Politecnico di Bari

Department of Electrical and Information Engineering

Master Degree in Automation Engineering

SIMULTANEOUS LOCALIZATION AND MAPPING USING MULTIPLE RADARS FOR AUTONOMOUS NAVIGATION

Supervisor
Prof. Eng. Saverio Mascolo

Co-supervisor
Eng. Vito Andrea Racanelli

Candidate
Nicola Pace

Academic Year 2022–2023

LIBERATORIA ALLA CONSULTAZIONE DELLA TESI DI LAUREA DI CUI ALL'ART.4 DEL REGOLAMENTO DI ATENEO PER LA CONSULTAZIONE DELLE TESI DI LAUREA (D.R. n. 479 del 14/11/2016).

Il sottoscritto **NICOLA PACE** matricola **580440**
Corso di Laurea **INGEGNERIA DELL'AUTOMAZIONE MAGISTRALE**

Autore della presente tesi di Laurea dal titolo:
SIMULTANEOUS LOCALIZATION AND MAPPING USING MULTIPLE RADARS FOR AUTONOMOUS NAVIGATION

Parole chiave: **autonomous navigation, millimeter wave, SLAM, localization, mapping**

Abstract: **Simultaneous localization and mapping (SLAM) is a crucial problem for autonomous navigation, as it allows an autonomous system to estimate its pose and build a consistent map of the environment in real time. However, traditional SLAM approaches can be prone to errors, especially in dynamic and challenging environments. This thesis presents a new approach to SLAM using multiple radars, in particular millimeter wave radars, which can provide complementary information and improve the robustness of the localization and mapping process. The proposed approach includes algorithms for fusing the measurements from multiple mmwave radars, modeling the uncertainties and correlations, and estimating the pose and map of the environment. The performance of the proposed approach is evaluated through experiments on real-world datasets and in real scenarios on different mobile robots, and the results demonstrate its accuracy and efficiency compared to other existing techniques. The thesis also discusses the limitations of the proposed approach and suggests directions for future work.**

☐ Autorizza
☐ Non autorizza

la consultazione della presente tesi, fatto divieto a chiunque di riprodurre in tutto o in parte quanto in essa contenuto.

Bari, 09/02/2023

Firma

Abstract

Simultaneous localization and mapping (SLAM) is a crucial problem for autonomous navigation, as it allows an autonomous system to estimate its pose and build a consistent map of the environment in real time. However, traditional SLAM approaches can be prone to errors, especially in dynamic and challenging environments. This thesis presents a new approach to SLAM using multiple radars, in particular millimeter wave radars, which can provide complementary information and improve the robustness of the localization and mapping process. The proposed approach includes algorithms for fusing the measurements from multiple mmwave radars, modeling the uncertainties and correlations, and estimating the pose and map of the environment. The performance of the proposed approach is evaluated through experiments on real-world datasets and in real scenarios on different mobile robots, and the results demonstrate its accuracy and efficiency compared to other existing techniques. The thesis also discusses the limitations of the proposed approach and suggests directions for future work.

Contents

Chapter 1

Introduction

Autonomous systems, such as self-driving cars, drones, and robots, are becoming increasingly prevalent in various applications. These systems rely on accurate and up-to-date maps of their surroundings to make informed decisions about their movements and actions. Simultaneous localization and mapping (SLAM) is a problem that addresses this need by allowing an autonomous system to estimate its pose (position and orientation) and build a consistent map of the environment in real time.

SLAM is a crucial problem in robotics and autonomous systems. It involves the creation of a map of the environment and the simultaneous determination of the pose (position and orientation) of the robot within the map. There are various sensors that can be used for SLAM, such as laser scanners, cameras, and radar. In this thesis, we will focus on using multiple millimeter wave (mmWave) radars for SLAM.

Millimeter wave radars operate in the frequency range of 30-300 GHz and have a number of advantages for robotics and autonomous systems. They can operate in challenging environments, such as those with poor lighting or occlusions, and they can detect objects at long ranges and high speeds. Millimeter wave radars can also provide high-resolution range and velocity measurements, which can be used for SLAM. However, traditional SLAM approaches with millimeter wave using a single sensor or without data processing can be prone to errors, especially in dynamic and challenging environments with different obstacles or changing conditions.

One way to address these issues is to use multiple sensors and applying a data-augmentation framework, to provide complementary information and improve the robustness of the localization and mapping process. However, using multiple mmWave radars for SLAM presents its own challenges. The radars may have different poses, fields of view, and measurement noise characteristics, and the data from the multiple radars must be fused in a consistent and accurate way. In this thesis, we will explore methods for using multiple mmWave radars for SLAM and address these challenges. The contributions of this thesis are as follows:

1. We will investigate various approaches to fuse the data from multiple mmWave radars for SLAM.

2. We will propose a new method for SLAM using multiple mmWave radars, which is designed to handle the challenges mentioned above.

3. We will evaluate the proposed method in simulations and experiments, and compare its performance to other state-of-the-art methods.

The rest of the thesis is organized as follows. In the literature review, we discuss the relevant background literature on SLAM and multiple radar systems, and survey the existing approaches and techniques for using millimeter wave radars for SLAM. In the problem formulation, we briefly pose some theoretical bases, and then formally define the problem and derive the mathematical model for the proposed approach. The proposed approach is then described in detail, including the algorithms and techniques used and the computational complexity and implementation considerations. The performance of the proposed approach is evaluated through real case scenario experiments and applications on real-world data sets, and the results are presented and discussed. Finally, the conclusion summarizes the main contributions and findings of the thesis, and discusses the limitations and future work.

1.1 Target

The target of this thesis is to study and find a way to define a Simultaneous Localization and Mapping (SLAM) framework to use with one or multiple mmWave sensors. The use of mmWave sensors for autonomous navigation has been gaining popularity in recent years due to their high-resolution range and velocity information. However, there are still many challenges to be addressed when it comes to integrating mmWave sensors into a SLAM system.

The main objective of this thesis is to investigate different techniques for integrating mmWave sensors into a SLAM framework. This will include an in-depth analysis of existing methods and the development of new techniques to improve system performance.

To achieve this objective, a series of experiments will be conducted using both simulated and real-world environments. These experiments will be carried out using a variety of mmWave sensors and different configurations of the SLAM system. The results will be analyzed and compared to determine the most effective methods for integrating mmWave sensors into a SLAM framework.

In addition to the technical aspects, this thesis will also focus on the practical implications of the proposed SLAM framework. This includes evaluating the real-world performance of the system and determining its potential applications in various fields such as autonomous vehicles, and robotics.

To sum up, the goal of this thesis is to study and find a way to define a SLAM framework to use with one or multiple mmWave sensors. By analyzing different techniques and conducting experiments, this thesis aims to improve the performance of the system and pave the way for the widespread use of mmWave sensors in autonomous navigation.

Chapter 2

Literature review

Simultaneous Localization and Mapping (SLAM) is a crucial problem in the field of robotics and autonomous systems, entailing the creation of a map of the environment and the simultaneous determination of the pose (position and orientation) of the robot within said map. SLAM is of paramount importance as it enables robots to navigate and operate in unknown or dynamic environments, which is vital for a wide range of applications, including autonomous vehicles, service robots, and drones.

The SLAM problem consists of two main components: mapping and localization. Mapping encompasses the creation of a representation of the environment, which can take the form of a 2D or 3D map depending on the specific application. Localization, on the other hand, involves determining the pose of the robot within the map. This can be represented by a set of coordinates (x, y, z) and orientation (roll, pitch, yaw) relative to a fixed frame of reference.

To solve the SLAM problem, the robot must be equipped with sensors that can measure the environment and its own pose, such as laser scanners, cameras, radars, and other sensors that can measure distances to nearby objects, environment appearances or the robot's motion. These sensor measurements are used to update the map and the robot's pose estimate in real-time as the robot traverses the environment.

A plethora of algorithms and techniques have been developed to solve the SLAM problem, including probabilistic approaches, geometric approaches, and machine learning-based approaches. The choice of algorithm depends on the sensor characteristics, the environment complexity and the computational resources available.

Different sensors can also be used for SLAM, including laser scanners, cameras and radars. In this chapter, we specifically focus on the use of millimeter wave (mmWave) radars for SLAM.

MmWave radars operate in the frequency range of 30-300 GHz and have several advantages for robotics and autonomous systems. They are capable of operating in challenging environments, such as those with poor lighting or occlusions, and can detect objects at long ranges and high speeds. Additionally, mmWave radars can provide high-resolution range and velocity measurements,

which are useful for SLAM.

However, using multiple mmWave radars for SLAM also brings its own set of challenges. The radars may have different poses, fields of view, and measurement noise characteristics, and the data from multiple radars must be fused consistently and accurately.

One approach for using multiple mmWave radars for SLAM is to utilize a probabilistic framework, such as a particle filter or a Kalman filter. These methods can fuse measurements from multiple radars in a probabilistic manner, taking into account the uncertainty in the measurements and the robot's pose. However, these methods can be computationally intensive and may not scale well to large environments or large numbers of radars.

Another approach is to use geometric techniques, such as multi-view stereo or simultaneous localization and mapping (SLAM). These methods exploit geometric constraints between the radars and the environment to jointly estimate the map and the robot's pose. However, these methods may be sensitive to initialization errors and may not handle dynamic environments well.

A third approach is to use machine learning techniques, such as deep neural networks or support vector machines, to learn the relationship between the radar measurements and the robot's pose. These methods can be trained on a large dataset of radar measurements and poses, and can generalize to new environments and configurations. However, these methods may require a large amount of training data and may not be interpretable.

In the following section, a review of the scientific literature on the argument will be presented with an in-depth view on each article.

2.1 Analyzed approaches

In this section various studies and approaches will be analyzed.

One of the first achievements in using millimeter wave radar for localization and mapping purposes is [12]. The article discusses a method for using a 77GHz millimeter wave radar in autonomous land vehicle navigation. The approach involves utilizing an extended Kalman filter to estimate both the vehicle state and map features. Radar reflectors are implemented for increased visibility and the polarisation of returned radar signals is used to identify natural features as navigational markers. This method does not require prior knowledge of the environment or any target infrastructure.

In the same line in time, another first approach on the problem is expressed in [13]. The article describes an implementation of the SLAM algorithm on a vehicle using millimeter-wave radar to provide map observations in an outdoor environment. The implementation is used to demonstrate how key issues such as map management and data association can be handled in a practical setting and the results are promising.simultaneously.

More modern approaches at the problem of using mmWave for SLAM application will now be explored.

Firstly, in the subject of dealing with mmWave data, to generate pointclouds, the authors in [9] present a novel method for generating high-quality radar point clouds using mmWave radar technology. The authors have discussed the challenges that currently exist with using mmWave radar point clouds in mobile robotics and autonomous driving, such as sparsity and the presence of "ghost" targets. Through an analysis of the reasons for these problems, they have proposed a new method for point cloud generation, along with a new evaluation metric. Experimental results on a new dataset in real-world scenes have shown that the proposed method outperforms other methods in terms of the quality of the radar point clouds generated. Additionally, by applying these high-quality point clouds to object detection, localization, and mapping tasks, the results have demonstrated that the proposed method can significantly improve the environment perception ability of mobile robots. Overall, this article represents a valuable contribution to the field of mmWave radar technology and its application in mobile robotics and autonomous driving.

Going more in depth in the SLAM problem, a good starting point is the work presented in [7], where the authors provide a comprehensive overview of the current state of the art in radar-based mapping and localization. They discuss the advantages and limitations of radar-based sensors and their potential applications in mobile robotics. Also, in "The Potential of Low-cost Millimeter-Wave Sensors for Mobile Radar Applications" [20], Grosswindhager et al. analyze the potential of low-cost millimeter-wave sensors for mobile radar applications. They present an overview of the current state of the art in millimeter-wave radar sensors, their characteristics and potential applications in mobile robotics. Another review of Indoor Millimeter Wave Device-based Localization and Device-free Sensing Technologies is presented in [43]. Here the authors provide a comprehensive overview of the current state of the art in indoor millimeter-wave device-based localization and device-free sensing technologies. They discuss the advantages and limitations of different millimeter-wave sensing technologies and their potential applications in mobile robotics. A recent review of the SLAM problem is also presented in "Millimeter-Wave Mobile Sensing and Environment Mapping: Models, Algorithms and Validation" [5], where the authors propose a comprehensive study of millimeter-wave mobile sensing and environment mapping. They present an overview of the current state of the art in millimeter-wave radar sensors, their characteristics, and potential applications in mobile robotics.

Starting from this, one interesting approach is MilliMap [31]. MilliMap is a single-chip mmWave radar-based indoor mapping system intended for use in low-visibility environments, such as emergency response scenarios. MilliMap is noteworthy for its utilization of a low-cost, off-the-shelf mmWave radar to generate a dense grid map with accuracy comparable to that of lidar, as well as providing semantic annotations of items on the map. Specifically, MilliMap makes two key technological contributions. Firstly, it addresses the sparsity and multi-path noise of mmWave signals through the integration of cross-modal supervision from a co-located lidar during training, as well as leveraging the strong geometric priors of interior areas. Secondly, it employs the spectrum

response of mmWave reflections as characteristics to robustly detect a variety of objects, such as doors and walls.

Like in the study presented before, in "Milli-RIO: Ego-Motion Estimation With Low-Cost Millimetre-Wave Radar" [1], the authors propose a novel ego-motion estimation method that uses a low-cost millimeter-wave radar sensor. The proposed method is tested in a number of simulated and real-world environments and it is showed that the algorithm could estimate the position of the robot with high accuracy and robustness.

Similarly, in [22], the authors explore the use of Deep Reinforcement Learning and propose a cross-modal contrastive learning for representation (CM-CRL) technique to address the common issues of noise and sparsity in mmWave radar signals. This technique maximizes agreement between mmWave radar data and LiDAR data during training. They evaluate their method against a baseline of two independent networks utilizing cross-modal generative reconstruction and an RL policy, as well as an Reinforcement Learning policy without cross-modal representation in a real-world robotic setting. The results demonstrate that their proposed end-to-end deep RL strategy with contrastive learning effectively navigates the robot through smoke-filled labyrinth environments and outperforms generative reconstruction approaches that create noisy artifacts or obstacles.

Another example is RadarHD [38], an end-to-end neural network that generates lidar-like point clouds from low-resolution radar inputs. This approach trains RadarHD on a large volume of raw I/Q radar data combined with lidar point clouds collected in a variety of indoor environments. The results indicate that rich point clouds can be generated even in situations not encountered during training and in the presence of significant smoke occlusion. Furthermore, the quality of RadarHD's point clouds is sufficient to be used with existing lidar odometry and mapping methods.

In [28] the authors propose a novel SLAM method that uses a millimeter-wave radar sensor and the RCS feature of the target to improve the performance of the SLAM algorithm. The proposed method is tested in a number of simulated and real-world environments and it is showed that the algorithm could estimate the position of the robot and the map of the environment with high accuracy and robustness.

The study in [46] presents an approach to indoor environment mapping based on the use of mmWave that includes Received Signal Strength (RSS), Time-Difference-of-Arrival (TDoA), and Angle-of-Arrival (AoA). The proposed method, called MOSAIC, leverages a map-based channel model to perform localization and environment inference through obstacle detection and dimensioning. The positions and dimensions of obstacles are estimated by detecting the zones of path obstructions, points of reflection, and obstacle vertices. The extended Kalman filter is then adapted to improve the estimation of these points of reflection, resulting in improved mapping accuracy. Simulation results demonstrate the effectiveness of the MOSAIC approach in achieving high localization accuracy and obstacle detection using mmWave technology. This study represents a significant contribution to the field of mmWave-based SLAM and offers promis-

ing prospects for its future development and applications in the realm of indoor environment mapping.

In [40] the authors presents a novel approach for simultaneous localization and mapping using a radar sensor based on the principle of Frequency-Modulated Continuous Wave (FM-CW) technology. In this approach, the radar sensor is used to scan the environment and match the radar images to build a map of the environment, without the use of odometric sensors. This approach allows the radar to be placed on any type of vehicle and can provide global and local coverage of natural environments. The article describes the development of a small-sized microwave radar sensor called K2Pi and its potential for environmental management through remote sensing techniques. The authors present an application of the sensor in simultaneous localization and mapping using scan matching of radar images, which they call the R SLAM algorithm. The article also highlights the potential of the proposed approach to overcome the limitations of traditional SLAM algorithms that rely on odometric sensors, which can be affected by visibility conditions and other environmental factors. The authors conduct experiments to evaluate the performance of the proposed approach and present results that demonstrate the ability of the R SLAM algorithm to reliably build a radar map of the environment. They also discuss the potential of the proposed approach to be used in various applications, such as environmental management, autonomous navigation, and search and rescue operations.

The authors in [21] RadarSLAM present a robust simultaneous localization and mapping system for all weather conditions. This paper studies the use of a Frequency Modulated Continuous Wave radar for SLAM in large-scale outdoor environments. The authors propose a full radar SLAM system, including a novel radar motion estimation algorithm that uses radar geometry for reliable feature tracking and estimates pose by joint optimization. The system's loop closure component captures and exploits the structural information of the surrounding environment. The experiments on three public radar datasets show competitive accuracy and reliability compared to LiDAR, vision and radar methods. The results demonstrate the potential of using radar for all-weather localization and mapping in extreme weather conditions such as heavy snow and dense fog.

In [36], it is presented by the authors a unique sensor system for 3D motion estimation in harsh environments where traditional camera and LiDAR systems may not perform well. The system uses two orthogonal radar sensors to secure returns from static objects on the ground and is encased in a plastic box for deployment in challenging environments. The authors introduce a new velocity-based ego-motion estimation method that outperforms existing point matching methods and integrate it into a SLAM framework for estimating 3D ego-motion. The proposed method was tested and found to perform reliably in both indoor and outdoor environments with various visibility and structural conditions.

Occupancy grid mapping is a widely used technique for perception of obstacles, where the environment is represented by evenly spaced cells and the probability of each cell being occupied is updated based on range sensor measurements. Classic approaches in occupancy grid mapping use ray casting to update cells

as occupied when an obstacle is encountered. However, these methods are limited in their ability to identify planar obstacles such as slippery, sandy or rocky surfaces. The paper in [10] presents a solution to this problem through the use of stereo cameras and a Convolution Neural Network-based pixel labeling technique. The presence of rocky obstacles in planetary environments is identified, and the obstacles are then converted into a scan-like model. Relative pose estimation between successive frames is achieved using the ORB-SLAM algorithm, and the occupancy grid map is updated using the Bayes' Update Rule. The proposed method was evaluated using the Martian analogous dataset and the ESA Katwijk Beach Planetary Rover Dataset, with the generated occupancy map being compared to a manually segmented orthomosaic map obtained through drone surveys. The results show the potential for the proposed method to effectively identify planar obstacles in a planetary environment.

The authors in [29] propose a novel method for estimating the ego-motion of a radar-equipped robot using only radar sensor data. They utilize the distribution of detected points on a two-dimensional plane to estimate the rotation angle of the robot. By correlating the distributions of the detected points over time, they are able to determine the rotation angle with an error of 3°. Additionally, they estimate the velocity of the robot by analyzing the trend line formed by the detected points on the 2D plane. The accuracy of the proposed method is demonstrated through comparisons with the ego-motion information received from the robot's motor, which show that the estimated yaw rate and velocity are within the error bounds of 3° and 0.073 m/s, respectively. This paper presents an innovative solution to a challenge in the field of SLAM with radar-based systems and highlights the importance of considering ego-motion information in this context. The results of this research have potential implications for improving the accuracy and reliability of SLAM algorithms in indoor and outdoor environments.

The paper in [8] presents a reliable and accurate radar-only motion estimation algorithm for mobile autonomous systems. The algorithm is based on the use of a FMCW radar, which is capable of extracting landmarks by accounting for unwanted effects in radar returns. The relative motion between subsequent scans is estimated through scan matching, where point correspondences are greedily added based on unary descriptors and pairwise compatibility scores. The results of the proposed radar odometry are robust in various conditions, including scenarios in which visual odometry and GPS/INS fail. This work advances the state of the art in radar-based motion estimation and highlights the potential of using radar technology as a primary sensor for autonomous systems. By providing a reliable and accurate solution to motion estimation, this research opens up new avenues for future research and applications in the field of autonomous systems.

On another hand, we think that it's also useful to analyze different approaches to the SLAM problem with different sensors other then lidars or mmWave radars.

As an example "Indoor mapping using kinect and ROS" [34], the authors propose a method for indoor mapping using a Kinect sensor and ROS. The proposed method uses the point cloud data from the Kinect sensor to construct

a 3D model of the environment and is tested in a number of indoor environments.

Another interesting approach that uses Sonar sensor data is shown in [41] that implements a sensor fusion layer to cope with reduced visibility in SLAM.

In [33], the authors propose a novel method for generating laser-quality 2D navigation maps from RGB-D sensors. The authors implemented a technique to transform RGB-D data to laserscan data using ROS. The proposed method is tested in a number of simulated and real-world environments and it is showed that the algorithm could generate high-quality 2D maps of the environment with high accuracy and robustness.

Similarly the authors in [48] propose a novel 3D reconstruction method that uses an RGB-D sensor and a grid map to construct a 3D model of the environment. The proposed method is tested on a mobile robot and demonstrates its effectiveness in a number of indoor environments.

Another interesting approach is sensor fusion. In "MilliEgo: Single-Chip MmWave Radar Aided Egomotion Estimation via Deep Sensor Fusion" [30], the authors propose a novel egomotion estimation method that uses a single-chip mmWave radar sensor and a deep sensor fusion algorithm. The proposed method is tested in a number of simulated and real-world environments and it is showed that the algorithm could estimate the position of the robot with high accuracy and robustness.

In the end, also approaches to enanche the performance and results of SLAM implementations have been explored. In "Improved Map Generation by Addition of Gaussian Noise for Indoor SLAM using ROS" [47], Yuen et al. propose a novel map generation method that uses the addition of Gaussian noise to improve the performance of Indoor SLAM using ROS. The proposed method is tested in a number of simulated and real-world environments and it is showed that the algorithm could estimate the position of the robot and the map of the environment with higher precision and robustness than traditional methods without added noise.

2.2 Conclusion on the review

In conclusion, this chapter on literature review has provided a comprehensive overview of the current state of the art in SLAM using mmWave radars. We have discussed the various approaches that have been proposed in the literature, including the use of different filters for the SLAM; the use of different tuning and data augmentation on the sensors and various machine learning techniques. We have also highlighted the challenges that arise when using mmWave radars for SLAM, such as the limited field of view, high noise levels, and the need for robust data association.

In this thesis, we will propose a new method for SLAM that combines elements of the above approaches and is specifically designed to address these challenges. The proposed method will make use of multiple mmWave radars to provide a more complete and robust representation of the environment. We

will evaluate the performance of the proposed method using simulations and experiments, and compare it to other state-of-the-art methods.

We believe that our proposed method has the potential to significantly improve the performance of SLAM using mmWave radars, and we are excited to contribute to the growing body of research in this field.

Chapter 3

Theoretical basis and problem formulation

3.1 SLAM

Building a map of an unfamiliar environment while also figuring out where the robot is in that environment is known as simultaneous localization and mapping (SLAM), and it is a basic issue in robotics. SLAM is essential for autonomous navigation and has numerous uses in industries such as robotics, computer vision, and self-driving cars.

Localization and mapping are the two main SLAM operations. Finding the robot's posture or its location and orientation on the map is a necessary step in the localization process. On the contrary, mapping involves creating a representation of the environment.

There are several techniques for solving the SLAM problem, including:

- **Odometry-based SLAM** : This approach uses odometry information, such as wheel encoders or inertial measurement units (IMUs), to estimate the robot's motion. The odometry estimates are then used to update the robot's pose within the map.

- **Feature-based SLAM**: This approach involves extracting features from sensor data, such as landmarks or corners, and using them to estimate the robot's pose and build the map.

- **Grid-based SLAM**: This approach involves representing the environment as a grid of cells, each with a probability of occupancy. Sensor data are used to update the probabilities of the cells, and the robot's pose is estimated based on the occupancy probabilities.

- **Graph-based SLAM**: This approach involves representing the environment as a graph, with the robot's poses and landmarks as nodes and the constraints between them as edges. Sensor data is used to update the graph, and the robot's pose is estimated by optimizing the graph.

Dealing with the ambiguity in the sensor data and the robot's mobility is one of the main difficulties in SLAM. The robot's pose and the map can be estimated using a variety of filtering techniques, including Kalman filters and Particle filters.

The Kalman filter is a recursive filter that gauges the state of a linear system. To determine the state of the system, a mathematical model of the system and sensor measurements are used.

A particle filter is a non-linear filter that makes an estimation of the state of a non-linear system. Sensor measurements are utilized to update a set of particles that the filter uses to represent the state of the system.

By offering range data for localization and mapping, MmWave (millimeter wave) sensors can be used to address the SLAM issue. MmWave sensors operate by emitted a beam of millimeter waves and detecting the waves' transmission or reflection after they interact with a scene or object. The measured waves can be used to deduce details about the object or scene, such as its size, shape, and material composition. mmWave sensors can pass through things that are opaque to other kinds of electromagnetic radiation, such as clothing and fog. MmWave sensors can therefore be used for a range of imaging and sensing tasks, such as security screening, business inspection, and imaging in the medical field.

