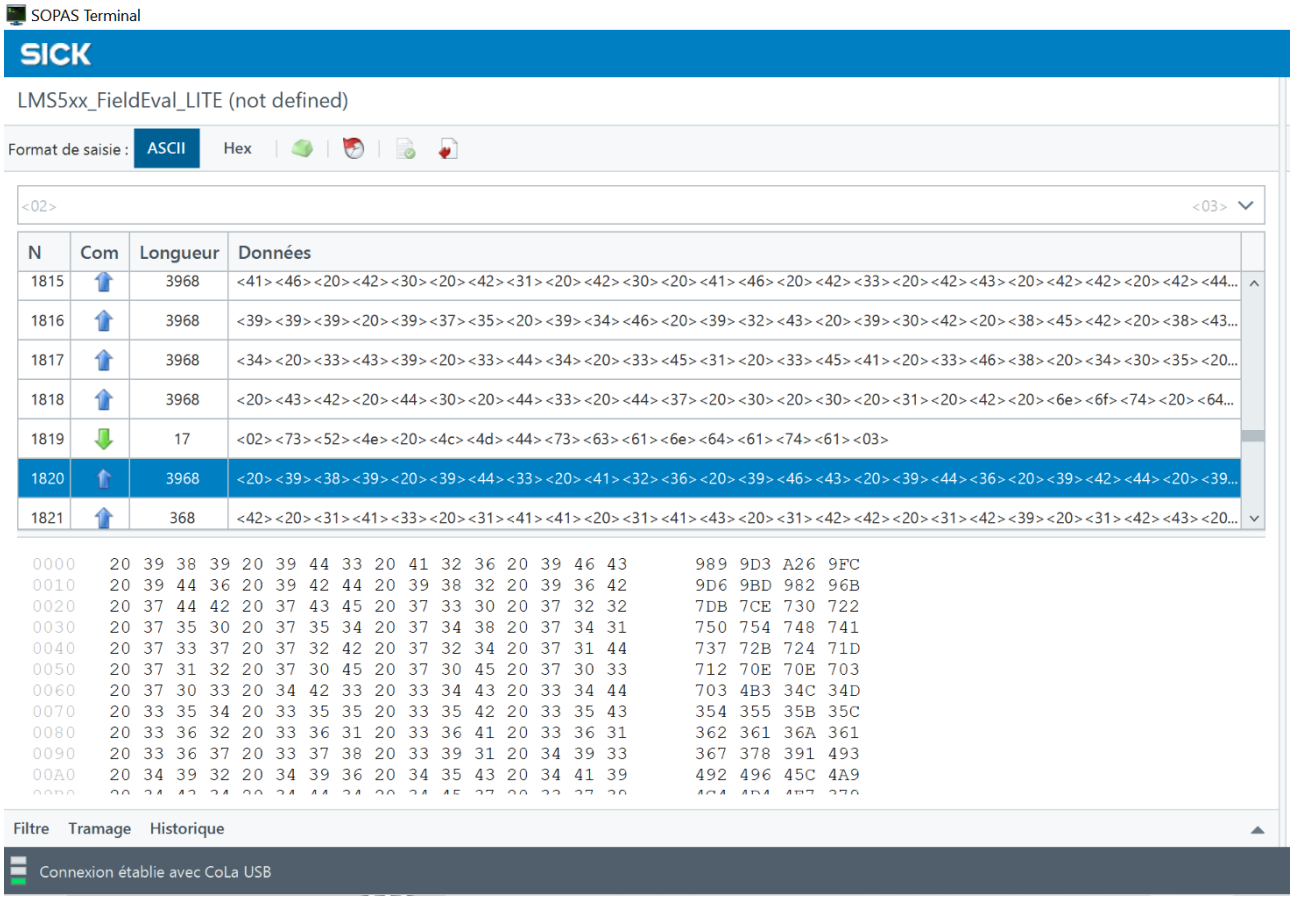


Figure 2.17 Data sent by SICK LMS5xx to PC (through SOPAS)

Matlab

An other alternative to access measurments data from Sick LMS5xx is to use Matlab software. In this case, we used USB interface to do that.Hence, we use some instructions in MATLAB which are of the following form :

```
vik=hex2dec({'02' , '73', '52', '4E', '20', '4C', '4D', '44', '73', '63',
'61', '6E', '64', '61', '74', '61','03'})% sRN LMDscandata

if exist('s')
    fclose(s)
    delete(s)
    clear s
end

s = serial('COM3');
set(s,'Timeout',0.1);
set(s,'InputBuffersize',40000);
set(s,'Terminator','CR');
set(s,'BaudRate',50000)

fopen(s)
s.RecordMode='index'
s.RecordDetail='verbose'
s.RecordName='dd.txt'

record(s)
fwrite(s,vik)
data=dec2hex(fread(s))
```

Figure 2.18 MATLAB instructions

Data Reading Results

After sending the 'sRN LMDscandata' telegram in hexadecimal format from the PC to the Sick LMS 5xx via the Serial interface, we successfully read and processed the hexadecimal measurements in MATLAB. The data measurement is shown in Figure 2.19 below

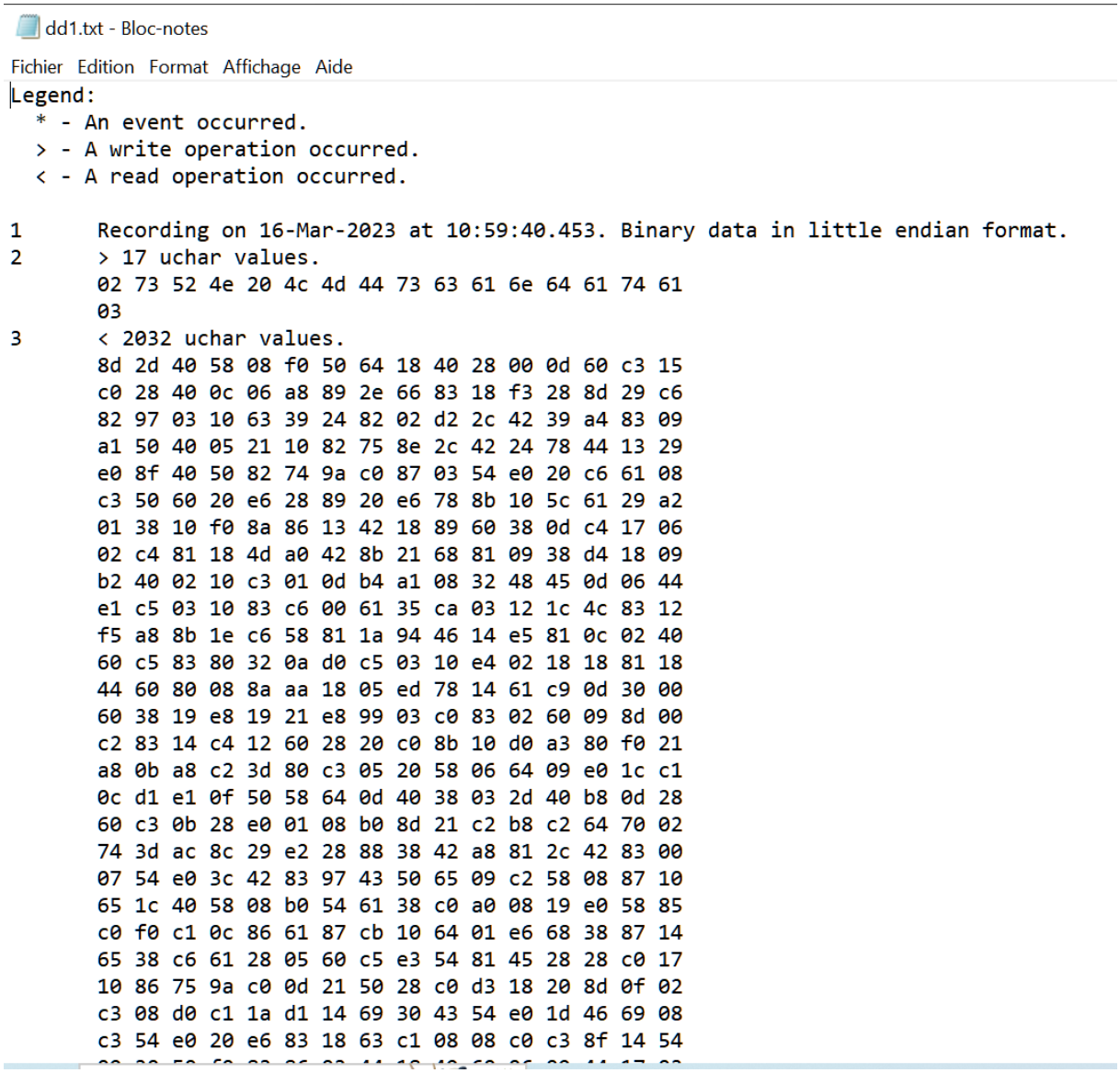


Figure 2.19 Data measurement read using Matlab.

2.5 Conclusion

In conclusion, this chapter provided a comprehensive overview of the Sick LMS5xx laser measurement sensor, its operation, and reading data. By delving into its principles and discussing its various applications, we have highlighted the importance of understanding this sensor for professionals in different fields.