So, SLAM is a critical robotics challenge that involves creating a map of an unknown area while also figuring out where the robot is in that environment. The SLAM problem can be solved using a variety of methods, such as odometry-based, feature-based, grid-based, and graph-based SLAM. To solve the SLAM challenge, millimeter wave sensors can be employed to provide range information for localization and mapping.

3.1.1 Odometric Localization

Odometric localization is a fundamental component of the SLAM problem, as it provides an estimate of the robot's pose within the map. Odometry information, such as wheel encoders or inertial measurement units (IMUs), is used to estimate the robot's motion and update its pose within the map.

There are several techniques for integrating odometry information to estimate the robot's pose, including:

Euler approximation: This approach uses the Euler method to approximate the integration of the robot's motion. It involves dividing the motion into small time steps and approximating the motion within each time step. The robot's pose at each time step is estimated using the following formulas:

$$x_{i+1} = x_i + v_i T_s cos\theta_i \tag{3.1}$$

$$y_{i+1} = y_i + v_i T_s sin\theta_i \tag{3.2}$$

$$\theta_{i+1} = \theta_i + \omega_i T_s \tag{3.3}$$

Where x, y, and θ are the position and orientation of the robot, v and ω are the linear and angular velocities of the robot, and T_s is the time step.

Runge-Kutta: This approach uses the Runge-Kutta method to integrate the robot's motion. It involves dividing the motion into small time steps and approximating the motion within each time step using a set of intermediate calculations. The robot's pose at each time step is estimated using the following formulas:

$$x_{i+1} = x_i + v_i T_s cos(\theta_i + \frac{\omega_i T_s}{2}) \tag{3.4}$$

$$y_{i+1} = y_i + v_i T_s sin(\theta_i + \frac{\omega_i T_s}{2}) \tag{3.5}$$

$$\theta_{i+1} = \theta_i + \omega_i T_s \tag{3.6}$$

Where variables are defined as before.

Exact integration: This approach uses the exact solution of the robot's motion to integrate the odometry information. It involves solving the differential equations that describe the robot's motion. The robot's pose at each time step is estimated using the following formulas:

$$x_{i+1} = x_i + v_i \int_{t_i}^{t_{i+1}} x(t)dt = x_i + v_i T_s(sin\theta_{i+1} - sin\theta_i) \tag{3.7}$$

$$y_{i+1} = y_i + v_i \int_{t_i}^{t_{i+1}} y(t)dt = y_i - v_i T_s(\cos\theta_{i+1} - \cos\theta_i) \tag{3.8}$$

$$\theta_{i+1} = \theta_i + \int_{t_i}^{t_{i+1}} \omega_i(t)dt = \theta_i + \omega_i T_s \tag{3.9}$$

Where x, y, and θ are the position and orientation of the robot, v and ω are the linear and angular velocities of the robot, and T_s is the time step.

Note that the first two are still defined for $\omega_i = 0$; in this case, they coincide with the corresponding formulae of the Euler and Runge-Kutta methods (which are exact over line segments). However, in the implementation, it is necessary to handle this situation with a conditional instruction.

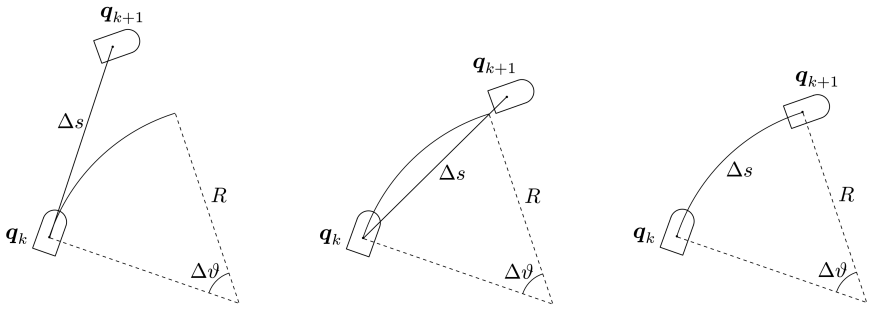

Figure 3.1: Odometric localization for a unicycle moving along a circle arc; left: Euler method, centre: Runge–Kutta method, right: exact integration. [11]

It is worth noting that the Euler approximation and Runge-Kutta methods are approximate methods, while the exact integration method gives the exact solution of the motion, but it could be computationally expensive.

Therefore, odometric localization is a fundamental component of the SLAM problem as it provides an estimate of the robot's pose within the map. As shown, there are several techniques for integrating odometry information, including the Euler approximation, Runge-Kutta, and exact integration. Each method has its own advantages and disadvantages, and the choice of method depends on the specific requirements of the application. The presented formulas allow the estimation of the odometry(position and angle) of the robot using the linear and angular velocities of the robot, and their integration over time. This technique is also known as dead reckoning and it is generally used in robotics to estimate the position and orientation of a robot based on the velocity commands reconstructed via proprioceptive sensors such as wheel encoders. As shown, this method involves iteratively using mathematical formulas to estimate the robot's motion and update its pose within the map, starting from an initial estimate of the configuration.

However, it has the limitation that the accuracy of the estimate cannot be better than the initial estimate, and the error tends to grow over time, known as drift. This error can be caused by various factors such as wheel slippage, inaccuracies in the calibration of the kinematic parameters, and numerical errors introduced by the integration method. Moreover, the performance of odometric localization also depends on the specific kinematic arrangement of the robot, for example, differential drive is usually more accurate than synchro drive.

A more robust solution for localization is to use exteroceptive sensors such as laser range finders, which can correct the estimate provided by dead reckoning by comparing the expected measures of the sensors with the actual readings. These techniques, which make use of tools from Bayesian estimation theory, such as the *Extended Kalman Filter* or the *Particle Filter*, provide greater accuracy than pure odometric localization and are essential for navigation tasks over long paths.

3.1.2 Probabilistic localization and Bayes Filtering

Probabilistic localization is a fundamental aspect of the SLAM problem, as it involves estimating the robot's pose within the map, using sensor measurements and odometry information. The use of probability theory and statistical methods allows to represent and reason about the uncertainty of the robot's perception and control. This is essential, as the robot's sensors and motion models are subject to noise and uncertainty. The key concept is that instead of considering a single hypothesis on the configuration, consider a probability distribution over the space of all possible hypotheses. A possible approach can be Bayesian filtering.

Bayesian filtering is a key technique used in probabilistic localization, as it provides a framework for fusing sensor measurements and odometry information to estimate the robot's pose. Bayesian filtering is based on Bayes' theorem, which states that the probability of a hypothesis given some observations is proportional to the probability of the observations given the hypothesis, multiplied by the prior

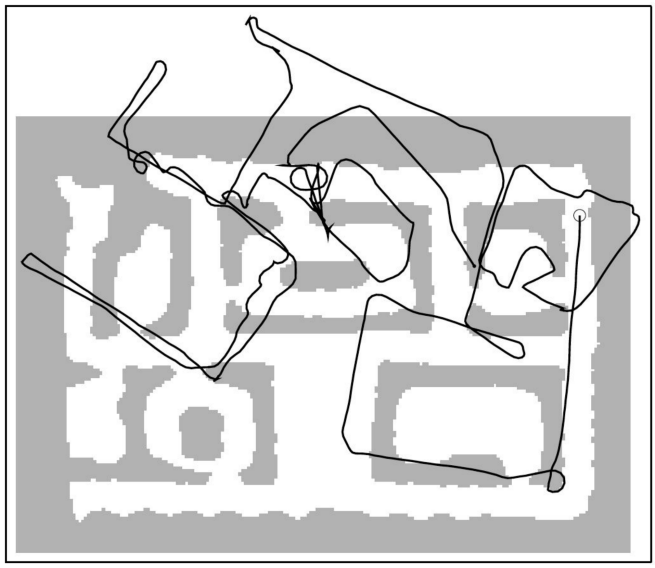

Figure 3.2: Example of drift in odometric measurements. [11]

probability of the hypothesis.

In the context of SLAM and odometric localization, Bayesian filtering is used to estimate the robot's pose, given a sequence of sensor measurements and odometry information. The Bayesian filtering approach can be applied through various methods such as the Extended Kalman filter (EKF) and the Particle filter.

3.1.2.1 Kalman Filter

Kalman filtering is a widely used estimation method in SLAM (Simultaneous Localization and Mapping) due to its ease of implementation for linear systems. The equations for Kalman filtering can be applied directly with minimal understanding of the underlying theory, making it accessible to a wide range of users. However, it is important to note that the underlying theory is still crucial, as modern applications of Kalman filtering often require modifications to address nonlinear sensor models and non-Gaussian noise models in robot localization and mapping problems. Additionally, modifications are often made to reduce computational complexity.

We can start our analysis considering a linear system with no noise:

$$x_{k+1} = A_k x_k + B_k u_k \tag{3.10}$$

$$y_k = C_k x_k \tag{3.11}$$

We need to build an observer[17] for this system with two steps. The first being the prediction: we need to generate a prediction $\hat{x}_{k+1|k}$ by propagating the previous estimate $\hat{x}_k$ according to the dynamics of the system. The second step is to do a correction correction of the previous prediction based on the measured and the predicted output.

To make the prediction we can give the state estimate as a state of our system model and we find:

$$\hat{x}_{k+1|k} = A_k\hat{x}_k + B_k u_k \tag{3.12}$$

Now, we need to wait for the measurement of the output of the system and calculate the correction. From the measured output y_{k+1} we can say that a possible state of the system x must respect the condition: $y_{k+1} = C_{k+1}x$. In other words, the state x_{k+1} has to lie on the hyperplane:

$$\Omega = \{x \in \mathbb{R}^n : C_{k+1}x = y_{k+1}\} \tag{3.13}$$

So our correction $\Delta x = \hat{x}_{k+1|k+1} - \hat{x}_{k+1|k}$ must respect the following:

$$y_{k+1} = C_{k+1}(\hat{x}_{k+1|k} + \Delta x) \tag{3.14}$$

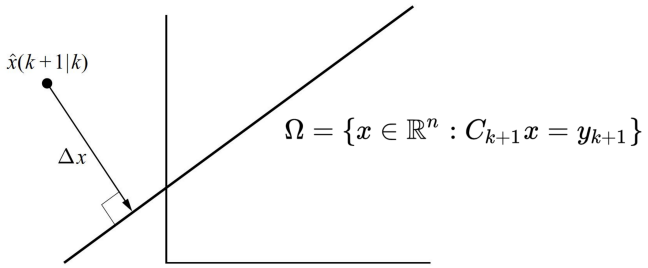

Figure 3.3: The set Ω corresponds to the states consistent with the current output y_{k+1}. The state after the correction must lie in this set , and, to be more precise, it's the closest state to the predicted estimate. [11]

As Figure 3.3 explains graphically the next estimate $\hat{x}_{k+1|k+1}$ has to be chosen to be a point on the Ω hyperplane that has the shorter distance to the previous prediction $\hat{x}_{k+1|k}$

Given C_{k+1}, which we assume to be full-row rank, its pseudoinverse is

$$C^+_{k+1} = C^T_{k+1}(C_{k+1}C^T_{k+1})^{-1} \tag{3.15}$$

And it can be shown that the shortest distance Δx is

$$\Delta x = C^+_{k+1}(y_{k+1} - C_{k+1}\hat{x}_{k+1|k}) = C^+_{k+1}\nu_{k+1} \tag{3.16}$$

where ν_{k+1} is called innovation.

So, the two-step observer that we built is:

$$\hat{x}_{k+1|k} = A_k\hat{x}_k + B_k u_k \tag{3.17}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + C^+_{k+1}\nu_{k+1} \tag{3.18}$$

The formulation given poses a problem: the state estimation x_{k+1} is not guaranteed to converge because the update we derived is naive. In fact estimate

errors in the direction parallel to are never corrected. The problem can be fixed by taking into account the presence of the noise in the process and by using a Kalman Filter with a Probability Distribution Function.

The estimate of the Kalman Filter (KF) is a multivariate Gaussian Probability Distribution over the state space. Let us consider

$$x_{k+1} = A_k x_k + B_k u_k + v_k \tag{3.19}$$

$$y_k = C_k x_k \tag{3.20}$$

where $v_k \in \mathbb{R}^n$ is a white zero-mean Gaussian noise and covariance matrix V_k.

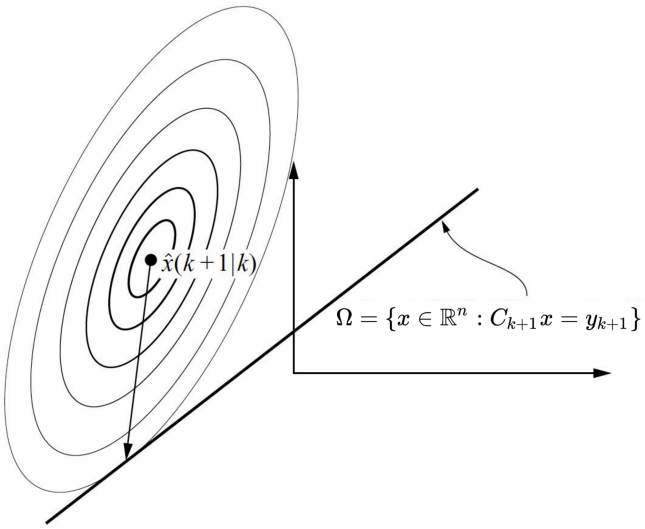

Figure 3.4: The correction determines the "closest" and "most likely" state on the set of states Ω. [11]

Since the process is random, in this case we need to find an estimate of the state $\hat{x}_{k|k}$ and the covariance matrix $P_{k|k}$ of the process. Since, v_k is zero mean the state prediction has no variation from the process without noise:

$$\hat{x}_{k+1|k} = A_k \hat{x}_k + B_k u_k \tag{3.21}$$

On the other hand, the covariance matrix is:

$$\begin{aligned} P_{k+1|k} =& E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T] \\ =& E[A_k(x_{k+1} - \hat{x}_k)(x_{k+1} - \hat{x}_k)A_k^T] + \\ & E[A_k(x_k - \hat{x}_k)v_k^T] + E[v_k v_k^T]] \end{aligned} \tag{3.22}$$

where $E[X]$ is the expectation. Since $E[X]$ is linear and v_k^T is a process with zero mean independent of both x_k and $\hat{x}_k$, we find that $E[A_k(x_k - \hat{x}_k)v_k^T] = 0$, so, knowing that the covariance $P_k = E[(x_{k+1} - \hat{x}_k)(x_{k+1} - \hat{x}_k)]$ we find:

$$\begin{aligned} P_{k+1|k} =& A_k E[(x_{k+1} - \hat{x}_k)(x_{k+1} - \hat{x}_k)]A_k^T + E[v_k v_k^T]] \\ =& A_k P_k A_k^T + V_k \end{aligned} \tag{3.23}$$

The Gaussian Distribution Probability Function, defined by $\hat{x}_{k|k}$ and $P_{k|k}$ is:

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |P_{k+1|k}|}} e^{-\frac{1}{2}(x-\hat{x}_{k+1|k})^T P_{k+1|k}^{-1}(x-\hat{x}_{k+1|k})} \tag{3.24}$$

We need to look for $x \in \Omega$ that maximizes the Gaussian Distribution in Equation 3.24. In other words, we need to find the most likely state of the system. We can derive that $p(x)$ is maximized when $(x-\hat{x}_{k+1|k})^T P_{k+1|k}^{-1}(x-\hat{x}_{k+1|k})$ is minimized. Now, we need to define the Mahalanobis [32] distance:

$$||x||_M^2 = x_T(P_{k+1|k}^{-1})x \tag{3.25}$$

Using the Mahalanobis [32] distance we now need to minimize the distance: $||\Delta x||_M$ with $(\hat{x}_{k+1|k} + \Delta x) \in \Omega$ As can be seen in Figure 3.4 the set of points with the same distance between $\hat{x}_{k+1|k}$ and Ω according to the norm $||\cdot||_M$ are ellipses. Now we need to choose $\hat{x}_{k+1}$ so that it is the closest point to $\hat{x}_{k+1|k}$ in Ω according to the ellipses we found from the Mahalanobis distance and the covariance estimate. In other words, we have to choose the point at which one of the equidistant ellipses intersects tangentially Ω. As before, to find this point we need to define the weighted pseudoinverse of C_{k+1}, that we define as:

$$C_{k+1,M}^{+} = Pk+1|kC_{k+1}^T(C_{k+1}P_{k+1|k}C_{k+1}^T)^{-1} \tag{3.26}$$

It can be shown that our corrections Δx are:

$$\Delta x = C_{k+1,M}^{+}(y_{k+1} - C_{k+1}\hat{x}_{k+1|k}) = C_{k+1,M}^{+}\nu_{k+1} \tag{3.27}$$

Where $\nu_{k+1}y_{k+1} - C_{k+1}\hat{x}_{k+1|k}$ is the innovation. We now also need a correction for the covariance. Using the definition of covariance matrix and the previously found state correction, we obtain:

$$P_{k+1} = P_{k+1|k} - C_{k+1,M}^{+}C_{k+1}P_{k+1|k} \tag{3.28}$$

In the end we find, for the prediction step:

$$\hat{x}_{k+1|k} = A_k\hat{x}_k + B_k u_k \tag{3.29}$$

$$P_{k+1|k} = A_k P_k A_k^T + V_k \tag{3.30}$$

And for the correction step:

$$\hat{x}_{k+1} = \hat{x}_{k+1|k}C_{k+1,M}^{+}\nu_{k+1} \tag{3.31}$$

$$P_{k+1} = P_{k+1|k} - C_{k+1,M}^{+}C_{k+1}P_{k+1|k} \tag{3.32}$$

To conclude the analysis on the Linear Kalman Filter we can briefly analyze the case with measurement noise in the system. The system equations considering noise in the sensor and in the system will be as follows:

$$x_{k+1} = A_k x_k + B_k u_k + v_k \tag{3.33}$$

$$y_k = C_k x_k + \omega_k \tag{3.34}$$

where ω_k is a white Gaussian noise of zero mean whose covariance matrix is W_k. Since the dynamic equation has not changed, the prediction step of the Kalman Filter is the same as the one previously shown.

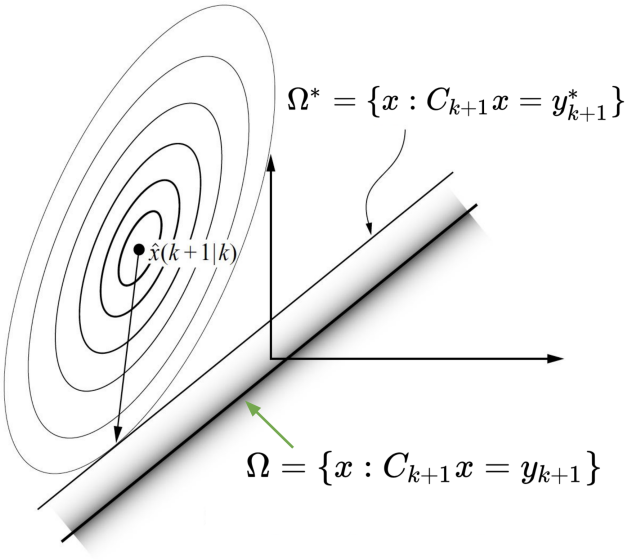

Figure 3.5: Hyperplane case with sensor noise [11]

In this case, we do not know exactly what the output is: we only know that it is drawn from a Gaussian distribution with mean value y_{k+1} and covariance matrix W_{k+1}. So, we now need to look for the most likely result y^*_{k+1} given the prediction $(\hat{x}_{k+1|k}, P_{k+1|k})$ and the measured result y_{k+1}. Therefore we define:

$$\Omega^* = \{x \in \mathbb{R}^n : C_{k+1}x = y^*_{k+1}\} \tag{3.35}$$

The problem is to find $\Delta x = \hat{x}_{k+1} - \hat{x}_{k+1|k}$ that minimizes $||\Delta x||_M$ so that, as before, $\hat{x}_{k+1} \in \Omega^*$. The solution that can be found is for the prediction step:

$$\hat{x}_{k+1|k} = A_k\hat{x}_k + B_k u_k \tag{3.36}$$

$$P_{k+1|k} = A_k P_k A_k^T + V_k \tag{3.37}$$

And for the correction step:

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} R^+_{k+1,M}\nu_{k+1} \tag{3.38}$$

$$P_{k+1} = P_{k+1|k} - R^+_{k+1,M} C_{k+1} P_{k+1|k} \tag{3.39}$$

where R_{k+1} is the Kalman gain matrix:

$$R_{k+1} = P_{k+1|k} C^T_{K+1}(C_{k+1}P_{k+1}C^T_{K+1} + W_{k+1})^{-1} \tag{3.40}$$

The Kalman gain matrix weights the trade-off between the accuracy of the prediction and the measurement noise. If is large, the sensor readings are more reliable than the prediction and the KF weights the sensor readings highly when computing the correction. If is small, the sensor readings are less reliable than the prediction, so sensor readings do not have much influence in the correction. A few key concepts of the Kalman filter are as follows.

- The Kalman Filter gives the best estimate, or, in statistical terms, at each instant k the Kalman Filter minimizes $E[x_{k+1} - \hat{x}_{k+1}]$.
- The Kalman Filter gives correct estimates for the state and covariance; in other words, given a Gaussian estimate of the state and covariance at instant k described by $(\hat{x}_k, P_k)$, then the Kalman filter gives the correct distribution at time $k+1$ (posterior Gaussian), described by $(\hat{x}_{k+1}, P_{k+1})$
- If the noise terms have non-Gaussian distributions, then the KF is still the best linear estimator, but there may be better nonlinear estimators.

3.1.2.2 Extended Kalman Filter

.

The Extended Kalman filter (EKF) is a popular method for estimating the state of a nonlinear system using a sequence of measurements. It is an extension of the Kalman filter which is designed for linear systems. The EKF algorithm uses a linearization of the nonlinear measurement and motion models to estimate the system's state.

The EKF algorithm, such as the Linear KF consists of two main steps: prediction and update. In the prediction step, the EKF algorithm uses the current state estimate and the control inputs to predict the next state of the system. This step is based on the motion model of the system. In the update step, the EKF algorithm uses the current measurement to correct the predicted state. This step is based on the measurement model of the system. Considering a nonlinear system with noise:

$$x_{k+1} = f_k(x_k, u_k) + v_k \tag{3.41}$$

$$y_k = h_k(x_k) + \omega_k \tag{3.42}$$

a possible Extended Kalman Filter can be obtained linearizing the equations around the current estimate and then apply the KF equations to the resulting linear approximation.

The EKF algorithm recursively updates the estimate of the system state on the basis of current measurements and control inputs. The EKF algorithm provides an estimate of the mean and covariance of the system's state. These estimates are used to represent the uncertainty of the state estimate.

The EKF algorithm has several advantages, such as its simplicity and computational efficiency. However, it also has some limitations, such as the requirement of a good initial estimate and the assumption of linearity of the system's motion and measurement models.

In summary, The Extended Kalman filter (EKF) is a method to estimate the state of a nonlinear system using a sequence of measurements, it linearizes the nonlinear measurement and motion models to estimate the system's state and recursively updates the estimate based on the current measurements and control inputs. The EKF algorithm is simple and computationally efficient but it has some limitations such as the requirement of a good initial estimate and the assumption of linearity of the system's motion and measurement models.

3.1.2.3 Other Techniques

The Particle filter (PF) is a method for estimating the state of a nonlinear system using a sequence of measurements. It is a generalization of the Kalman filter, which is designed for linear systems, and does not rely on linearization of the system's models. The PF algorithm represents the probability distribution over the system's state as a set of samples, called particles, and updates the particles based on the current measurements.

In the context of SLAM, the Particle filter can be used to estimate the robot's pose within the map, using sensor measurements and odometry information. The algorithm represents the probability distribution over the robot's pose as a set of particles and updates the particles based on the sensor measurements and odometry information. An example of the Particle filter algorithm for SLAM is as follows:

Algorithm 1 Particle Filter Algorithm

1: Initialize the particles with a random distribution over the robot's pose.
2: **for all** k timestamps **do**
3: Predict the next pose of each particle based on the odometry data.
4: Resample the particles based on their weights to obtain a new set of particles.
5: Compute the weight of each particle based on the sensor measurements and the likelihood function.
6: Update the estimate of the robot's pose as the mean of the particles.
7: **end for**

The Particle filter algorithm has several advantages, such as its ability to handle nonlinear models and its ability to represent the uncertainty of the state estimate through a set of particles. However, it also has some limitations, such as the high computational cost and the requirement of a large number of particles to obtain a good estimate.

In summary, The Particle filter (PF) is a method for estimating the state of a nonlinear system using a sequence of measurements, it represents the probability distribution over the system's state as a set of samples, called particles, and updates the particles based on the current measurements, it does not rely on linearization of the system's models. The Particle filter algorithm for SLAM is an example of how it can be used to estimate the robot's pose within the map, using sensor measurements and odometry information. The algorithm has several advantages, such as its ability to handle non-linear models and its ability to represent the uncertainty of the state estimate through a set of particles. However, it also has some limitations, such as the high computational cost and the requirement of a large number of particles to obtain a good estimate.

The Unscented Kalman Filter (UKF) is another popular method for estimating the state of non-linear systems using a sequence of measurements. It is an extension of the Kalman filter and the EKF, and it does not rely on linearization of the system's models. The UKF algorithm uses a set of carefully

chosen points, called sigma points, to approximate the non-linear measurement and motion models.

In the context of SLAM, the UKF can be used to estimate the robot's pose within the map, using sensor measurements and odometry information. The algorithm uses sigma points to approximate the non-linear measurement and motion models and updates the estimate of the robot's pose based on the sensor measurements and odometry information.

An example of the UKF algorithm for SLAM is as follows:

Algorithm 2 UKF Algorithm

1: Initialize the estimate of the robot's pose and the covariance matrix.
2: **for all** k timestamps **do**
3: Generate sigma points based on the current estimate of the robot's pose and the covariance matrix.
4: Predict the next pose of each sigma point based on the odometry information.
5: Compute the mean and covariance of the predicted sigma points.
6: Compute the weight of each sigma point based on the sensor measurements and the likelihood function.
7: Update the estimate of the robot's pose and the covariance matrix based on the weighted sigma points.
8: **end for**

The UKF algorithm has several advantages, such as its ability to handle non-linear models, its computational efficiency compared to the particle filter, and its ability to represent the uncertainty of the state estimate through a set of sigma points. However, it also has some limitations, such as the requirement of a good initial estimate and the selection of the sigma points.

Another technique is the Graph-Based SLAM, it consists of constructing a graph where the nodes represent the robot's poses and the edges represent the constraints between the poses. The graph is updated as the robot moves and obtains new sensor measurements. As the Fig. 3.6 shows, an edge between two nodes can be created when the agent moves between two consecutive poses, for example from x_i to x_{i+1}, corresponding to odometry measurement or when the agent observes the same part of the environment from x_i and x_j. The goal of the algorithm is to find the best estimate of the robot's poses and the map by solving an optimization problem over the graph. The main advantage of this approach is that it can handle loops and non-linear constraints.

In summary, The Unscented Kalman Filter (UKF) is another popular method for estimating the state of nonlinear systems using a sequence of measurements; it does not rely on linearization of the system's models and uses a set of carefully chosen points, called sigma points, to approximate the nonlinear measurement and motion models. In the context of SLAM, the UKF can be used to estimate the robot's pose within the map, using sensor measurements and odometry information. Another technique is the Graph-Based SLAM, it consists of con-

structing a graph where the nodes represent the robot's poses and the edges represent the constraints between the poses, the graph is updated as the robot moves and obtains new sensor measurements. The goal of the algorithm is to find the best estimate of the robot's poses and the map by solving an optimization problem over the graph. The main advantage of this approach is its ability to handle loops and non-linear constraints, making it more robust and adaptable to different environments.

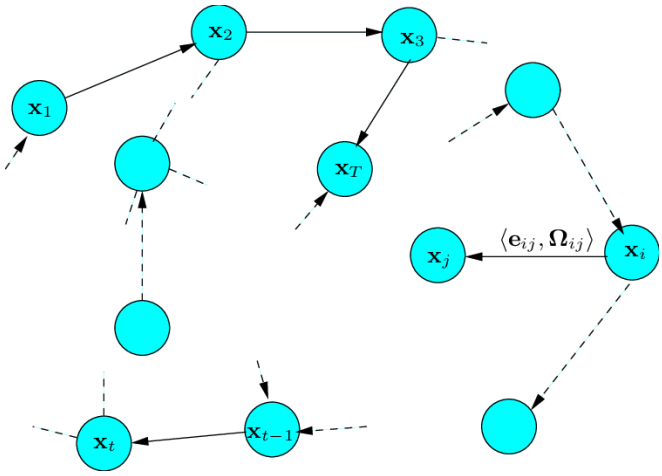

Figure 3.6: A pose graph representation of the SLAM problem. [19]

However, it also has some limitations, such as high computational cost, complexity of the optimization problem, and requirement of a good initialization. Overall, there are several techniques available for localization and mapping in the context of SLAM, including the Kalman filter, Extended Kalman filter, Unscented Kalman filter, Particle filter and Graph-Based SLAM. Each technique has its own advantages and limitations, and the choice of technique depends on the specific requirements of the application and the characteristics of the system.

3.1.3 SLAM implementations

Various SLAM implementations exists, each with their own strengths and weaknesses. In this chapter, we will focus on SLAM implementations that are compatible with the Robot Operating System (ROS) ecosystem.

One of the most popular SLAM implementations for ROS is the Hector SLAM [25] algorithm. This algorithm is a hybrid approach that combines the strengths of both grid-based and graph-based SLAM methods. It uses a 2D occupancy grid map to represent the environment and a scan matching algorithm to determine the robot's location within the map. This approach consumes low computational resources and can be used on low-weight, low-power, and low-cost processors, making it suitable for small-scale autonomous systems.

It is designed to enable sufficiently accurate environment perception and self-localization while keeping computational requirements low. The system has been successfully used on unmanned ground vehicles, unmanned surface vehicles, and

a small indoor navigation system. The system is particularly useful in scenarios where large loops do not have to be closed and where leveraging the high update rate of modern LIDAR systems is beneficial, such as the RoboCup Rescue competition and indoor navigation of agile aerial vehicles.

Another popular SLAM implementation for ROS is Cartographer. Cartographer is a 2D and 3D SLAM library that is developed and maintained by Google. It uses a combination of scan matching and graph optimization to determine the robot's location within the map. It also supports submap-based loop closure, which allows the algorithm to detect when the robot has returned to a previously visited location. Cartographer is also designed to work with a wide variety of sensors, including LIDAR, IMU, and cameras. [2]

Finally, there is the Karto SLAM library. This algorithm is based on a scan matching method and a graph-based data structure to represent the map. The library also includes a feature-based loop closure detection algorithm, which allows the robot to detect when it has returned to a previously visited location. The library is designed to be computationally efficient and can run on a variety of platforms, including small robots and drones. [3]

3.1.3.1 OpenSLAM GMapping

The ROS GMapping node is a popular open-source implementation of the SLAM problem [35], specifically using a technique called grid-based SLAM. It is a part of the ROS (Robot Operating System) navigation stack and can be used to build 2D occupancy grid maps of an unknown environment using laser range finder data. The `gmapping` node is implemented in C++ and can be used with a wide range of robots and platforms.

Gmapping uses a variant of the Rao-Blackwellized particle filter to estimate the pose of the robot and the occupancy of the cells in the grid map. The laser range finder data is used to update the particles and the occupancy of the cells. The node uses a likelihood field model to calculate the likelihood that each cell will be occupied on the basis of sensor measurements. The `gmapping` node also uses a scan matcher algorithm to align the laser scans with the map.

The gmapping node is designed to be robust against sensor noise and to handle large amounts of data. It can be used with a wide range of laser range finders and can also handle data from other sensors such as odometry and IMU.

The `gmapping` node can be launched using a launch file provided by the ROS navigation stack, which sets the parameters for the node such as the laser topic, the `odom` topic, and the *frame_id*. Once the `gmapping` node is running, it can be controlled through ROS services, such as starting and stopping the mapping process, and through ROS topics, such as publishing the generated map.

The `gmapping` node provides several features that make it a powerful tool for building maps in robotics. It includes a feature called Dynamic Window Approach (DWA) for motion planning, which is used to avoid obstacles in the environment. It also includes an Adaptive Monte Carlo localization (AMCL) algorithm that estimates the robot's pose based on the map and sensor data.

The `gmapping` node also provides a feature called loop closure detection, which is used to detect when the robot revisits a previously visited location. This allows the node to correct the map and improve its accuracy over time.

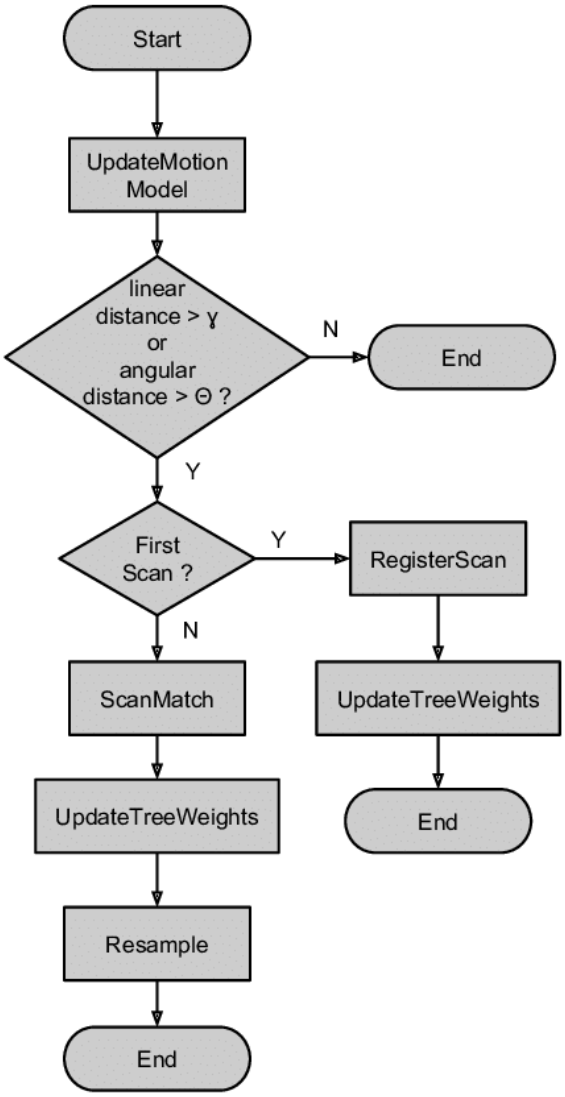

Figure 3.7: Flowchart of the GMapping algorithm, which is executed every time a new scan is received. [18]

The node is designed to be robust against sensor noise and can handle data from a wide range of sensors. The `gmapping` node can be easily integrated into a robotics platform and provides useful features such as DWA for motion planning, AMCL for localization, and loop closure detection to improve map accuracy over time.

3.2 Motion planning

Motion planning is a crucial aspect of autonomous navigation, as it involves determining a safe and efficient path for a robot to move from its current location to a desired goal location while avoiding obstacles.

Obstacle avoidance involves generating a path that keeps the robot a safe distance from any obstacles in its environment. This can be achieved through various techniques, each with its own strengths and weaknesses. In this chapter, we will discuss several popular methods for motion planning, including Voronoi diagrams, cell decomposition, probabilistic planning, and artificial potential fields, before concluding with a discussion of the Dynamic Window Approach (DWA) planner.

Voronoi diagrams are another popular method for motion planning and are used to divide a space into regions based on the distance to a set of points or obstacles. The result is a map of polygons, with each polygon representing the area closest to a specific point or obstacle. This can be used to create a roadmap of the environment which can then be used to plan a path for the robot.

Cell decomposition is a method that involves dividing the environment into a grid of small cells, each cell representing a possible location for the robot. This method can be used to quickly check for collisions between the robot and obstacles and can also be used to plan a path for the robot.

Probabilistic planning involves using probability distributions to model the uncertainty in the robot's position and the location of obstacles. This can be used to plan a path that takes into account the likelihood of the robot's true position, as well as the likelihood of obstacles being present in a certain location.

Artificial potential fields are a method for obstacle avoidance in which the environment is represented by a scalar field, with the value of the field representing the potential energy of a point. The idea is to assign a repulsive force to obstacles and an attractive force to the goal location. The robot can then move towards the goal by following the gradient of the field, while avoiding obstacles by moving away from areas with high potential energy.

Dynamic Window Approach (DWA) planner is a method for motion planning that involves discretely sampling in the robot's control space, performing forward simulation for each sampled velocity, and then evaluating each trajectory using a set of metrics. The highest-scoring trajectory is then selected and sent to the robot's mobile base. DWA has been shown to be effective in dealing with dynamic environments, as it can quickly adapt to changes in the robot's surroundings.

3.2.1 DWA Planner

The DWA (Dynamic Window Approach) [14] [24] algorithm is a popular method for autonomous navigation in mobile robots.

It is a popular motion planning algorithm for mobile robots. It is a reactive algorithm that can be used for online decision making in environments with dynamic obstacles. DWA has gained widespread popularity due to its ability to handle real-time situations and generate smooth, feasible, and safe motions in cluttered environments.

The algorithm determines the best trajectory for the robot considering its current speed and the limits of its dynamic capabilities, such as its maximum linear and angular velocity.

The algorithm has been extensively tested and validated in various environments, making it a reliable and widely used tool for motion planning in mobile robotics. In this section, we will present a detailed overview of the DWA algorithm, including its underlying principles, algorithms, and applications.

The basic principle of the DWA algorithm is to discretely sample in the robot's control space, perform forward simulation for each sampled velocity, evaluate the resulting trajectory using a metric that incorporates characteristics such as proximity to obstacles, proximity to the goal, and proximity to the global path, and finally select the highest scoring trajectory to send to the mobile base. This process is then repeated in a loop to continually update the robot's trajectory. Expressed as a simple step list, the algorithm goes like this:

1. **Dynamic Window Selection:** The first step of the DWA algorithm is to select the dynamic window, which defines the set of possible velocities and accelerations that the robot can take. The dynamic window is defined by the maximum and minimum values of the velocity and acceleration that are physically possible for the robot to achieve, taking into account its current state and the constraints imposed by the environment.

2. **Sampling of Trajectories:** Once the dynamic window is defined, the next step is to sample a set of possible trajectories that the robot can take within the dynamic window. The trajectories are represented by a set of points that correspond to the velocity and acceleration at different points in time.

3. **Evaluation of Trajectories:** In this step, the DWA algorithm evaluates the sampled trajectories based on some performance criteria, such as the distance from the robot to the goal, the proximity to obstacles and the smoothness of the trajectory. The goal is to find the best trajectory that satisfies the performance criteria.

4. **Final Trajectory Selection:** The best trajectory, as determined in the previous step, is then selected as the final trajectory for the robot to follow.

5. O**bstacle Avoidance:** Finally, the DWA algorithm takes into account any obstacles in the environment and adjusts the final trajectory to avoid collision with obstacles.

In general, the DWA algorithm is a real-time, online algorithm that can handle a range of scenarios and environments, and its dynamic window approach

allows it to generate a set of feasible and optimal trajectories for the robot to follow.

This algorithm, implemented in the ROS `dwa_local_planner` package, connects the path planner to the robot by creating a kinematic trajectory that guides the robot from a start to a goal location. Using a map, the planner generates a value function, represented as a grid map, that encodes the costs of traversing through the grid cells. The DWA controller's job is to use this value function to determine the dx, dy, and $d\theta$ velocities that will be sent to the robot.

One key feature of DWA is its ability to handle dynamic environments by constantly updating the robot's trajectory based on the current state of the environment. This is achieved by continuously sampling and evaluating different velocities, discarding illegal trajectories that collide with obstacles, and selecting the highest-scoring trajectory that avoids obstacles while still moving towards the goal.

It is important to note that the DWA algorithm assumes a 2D environment and is not suitable for 3D environments.[15] Additionally, the DWA algorithm assumes that the robot's motion is continuous, which may not be true for all robots, and it may not be the best approach for all scenarios. Therefore, it is crucial to carefully assess the suitability of DWA for a specific application before using it for autonomous navigation.

The DWA algorithm is illustrated in Figure 3.8, showing the current position of the robot, the target location, and obstacles in the environment. The algorithm samples different velocities, performs forward simulation, evaluates the trajectories, and selects the highest-scoring trajectory to send to the robot, allowing it to navigate towards the goal while avoiding obstacles.

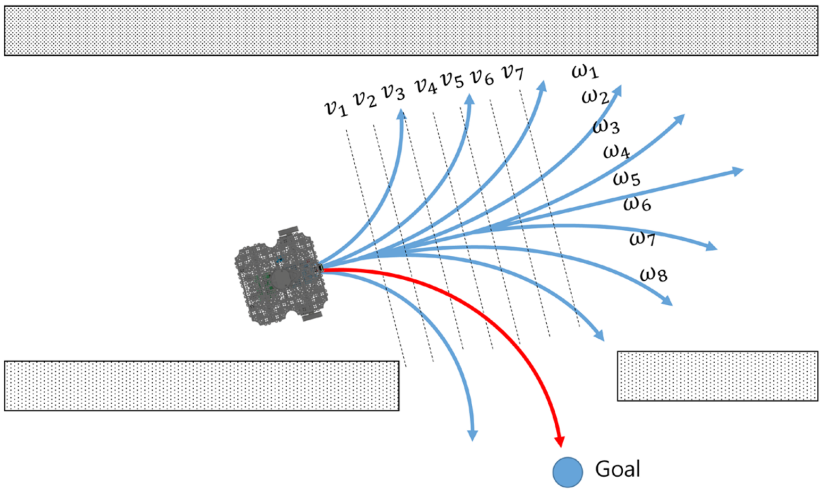

Figure 3.8: Dynamic Window Approach algorithm example.

3.3 Millimeter wave radars

Millimeter wave (mmWave) sensors are devices that operate in the millimeter wave frequency range, typically between 30 GHz and 300 GHz. They are used in a variety of applications, including imaging, sensing, and communication.

mmWave sensors work by emitting a beam of millimeter waves and measuring the reflection or transmission of the waves after they interact with an object or scene. The properties of the object or scene, such as its shape, size, and material composition, can be inferred from the measured waves.

One of the key advantages of mmWave sensors is their ability to penetrate certain materials, such as clothing and fog, that are opaque to other types of electromagnetic radiation, such as visible light and infrared radiation. This makes mmWave sensors useful for a variety of imaging and sensing applications, including security screening, industrial inspection, and medical imaging.

The basic principle of operation of mmWave sensors is the same as that of radar and radio communication systems. Sensors transmit a signal and measure the reflection or transmission of the signal after it interacts with an object or scene. The reflection or transmission characteristics of the signal are a function of the properties of the object or scene and can be used to infer information about the object or scene.

3.3.1 FMCW Radars

Frequency Modulated Continuous Wave (FMCW) radar is a common type of millimeter wave sensor that uses a continuous wave signal that is modulated with a linear frequency ramp. The FMCW radar transmits the modulated signal and receives the reflected signal from an object. The frequency difference between the transmitted and received signals is proportional to the distance of the object from the sensor. By analyzing the frequency difference via demodulation, the distance of the object can be determined.

The operation of FMCW radar can be broken down into several steps:

1. A continuous wave signal is generated at a starting frequency, f0.

2. The continuous wave signal is modulated with a linear frequency ramp, resulting in a signal with a frequency that increases or decreases linearly over time.

3. The modulated signal is transmitted and reflected by an object.

4. The reflected signal is received by the sensor and mixed with a locally generated copy of the transmitted signal.

5. This process is known as heterodyning, and results in a beat frequency that is proportional to the frequency difference between the transmitted and received signals.

6. The beat frequency is demodulated, resulting in a signal that is proportional to the distance of the object from the sensor.

FMCW radar has several advantages over other types of radar, such as pulsed radar. One of the main advantages is the ability to measure range and velocity simultaneously. This is achieved by analyzing the frequency shift of the received signal, which is proportional to the range, and the slope of the frequency ramp, which is proportional to the velocity. Additionally, FMCW radar has a lower peak power than pulsed radar, which results in lower transmitted power and a reduced risk of interference with other systems.

FMCW radar is widely used in many applications, including automotive, industrial, and military applications. In automotive applications, FMCW radar is used in advanced driver assistance systems (ADAS) such as adaptive cruise control, lane departure warning, and automatic emergency braking. In industrial applications, FMCW radar is used for level and distance measurement in a wide range of industries including the oil and gas, chemical, and food and beverage industries. In military applications, FMCW radars are used to detect and track aircraft and missiles.

In robotics, FMCW radar can be used for rangefinding and obstacle detection. By transmitting a modulated signal and measuring the time delay of the reflected signal, the distance to an object can be determined. This information can be used by a robot for navigation and obstacle avoidance. Additionally, FMCW radar can be used for sensing the velocity of objects, which can be useful for tasks such as tracking and following moving objects.

3.3.1.1 Range Measurement

The main concept in radar systems is the transmission of an electromagnetic signal that objects reflect in its path [23]. In particular, as explained previously, the signal used in FMCW radars is a signal in which the frequency increases linearly with time. An FMCW radar transmits a signal called a "chirp". A chirp is a sinusoidal signal whose frequency increases linearly with time, as shown in Figure 3.9 that shows a representation of a chirp signal, with magnitude (amplitude) as a function of time.

Therefore, knowing the nature of the chirp signal is convenient to use a frequency vs. time plot (or f-t plot) to represent a chirp. A chirp is generally characterized by a start frequency (f_c), Bandwidth(B) and duration The Slope (S) of the chirp defines the rate at which the chirp ramps up. In the example in Figure 3.10 the chirp sweeps a bandwidth of 4 GHz in $40\mu\ s$ which corresponds to a slope of 100 MHz/μs

An FMCW works as described in Figure 3.11. A synthesizer (synth) generates a chirp and then the chirp is transmitted by the TX antenna. The chirp is

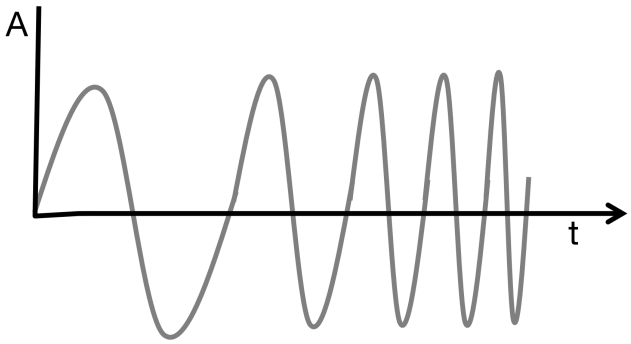

Figure 3.9: Chirp signal, with amplitude as a function of time. [23]

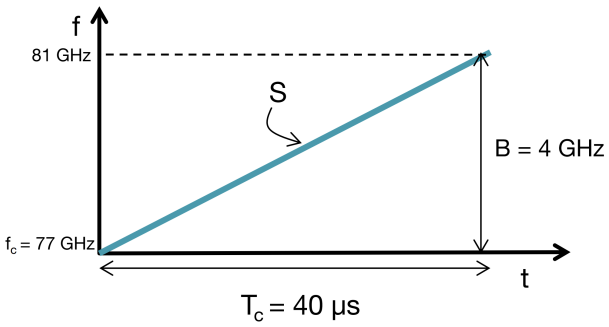

Figure 3.10: Chirp signal, with frequency as a function of time. [23]

reflected off an object, and the reflected chirp is received at the RX antenna. The RX signal and TX signal are 'mixed' and the resulting signal is called an 'IF signal'.

The FMCW Radar uses a frequency mixer. A frequency mixer is an electronic component that combines two signals to create a new signal with a new frequency. Given two sinusoidal inputs x_1 and x_2 (Equations 3.43 and 3.44):

$$x_1 = sin(\omega_1 t + \Phi_1) \tag{3.43}$$

$$x_2 = sin(\omega_2 t + \Phi_1) \tag{3.44}$$

The output x_{out} has an instantaneous frequency equal to the difference of the instantaneous frequencies of the two input sinusoids. The phase of the output x_{out} is equal to the difference of the phases of the two input signals.

$$x_{out} = sin[(\omega_1 - \omega_2)t + (\Phi_2 - \Phi_1)) \tag{3.45}$$

The operation of the frequency mixer can be understood graphically by looking at the representation of the chirp frequency of TX and RX as a function of time (the $f - t$ plot shown in Figure 3.12.

The top figure in Figure 3.12 shows the TX-signal transmitted by the radar and the RX-signal that is reflected from an Object detected, as a function of time. Note that the RX signal is just a delayed version of the TX signal. (τ

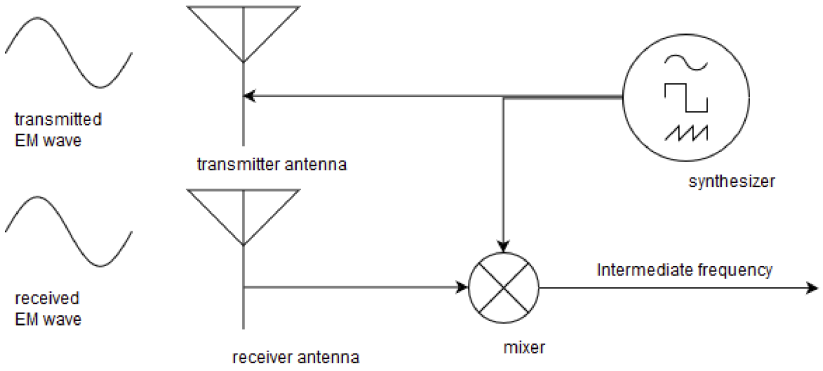

Figure 3.11: FMCW radar block diagram. [23]

denotes the round-trip time between the radar and the Object. Also S denotes the slope of the chirp). Recall that the frequency of the signal at the mixer output is the difference of the instantaneous frequency of the TX chirp and the RX chirp. As shown in the figure below in Figure 3.12, this is a straight line.