With the introduction of the SOPAS Engineering Tool, this chapter has also highlighted the convenience and efficiency it offers in configuring and accessing data from the sensor. Overall, this chapter has shed light on the operation and reading of data from the Sick LMS5xx laser measurement sensor, enabling professionals to harness its full potential for a wide range of purposes and more particularly in Lidar-based SLAM systems and autonomous navigation of mobile robots.

CHAPTER 3

ROS Implementation of SLAM and Results

3.1 Introduction

In this Chapter, we will focus on the implementation of Simultaneous Localization and Mapping (SLAM) using the Robot Operating System (ROS) framework. We explore the hardware and software components involved in SLAM and present the results from simulations and real-world experiments.

In the following sections, we briefly touch upon the Pioneer P3DX robot, our platform for SLAM implementation. ROS, our chosen framework, offers a flexible architecture for developing complex robotic systems. We discuss the packages and drivers used, including Teleop, Gmapping, ROSARIA, and the Sick-Scan sensor driver. These components enable teleoperation, mapping, communication, and data respectively.

Gazebo, a popular robotics simulator, is utilized for testing the SLAM algorithm, while Rviz is used for visualization. We present then the results from simulations in Gazebo and real-world experiments. Through this chapter, we hope to discover the world of robotics and SLAM techniques.

3.2 Pioneer P3DX robot

The Pioneer 3-DX mobile robot used in this work is presented in figure (3.1). It is a wheeled differential drive mobile robot employed to evaluate the proposed control methods.

Figure 3.1 P3-DX mobile robot.

[16]

Robots may be smaller than most of other robots, but they pack an impressive array of intelligent mobile robot capabilities that rival bigger and much more expensive machines. The P3- DX with onboard PC is a fully autonomous intelligent mobile robot. Pioneer's modest size lends itself very well to navigation in tight quarters and cluttered spaces, such as classrooms, laboratories, and small offices. The P3-DX is fully capable of mapping its environment, finding its way home and performing other sophisticated path-planning tasks.

Weighing only 9 kg (with one battery), the basic P3-DX mobile robot is lightweight, but his strong aluminum body and solid construction makes it virtually indestructible. These characteristics also permit it to carry extraordinary payloads, the P3-DX can carry up to 23 Kg additional weight [16]. The physical dimensions are shown in figure (3.2).

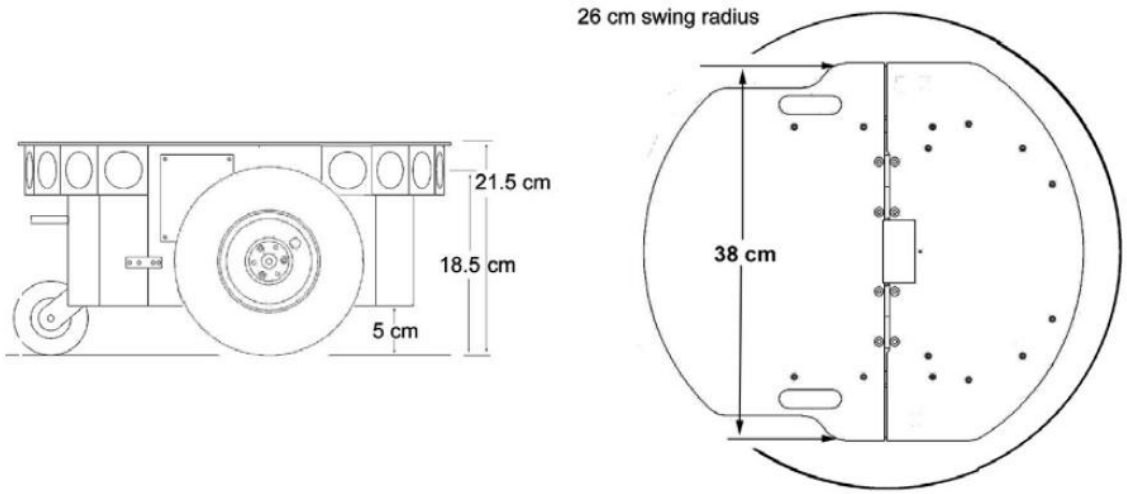


Figure 3.2 Physical dimensions of the robot.

[16]

3.3 Robot Operating System (ROS)

The Robot Operating System (ROS) is a set of software libraries and tools that help us build robot applications from drivers to state-of-the-art algorithms, and with powerful developer tools.

In the field of robotics, there exist various versions of the Robot Operating System (ROS) that cater to different needs and requirements. However, for the purpose of this project, our focus was on utilizing ROS Noetic, which proved to be the most suitable choice for our simulation and real-world experimental endeavors.

Ros noetic ROS Noetic, released in 2020, is a popular version of the Robot Operating

System (ROS). It offers improved compatibility with modern hardware and software frameworks, supports Python 3 as the default language, and provides extensive documentation and tutorials. With an active community and a wide range of open-source packages, ROS Noetic simplifies the development of robust robotic applications [26].

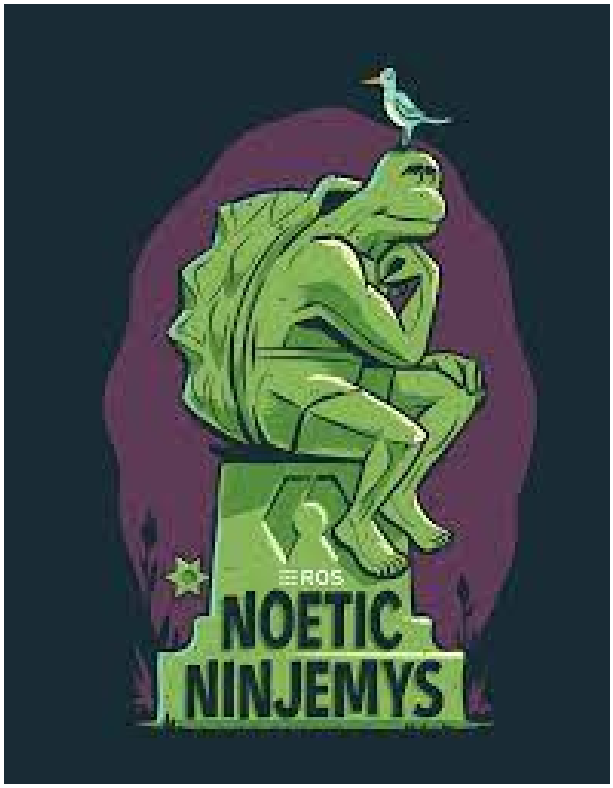


Figure 3.3 Ros noetic ninjemys.

[26]

3.4 Packages and Drivers

In the following subsections, we will discuss the installation procedures and functionality of the packages and drivers utilized in this project, providing a comprehensive overview of their integration within the ROS .

3.4.1 Teleop keyboard package

The package teleop-twist-keyboard authored by Graylin Trevor Jay offers the option of controlling the robot via the keyboard. Figure (3.4) illustrate the essential Command Lines to install and run this package.

```
# Installing
$ sudo apt-get install ros-noetic-teleop-twist-keyboard

#Running
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/
   cmd_vel

#Controls
  Reading from the keyboard  and Publishing to Twist!
  ---------------------------
  Moving around:
   u    i    o
   j    k    l
   m    ,    .

   q/z : increase/decrease max speeds by 10%
   w/x : increase/decrease only linear speed by 10%
   e/c : increase/decrease only angular speed by 10%
  anything else : stop
  CTRL-C to quit
```

Figure 3.4 Command Lines to install and run *teleop_twist_keyboard*. [14]

3.4.2 Gmapping package

The Gmapping package, widely used in the field of robotics, allows for the creation of occupancy grid maps from laser and odometry data. By employing probabilistic techniques, it enables simultaneous localization and mapping (SLAM) capabilities, aiding in autonomous navigation and environment understanding. Figure (3.5) illustrate the essential Command Lines to install and run this package.

```
# Bring Gmapping into the workspace
  $ cd ~/ catkin_ws / s r c
  $ git clone https://github.com/ros-perception/slam_gmapping.git
  $ cd . .
  $ catkin_make
#Running
$ rosrun gmapping slam_gmapping scan:=/laser/scan
```

Figure 3.5 Command Lines to install and run *Gmapping* . [8]

3.4.3 RosAria stack

The RosAria node authored by SrećkoJurić- Kavelj is available for ROS distributions. The node provides an interface with ROS for Adept MobileRobots which use the open source ARIA library. First, the ARIA library must be installed from Adept MobileRobots or its alternative version AriaCoda. The ARIACoda software for Ubuntu 20.04.4, for a 64- bit architecture was downloaded. The RosAria node was cloned from source using git. By using rosdep, we also installed the necessary dependencies. Lastly, the node was built using catkin. To test and run the RosAria node, the "How to use ROSARIA" tutorial is referenced [9].

The RosAria node publishes on multiple topics including pose and battery voltage. In this section we only consider topics relevant to the control objective. RosAria publishes the current pose estimation received from the robot in **`/RosAria/pose`**. The type of this data is **`nav_msgs/Odometry`**.

The message contains the position in xyz coordinates as well as the rotation of the robot. The pose of the robot is set to [0, 0, 0] at the start of the RosAria node. The desired velocity of the robot is set via the **`/RosAria/cmd_vel`** topic. This desired velocity state is set in ARIA, which sends commands every robot cycle (100ms by default) to effect this state in the robot's embedded motion controller, which performs motor control to achieve then maintain the robot velocity automatically (using previously configured acceleration parameters).

In our case, we have connected the mobile robot to our computer using a RS232 serial communications link. Figure (3.6) clarify the necessary Terminal Window Command Lines to build and run the RosAria node.

```
 # Bring ROSARIA into the workspace
$ cd ~/catkin_ws/src
$ git clone https://github.com/amor-ros-pkg/rosaria.git
$ cd ..
$ catkin_make

# Running RosAria
# Tab 1 (roscore)
$ source catkin_ws/devel/setup.sh
$ sudo chmod 777 -R /dev/ttyUSB0    #or /dev/ttyS0
$ roscore
# Tab 2 (rosaria node)
$ source catkin_ws/devel/setup.sh
$ rosrun rosaria RosAria
```

Figure 3.6 Terminal Window Command Lines to build and run the *RosAria* node. [9]

Rosaria client interface: This command launches the RosAria client interface, which provides control and monitoring of the P3dx robot.

```
# Running RosAria client interface
$ source catkin_ws/devel/setup.sh
$ rosrun rosaria_client interface
```

Figure 3.7 Terminal Window Command Lines to Launch $RosAriaclientinterface$. [9]

3.4.4 Package Small House World

The Small House World package is a downloadable resource designed for testing in Gazebo. It provides pre-built assets and models that resemble a small house environment. By using this package, we can create a custom map in Gazebo for simulating various robotic scenarios within a realistic indoor setting.

It offers convenient and efficient tools for evaluating and refining our robot's performance in confined spaces.Figure (3.7) clarify the necessary Terminal Window Command Lines to build and run the AWS Robomaker small house world .

Figure 3.8 AWS Robomaker Small House World on Gazebo.

```
 # Bring AWS Robomaker Small House World into the workspace
$ cd ~/catkin_ws/src
$ git clone https://github.com/aws-robotics/aws-robomaker-small-
   house-world.git
$ cd ..
$ catkin_make

# Running
$ source catkin_ws/devel/setup.sh
$ roslaunch aws_robomaker_small_house_world view_small_house.
   launch
```

Figure 3.9 Installing and running Small House World Package.

3.4.5 Sick-Scan sensor driver

The Sick-Scan sensor driver is a crucial component that enables communication and data acquisition from Sick LIDAR sensors. It provides a seamless interface between the sensor and the Robot Operating System (ROS), allowing for real-time access to laser data. The driver facilitates accurate perception and mapping capabilities by providing high-quality sensor measurements for further processing and analysis within the robotic system.

Figure (3.10) illustrate the essential Command Lines to install and run this package.

```
# Bring Sick_scan into the workspace
$ source /opt/ros/<rosdistro>/setup.bash
$ mkdir -p ~/ros_catkin_ws/src/
$ cd ~/ros_catkin_ws/src/
$ git clone -b devel --single-branch git://github.com/SICKAG/
   sick_scan.git
$ cd ..
$ catkin_make
$ source ~/ros_catkin_ws/install/setup.bash
#Running
$ roslaunch sick_scan sick_lms_5xx.launch
```

Figure 3.10 Terminal Window Command Lines to install and launch sick scan. [1]

3.5 Used tools

Two important tools that are commonly employed in robotics Simultaneous Localization and Mapping are Gazebo and RViz. In the subsequent subsections, we will discuss and define these tools in detail, along with explaining the commands required to launch and install Gazebo and RViz.

3.5.1 Gazebo

Gazebo is a versatile software tool that enables the simulation of robot models before implementing them on hardware. It provides a virtual world where developers can thoroughly test the functionality and performance of their robots. With an easy-to-use programming interface and high-quality graphics, Gazebo offers a user-friendly environment for designing and evaluating robotic systems [7].

Furthermore, Gazebo serves as a multi-robot simulator capable of simulating a wide range of robots, sensors, and objects in a three-dimensional world. It goes beyond simply generating sensor feedback by also simulating realistic physical interactions between objects, including accurate rigid-body physics simulations. This realistic simulation environment allows developers to execute code intended for physical robots on an artificial version, ensuring that the software functions as expected before deployment.

To install Gazebo along with ROS Noetic, we can follow these steps:

- Open a terminal and run the following command:

  $sudo apt-get install ros-noetic-gazebo-ros-pkgs

- Source the ROS setup files in every new terminal session before using ROS or Gazebo:

  $source /opt/ros/noetic/setup.bash

- Once Gazebo is installed, we can launch it by running the following command in a terminal:

  $gazebo

- Alternatively, we can launch Gazebo with Pioneer 3-DX by running the following commands in a terminal:

  $source ~/catkin_ws/devel/setup.sh

  $roslaunch p3dx_gazebo pioneer3dx.launch

In Figure 3.11, we present the P3DX mobile robot situated in an empty Gazebo environment.

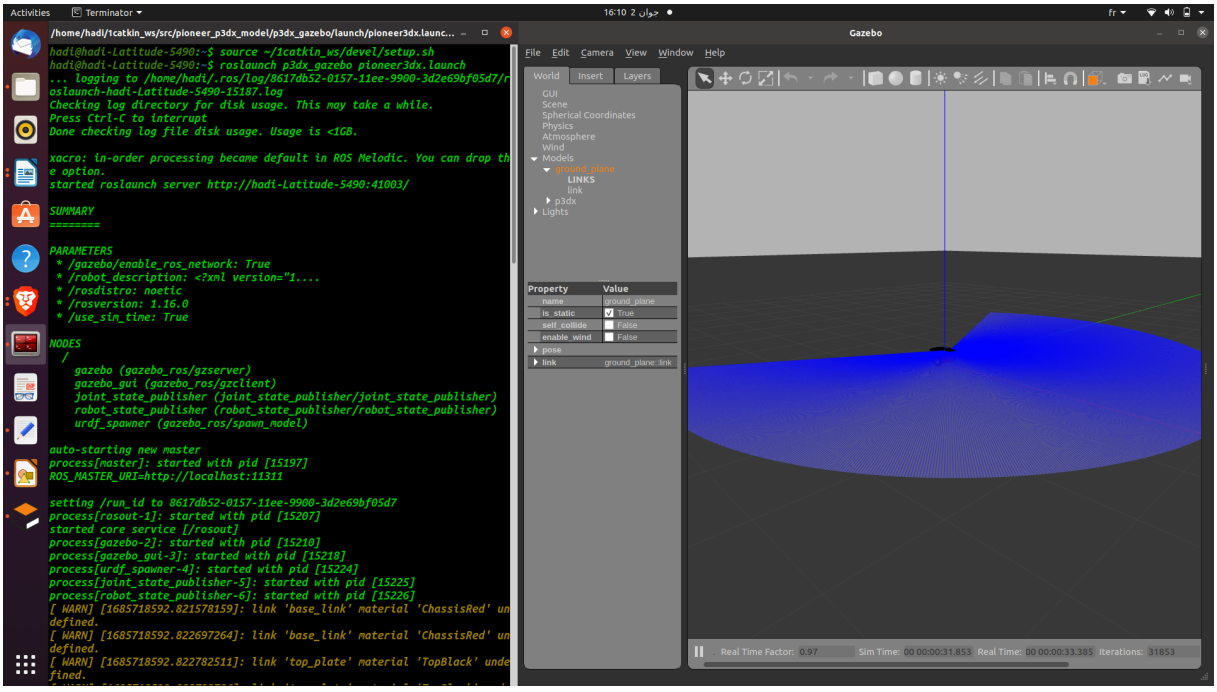


Figure 3.11 P3dx mobile robot in the Gazebo environment.

3.5.2 Ros visualization

Rviz, short for ROS visualization, is an effective 3D visualization tool used in robotics. It offers various capabilities that aid in visualizing and analyzing the data generated by a robot. One of its primary applications is visualizing maps generated by a robot in an unknown environment. Using the gmapping package, Rviz allows the generated map to be saved and used for autonomous navigation. Key topics such as the /scan topic from the laserscan node and the /map topic from the gmapping node are utilized prominently in this process.

Additionally, Rviz provides a comprehensive view of the simulated robot model and enables logging of sensor data from the robot's sensors. This logged data can be replayed, allowing for detailed analysis and debugging of the robot's behavior. By visualizing what the robot perceives, thinks, and does, users can effectively debug and refine their robot applications, from sensor inputs to planned or unplanned actions [?] .

Rviz supports the display of 3D sensor data, such as point clouds generated by stereo cameras, lasers, Kinects, and other devices. It can also visualize 2D sensor data, such as images from webcams or RGB cameras. With these capabilities, Rviz serves as a valuable tool for visualizing and interpreting sensor data, aiding researchers and developers in the analysis and improvement of their robotic systems.