The time delay (τ) can be mathematically derived as described in Equation 3.46:

$$\tau = \frac{2d}{c} \tag{3.46}$$

where c is the speed of light.

Hence: a single object in front of the radar produces an IF signal that is a constant frequency tone. The frequency of this tone is.

$$f_\tau = \frac{S2d}{c} \tag{3.47}$$

Note that τ is typically a small fraction of the total chirp time. For a radar with a maximum distance of 300 m and $T_c = 40\ \mu s$ we have $t/T_c = 5\%$

Without going more in depth, is possible to demonstrate that the phase of this tone is:

$$\phi_0 = \frac{4\pi d}{\lambda} \tag{3.48}$$

with c being speed of light and λ wavelength of the electromagnetic wave.

And the resulting formula for this tone will be:

$$Asin(2\pi f_\tau t + \phi_0) \tag{3.49}$$

So far we have talked about a single object in front of the radar.In real case scenarios the radar detects multiple objects.It is easy to extend this to the case where there are multiple objects in front of the radar.

So, in Figure 3.13 we have a radar transmitting a single chirp and there are multiple reflected chirps from different objects. Each chirp is delayed by a different amount depending on the distance to that object.

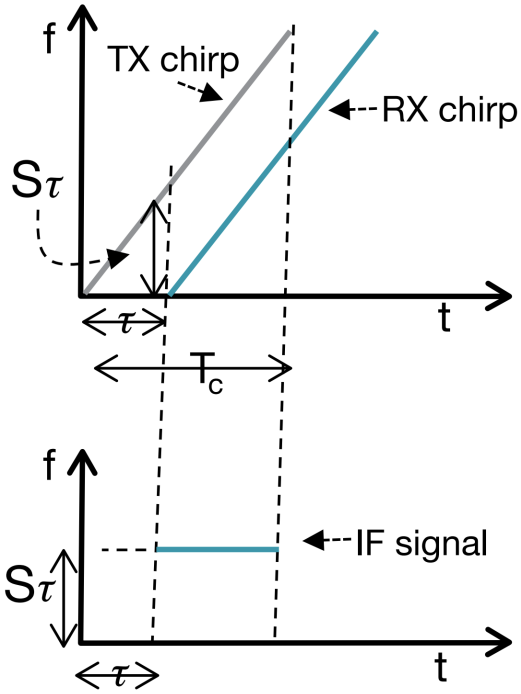

Figure 3.12: IF is a constant signal. [23]

A frequency spectrum obtained using the Fourier Transform of the IF signal will reveal multiple tones, the frequency of each being proportional to the range of each object from the radar.

An important issue to solve is the range resolution of the radar. So, we want to calculate how close any two objects can still be resolved as two peaks in the i-f spectrum.

We can intuitively assume that a longer chirp with a greater bandwidth will have a better resolution than one with smaller bandwidth. In particular given Equation 3.47 and knowing that rwo signals can be resolved if $\Delta f > 1/T_c$ where T_c is the chirp duration, we can find that the range resolution is:

$$d_{res} = \frac{c}{2ST_c} = \frac{c}{2B} \tag{3.50}$$

where S is the slope of the chirp, so the Bandwidth B is $B = S \cdot T_c$

3.3.1.2 Velocity Estimation

An FMCW radar must send at least two chirps separated by a specific time T_c to measure velocity. As seen in the preceding section, the range of the object is determined by processing each reflected chirp using FFT (range-FFT). The FFT will show peaks in the same position, but with distinct phases for each chirp. The calculation of the object's velocity may be done using the measurement of the different phases.

The phase difference can be derived from Equation 3.48 as follows:

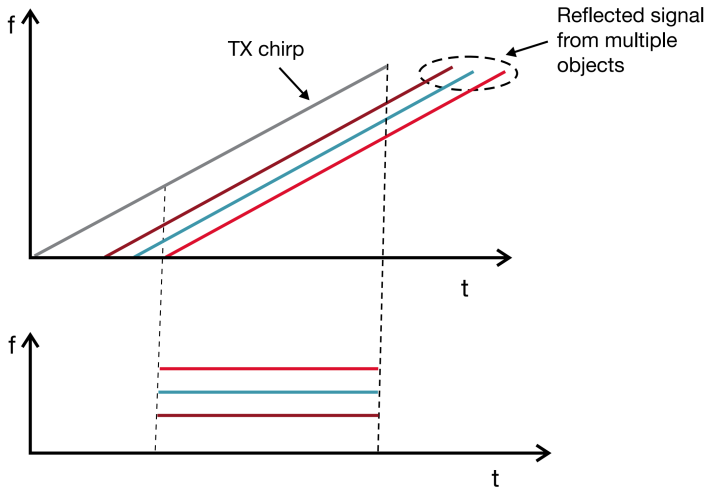

Figure 3.13: Multiple IF tones for multiple object detection. [23]

$$\Delta\phi = \frac{4\pi v T_c}{\lambda} \tag{3.51}$$

It is then straightforward to calculate the velocity:

$$v = \frac{\lambda \Delta\phi}{4\pi T_c} \tag{3.52}$$

The velocity measurement is based on a phase difference, so there will be ambiguity. The measure is non-ambiguous if $|\Delta\phi| < \pi$. So, the maximum relative speed measured by two chirps spaced T_c is:

$$v_{max} = \frac{\lambda}{4T_c} \tag{3.53}$$

If several moving objects with various speeds are present at the moment of measurement and are all located within an equal range of the radar, the two-chirp velocity measurement method will not be accurate. These objects will produce reflecting chirps with the same IF frequencies, since they are at the same distance from one another. As a result, the range-FFT will produce a single peak that is the signal from all of these equal-range objects combined. Simple phase comparison methods are ineffective. In this instance, the radar system needs to send out more than two chirps in order to measure the speed. It sends N chirps that are spread out evenly. Chirp frame refers to this collection of chirps. Figure 3.15 depicts the frequency of a chirp frame as a function of time.

Using the example of two objects that are equally spaced from the radar but have differing velocities, v1 and v2, the processing technique is detailed below. After applying Range-FFT to the reflected set of chirps, a set of N identically located peaks with distinct phases that each incorporate the phase contributions from both of these objects are produced (the red and blue phasors in Figure 3.16 show the individual phase contributions from each of these objects, respectively).

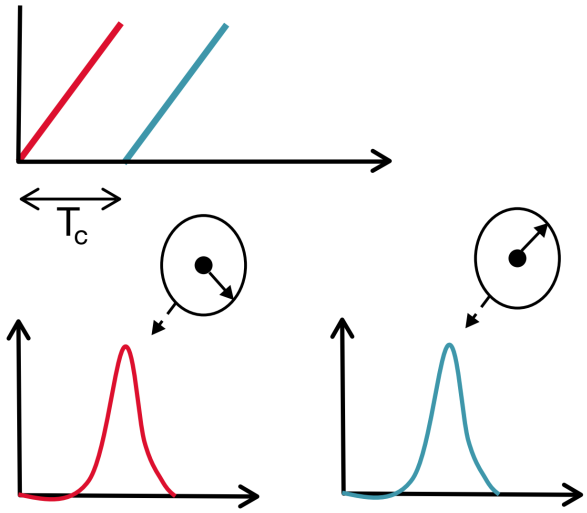

Figure 3.14: Two-chirp velocity measurement. [23]

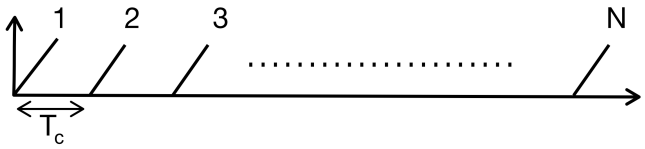

Figure 3.15: Chirp frame. [23]

So, to resolve the two objects, a second FFT, named Doppler-FFT, is performed on the N phasors as shown in Figure 3.17

Then, from the Figures 3.17 ω_1 and ω_2 both correspond to the phase difference between consecutive chirps for the objects.

$$v_1 = \frac{\lambda\omega_1}{4\pi T_c}, v_2 = \frac{\lambda\omega_2}{4\pi T_c} \tag{3.54}$$

3.3.1.3 Angle of Arrival Estimation

An FMCW radar system can estimate the angle of a reflected signal with the horizontal plane, as shown in Figure 3.18. This angle is also called the angle of arrival (AoA).

A slight change in an object's distance causes a phase change in the peak of the range-FFT or Doppler-FFT, which is the basis for angular estimate. As seen in Figure 3.19, this outcome is used to execute angle estimate utilizing at least two RX antennas. The FFT peak has a phase change as a result of the varied distances between the object and each of the antennas. We can estimate the Angle of Arrival thanks to the phase of these changes.

Therefore, the phase difference in the configuration of Figure 3.19 is derived as follows:

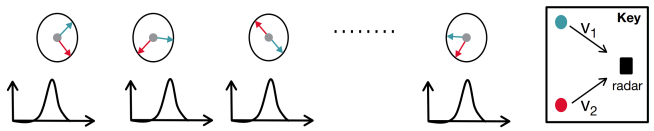

Figure 3.16: The range-FFT of the reflected chirp frame results in N phasors. [23]

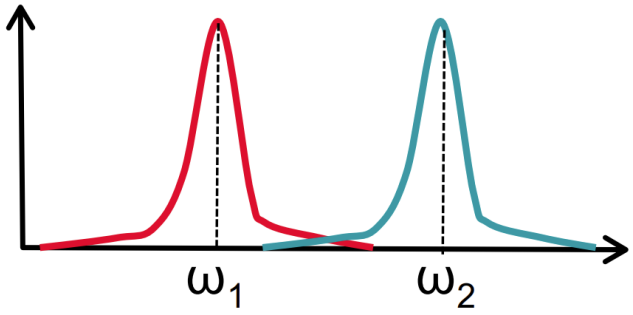

Figure 3.17: The range-FFT of the reflected chirp frame results in N phasors. [23]

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda} \tag{3.55}$$

Assuming a planar wavefront basic geometry, so that $\Delta d = lsin(\theta)$, where l is the distance between the antennas, the angle of arrival can be computed from the measured phase difference $\Delta\phi$ as described:

$$\theta = sin^{-1}(\frac{\lambda\Delta\phi}{2\pi l}) \tag{3.56}$$

Note that $\Delta\phi$ depends on $sin(\theta)$. This is called a nonlinear dependency. $sin(\theta)$ is approximated by a linear function when θ has a small value $sin(\theta)$ θ. As a result, the accuracy of the estimation depends on the Angle of Arrival and is more accurate when θ has a small value.

3.3.1.4 Multiple Input Multiple Output Radar

One of the ways to improve the performance of mmWave radars for SLAM is to use a Multiple Input Multiple Output (MIMO) configuration. MIMO is a signal processing technique that uses multiple antennas at both the transmitting and receiving ends to increase the capacity and reliability of wireless communications. In the context of mmWave radars, a MIMO configuration can be used to improve the range resolution, angular resolution, and robustness of the radar system.

A MIMO mmWave radar system typically consists of multiple transmitting antennas and multiple receiving antennas. Transmitting antennas are used to transmit multiple signals at the same time, whereas receiving antennas are used

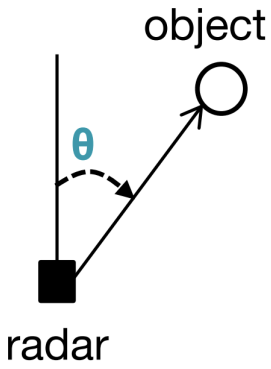

Figure 3.18: Angle of Arrival.

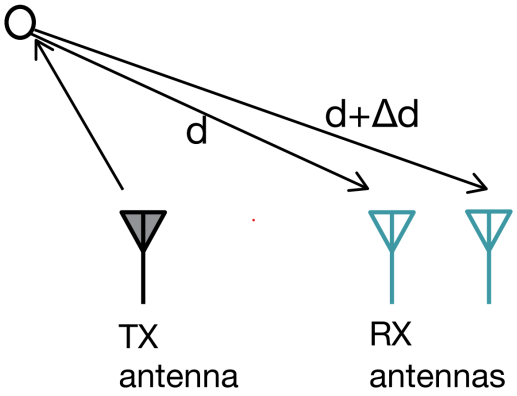

Figure 3.19: Two antennas are required to estimate the Angle of Arrival. [23]

to receive the signals reflected by objects in the environment. By using multiple antennas, the system can achieve a higher spatial resolution, which can be used to improve the accuracy and reliability of the radar measurements.

The MIMO configuration also allows for the transmission of multiple signals at different angles, which can be used to improve the angular resolution of the radar system. This is particularly useful in environments with occlusions or limited visibility, where a single sensor may not be able to provide a complete map of the environment.

Additionally, MIMO mmWave radar systems can also improve the robustness of the system by using multiple signals to mitigate the effects of multipath propagation, interference, and noise. Multipath propagation occurs when a signal is reflected by multiple objects before reaching the receiver, causing delayed versions of the same signal to arrive at the receiver at different times. This can cause constructive and destructive interference, which can degrade radar measurements. By using multiple signals, the MIMO configuration can mitigate the effects of multipath propagation by providing multiple measurements of the same object.

Furthermore, the MIMO configuration can also improve the robustness of the

system against interference and noise. By transmitting and receiving multiple signals, the system can use signal processing techniques, such as beamforming, to enhance the desired signals and suppress interference and noise.

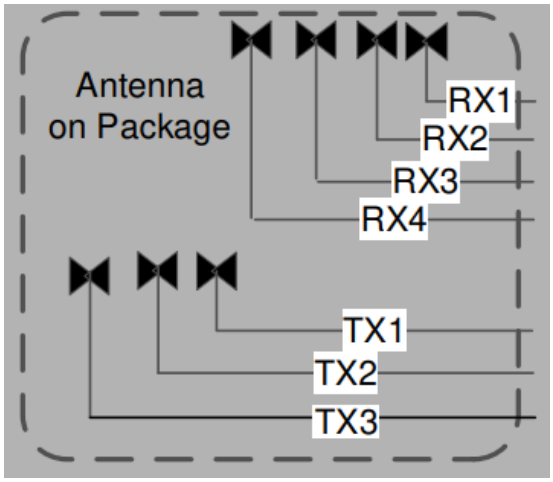

Figure 3.20: MIMO configuration of the TI IWR1843 sensor with 3 TX antennas and 4 RX antennas)

Chapter 4

Proposed Approach

4.1 Used Technologies

In this section, we will discuss the various technologies used in our research on SLAM using mmWave radars. Specifically, we will delve into the mmWave sensor, the Turtlebot3 and our custom robot platforms, the software frameworks utilized, including ROS and various ROS packages such as gmapping and Hector SLAM. We will also cover the sensor tuning process, as well as custom algorithms that were used in our implementation. The purpose of this chapter is to provide a comprehensive overview of the technologies used in our research, including the methods and techniques employed to optimize their performance. Additionally, this chapter will provide insights into the challenges and considerations that were taken into account when selecting and implementing the technologies used in our research.

All the source code that was made and used for the simulation, experiments and implementations for this thesis can be found on `https://github.com/nicolapace/mmwave_slam`.

4.1.1 mmWave sensor

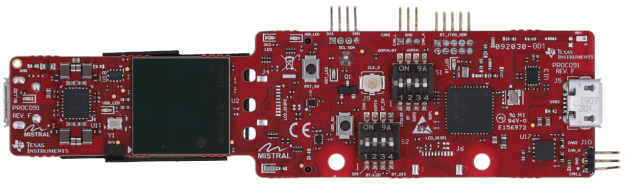

Figure 4.1: IWR1843 evaluation module for integrated AoP intelligent mmWave sensor.

The sensor utilized for the experiments has been the IWR1843AOPEVM [3] evaluation module for an integrated antenna-on-package (AoP) intelligent mmWave sensor with an FMCW transceiver by Texas Instruments. The IWR6843 antenna-on-package (AoP) evaluation module (EVM) is single-chip 77 and 79 GHz FMCW mmWave Sensor Antennas-On Package , which integrates wide field-of-view (FoV) antennas. This FMCW radar enables access to point-cloud data and power over a USB interface. The sensor has 4 receivers and 3 transmitters integrated in the and also has integrated PLL, transmitter, receiver, baseband, and ADC. It has a 76 GHz to 81 GHz bandwidth coverage with 4-GHz continuous. Supports 6-bit phase shifter for TX Beam and has an ultra-accurate chirp engine based on fractional-N PLL.

The IWR1843AOPEVM sensor was chosen among other sensors for several reasons. Firstly, Texas Instruments provides an API and a package for ROS, which allows for easy integration with the autonomous navigation system. Additionally, the sensor's integrated antenna-on-package (AoP) design provides a wide field-of-view (FoV) and enables access to point-cloud data and power over a USB interface. This makes the sensor a good choice for Simultaneous Localization and Mapping (SLAM) experiments as it allows for accurate and efficient data collection.

Furthermore, the sensor's FMCW (Frequency-Modulated Continuous Wave) radar technology provides high-resolution imaging capabilities, making it well suited for obstacle detection and avoidance in autonomous navigation. Additionally, the sensor's integrated PLL, transmitter, receiver, baseband, and ADC, allows for precise and accurate data processing and collection. With its 76 GHz to 81 GHz bandwidth coverage and 4-GHz continuous support, the sensor is able to cover a wide range of frequencies and provide high-resolution imaging.

In our application, sensors are connected to the RaspberryPi via USB.

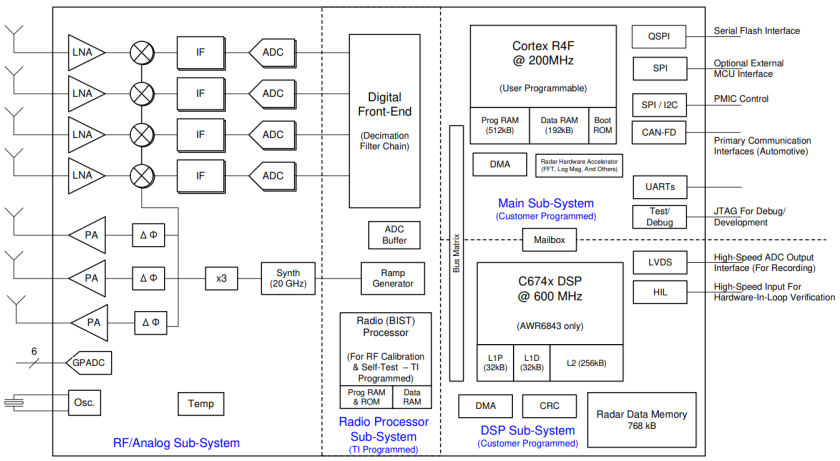

Figure 4.2: Functional Block Diagram of the IWR1843 mmWave sensor.

4.1.1.1 Sensor tuning

Millimeter wave sensors are complex devices that typically require programming of the onboard MCU and DSP to process data and generate appropriate output for the task at hand. Texas Instruments (TI) provides several demonstration labs within their Industrial and Automotive Toolboxes that can be used for mobile robotics applications. The starting firmware, called Out of Box Demo (OOB), is introduced as a generic form in the Industrial Toolbox. This demo is used to demonstrate the capabilities of the mmWave sensors and provides an easy way to modify their configuration parameters and acquire data through a USB port. In particular, the AWR1843AOP sensor used in this work has a single lab in the Automotive Toolbox, the AoP Obstacle Detection. This lab focuses on obstacle recognition and provides an interactive Matlab interface to visualize the sensor captured points. The OOB demo is sufficient for using mmWave sensors in mobile robotics applications, however, since the release of the AWR1843AOP sensor is very recent, an OOB demo dedicated to it is not officially available from TI. However, it is possible to adapt the OOB developed for xWR1843 sensors to xWR1843AOP devices. A version of OOB developed for AWR1843AOP has been made available by the developer community on the TI forum. It should be noted that using this demo, fine-tuning and a good configuration of parameters have been found, resulting in improved performance.

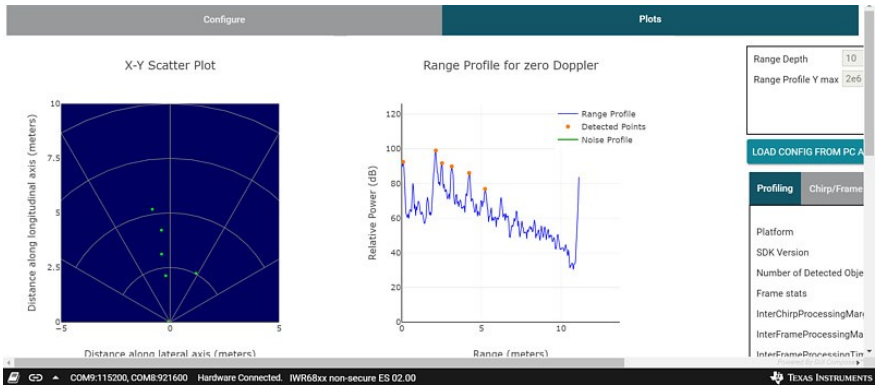

Figure 4.3: Tuning process of the IWR1843 mmWave sensor.

4.1.2 Turtlebot3

The TurtleBot3 [39] Waffle Pi is a low-cost open-source robot platform that is widely used in research and education. It is based on the Raspberry Pi 3 Model B+ and an OpenCR platform to pilot motors and read data from sensors. The robot is also equipped with various sensors and actuators, including a Lidar sensor, a 9-axis IMU, encoders on each wheel, and a stereo camera. It also uses ROS (Robot Operating System) as a software framework, which allows easy integration of various sensors and actuators.

The TurtleBot3 Waffle Pi's small size and lightweight design, at only 138mm × 178mm × 192mm (l×w×h), makes it an ideal platform for testing the mmWave

sensor in different environments. Its compact design also allows for easy maneuverability in tight spaces and its small footprint makes it easy to transport and set up in various locations. The Turtlebot3 can also be easily piloted via an infrared joystick.

In our experiments, we used the TurtleBot3 Waffle Pi as one of the platforms to evaluate the performance of the SLAM system with mmWave radars. The Lidar sensor provides range measurements which can be used for localization and mapping, while the 9-axis IMU provides information about the robot's orientation. The stereo camera can be used for visual odometry, which can provide additional information for localization and mapping.

Additionally, we equipped the TurtleBot3 Waffle Pi with a mmWave sensor positioned on the front of the robot and used both encoders on the wheels the robot. The Raspberry Pi can be used to read data from the encoders via the CAN bus and from the sensors via the OpenCR and to run all the ROS nodes for SLAM. The Raspberry Pi also has built-in WiFi that can be used to launch remote ROS nodes, which makes it a powerful and versatile component of the robot.

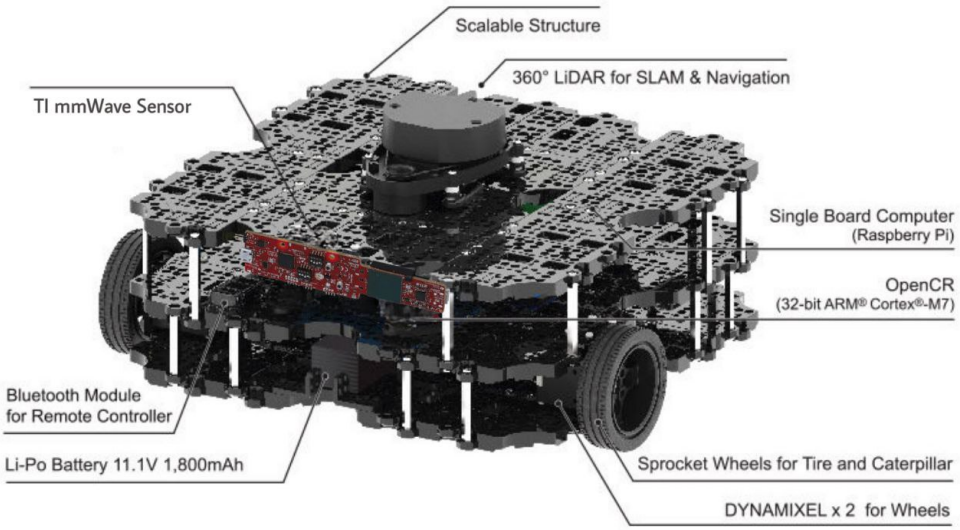

Figure 4.4: Turtlebot3 waffle-pi.