To install RViz (ROS Visualization), we can follow these steps:

- Open a terminal and run the following command to install RViz:

  ```
  sudo apt-get install ros-noetic-rviz
  ```

- Source the ROS setup files in every new terminal session before using ROS or Gazebo:

  ```
  $source /opt/ros/noetic/setup.bash
  ```

- Once RViz is installed, you can launch it by running the following command in a terminal:

  ```
  $rviz
  ```

- Alternatively, you can launch RViz with the Pioneer 3-DX configuration by running the following command in a terminal:

  ```
  $source ~/catkin_ws/devel/setup.sh
  roslaunch p3dx_description rviz.launch
  ```

In Figure 3.12, we present the P3DX mobile robot situated in an Rviz. This figure show also the trajectory of the robot when controlled in gazebo using `teleop_twist_keyboard.py`.

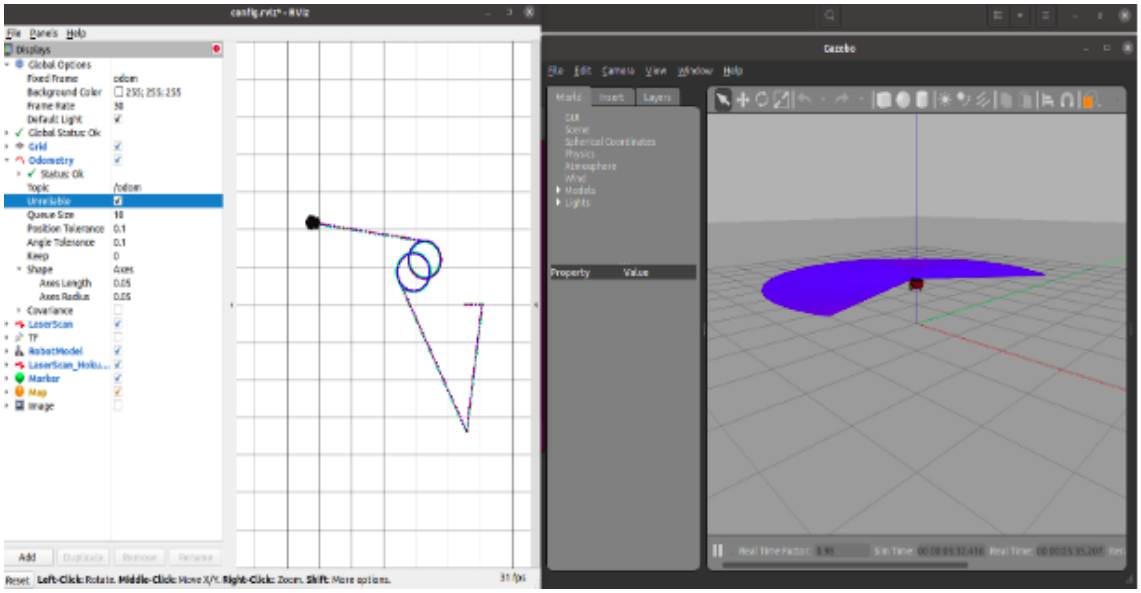


Figure 3.12 P3dx mobile robot in the Rviz

3.5.3 MobileSim

MobileSim is a simulator for use with ARIA. It provides a simple 2D simulation scalable to many robots, and useful for software testing. ARIA automatically detects MobileSim if it is running on the same computer, and will connect to it.

MobileSim will emulate the robot base and sonar , but connection to other devices such as laser, camera, etc. are through separate nodes in ROS and cannot currently be connected to MobileSim.

1. Download the MobileSim package `mobilesim` from [12].
2. Install the MobileSim package using the following command:

   ```
   sudo dpkg i mobilesim_0.9.8+ubuntu16_amd64.deb
   ```

3. Launch MobileSim from the terminal using the following command:

   ```
   MobileSim nomap
   ```

3.6 Results and discussion

As has been mentioned previously, our objective is to build a map of the mobile robot environment first in gazebo and after that in real world environment. Thereby, in order to do that, we considered the webots platform of the simulation to test and evaluate the Gmapping algorithm preferences in terms of building a 2D map , then we take advantage of having the sensor sick lms 5xx and the real P3DX robot to do an experiment test with SLAM in order to build a 2D map.

3.6.1 Simulation Results in Gazebo

After we finished adding those packages, we are now ready to launch our project. To launch the necessary packages for performing SLAM (Simultaneous Localization and Mapping), we can follow the steps below:

In a new terminal:

1. Execute the command:

   `source ~/1catkin_ws/devel/setup.bash`

   This command sets up the environment variables for our workspace.

2. Launch the package `aws_robomaker_small_house_world` by running the command:

   `roslaunch aws_robomaker_small_house_world view_small_house.launch`

   This command starts the visualization of the small house world in Gazebo.

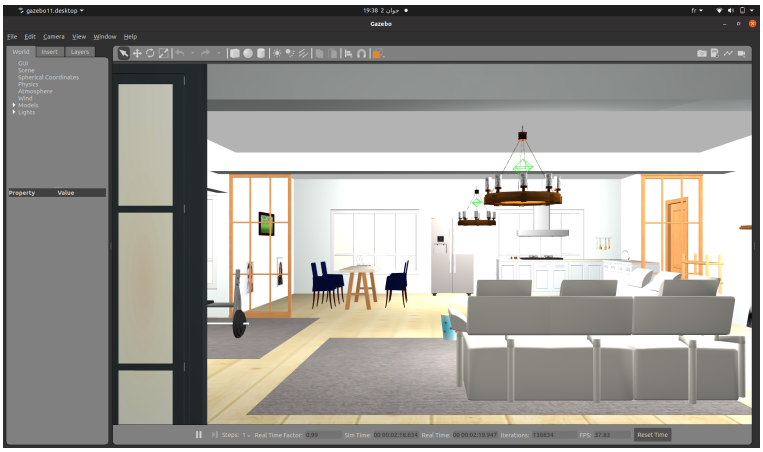


Figure 3.13 Small house world environment gazebo.

In a new terminal:

1. Execute the command:

   `source ~/1catkin_ws/devel/setup.bash`.

2. Launch the package `p3dx_gazebo` by running the command:

   `roslaunch p3dx_gazebo pioneer3dx.launch`.

This command launches the Gazebo simulation of the Pioneer 3-DX robot.

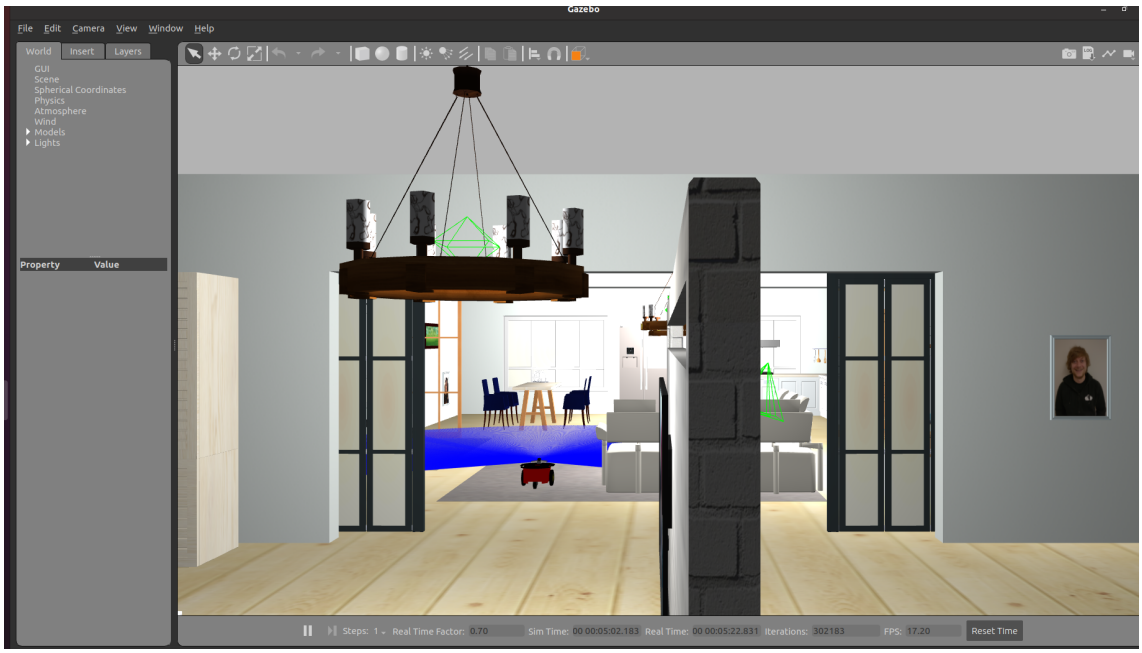


Figure 3.14 Launch p3dx in gazebo.

In a new terminal:

1. Execute the command:

   `source ~/1catkin_ws/devel/setup.bash`.

2. Set the parameter `use_sim_time` to `true` by running the command:

   `rosparam set use_sim_time true`.

   This command enables the use of simulated time in ROS.

3. Start the SLAM algorithm using the `gmapping` package with the following command:

   `rosrun gmapping slam_gmapping scan:=/laser/scan`.

   This command runs the SLAM algorithm using laser scan data from the robot.