4.1.3 Custom Robot

The custom robot used for SLAM experiments with multiple mmWave radars is shown in Figure 4.6. The robot has a dimension of 70cm × 80cm and is equipped with 4 mmWave sensors, one on each side as shown in figure 4.5. The robot has four wheels, two free swedish wheels and two motorized wheels. The radius of

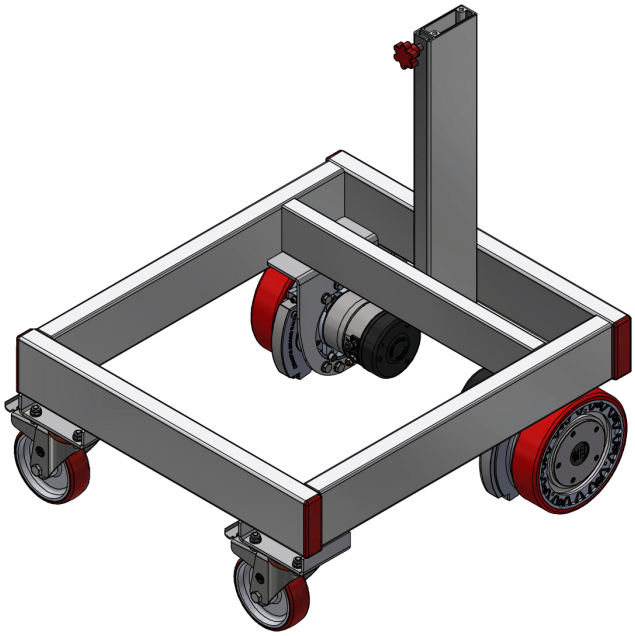

Figure 4.5: 3D CAD model of custom robot used for experiments.

the motorized wheels is 11.6 cm and each wheel is equipped with an AMT10 Modular Incremental Encoder [2].

The encoder used in the robot's wheels is also an important technology that is used to measure the robot's motion. The AMT10 Modular Incremental Encoder [2] is a high-resolution encoder that can provide precise measurements of the robot movement. This is crucial for accurate localization and mapping in SLAM applications.

The custom robot used for the experiments is also equipped with a Raspberry Pi 4 Model B 4.7 with 8 GB of RAM, which was used to read the data from the encoders through the CAN bus and from the sensors through USB.

Additionally, the Raspberry Pi was used to run all the ROS nodes for SLAM. The Raspberry Pi also has built-in WiFi, which can be used to launch remote ROS nodes.

The Raspberry Pi is a powerful and cost-effective single-board computer that can be easily integrated into robotic systems. It's capable of running Linux operating systems, including ROS, and can be used to interface with various sensors and actuators.

In this case, the Raspberry Pi was used to read data from the encoders via the CAN bus and from the sensors via USB, which allowed the robot to obtain the necessary data for accurate localization and mapping in the SLAM application. To read data from the CAN bus an add-on to add CAN functionalities to the RaspberryPi was used.

By using the Raspberry Pi, the robot was able to use the ROS operating system to process sensor data and make decisions about its location, and also communicate with other devices. Additionally, the built-in WiFi capability of

the Raspberry Pi allowed for remote access to the robot, making it easy to launch ROS nodes from a remote machine.

Figure 4.6: Custom robot used for experiments.

In addition, a custom board is used to manage power and communication, in particular to manage the CAN bus network. In fact, a joystick is mounted on the robot that was used in experiments to pilot the robot manually.

Figure 4.7: AMT10 Modular Incremental Encoder

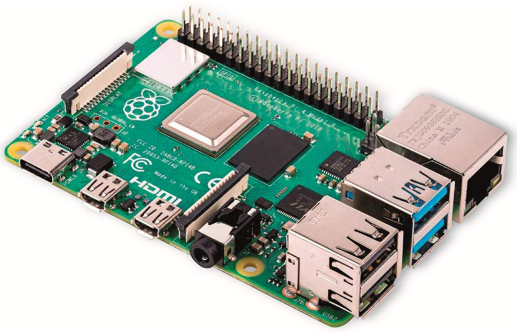

Figure 4.8: Raspberry Pi 4 Model B

4.1.4 ROS

Robot Operating System (ROS) is a widely used framework in research for developing robotic applications. It provides a set of tools and libraries for robot control, perception, and communication, as well as a powerful development environment for creating, testing, and deploying robotic systems. In this section, we will discuss how ROS can be used for simultaneous localization and mapping (SLAM) using millimeter-wave (mmWave) radars.

ROS provides a set of libraries and tools for working with mmWave radars, including drivers for popular sensor models, message definitions for radar data, and visualization tools. In particular, the code utilized in the experiments is heavily borrowed from the open source `ti_mmwave_rospkg` [16] that provides an RTOS wrapper for the TI mmWave sensors driver. This package provides the output of the radars as a `PointCloud2` data that with the PointCloud Library (PCL) [37] are easy to work with.

In addition, ROS provides a number of packages for SLAM, including the `gmapping` package, which provides a wrapper for the popular `gmapping` library and allows for real-time SLAM using "laser-like" data. This package can be adapted to work with mmWave radar data by modifying the sensor driver and message definition to match the radar data format (PointCloud2).

Another ROS package for SLAM is `hector_slam`, which is another laser-based SLAM solution that can also be adapted to work with radar data. It uses an extended Kalman filter (EKF) to estimate the robot's pose and a scan-matching algorithm to align the scans.

To use ROS for SLAM with mmWave radars, the first step is to set up the sensor driver and message definition to match the radar data format. Then, the appropriate SLAM package can be used to perform the localization and mapping. It is important to note that the performance of the SLAM solution may depend on the specific characteristics of the radar sensor and the environment, and some adjustments may be necessary to optimize the performance.

ROS provides a set of libraries and tools for working with mmWave radars, including drivers for popular sensor models, message definitions for radar data, and visualization tools. ROS also provides a number of packages for SLAM, including the `gmapping` and `hector_slam` packages. To use ROS for SLAM with mmWave radars, it is necessary to adapt the sensor driver and message definition to match the radar data format. The chosen SLAM package can then be used to perform localization and mapping. It is important to note that the performance of the SLAM solution may depend on the specific characteristics of the radar sensor and the environment, and adjustments may be necessary to optimize the performance.

Additionally, it is important to note that mmWave radars have unique characteristics that can affect SLAM performance, such as the limited field of view and range of the radar, the presence of reflections and interference, and the effect of weather and atmospheric conditions. Therefore, it is crucial to consider these factors when designing and implementing a SLAM system using MMWave

radars.

Another key aspect to consider when using mmWave radars for SLAM is the choice of the right frequency band; this will depend on the specific application, the environment, and the accuracy required. Furthermore, it is important to properly calibrate the radar sensor to ensure accurate measurements and to filter out noise and interference.

In conclusion, while ROS provides a powerful framework for developing robotic applications, it is important to consider the unique characteristics of mmWave radars when using them for SLAM, including the sensor characteristics, the environment, and the chosen frequency band. Careful consideration of these factors and proper adaptation of the sensor driver and message definition will lead to an effective and efficient SLAM solution using mmWave radars.

4.1.5 ROS packages used

Various ROS packages have been used for the scope of the thesis. To access data from the mmwave sensors, a modified version of the `ti_mmwave_rospkg` was used. For experiments with the Turtlebot3 the available navigation stack was used. For the custom robot, those Turtlebot3 packages were adapted and modified. Also, in the following subsection an inn depth view of the SLAM packages that we chose to use will be provided.

4.1.5.1 gmapping

In this thesis, as the main SLAM framework for the experiments, the `gmapping` package is employed. As discussed before, the `gmapping` package is a ROS package that provides an implementation of SLAM using a laser rangefinder and odometry data.

One of the key reasons for selecting the `gmapping` package is its widespread utilization and active community of developers and users. This affords easy access to support and resources, as well as the capability to seamlessly integrate the package with other ROS packages and tools.

The `gmapping` package utilizes a laser rangefinder to acquire range data, and odometry data to estimate the robot's pose. This information is then used to construct a 2D occupancy grid map of the environment, which can be used for navigation and localization. The package employs a probabilistic approach, which is based on the FastSLAM algorithm, to estimate the robot's pose and construct the map.

However, to use the `gmapping` package with the millimeter wave sensor, Pointcloud data needs to be converted to laserscan data. This conversion process can be achieved using the `pointcloud_to_laserscan` package, which is a ROS package that converts 3D point cloud data to 2D laser scan data. This package uses the `sensor_msgs/PointCloud2` message type as input and produces the `sensor_msgs/LaserScan` message type as output. It also allows you to set the min and max range of the scan and the number of scans per revolution.

For these reasons, the `gmapping` package is an appropriate choice for this thesis due to its robust and reliable implementation of SLAM, its wide range of features, and its flexibility in adapting to different experimental setups. Additionally, by using the `pointcloud_to_laserscan` package, the `gmapping` package can be used with millimeter wave sensor data, enabling the conversion of PointCloud data to laserscan data, and thus allowing the integration of mmWave sensor data into the SLAM process.

4.1.5.2 Hector SLAM

In this thesis, in addition to using the `gmapping` package, another SLAM implementation called Hector SLAM was also employed in the experiments to compare the results and evaluate the robustness of the system.

Hector SLAM is a 2D SLAM algorithm that is based on scan matching, which is a technique used to match the laser scan data with the previous scans in order to estimate the robot's pose. The package is also a ROS package that provides an implementation of SLAM using a laser rangefinder and odometry data. It is designed to work with both 2D and 3D laser scanners, as well as RGB-D cameras.

One of the key advantages of Hector SLAM is its ability to handle dynamic environments and perform well in large-scale environments. It uses a scan matching algorithm that is based on the ICP (Iterative Closest Point) algorithm, which can handle large amounts of data and estimate the robot's pose with high accuracy.

Hector SLAM also offers a number of advanced features such as loop closure detection, which allows the robot to detect when it has returned to a previously visited location, and graph-based SLAM, which allows the robot to construct a graph of the environment and use it for navigation and localization.

4.2 Implementation

4.2.1 Architecture of the system

In this chapter, we present the methodology and architecture of the system used in this thesis for the implementation of SLAM using mmWave radars for autonomous navigation.

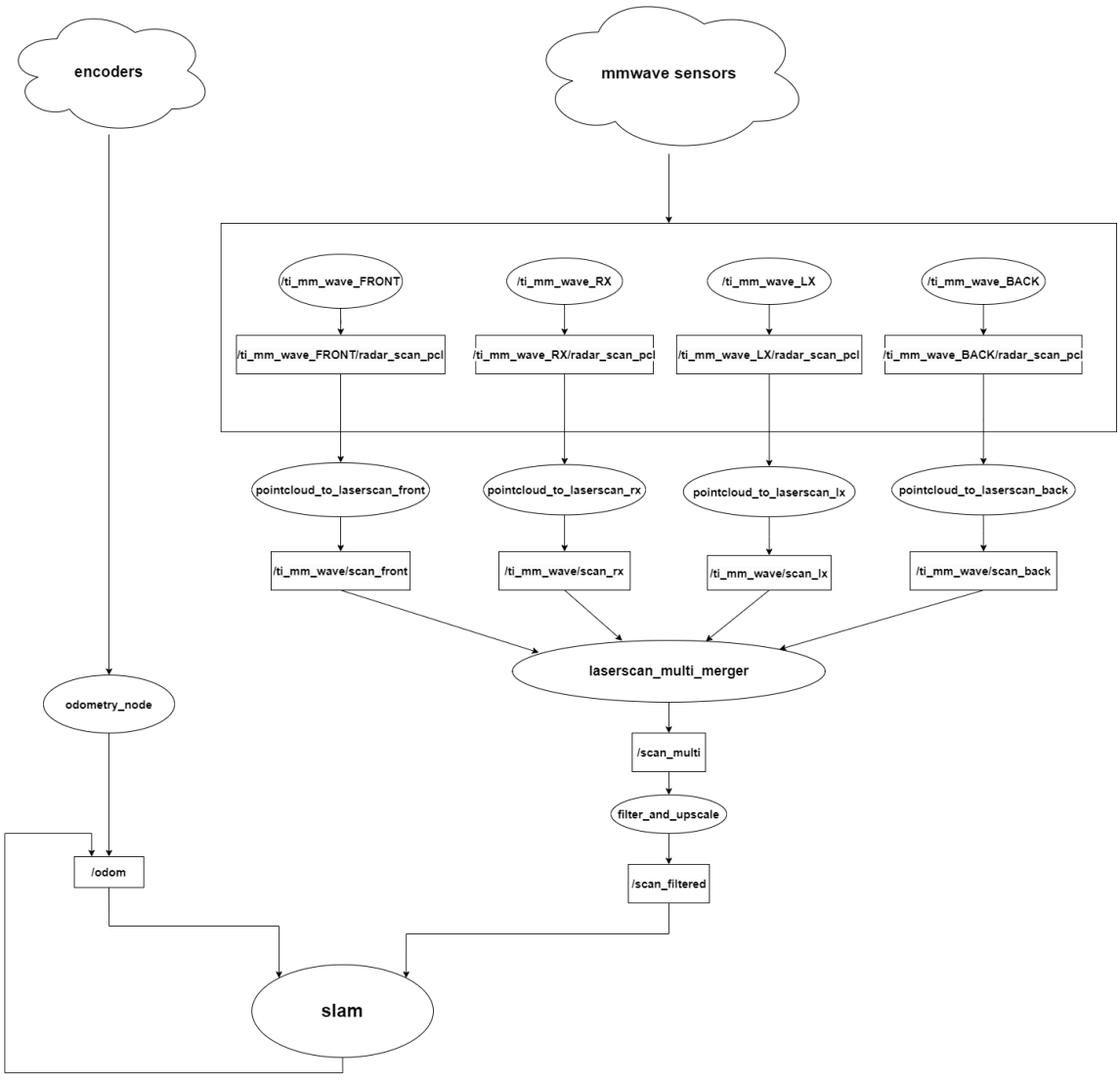

Figure 4.9: Graph of ROS nodes and topics.

The architecture of the system is based on the Robot Operating System (ROS) and consists of several nodes that work together to perform SLAM. The primary sensors used in the system, as discussed above, are the TI mmWave

sensors, which provide PointCloud2 data. This data is sampled using the ROS API package `ti_mmwave_rospkg`.

The first step in the architecture is the real-time conversion of each of the four PointCloud2 data streams into four different LaserScan data streams using the`pointcloud_to_laserscan` ROS node. The `pointcloud_to_laserscan` node converts the 3D point cloud data to 2D laser scan data and allows setting the min and max range of the scan, as well as the number of scans per revolution.

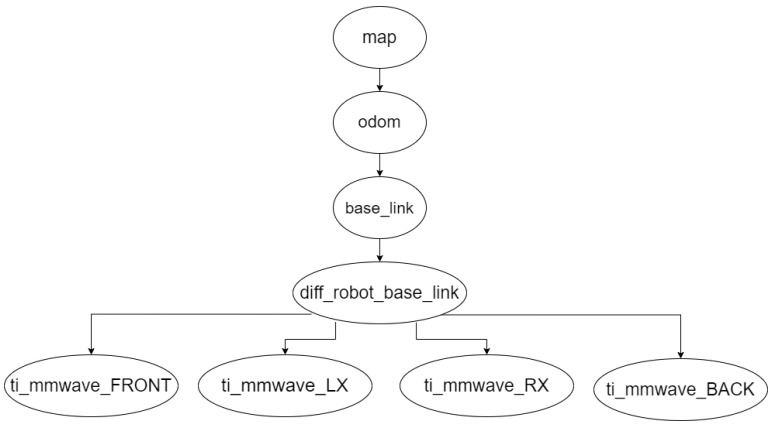

Figure 4.10: tf tree of the system.

The next step is the merging of the four `LaserScan` data streams into a single LaserScan data stream using `ira_laser_tools` [4], and in particular the `laserscan_multi_merger` node. This merged scan is then published on the ROS topic `/scan_multi` and is used as input for the following processing steps.

A custom node is used to filter and upsample the merged `LaserScan` data. This node uses a custom clustering and filtering technique, which improves the accuracy and reliability of the sensor data. The filtered and upsampled `LaserScan` data is then given to the SLAM node (such as `gmapping`). `gmapping` has been fine-tuned to work with the non-standard `LaserScan` data, which improves its performance.

In addition to the `LaserScan` data, the SLAM node also receives odometry data via a custom odometry node named `/odom`. This node reads the angular velocity of the wheels data via a Controller Area Network (CAN) bus from the encoders and calculates the odometry in real time. The computed odometry is then published on the `/odom` topic and is used to correct the odometry in real time by the SLAM node.

The odometry node is an important part of the system as it provides a measure of the robot's position and orientation, which is used by the SLAM node to construct the map and estimate the robot's pose. The use of a custom odometry node allows for the integration of the mmWave radar data with the odometry data, which improves the accuracy and reliability of the SLAM process.

In the following sections, each part of the system will be discussed in details.

4.2.2 *ti_mmwave_rospkg*

The `ti_mmwave_rospkg` is a package developed by Texas Instruments (TI) to work with mmWave radar sensors in the Robot Operating System (ROS) environment. It provides a set of tools and interfaces to configure, control and process the data from mmWave radar sensors in a ROS-based system. This package is designed to work with the TI mmWave sensor family, which includes sensors operating at various frequency bands, such as 76-81GHz and 60-64GHz.

The `ti_mmwave_rospkg` provides a wide range of features for working with mmWave radar sensors in ROS, including:

- Sensor configuration: The package includes a set of tools for configuring various parameters of the mmWave sensor, such as frame rate, range resolution, and azimuth resolution.
- Data acquisition: The package provides interfaces for acquiring data from the mmWave sensor, including raw data and point cloud data.
- Data processing: The package includes a set of tools for processing the acquired data, such as range-Doppler processing, clutter removal, and point cloud generation.
- Visualization: The package provides a set of tools for visualizing the acquired data, including a range-Doppler plot and a point cloud viewer.

Once installed, the package can be used by launching the appropriate launch files provided in the package and configuring the sensor as needed. The package also provides a set of ROS nodes for acquiring data from the sensor, processing the data, and visualizing the results.

The `ti_mmwave_rospkg` has been evaluated in various environments and tasks, such as autonomous navigation, object detection, and mapping. The package has shown good performance in terms of data acquisition, processing, and visualization. The package also provides a good level of flexibility in terms of sensor configuration and data processing.

The package was also modified to work with multiple sensors, enhance the performance of the system, and correct bugs encountered. For example, an Agglomerative Clustering algorithm was tried to upscale the number of Points in the point clouds detected by the sensors.

This package has been an important tool in this thesis for the implementation and evaluation of the proposed SLAM method using multiple mmWave radars.

4.2.3 Pointcloud to laserscan

The pointcloud to laserscan node is a ROS package that provides a component to convert `PointCloud2` messages to laserscan data `LaserScan` messages. This package is particularly useful when working with mmWave sensors, as it allows

for the transformation of mmWave data into laserscan data, which can then be used with laser-scan based algorithms such as laser-based SLAM.

The pointcloud to laserscan node subscribes to a topic called `cloud_in` which is of type `PointCloud2`, and publishes to a topic called "scan" which is of type `LaserScan`. The conversion process is based on the provided parameters, which include the minimum height of the point cloud, the angle increment, the input point cloud queue size, and the scan rate in seconds.

The pointcloud to laserscan node offers several parameters that can be adjusted to suit the specific needs of the application. For example, the angle increment parameter, which is set by default to $\pi/180$, determines the resolution of the laser scan in radians per ray. The `queue_size` parameter determines the size of the input point cloud queue and the `scan_time` parameter determines the scan rate in seconds.

The package also provides components to convert `LaserScan` messages back to `PointCloud2` messages. This is a port of the original ROS 1 package and was developed by the ROS Perception Team.

4.2.4 Laserscan data merger

For the task of merging multiple laser scan data into one single one, the approach implemented by `ira_laser_merger`[4] was used. The laser merger node used is a ROS package that allows for the merging of multiple single-plane laser scans into a single scan. This package is particularly useful when working with multiple laser scanners and interfacing with ROS nodes that require inputs in the form of LaserScan messages. The package offers a Dynamic Reconfigure interface, which allows for easy and on-line configuration of the node parameters.

The ira laser merger node has the following parameters:

- destination frame
- cloud destination topic
- scan destination topic
- laserscan topics

The destination frame is the frame to which the merged scan is referred. The cloud destination topic is the topic where the merged scan is published as a point cloud, which is useful for debugging. The scan destination topic is the topic where the merged scan is published as a laser message. The laserscan topics parameter is the list of the topics to which the node is subscribed.

4.2.4.1 Limitations of the merger

The laser merger node has a few limitations to take into account when using it for experiments. One limitation is that the generation of the merged laser scan follows an initial phase in which all the laser scans to be merged are flattened

to a single plane, which is the scanning plane of the destination (merged) laser scan. This can lead to errors in mapping, localization and navigation, as objects that would not be observable in the destination (merged) laser scan, but that are in some of the input scans, will be present in the final merged scan.

Another limitation is that the output scan must be in the form of the ROS LaserScan message, which implies that the points of the merged scan are generated as if they were measured from the measuring center of the destination laser scanner. This can lead to the generation of data that would not be physically feasible. An example of this would be a scenario where two scanners are mounted on the front corners of a rectangular vehicle, and the merged scan appears as if it were generated from a virtual scanner positioned halfway between the two. This could present data that would not be physically possible due to vehicle occlusions.

Knowing those limitations of the laser merger, it was may necessary to adjust the parameters accordingly and those limitations where taken into account when interpreting the results of the merged scans.

4.2.5 Data augmentation

In this section, we present the development and implementation of a custom filter and up-sampler for LaserScan data, specifically designed to address two common issues found in LaserScan data obtained from PointClouds generated by mmWave sensors: the presence of outliers and the sparsity of data.

LaserScan data, obtained from the conversion of PointCloud data, can often contain outliers, which are measurements that do not correspond to real objects in the environment. These outliers can be caused by reflections from objects that are not part of the environment or by errors in the sensor hardware. Additionally, LaserScan data obtained from mmWave sensors can be sparse, meaning it does not provide enough information to accurately represent the environment.

To address these issues, we developed a custom filter for LaserScan data that includes two main components: outlier removal and data up-sampling. The filter is based on a rotating moving average algorithm that groups non-zero nearby measures together, even if there are "holes" between them. This enables us to group nearby measures together.

The outlier removal component of the filter uses a statistical method to identify and remove measurements that do not conform to the overall distribution of the data. This is done by comparing each measurement to the mean and standard deviation of the entire dataset. Measurements that are more than a certain number of standard deviations away from the mean are considered outliers and are removed from the dataset.

The data-upsampling component of the filter has been implemented to improve the LaserScan points used by our SLAM implementation. The algorithm 3 describes the up-sampling process in detail. It starts by converting the 1D LaserScan data to 2D Cartesian data, and then finds the convex hull of each non-zero cluster using the Quickhull algorithm [6]. The Quickhull algorithm is a tech-

nique used to calculate the convex hull of a finite set of points in n-dimensional space, using a divide-and-conquer strategy similar to the quicksort algorithm. The worst-case time complexity of Quickhull for 2-dimensional non-random data is $O(n \log(n))$.

After finding the convex hull, the algorithm then adds random points using a normal distribution in place of the "holes" in the 1D LaserScan data, so that each added point lies inside the convex hull within each cluster. This step improves the realism of the generated data.

The performance of the custom filter was evaluated by comparing the filtered LaserScan data with the original data and with data filtered by other methods such as Agglomerative clustering. The evaluation showed that the custom filter was able to effectively remove outliers and increase the density of data, resulting in a more accurate representation of the environment.

Algorithm 3 Filtering and Upsampling Algorithm

```
1: Find non-empty cluster of point in 1D LaserScan data using rotating
   moving-average clustering
2: for all cluster in clusters do
3:    Remove outliers
4:    Convert cluster to cartesian 2D data
5:    Compute convex hull of cluster using Quickhull algorithm
6:    for all points in cluster do
7:       if point empty then
8:          Generate a random point in 1D LaserScan data that lies inside the
            convex hull
9:       end if
10:   end for
11: end for
```

Additionally, to further enhance the performance of the filter, we implemented an up-sampling component. This component was designed to increase the density of data points within each cluster by adding new points based on a normal distribution. The idea behind this is that, by adding more points within the cluster, we can increase the accuracy of the representation of the environment.

To achieve this, we developed a custom clustering algorithm that uses a moving average approach to group nearby measures. This algorithm is able to identify clusters of points even if there are gaps between them. Once the clusters are identified, the algorithm converts the 1D `LaserScan` data to 2D Cartesian data and then proceeds to find the convex hull of each cluster using the Quickhull algorithm [6].

The Quickhull algorithm is a widely used technique for computing the convex hull of a finite set of points in n-dimensional space. It utilizes a divide-and-conquer approach similar to that of quicksort, from which its name derives. The worst-case time complexity of Quickhull for 2-dimensional non-random data is $O(n \log(n))$. This makes it an efficient and effective algorithm for our application.

Once the convex hull of each cluster is computed, the algorithm generates new points within the cluster by adding random noise with a normal distribution. These new points are added in the "holes" in the 1D `LaserScan` data, ensuring that each added point lies inside the convex hull of the cluster.