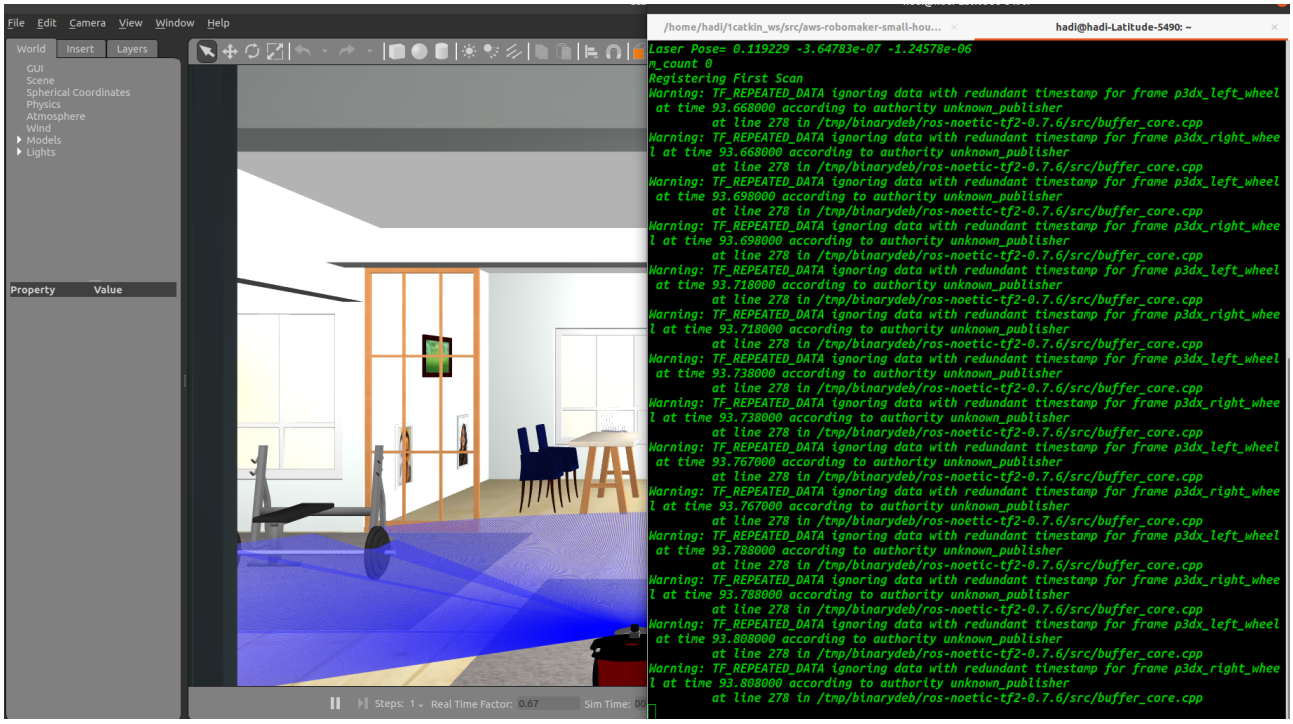


Figure 3.15 Launch Gmapping package

In a new terminal:

1. Execute the command:

   `source ~/1catkin_ws/devel/setup.bash`.

2. Launch the package `p3dx_description` to start RViz visualization by running:

   `roslaunch p3dx_description rviz.launch`.

   This command opens RViz with the robot model and sensor data visualization.

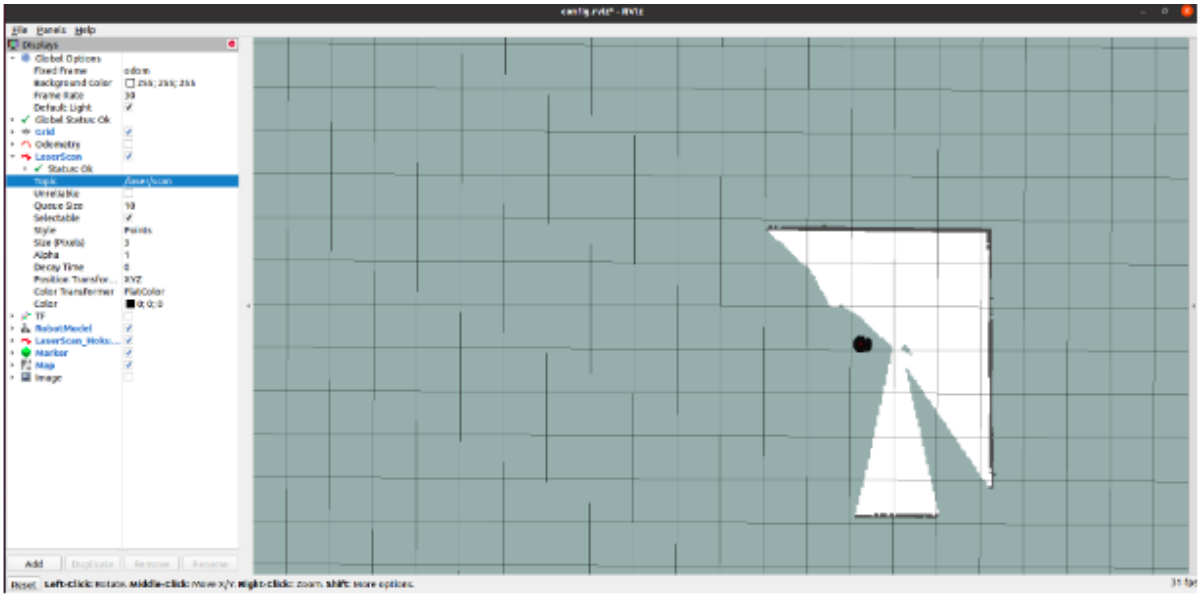


Figure 3.16 Rviz

In a new terminal:

1. Execute the command:

   `source ~/1catkin_ws/devel/setup.bash`.

2. Control the robot using the `teleop_twist_keyboard` package by running:

   `rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/cmd_vel`.

   This command allows us to use the keyboard to send velocity commands to the robot.

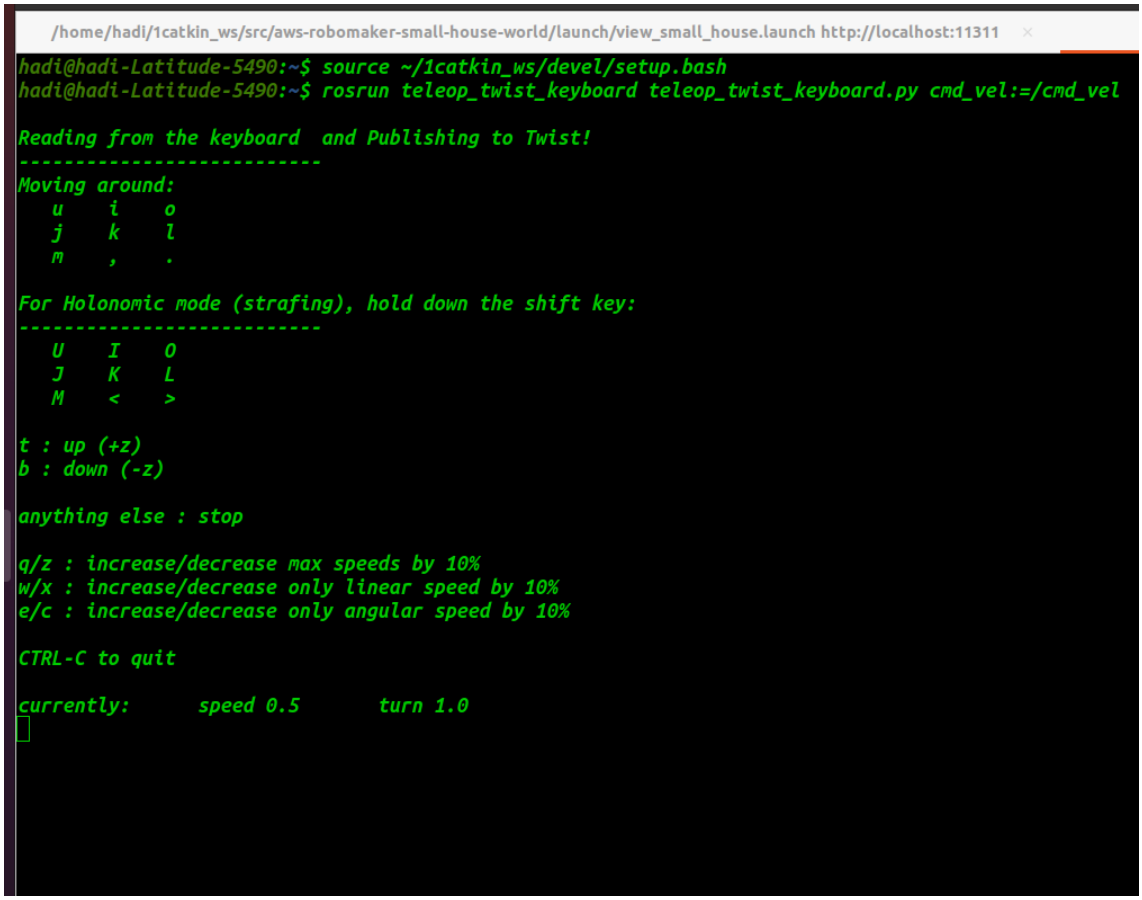


Figure 3.17 Teleop twist keyboard

To simplify the process and reduce the time required, we can use a launch file or a Python script to launch the necessary packages. Instead of running multiple commands in separate terminals, we can create a Python script named 'p3dx.py' and execute it using the command python3 p3dx.py in the terminal.

In Figure 3.18, we present the Python script (p3dx.py) that simplifies the process of launching the necessary packages for SLAM.

Open p3dx.py Save

```
#!/usr/bin/env python3
import os


tab1_command1 = "source ~/1catkin_ws/devel/setup.bash"
tab1_command2 = "roslaunch aws_robomaker_small_house_world view_small_house.launch"


tab2_command1 = "sleep 30 && source ~/1catkin_ws/devel/setup.sh"
tab2_command2 = "roslaunch p3dx_gazebo pioneer3dx.launch"


tab3_command1 = "sleep 30 && source ~/1catkin_ws/devel/setup.bash"
tab3_command2 = "rosparam set use_sim_time true && rosrun gmapping slam_gmapping scan:=/laser/scan"


tab4_command1 = "sleep 30 && source ~/1catkin_ws/devel/setup.bash"
tab4_command2 = "roslaunch p3dx_description rviz.launch"


tab5_command1 = "sleep 30 && source ~/1catkin_ws/devel/setup.bash"
tab5_command2 = "rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/cmd_vel"


os.system(f"gnome-terminal --tab -e 'bash -c \"{tab1_command1}; {tab1_command2}; exec bash\"' --
tab -e 'bash -c \"{tab2_command1}; {tab2_command2}; exec bash\"' --tab -e 'bash -c
\"{tab3_command1}; {tab3_command2}; exec bash\"' --tab -e 'bash -c \"{tab4_command1};
{tab4_command2}; exec bash\"' --tab -e 'bash -c \"{tab5_command1}; {tab5_command2}; exec bash\"'")

#credit: Abdelhadi Bousaid
```

Figure 3.18 Python script to launch Slam

We achieved successful outcomes in our simulation and mapping experiments using the Pioneer P3DX robot. Firstly, we successfully launched the P3DX robot in the Gazebo simulation environment, specifically utilizing the Small House World environment. This allowed us to simulate the robot's movements and interactions with the environment. Next, we employed the Gmapping SLAM algorithm to generate a 2D map of the environment.

By collecting data from the robot's sensors and utilizing the Gmapping package, additionally, we achieved successful control of the Pioneer P3DX robot using the Teleop Twist Keyboard package. This allowed us to control the robot's movements manually using the Keyboard interface.This allowed us to simulate the robot's movements and interactions with the environment. Figure 3.19 provides a visual representation of the successful launch, showcasing the robot operating within Gazebo.

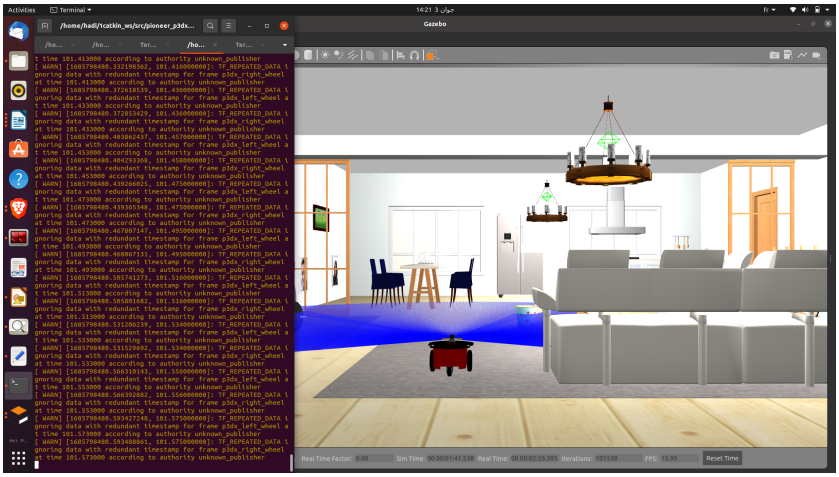


Figure 3.19 The beginning of building the map

Finally , the resulting 2D map demonstrated the robot's exploration and the successful mapping process. Figure 3.20 presents this map within the RViz visualization tool.

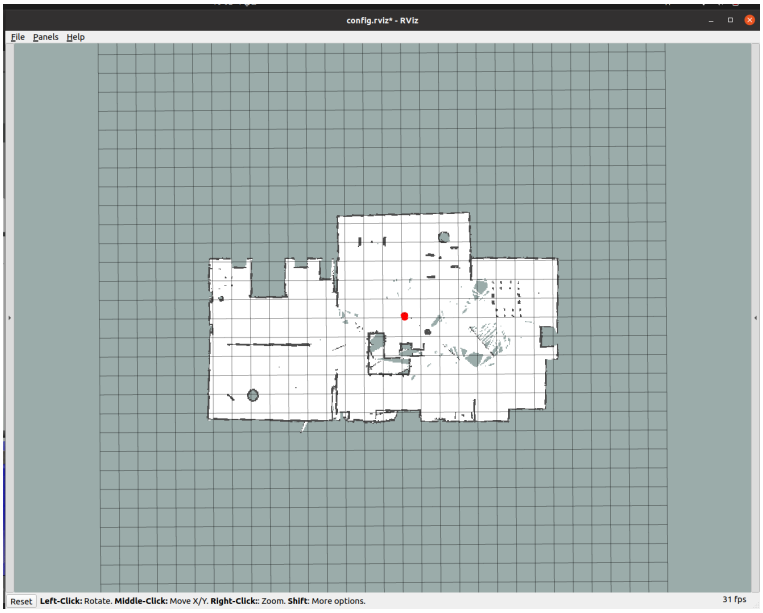


Figure 3.20 The end of building the map

To save the map in RViz using a command, we can use the following command:

`rosrun map_server map_saver -f <map_name>`.

We replace <map_name> with the desired file path

Results for a created environment in Gazebo

Gazebo also offers us the possibility of creating our own environment. Figure 3.20 shows a simple environment we created . We can also add obstacles in it. The map corresponding to that environment is shown in Figure 3.22.

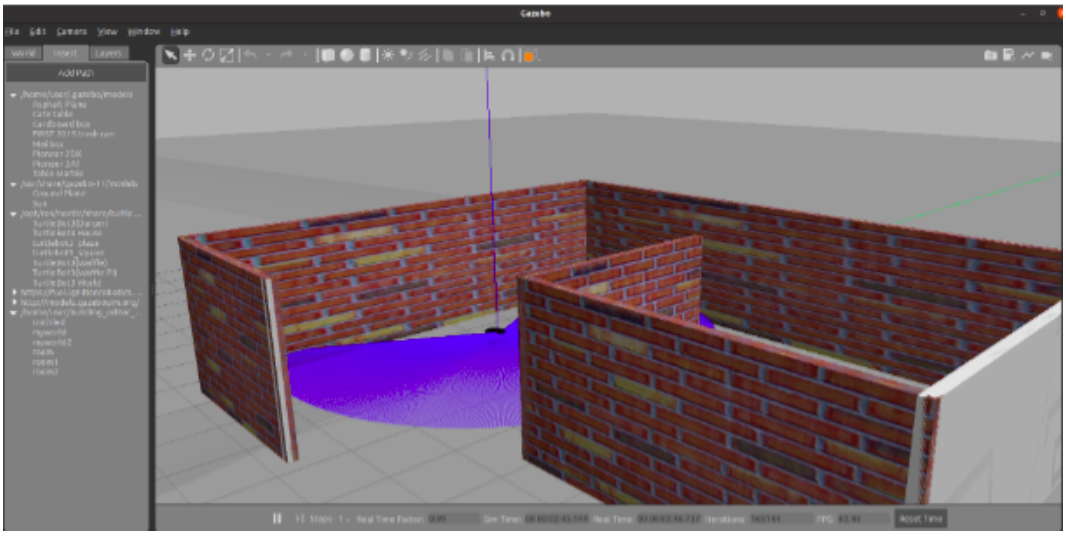


Figure 3.21 Empty environment with Gazebo

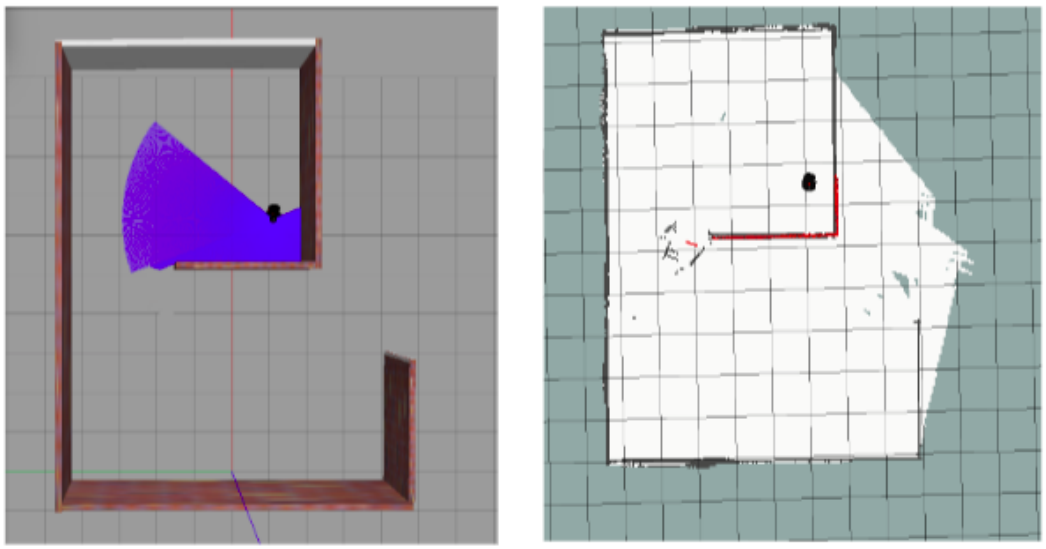


Figure 3.22 The environment we created: (a) in gazebo (b) the obtained map in rviz