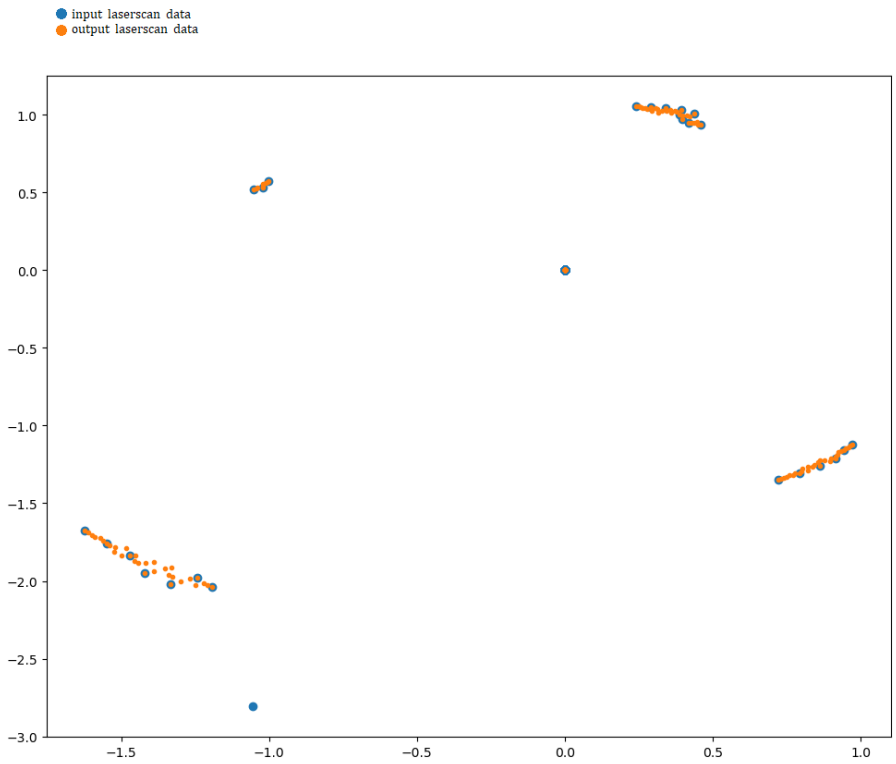

Figure 4.11: Example of the filtering and on real LaserScan data.

The performance of the custom filter was evaluated by comparing the filtered `LaserScan` data with the original data and the data filtered by other methods, such as using Agglomerative clustering. The evaluation showed that the custom filter was able to effectively remove outliers and increase the density of data, resulting in a more accurate representation of the environment. Additionally, the noise addition step improved the realism of the generated data.

In conclusion, the custom filter and up-sampler for the `LaserScan` data developed in this work addresses the issues of outliers and sparse data commonly found in the `LaserScan` data obtained from mmWave sensors. The use of a rotating moving-average clustering algorithm and the Quickhull algorithm allows for an efficient and effective outlier removal and data upsampling process. The resulting filtered data provides a more accurate representation of the environment and can improve the performance of SLAM and other robotics applications.

4.2.6 Navigation stack

The ROS Navigation Stack is a package of the ROS (Robot Operating System) that performs simultaneous localization and mapping (SLAM) and path planning, along with other functionalities for navigation. The Navigation Stack takes in information from odometry and sensor streams and outputs velocity commands to send to a mobile base. The stack is built on top of various other ROS packages, such as the `costmap_2d` package, which is responsible for building and maintaining a map of the environment.

One of the key features of the Navigation Stack is the use of a global planner, which is responsible for creating a high-level path for the robot to follow. The global planner takes in the current location of the robot, the location of the goal and the map of the environment, and outputs a path for the robot to follow. The ROS Navigation Stack also includes a local planner, which is responsible for fine-tuning the robot's path and ensuring that it avoids obstacles.

The Dynamic Window Approach (DWA) planner is one of the most popular local planners for the ROS Navigation Stack; it provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map.

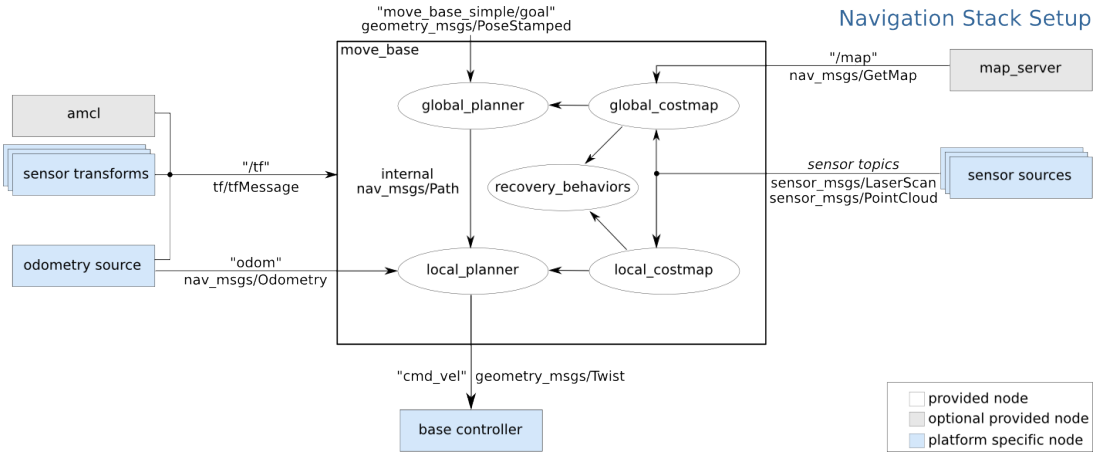

Figure 4.12: Description of the ROS `move_base` node and its configuration.

4.2.7 Map post-processing

For map post processing various techniques from the scientific literature have been explored. The main technique that has been explored is a machine learning framework implemented in Pytorch for image synthesis and semantic Manipulation with Conditional GANs, presented in [31].

To use the above post-processing implementation, the user will need a Linux or macOS operating system, Python 2.7 or 3.6, and an NVIDIA GPU with 12G or 24G memory, as well as CUDA and cuDNN. To make this work on Windows

and on devices without NVIDIA GPUs (such as the RaspberryPi) at inference time, the code was slightly modified.

The pre-trained model on the data-set presented in [31] was used. Results are show in the next chapter.

4.2.8 Performance of the system

The performance of the autonomous navigation system using the simultaneous localization and mapping (SLAM) algorithm with millimeter wave sensors and the dynamic window approach (DWA) was evaluated through a series of experiments. The aim was to assess the accuracy and reliability of the system in real-world scenarios.

In terms of hardware, the system consisted of a mobile robot equipped with millimeter wave sensors, encoders, and a microcontroller. The millimeter wave sensors were used for obstacle detection and mapping of the environment, while encoders were used for odometry calculation. The microcontroller was responsible for processing the sensor data and executing the SLAM and DWA algorithms in real time.

In the experiments, the robot was placed in different environments, and tasked with reaching a predetermined goal location. The system was able to successfully locate the robot and find a path to the goal in all experiments.

In terms of computational requirements, the system was designed to be efficient and lightweight, ensuring that it could be run in real-time on a single board computer such as the RaspberryPi or the Nvidia SHIELD. This was critical for the system to be practical and useful in real-world applications, as it meant that it could be deployed on a range of different platforms and in various environments.

Chapter 5

Results and Evaluation

In this chapter, we present the results of our experiments that evaluate the use of multiple mmWave radar sensors for autonomous navigation. We conducted experiments in two different scenarios: one with a Turtlebot3 equipped with a single mmWave radar inside a large laboratory environment and a defined track, and the second scenario using a custom robot equipped with three mmWave radars in an indoor corridor-like and room-like environment.

In the first scenario, the Turtlebot3 was able to navigate, localize, and map the environment using the single mmWave radar sensor. In this scenario, a ground truth of the map with a lidar sensor was taken to confront results. However, the results were not as accurate as expected, with the map not beeing that precise and with lot of noise. This deviation can be attributed to the limited field of view of the single mmWave radar sensor (especially in the open map scenario compared to the corridor-like scenario), the presence of many obstacles, and the absence of postprocessing of the mmWave data.

On the contrary, the second scenario using the custom robot equipped with three mmWave radar sensors yielded much better results. The robot was able to navigate the interior corridor environment with much greater accuracy and precision. This can be attributed to the increased field of view provided by the three mmWave radar sensors, which allowed for a more comprehensive mapping of the environment. Additionally, we implemented some filtering and data augmentation techniques to improve the performance of the system. These techniques improved the accuracy and robustness of the system, resulting in a more stable and reliable navigation.

In the end, our experiments have demonstrated that the use of multiple mmWave radar sensors for autonomous navigation improves the accuracy and robustness of the system. The results obtained in the second scenario, using a custom robot equipped with three mmWave radars, were particularly promising. We believe that this approach has the potential to be used in a wide range of applications and environments.

In the following section an in-depth view of all the experiments and results will be presented, with focus on the mapping and localization task, and also showing some map post-processing results and some experiments of autonomous

navigation using the on-line localization and mapping obtained with SLAM using only mmWave sensors.

5.1 Scenario 1: One sensor

In the first scenario of our experiments, we utilized the Turtlebot3 robot, which was equipped with a single mmWave sensor. The experimental setup is depicted in Figure 5.1. The area used for these experiments was a large, square open room with a variety of obstacles.

The purpose of these experiments was to compare the maps generated by the Turtlebot3's lidar sensor with those generated by a single mmWave sensor, placed in front of the robot.

Figure 5.1: Scenario of the experiment.

The results of the initial experiments, without any data filtering or enhancement, are shown in Figure 5.3. Despite the poor performance of the mmWave in this case, as seen in Figure 5.2, the mmWave sensor demonstrated its ability to detect obstacles such as glass doors and obstacles at height. This is particularly noteworthy as the lidar was unable to accurately detect the obstacle even when the glass door was covered with paper, while the mmWave sensor was able to do so. Furthermore, the mmWave sensor was able to detect obstacles in three dimensions, allowing it to detect obstacles at various heights, such as the table shown in Figure 5.2. The results of this comparison, shown in Figure 5.3 and Figure 5.2, demonstrated that even in an out-of-the-box configuration without

any enhancements or filtering, the mmWave sensor was capable of recognizing obstacles such as glass doors and objects at different heights that the lidar sensor was not able to detect accurately.

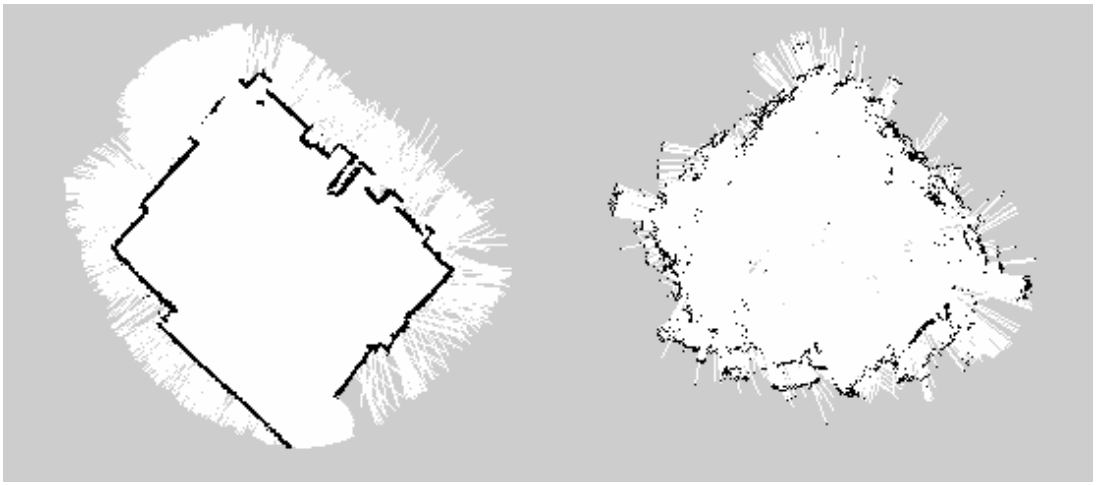

Figure 5.2: Cost-map of the lab environment taken with lidar sensor vs. cost-map of the lab environment taken with one mmWave sensor.

In Figure 5.4, a comparison of the maps generated by the lidar and mmWave sensors with data filtering and augmentation was presented. The results showed a significant improvement in the precision of the maps, making the results considered promising. It is also important to note that the mmWave sensor was able to see obstacles behind other objects, adding to its capability in mapping tasks.

In another experiment, a corridor-like path with a 90° turn was created in the lab, with the aim of creating an easier mapping task for the Turtlebot3. The scenario and the results are shown in Figure 5.5. In this case, the results clearly demonstrate that the map generated using a single mmWave sensor was far more precise than the map generated in an environment with many obstacles, as was the case in the previous experiment. This highlights the versatility of mmWave sensors in different environments and their potential for use in mapping tasks, especially in indoor scenarios with glass or smoke where lidar sensors can have difficulty detecting objects accurately.

The results of these experiments not only showed the potential for mmWave sensors in mapping tasks but also emphasized the importance of data filtering and enhancement in improving the accuracy of the maps generated. With the ability to detect obstacles and objects at different heights, as well as behind other obstacles, the mmWave sensor has proven to be a valuable addition to Turtlebot3's mapping capabilities.

These results also demonstrate the potential for mmWave sensors, especially in challenging environments such as indoor spaces with glass or smoke. The experiments show that the mmWave sensor can provide more accurate maps than the traditional lidar sensor, making it a valuable tool for various applications. The findings of these experiments serve as a stepping stone for further research

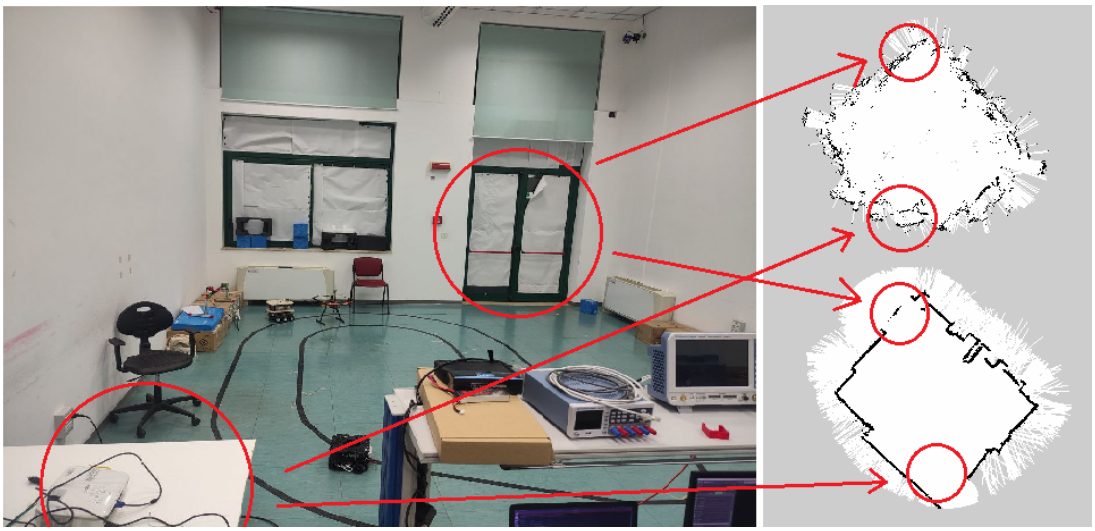

Figure 5.3: Map details: the mmWave sensor can see glass and obstacles at height.

and development of mmWave sensors for mapping tasks, and highlight the importance of data filtering and augmentation in improving the accuracy of the maps generated.

These experiments provide insight into the potential of mmWave sensors for use in SLAM. The mmWave sensor's ability to detect obstacles, even behind other objects, and to recognize objects at different heights, makes it a promising alternative or a companion to traditional lidar sensors in mapping tasks. The improvements in map precision seen with data filtering and enhancement further demonstrate the potential of the mmWave sensor for use in SLAM. The results of the corridor-like experiment also indicate that the mmWave sensor is versatile and can be used in different environments with varying levels of complexity. It is important to note that these experiments were performed with a single mmWave sensor. The addition of multiple mmWave sensors or the use of mmWave sensors in combination with other sensors could provide even greater improvements in mapping tasks. Further experiments and research will be necessary to fully understand the potential of mmWave sensors in SLAM and their ability to replace or augment traditional lidar sensors.

Overall, the results of these experiments demonstrate the potential of using a single mmWave sensor for SLAM tasks with the Turtlebot3 robot. The mmWave sensor was able to detect with accuracy obstacles that the lidar sensor was unable to recognize, and the maps generated with the mmWave sensor showed a significant improvement in precision even in cluttered environments. These experiments provide evidence of the ability of mmWave sensors to perform well in a variety of indoor scenarios, including those with glass or smoke, and highlights their potential for use in mapping tasks.

Further experiments shown in the next section with multiple mmWave sensors lead to even greater results. Additionally, the use of other data filtering and

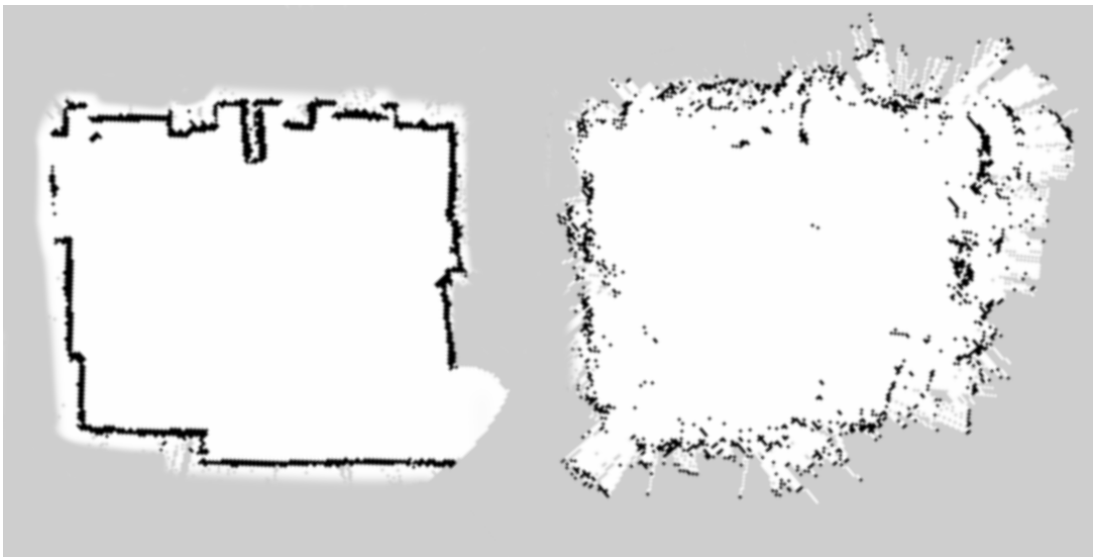

Figure 5.4: Cost-map of the laboratory environment taken with a lidar sensor vs. cost-map of the lab environment taken with one mmWave sensor, with data augmentation.

augmentation techniques could also further improve the precision and accuracy of the maps generated by the mmWave sensor. These results demonstrate the potential of mmWave sensors for use in SLAM tasks and pave the way for further exploration and development in this field.

Figure 5.5: Scenario of the experiment with walls and cost-map obtained with mmWave sensors.

5.2 Scenario 2: Multiple sensors

In this section, we present the results of another major set of series of experiments aimed at evaluating the performance of multi-sensor configurations using mmWave radars for autonomous navigation. The experiments were carried out in a real-world indoor environment, a corridor with a room at one end, with a width of 1.5 meters and a mapped length of more than 30 meters.

Figure 5.6: Scenario of the experiments: corridor

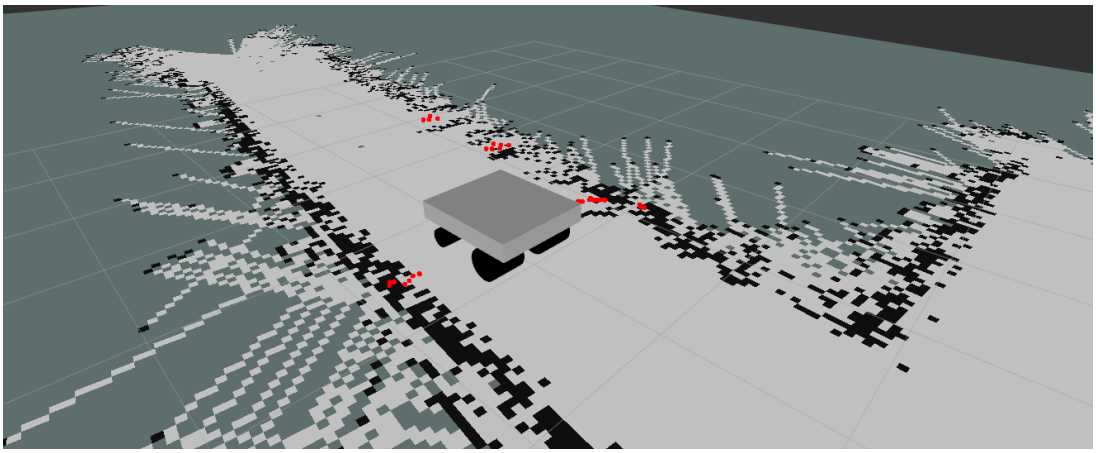

Figure 5.7: Example of the mapping process with multiple sensors in Rviz

The environment also included obstacles, glass doors, and other features that challenged the mapping and localization capabilities of the robot. For this series of experiments a custom robot was utilized that was equipped with 2 and later 3 mmWave sensors. In the initial phase, two sensors were placed on the robot, one on the right side and one on the left side. The goal of these experiments was to examine the capability of multi-millimeter wave sensors to create a more comprehensive map of the environment. The scenario in which all these experiments were conducted is shown in Figure 5.6. In the photos the corridor and the room that will be evident in the maps taken are shown from the view of the robot.

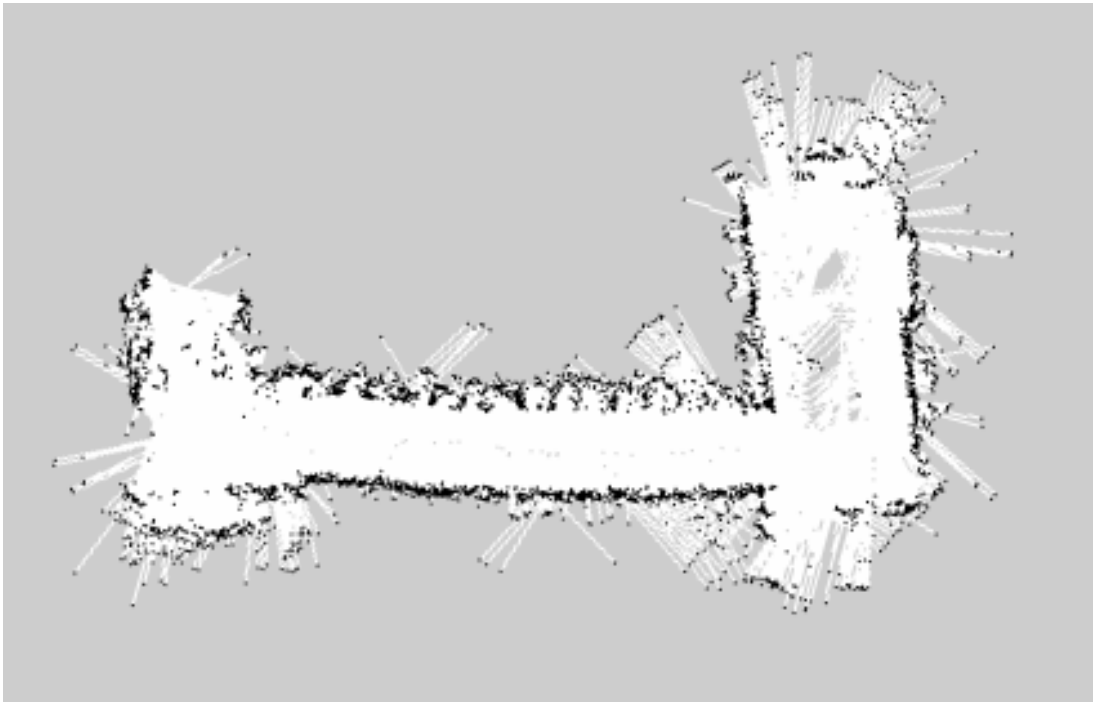

Figure 5.8: Map of the scenario taken with two sensors

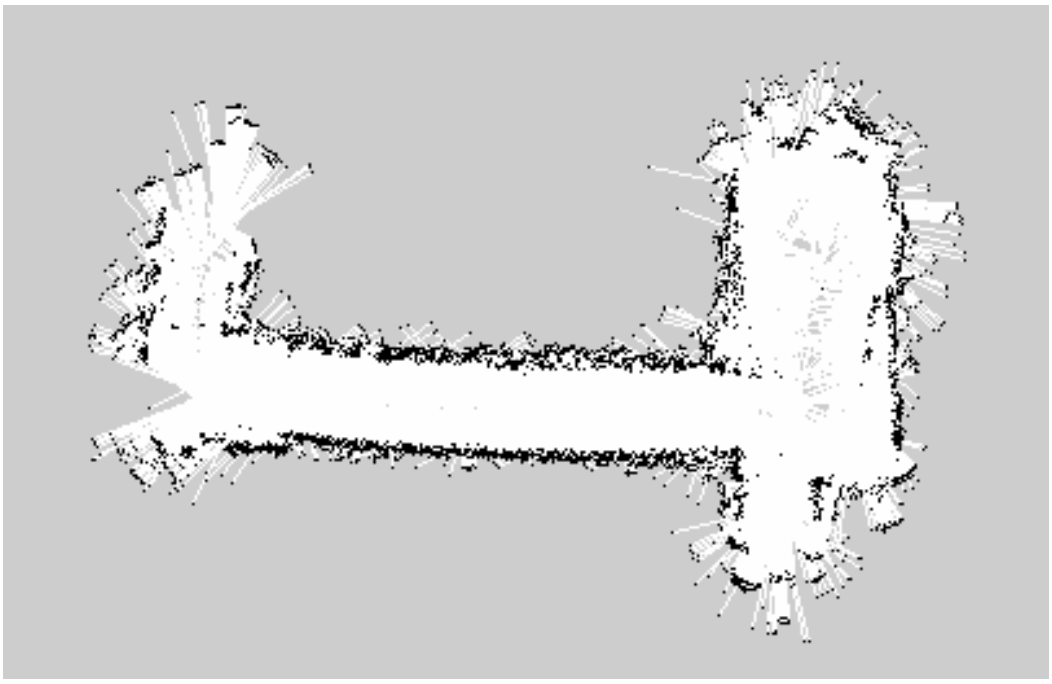

Figure 5.9: Map of the scenario taken with two sensors

It is a corridor with a room at one end, the corridor is 1.5 meter wide and the max length mapped is more then 30 meters. There were also some obstacles and glass doors in the corridor. In some parts the corridor expands in in width and then returns to 1.5 meters as we captured in maps shown later.