3.6.2 Results using MobileSim

To test the fonctionnality of Rosaria and rosaria_client interface, we performe a simulation using MobileSim. We teleoperate the simulated P3DX mobile robot using rosaria_client interface with the commands below:

1. Start the mobilesim Application
2. New window, 3 new Tabs:
   (a) In the first Tab:

```
hadi@Latitude-5490:~$ source ~/catkin_ws/devel/setup.bash
hadi@Latitude-5490:~$ roscore
```

   (b) In the second Tab:

```
hadi@Latitude-5490:~$ source ~/catkin_ws/devel/setup.bash
hadi@Latitude-5490:~$ rosrun rosaria RosAria
```

   (c) In the third Tab:

```
hadi@Latitude-5490:~$ source ~/catkin_ws/devel/setup.bash
hadi@Latitude-5490:~$ rosrun rosaria_client interface
```

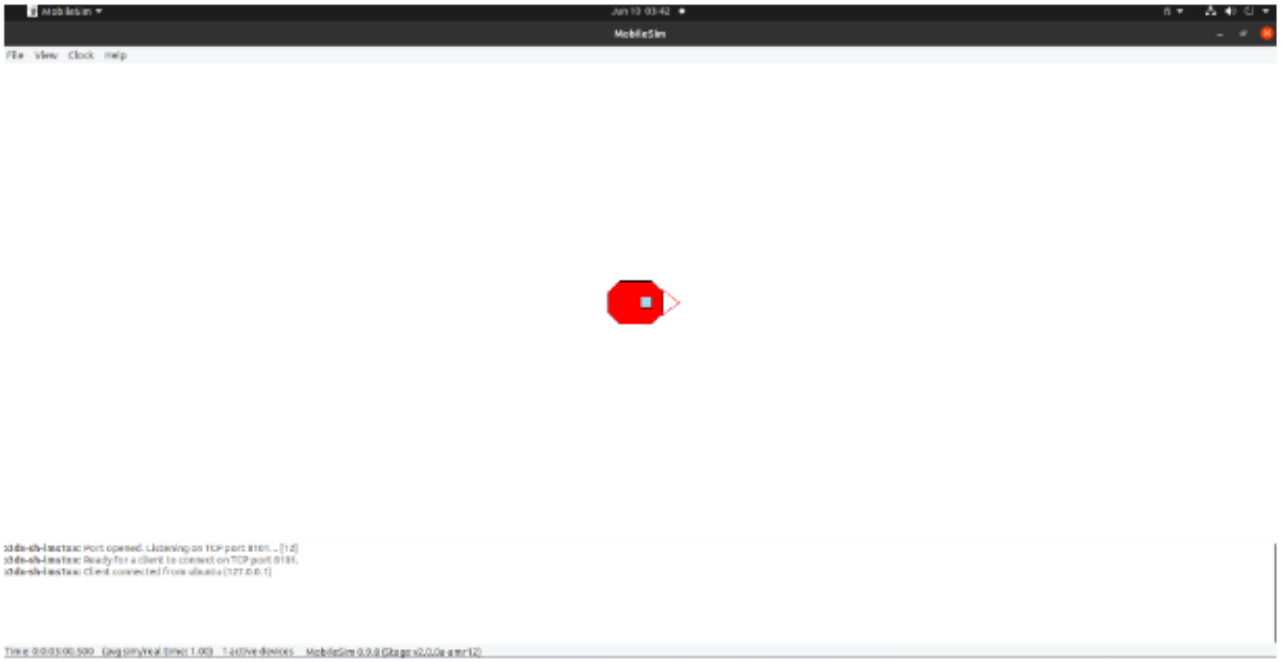


Figure 3.23 Initial position of the robot

Figure 3.24 New pose of the robot after teleoperation

By observing the successful movement of the Pioneer P3DX robot within the mobile simulator with ROSARIA client interface , we can confirm that ROSARIA is functioning correctly.

3.6.3 Experimental Results

First of all, we have teleoperated successfully the real P3dx robot using rosaria and rosaria_client interface . Second , we used the sick_scan driver with the previous commands which allows us to read data (scans) from the sick LMS5xx lidar placed on the P3dx robot.

Figure 3.25 Sick lms5xx and robot P-3DX

To establish a communication between our PC and the Sick LMS5xx, we used we used the Ethernet interface and found some problems (troubleshouting) to do so , but finally we managed to access the measures.

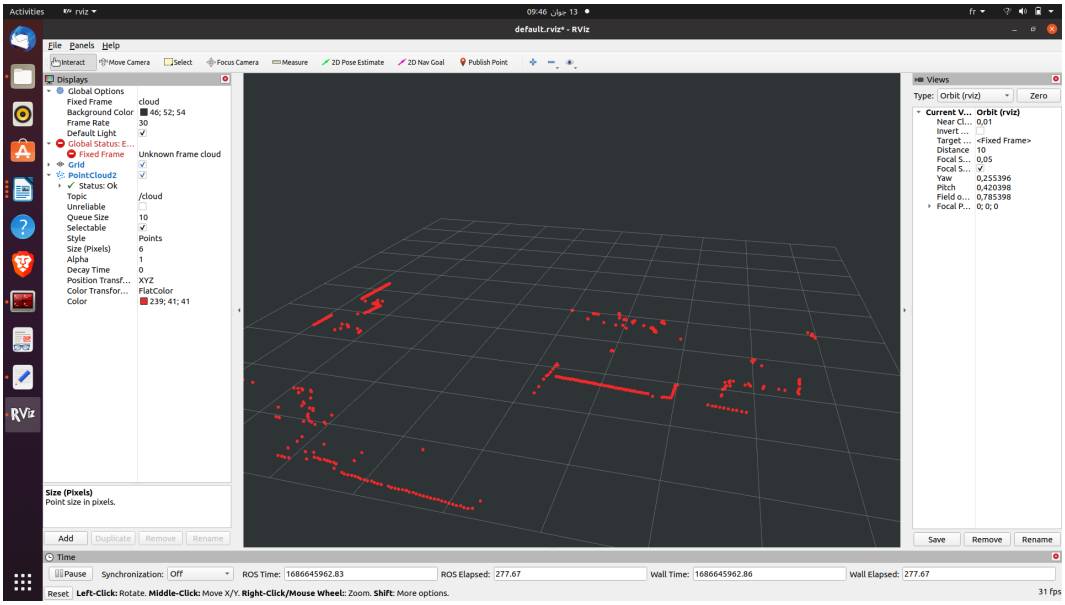


Figure 3.26 A scan measured by the Sick LMS5xx scanner

Figure 3.25 shows some simple test results. a 2D map scan was displayed in Rviz the une part of the environment world in front of it . all these results pave the way for the next experiment.

Now, we had this experiment to gather the Sick lms5xx with the P-3DX robot and scan a big room environment in LTSS laboratory to build a 2D maps. Let's go through the commands to set up SLAM using gmapping with our Pioneer P3DX robot and the Sick LMS5xx laser scanner.

In a terminal:

1. Execute the command: `source /catkin`$_w$*s/devel/setup.bash*

   This command sets up the environment variables for your ROS workspace.

2. Execute the command: `roscore`

   This command starts the ROS core, which provides essential services for communication between ROS nodes.

In a new terminal:

1. Execute the command: `source ~/catkin`$_w$*s/devel/setup.bash.*

2. Execute the command: `sudo usermod -a -G dialout $Hadi` OR: `sudo chmod 777 -R /dev/ttyUSB0`

   This command adds your user to the "dialout" group, granting access to serial ports.

3. Execute the command: `rosrun rosaria RosAria`

   This command launches the RosAria node, which connects to the Aria software for communication with the P3dx robot.

In a new terminal:

1. Execute the command: `source ~/catkin`$_w$*s/devel/setup.bash.*

2. Execute the command: `roslaunch sick`$_s$*cansick*$_l$*ms*$_5$*xx.launch*

   This command launches the driver for the Sick LMS 5xx laser scanner, used for obtaining laser scan data from the robot.

In a new terminal:

1. Execute the command: `source ~/catkin_ws/devel/setup.bash`.

2. Execute the command: `rosrun tf static_transform_publisher 0.17 0 0.4 0 0 0 base_link cloud 33`

   This command publishes a static transform between the "base_link" frame and the "cloud" frame. Adjust the translation values (0.17, 0, 0.4) as needed.

In a new terminal:

1. Execute the command: `source ~/catkin_ws/devel/setup.bash`.
2. Execute the command: `rosrun gmapping slam_gmapping scan:=scan`

   This command runs the GMapping package, which performs SLAM using the laser scan data from the robot. The scan:=scan argument sets the topic name for the laser scan data.

In a new terminal:

1. Execute the command: `source ~/catkin_ws/devel/setup.bash`.
2. Execute the command: `rosrun rosaria_client interface`

   This command launches the RosAria client interface, which provides control and monitoring of the P3dx robot.

In a new terminal:

1. Execute the command: `source ~/catkin_ws/devel/setup.bash`.
2. Execute the command: `rosrun map_server map_saver f carte1`

This command saves the map generated by GMapping. Adjust the filename (carte1) as needed. We can repeat the same process to save additional maps by changing the filename in the map_saver command.

Figure 3.27 The environment of the scan .

We initiated the scanning process in the LTSS environment using the aforementioned commands. The SLAM algorithm executed successfully, resulting in the desired outcome, and we obtained the corresponding map with Rviz, as depicted in Figures 3.28 and 3.29.

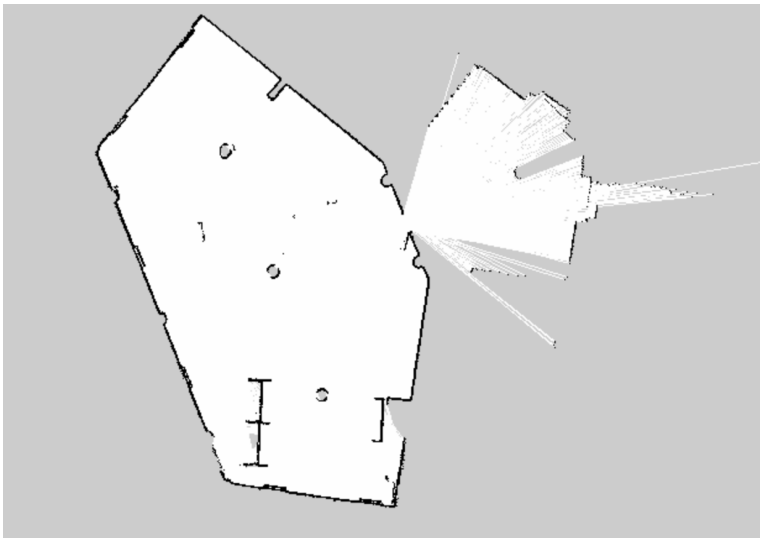


Figure 3.28 Map of LTSS room .

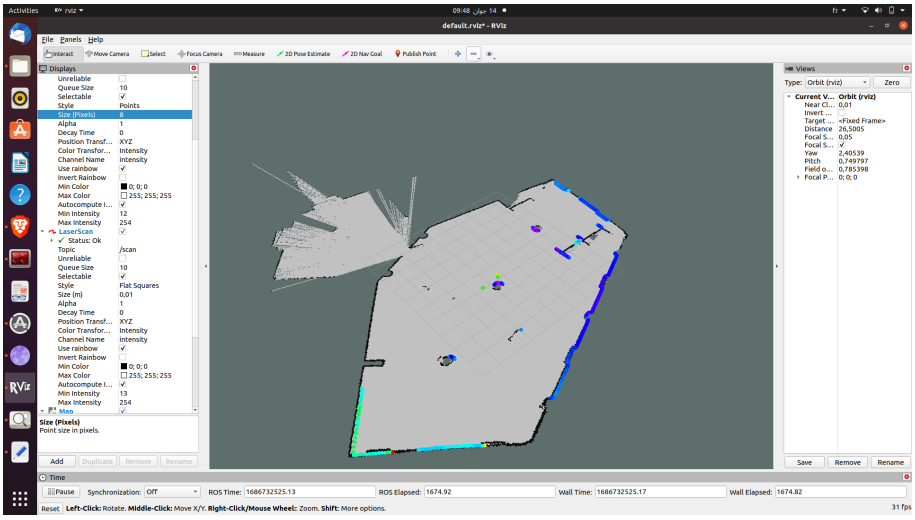


Figure 3.29 Map of LTSS room in Rviz .

3.7 Conclusion

In conclusion, Chapter 3 presented the ROS implementation of SLAM on the Pioneer P3DX robot. We discussed the packages, drivers, and tools used for the implementation, including Teleop keyboard, Gmapping, RosAria, Small House World, and the Sick-Scan sensor driver.

We also showcased simulation results in Gazebo, results using MobileSim, and experimental results. The implemented SLAM system demonstrated reliable mapping and localization capabilities, highlighting its potential for real-world applications . The obtained map could be used for autonomous navigation of the robot in future works.

GENERAL CONCLUSION

In this thesis, we have explored the field of Lidar-based SLAM for mobile robots, with a specific focus on the Sick LMS5xx laser measurement sensor. We started by providing an overview of SLAM and its significance in enabling autonomous robot navigation and mapping. We discussed the importance of perception in robotic systems, particularly exteroceptive sensors, and highlighted the role of Lidar in accurately capturing the environment. Our study focused on understanding the operation and reading of data from the Sick LMS5xx sensor.

We explored its operating principles, including distance and direction measurements, as well as data interfaces and communication protocols. Through the use of the SOPAS Engineering Tool, we gained insights into the software aspects of the sensor and its integration with the Robot Operating System (ROS). By leveraging packages and drivers such as Teleop keyboard, Gmapping, RosAria, and Sick-Scan sensor driver, we were able to perform SLAM and generate maps of simulated environments using Gazebo and MobileSim. We also conducted real-world experiments to validate the performance of our SLAM implementation.

The results of our simulation experiments and real-world tests demonstrate the effectiveness of the Lidar-based SLAM approach using the Sick LMS5xx sensor. We successfully achieved our goal of implementing SLAM in both simulated and real-world environments, showcasing the sensor's capabilities for accurate mapping and localization. Looking ahead, the maps generated through our SLAM system provide a foundation for future navigation tasks. By utilizing the acquired maps, mobile robots can navigate autonomously and make informed decisions based on their surroundings.

This opens up opportunities for various applications, such as path planning, obstacle avoidance, and efficient exploration of unknown environments. In conclusion, this thesis has contributed to the field of mobile robotics by providing insights into Lidar-based SLAM and showcasing the capabilities of the Sick LMS5xx sensor. The knowledge and findings presented here can serve as a valuable resource for researchers and practitioners working in the field. With further advancements in SLAM algorithms and sensor technologies, we can expect even greater autonomy and navigational capabilities for mobile robots in the future.

References

[1] S. AG. Sick-scan sensor driver - ros wiki. Available at http://wiki.ros.org/sick_scan.

[2] T. S. . author. *Probabilistic Robotics*. The MIT Press, 2005.

[3] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng, and Y. Chong. Sensor technologies and simultaneous localization and mapping (slam). *Procedia Computer Science*, 76:174–179, 2015.

[4] I. Dobriakov. Design and development of a ros based lidar platform to scan environment for a mobile robot's autonomous motion. 2019.

[5] H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, 4(1):23–31, 1988.

[6] D. Filliat*. Vers une cartographie sémantique d'environnements intérieurs. *Réalités industrielles*, (1):49–55, 2012.

[7] O. S. R. Foundation. Gazebo - the robot simulation toolbox. Available at http://gazebosim.org.

[8] G. Grisetti, C. Stachniss, and W. Burgard. gmapping - ros wiki, 2011. Available at http://wiki.ros.org/gmapping.

[9] M. Henning. rosaria - ros wiki, 2021. Available at http://wiki.ros.org/rosaria.

[10] K. H. Low, W. K. Leow, and M. H. Ang Jr. A hybrid mobile robot architecture with integrated planning and control. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 219–226, 2002.

[11] N. Mehendale and S. Neoge. Review on lidar technology. *Available at SSRN 3604309*, 2020.

[12] MobileRobots. Mobilerobots research and academic customer support rss. MobileRobots Research and Academic Customer Support RSS, Accessed 2023.

[13] J. Nieto, T. Bailey, and E. Nebot. Recursive scan-matching slam. *Robotics and Autonomous systems*, 55(1):39–49, 2007.

[14] M. Purvis. teleop-twist-keyboard - ros wiki, 2020. Available at `http://wiki.ros.org/teleop_twist_keyboard`.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[16] A. M. Robots. Pioneer 3 operations manual, 2010.

[17] SICK AG. Operating instructions lms5xx 2d lidar sensors. SICK AG · Germany, May 2015. Accessed on June 17, 2023.

[18] SICK AG. Laser measurement technology. LMS5xx Laser Measurement Technology - SICK AG, Accessed 2023.

[19] SICK AG. SICK SOPAS Engineering Tool, Accessed 2023.

[20] SICK AG. Technical data. SICK, Accessed 2023.

[21] SICK AG. Technical information telegram listing - ranging sensors lms5xx. Technical Information Telegram Listing - SICK, Accessed 2023.

[22] H. Taheri and Z. C. Xia. Slam; definition and evolution. *Engineering Applications of Artificial Intelligence*, 97:104032, 2021.

[23] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[24] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the national conference on artificial intelligence*, pages 944–951, 1996.

[25] H. D. Whyte. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.

[26] R. Wiki. Documentation-ros wiki. *URL: http://www. ros. org*.

[27] D. Wilbers, C. Merfels, and C. Stachniss. A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 220–225. IEEE, 2019.