The environment in which the experiments where conducted was a classic indoor environment, but the experiments where conducted in long runs to better prove the accuracy of the mapping but especially the precision of the localization.

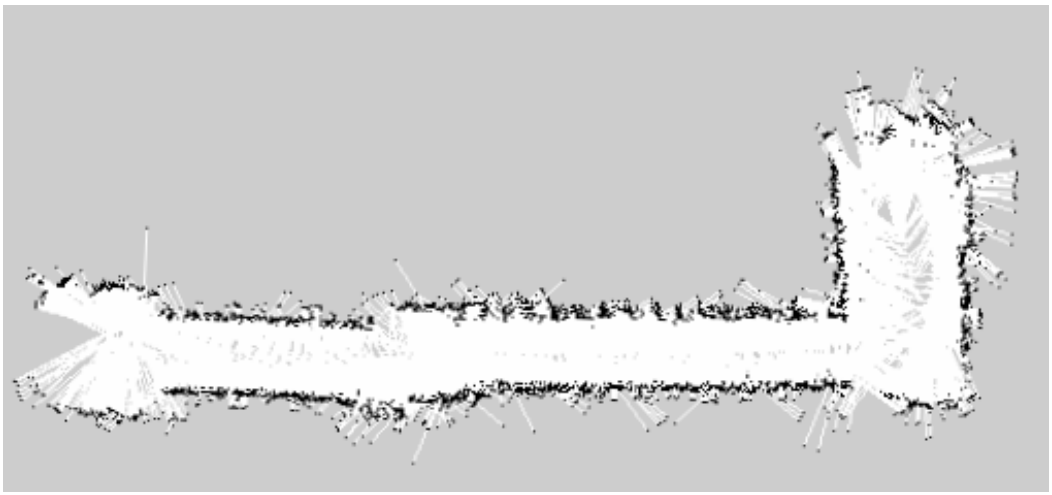

Figure 5.10: Map of the scenario taken with three sensors

It was important to prove that the system keeps the robot localized in the

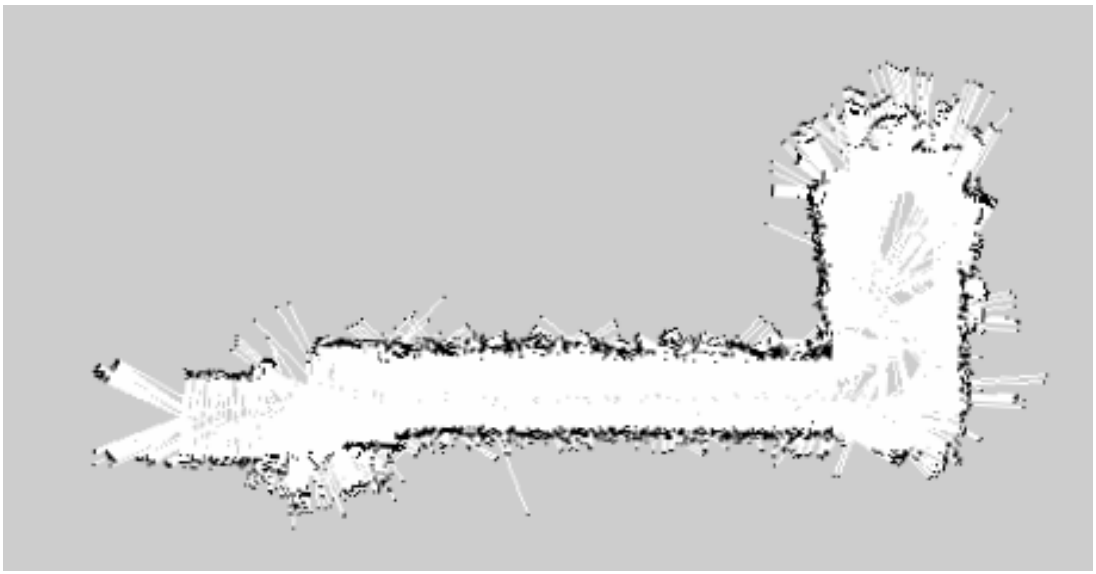

Figure 5.11: Map of the scenario taken with three sensors

environment even after a lot of time and movement, where the impact of the divergence of the encoder-based odometry is more relevant.

A first session of experiments was conducted using the `gmapping` package, with no filtering and no up-scaling, and just using two millimeter wave sensors. The two sensor where mounted one on the left and one on the right of the robot. Maps taken using this setup are shown in Figure 5.8 and Figure 5.9.

As can be seen in Figure 5.8 the use of two mmWave sensors allowed greater coverage of the environment, providing a more detailed map of obstacles and surroundings compared to the results obtained with one sensor shown before.

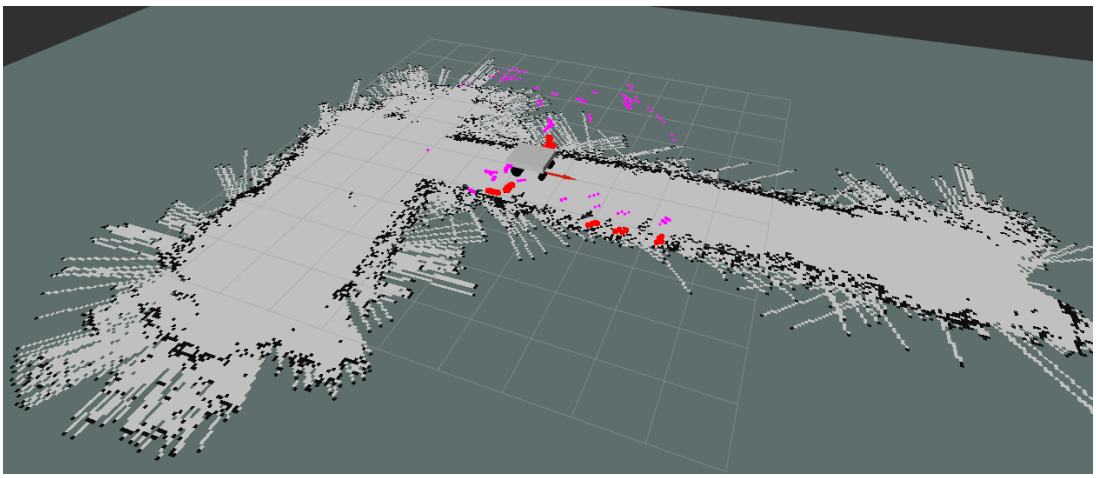

Figure 5.12: Robot localizing itself in the known map using mmWave sensor data

In more details, the map generated using two millimeter-wave (mmWave) sensors presents a higher level of detail and accuracy in terms of environment

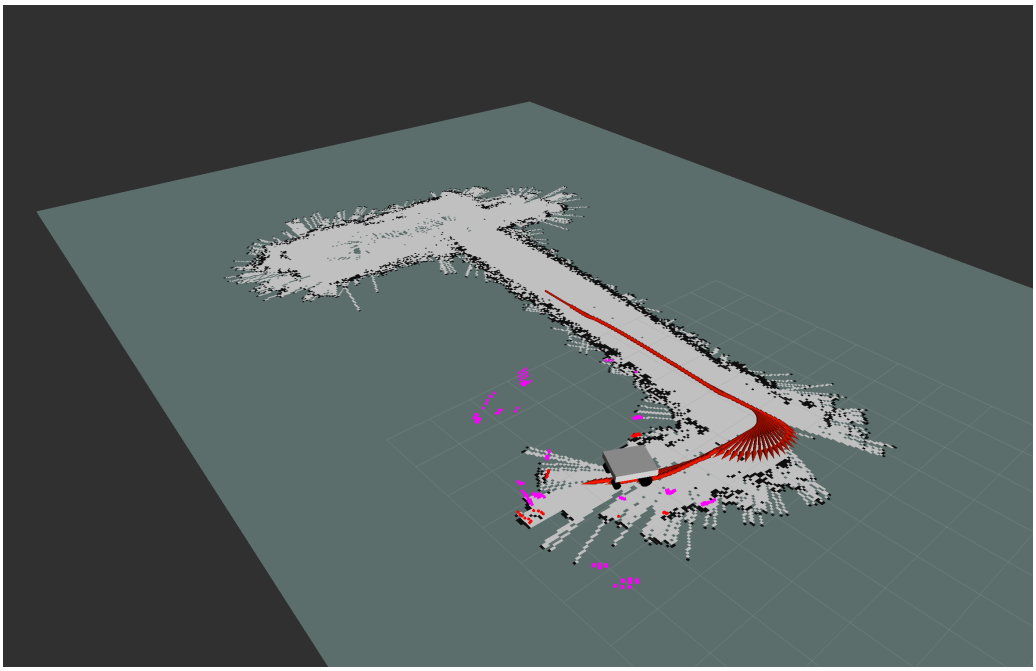

Figure 5.13: Example of the mapping process with multiple sensors in Rviz, with odometry vector from the encoders corrected by the SLAM algorithm shown in red.

mapping in the environment shown. The part of the corridor that appears to be less uniform in the map is likely due to the presence of a wall made of drywall. This material may have caused interference with the mmWave sensors, resulting in less precise measurements and less uniform representation of the environment.

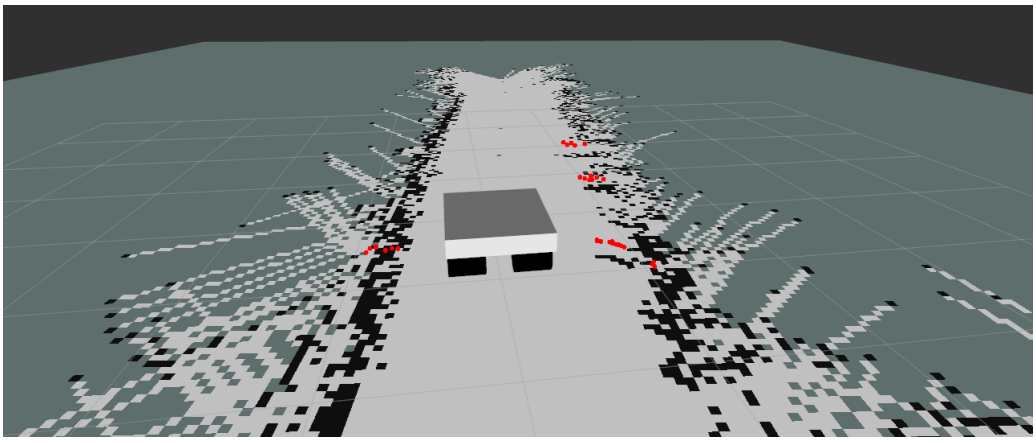

Figure 5.14: Example of the mapping process with multiple sensors in Rviz

However, with the integration of additional sensors and the use of up-scaling and data filtering techniques, there is significant room for improvement in the accuracy and uniformity of the map.

The use of multiple sensors allows for a more comprehensive view of the

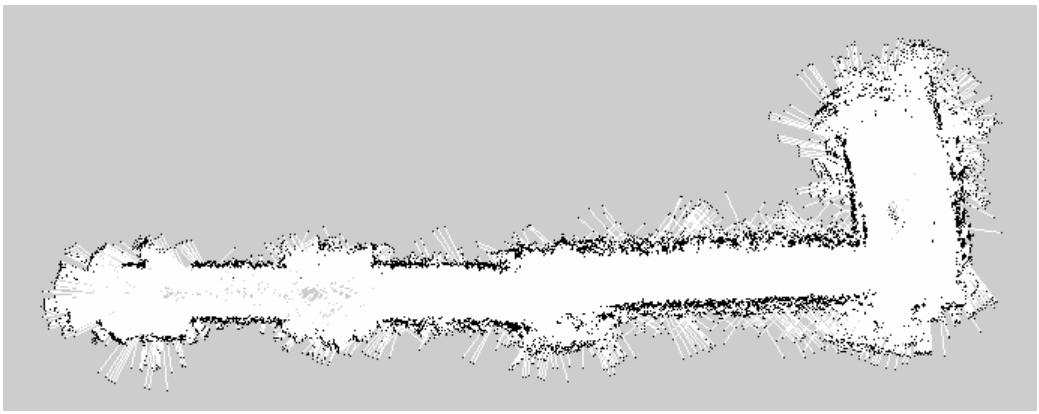

Figure 5.15: Map obtained with data augmentation and filtering using three sensors

environment, reducing the risk of missing important features and at the same time making it possible to map the environment and localize in it in a faster, and more importantly, more reliable way. Advanced data filtering and upscaling techniques enable an effective reduction of noise and increase the resolution of the map, leading to improved accuracy and detail.

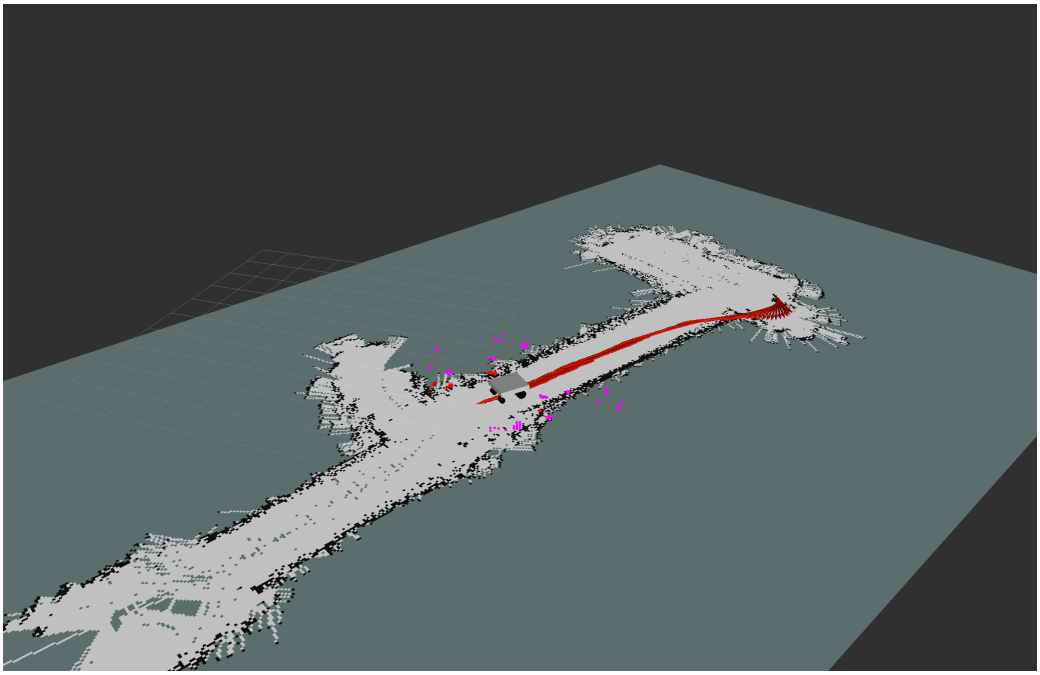

Figure 5.16: Example of the mapping process with multiple sensors in Rviz with odometry vector in red

As an example, the map reported in Figure 5.11 and Figure 5.10 are taken always in the same scenario but just adding the third mmWave sensor on the front of the robot. This sensor proved to be necessary even if obstacles are on the sides of the robot. In fact, the addition of a third mmWave sensor further

improved the mapping capabilities of the robot, and especially the localization capability of the robot.

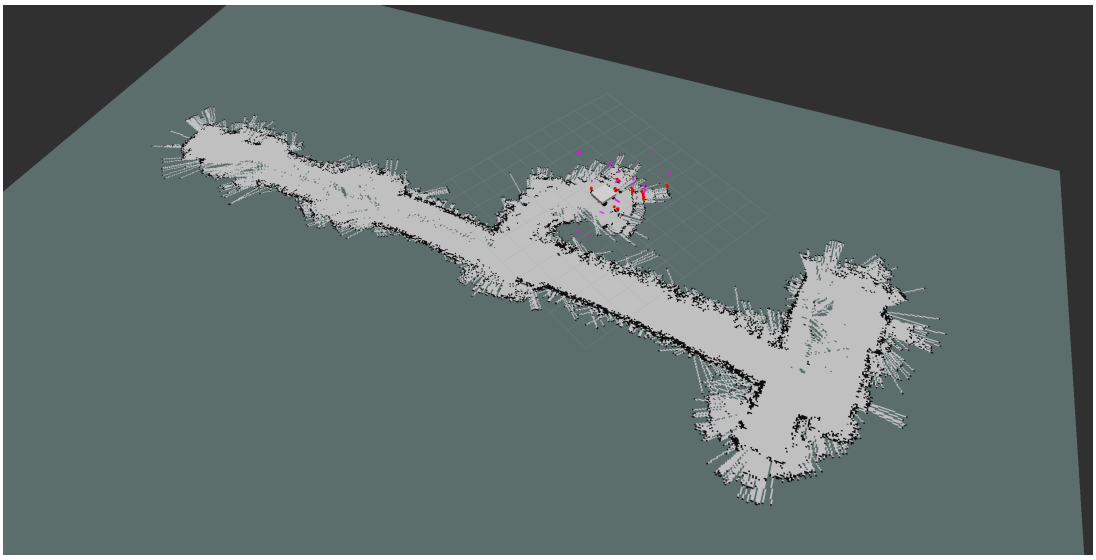

Figure 5.17: Example of the mapping process with multiple sensors in Rviz

It is important to say that the mapping experiments in those cases are notably faster compared with the experiments with just one sensor or even two and the time taken for those.

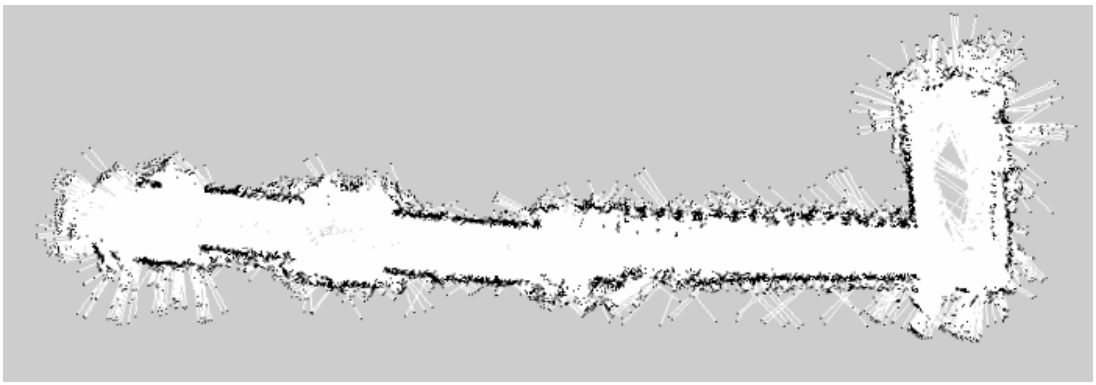

Figure 5.18: Map obtained with data augmentation and filtering using three sensors

It is also worth to note that it isn't still as fast as a lidar. In the experiments with three sensors the robots just needs to explore the environment one time to produce the maps shown in Figure 5.10 and Figure 5.10. In those experiments it was also evidently improved the capability of the SLAM algorithm in making it possible for the robot to localize itself in a know map. This is show in Figure 5.12.

The robot spawns in a random place on the known map and in some time end with some movements (depending on the position and the homogeneity or

heterogeneity of the environment) around it is able to correctly localize itself. In Figure 5.13 and Figure 5.16 in red arrows is shown the odometry computed by the encoder sensors on the wheels. It is easy to see that the SLAM algorithm corrects the position computed by the odometry with the Map matching of the Bayesian filter of the SLAM algorithm, making it possible to localize and correct odometry in a very good way. In Figure 5.16 it is shown how the odometry with no correction computed a localization and movement of the robot that evidently has errors how is normal to expect for a dead-reckoning localization system.

In practice, the SLAM algorithm creates the initial map of the environment and then continuously updates the map as the robot moves through it. At each time step, the odometry information is combined with the mmWave sensor data to update the estimate of the robot's location. This estimate is then compared to the map to determine the most likely location of the robot, which is then corrected and refined by the Bayesian filter. The Bayesian filter combines the information from the odometry and mmWave sensors to make a probabilistic estimate of the robot's location. This filter takes into account both the uncertainty in the odometry information and the accuracy of the mmWave sensor data to provide a robust estimate of the robot's location. The result of this process explains how the corrected estimate of the robot's location is significantly more accurate than the estimate based solely on the odometry information, as shown in Figure 5.13 and in Figure 5.16.

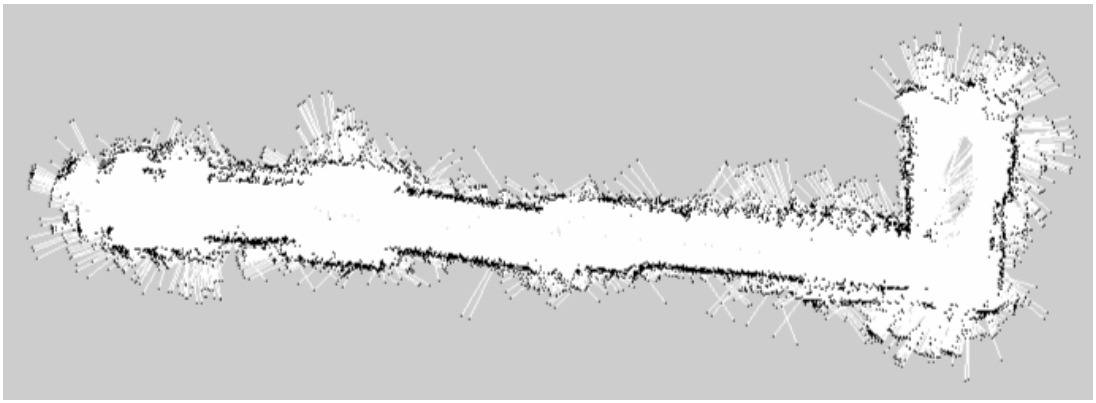

Figure 5.19: Map obtained with data augmentation and filtering using three sensors

In the end the experiments where conducted with the complete system implemented. All three mmWave where activated and data from the sensors was opportunely filtered and up-sampled with the custom clustering algorithm discussed in 3. Maps obtained from the last experiments are shown in Figure 5.15, Figure 5.18 and Figure 5.18. These last results are really promising, the system is capable of mapping the entire environment without losing the precise localization, and the maps are even more clear and precise. The three maps shown in the Figure where taken in different runs of the experiments, with different starting point and different SLAM implementations or tuning, and the results shows that the tool-chain developed for the purpose and clustering algorithm

implemented are a reliable and well grounded base system to implement SLAM using mmWave radar.

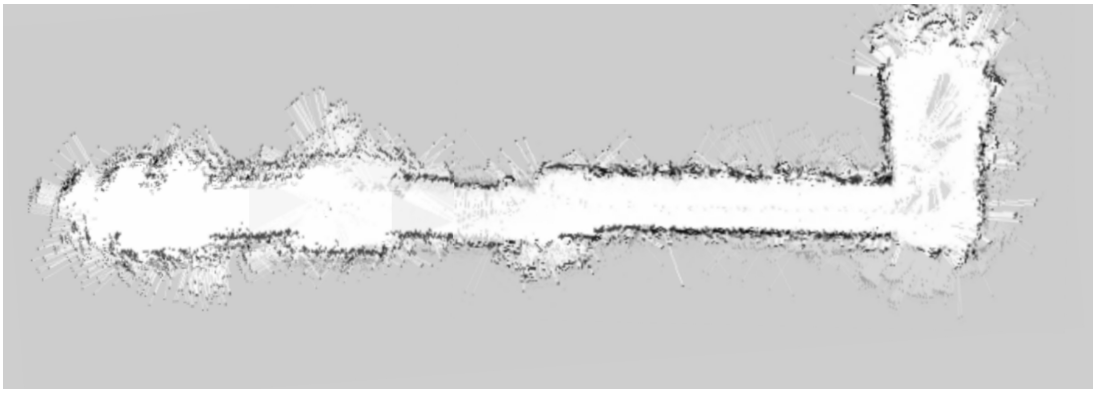

Figure 5.20: Overlapped maps

To further explore the reliability of the system all map taken have been overlapped to show the stability and steadiness of the implementations. This is shown in Figure 5.20. The Figure shows clearly how the system correctly mapped the environment in each experiment, proving its reliability.

The results of these experiments indicate the potential for mmWave sensors to be used effectively in multi-sensor configurations for mapping tasks. The improved mapping capabilities of the robot demonstrate the benefits of using multiple sensors in terms of increased coverage and accuracy of the generated maps. These results suggest that further development and experimentation with multi-sensor configurations using mmWave sensors could lead to even greater improvements in the mapping capabilities of robots in various environments.

5.3 Autonomous navigation

To further explore the capability of the SLAM implementation we developed, in this section of the thesis we present the results of a series of experiments that were conducted to demonstrate the ability of the system to enable autonomous navigation of a robot. The algorithm used the Dynamic Window Approach (DWA) for path planning as discussed before and was able to successfully localize the robot and find a path to the goal in multiple experiments.

In order to conduct the experiments, the robot described in previous sections equipped with three mmWave sensors and encoders was used.

The experiments were conducted in the same indoor environment as the experiments discussed in the previous section. The environment was an indoor environment with a variety of obstacles and different layouts. The goal of the experiments was to test the ability of the SLAM algorithm to localize the robot and find a path to the goal in different scenarios.

For these experiments, as shown in Figure 5.21, we used Rviz to assign goal to the planner. In the Figure, in green in shown the planned path calculated by the DWA planner. To make the robot move based on the velocities computed by the `/move_base` node, we also implemented a custom driver based on the kinematics of the robot, the driver is a ROS node that subscribes to the `/cmd_vel` topic and converts the velocities provided by the `move_base` node to RPM set-point for the wheel motors.

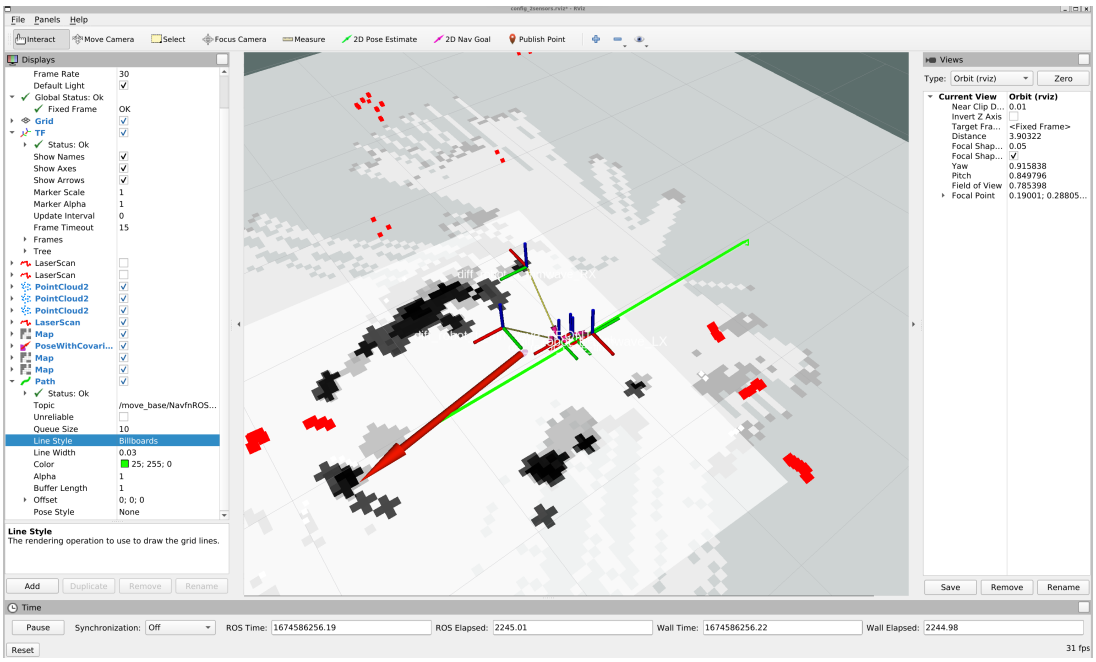

Figure 5.21: Autonomous navigation on Rviz with planned path in green

The results of the experiments showed clearly that the SLAM algorithm implemented with mmWave sensors was able to successfully localize the robot and find a path to the goal in all of the experiments. The algorithm was able to correct the robot odometry, computed via the encoders, by incorporating the range information from the mmWave sensors. This allowed the algorithm to accurately estimate the position of the robot in the environment, even in the presence of obstacles.

Furthermore, the DWA algorithm was able to find a safe and efficient path to the goal. The algorithm took into account the range information collected by the mmWave sensors to avoid obstacles and plan a path that was optimized for speed and safety. This allowed the robot to navigate to the goal without hitting any obstacles or taking inefficient routes. An example can be seen in Figure 5.22, which shows an Rviz representation of the path planned by the DWA algorithm in green and the obstacles obtained by mapping and sensing the environment.

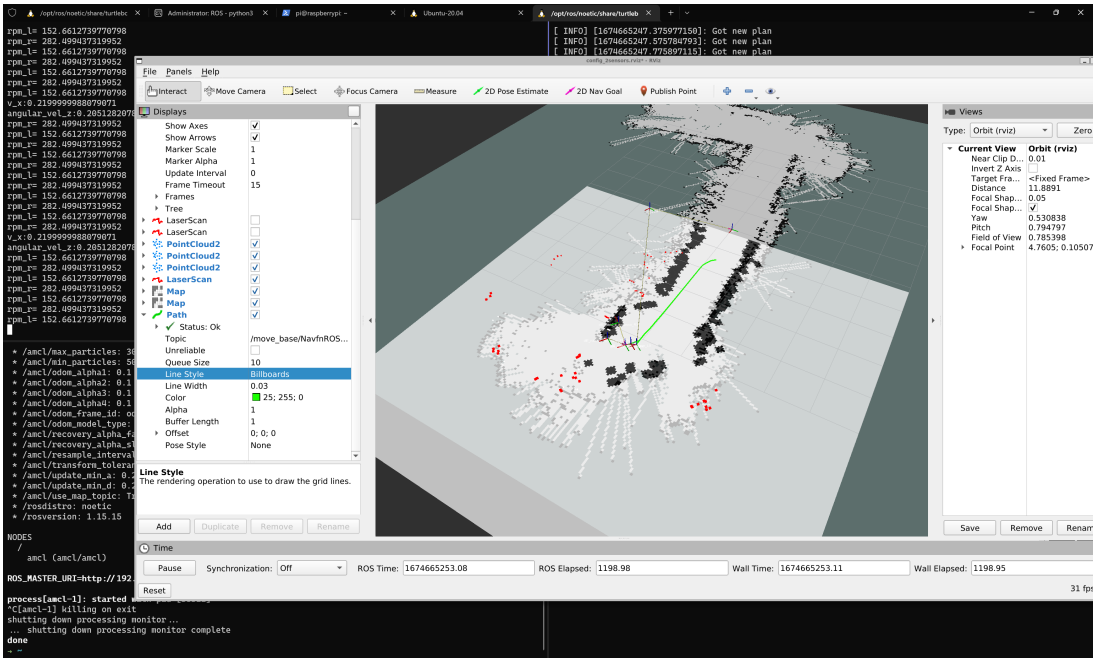

Figure 5.22: Autonomous navigation on Rviz with planned path in green

In conclusion, the experiments demonstrated the ability of the SLAM algorithm with mmWave sensors to enable autonomous navigation of a robot. The algorithm was able to accurately localize the robot and find a safe and efficient path to the goal in multiple experiments. This highlights the potential of mmWave sensors for use in autonomous navigation systems and provides a foundation for future research in this area.

5.4 Map post-processing

Autonomous navigation systems rely on the accuracy of the map generated by the SLAM algorithm. In order to improve the map accuracy, several techniques have been developed, in particular using different machine learning techniques such as Contrastive Learning or Auto-Encoders. In this section, we will evaluate the performance of map post-processing using a Contrastive Learning based machine learning technique.

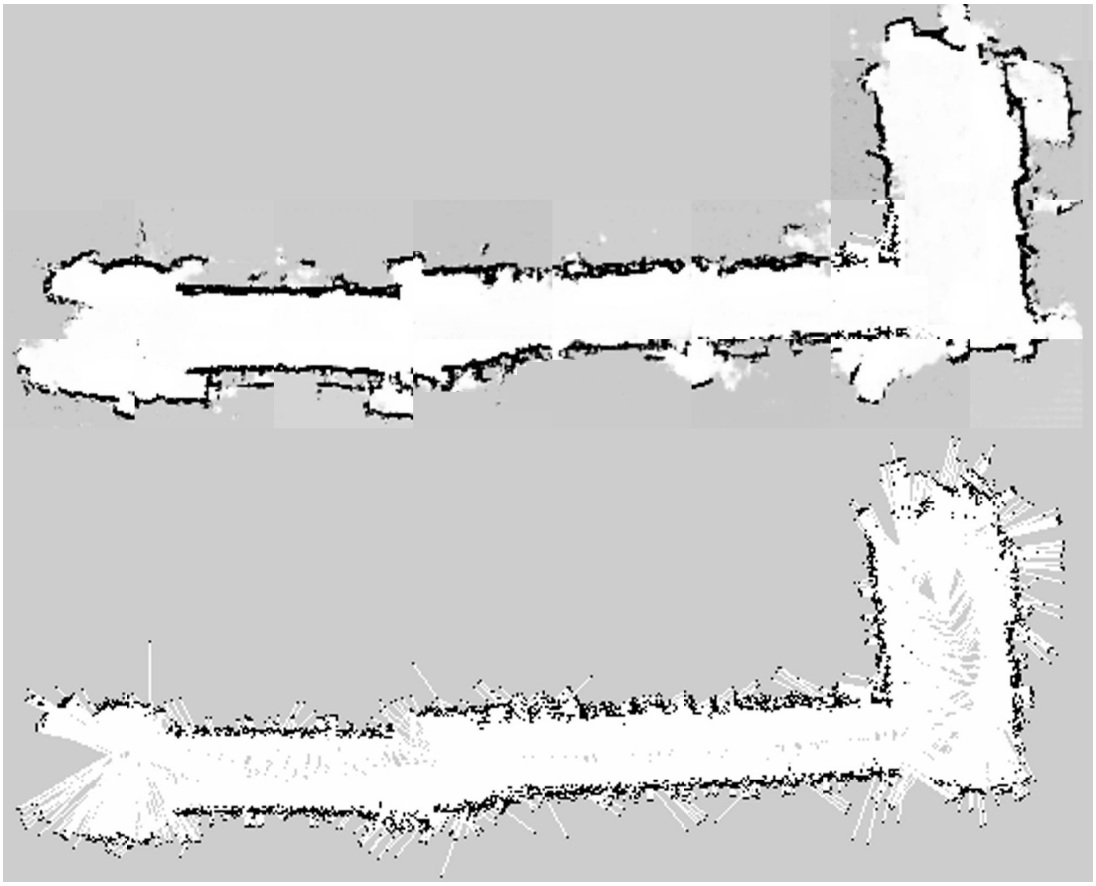

Figure 5.23: Map obtained with contrastive learning compared to real-time map

In a contrastive learning approach, the system compares the current map to a known reference map and corrects any differences. This approach has previously been presented in several papers[31], where the results seemed promising. However, the actual results shown in Figure 5.23 and Figure 5.24, still require a lot of work to achieve the desired level of precision.

In our experiments, we used comparative learning to correct the map generated by the SLAM algorithm. The system was trained using a dataset of maps collected from multiple experiments conducted in [31] and others such as

[26]. The training process consisted of comparing the maps generated by the SLAM algorithm with the reference maps and updating the system parameters to minimize the differences between the two maps.

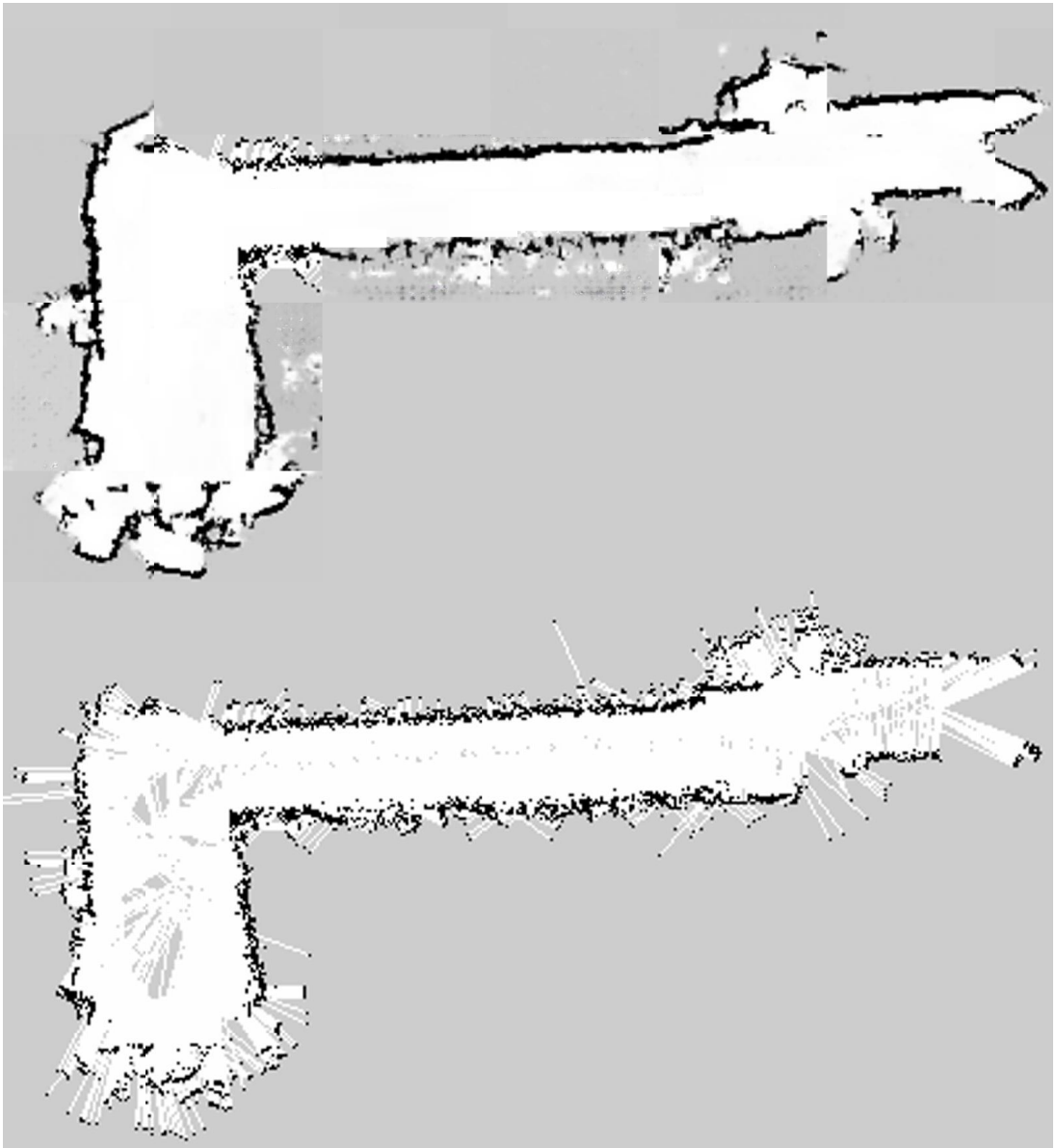

Figure 5.24: Map obtained with contrastive learning compared to real-time map

The results of our experiments showed that the map post-processing using comparative learning was able to improve the accuracy of the map generated by the SLAM algorithm. However, the improvement was not as significant as expected and more work is needed to achieve the desired level of accuracy. This may be due to the limitations of the SLAM algorithm, the complexity of the environment, and the limitations of the comparative learning algorithm.

In particular, as can be seen in the presented Figures, in our evaluation of the machine learning algorithm for map post-processing, we have found that it holds

promise for enhancing the quality of the maps produced by our autonomous navigation system. However, the current implementation does not meet the reliability requirements for real-time application.

Despite these limitations, we believe that this approach has significant potential and that further improvements can be made to increase its reliability. For instance, the model could be further trained to better recognize different environments and to improve its ability to process the costmap information, which only reports three values (grey for unknown, black for occupied, and white for empty). Modifying the image processing technique to better accommodate these limitations could improve the overall performance of the system.

In the end, While the results of our experiments were encouraging, there is still a lot of work to be done to achieve the desired level of accuracy in map post-processing using comparative learning. More research is needed to improve the SLAM algorithm, develop better algorithms for comparative learning, and collect more data to train the system. Additionally, the hardware used in the experiments should be optimized to achieve the best possible performance. With these improvements, we believe that map post-processing using comparative learning has the potential to significantly improve the accuracy of the maps generated by the SLAM algorithm, thus improving the overall performance of autonomous navigation systems.

5.5 Final results

The analysis of the results of the experiments and navigation, as well as the post-processing of maps, has revealed the potential and limitations of the system developed in this study.

In terms of SLAM, the robot was able to successfully provide high-accuracy maps in fast time and also provide a reliable real time localization. In autonomous,navigation the system was able find a path using the Dynamic Window Approach and reach its goal while localizing itself within the environment. The use of millimeter wave sensors and a SLAM algorithm were key in correcting the odometry of the robot computed via encoders, allowing for more accurate localization.

In terms of map post-processing, the approach using contrastive learning showed promising results. However, in practice, further work is still needed to fully realize the potential of this approach. This highlights the ongoing nature of research in the field of autonomous navigation and map post-processing.

The hardware used in this system included millimeter wave sensors, encoders, and a powerful processor to handle the computational demands of the algorithms. The performance of the system was deemed satisfactory, but there is room for improvement in terms of speed and accuracy.

In conclusion, this study highlights the potential for using millimeter wave sensors and machine learning techniques in the field of autonomous navigation. While the results are encouraging, there is still much work to be done in order to fully realize the potential of these technologies. Further research and development will be necessary in order to improve the performance and accuracy of the system, and to address the limitations that have been revealed through this study.

Chapter 6

Conclusions

In this master thesis, the use of multiple millimeter wave radars for simultaneous localization and mapping (SLAM) was investigated as a means to improve the performance of autonomous navigation systems. The thesis presented a novel algorithm for SLAM that leveraged the unique characteristics of millimeter wave radar to improve accuracy and robustness.

Experimental results demonstrated the effectiveness of the proposed algorithm in various environments, including urban and indoor scenarios. The algorithm was implemented on an autonomous platform and the real-time performance was shown to be suitable for autonomous navigation.

The use of multiple millimeter wave radars was shown to provide significant improvements in terms of accuracy and robustness compared to the use of a single radar. Furthermore, the integration of multiple millimeter wave radars with other sensor modalities, such as cameras and lidar, was shown to result in more robust and reliable autonomous navigation systems.

The results of this research provide valuable insight into the potential of multiple millimeter wave radar in SLAM, and open up new possibilities for the development of autonomous navigation systems. However, it is acknowledged that there is still room for improvement in terms of specific challenges, such as dealing with dynamic objects or occlusions. Therefore, it is recommended that future research continue to explore the use of multiple millimeter wave radar in SLAM, incorporating new techniques such as deep learning for enhanced performance.

In conclusion, this research provided a novel algorithm for SLAM based on multiple millimeter wave radars, which demonstrated superior performance and real-time capabilities. It contributes to the growing field of autonomous navigation and opens the door to further research on this topic.

In addition to the contributions outlined above, this thesis also highlighted the potential of millimeter wave radar technology in the context of autonomous navigation. Millimeter wave radar has unique characteristics such as high resolution, robustness to environmental conditions, and the ability to penetrate some non-metallic objects. These characteristics make it an attractive sensor modality for SLAM and autonomous navigation applications.

Furthermore, the research reported in this thesis demonstrates the feasibility of using multiple millimeter wave radars for SLAM, which has the potential to overcome some of the limitations of using a single radar. This is particularly important in environments with dynamic objects, occlusions, or other challenges that can degrade the performance of a single radar-based SLAM system.

In general, this thesis represents an important step forward in the use of millimeter wave radar technology for SLAM and autonomous navigation. It contributes to the understanding of the capabilities and limitations of this technology and provides a foundation for further research in this area. With the rapid development of millimeter wave radar and other sensor technologies, it is expected that autonomous navigation systems will continue to improve in terms of performance and functionality in the future.

6.1 Future works

In conclusion, this thesis has presented research on the application of SLAM using mmWave radars for autonomous navigation. The results are promising and solid. However, there is still ample room for further research in order to improve the performance of the proposed method.

One area of future research is exploring different SLAM algorithms using different filters and data types. The gmapping package relies on the use of a laser rangefinder to acquire range data, but other SLAM algorithms may use pointcloud data directly, without the need for conversion. This could lead to improvements in the accuracy and efficiency of the SLAM process.

Another area of focus is to improve data augmentation and filtering techniques. In particular, techniques such as clustering and machine learning could be used to better filter and process the sensor data, resulting in more accurate and reliable maps of the environment.

Sensor fusion techniques could also be investigated to combine the mmWave radar data with other types of pointcloud or laserscan data from other sensors such as lidars, cameras, and other sensing devices. This could lead to better performance by combining the strengths of multiple sensor modalities.

Additionally, a 3D SLAM algorithm could be developed that uses point-cloud data to build 3D maps of the environment. This would enable the use of the proposed method in more complex and dynamic environments.

Lastly, the proposed method could be tested in an autonomous navigation scenario using the Adaptive Monte Carlo Localization (AMCL) algorithm. This would allow one to evaluate the performance of the proposed method in real-world scenarios and compare it to other existing methodologies.

Overall, these future research directions hold great promise for advancing the state of the art in the field of SLAM using mmWave radars for autonomous navigation

and for enabling more effective and efficient autonomous navigation in various environments.

Appendix

gmapping params

```
map_update_interval: 0.5
maxUrange: 20.0
sigma: 0.05
kernelSize: 1
lstep: 0.05
astep: 0.05
iterations: 5
lsigma: 0.075
ogain: 3.0
lskip: 0
minimumScore: 10000
srr: 0.1
srt: 0.2
str: 0.1
stt: 0.2
linearUpdate: 0.2
angularUpdate: 0.2
temporalUpdate: 0.5
resampleThreshold: 1
particles: 100
xmin: -10.0
ymin: -10.0
xmax: 10.0
ymax: 10.0
delta: 0.05
llsamplerange: 0.01
llsamplestep: 0.01
lasamplerange: 0.005
lasamplestep: 0.005
```

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor Prof. Saverio Mascolo and my co-supervisor Vito Andrea Racanelli.

They have provided me with invaluable guidance, support, and encouragement throughout my research journey. Their expertise, patience, and unwavering belief in my abilities have been a source of inspiration and motivation. I am grateful for their contributions and insight into my research. Their feedback and suggestions have been instrumental in shaping my work into its final form.

I would like to extend my appreciation to all the people at the C3 Lab for their support, camaraderie, and for staying with me until late hours to help me with my research. Their presence and assistance have made my time in the lab a truly memorable and enriching experience.

Finally, I would like to express my deepest gratitude to my family for their love, support, and understanding. Without their constant encouragement, sacrifices, and belief in me, this journey would not have been possible. This achievement is a testament to their unwavering love and support, and I will be forever grateful to them.

I am immensely grateful to all of the above-mentioned individuals and entities who have been an integral part of my research journey and have made this thesis possible.

Bibliography

[1] Yasin Almalioglu et al. "Milli-RIO: Ego-Motion Estimation With Low-Cost Millimetre-Wave Radar". In: *IEEE Sensors Journal* 21.3 (2021), pp. 3314–3323. DOI: 10.1109/JSEN.2020.3023243. URL: https://doi.org/10.1109/JSEN.2020.3023243.

[2] *AMT10 Series Modular Incremental Rotary Encoders - cuidevices.com*. https://www.cuidevices.com/product/motion-and-control/rotary-encoders/incremental/modular/amt10-series. [Accessed 22-Jan-2023].

[3] *AWR1843AOPEVM millimeter wave sensor by Texas Instruments*. URL: https://www.ti.com/tool/AWR1843AOPEVM.

[4] Augusto Luis Ballardini et al. *"ira_laser_tools": a ROS LaserScan manipulation toolbox*. 2014. DOI: 10.48550/ARXIV.1411.1086. URL: https://arxiv.org/abs/1411.1086.

[5] Carlos Baquero Barneto et al. "Millimeter-Wave Mobile Sensing and Environment Mapping: Models, Algorithms and Validation". In: *IEEE Transactions on Vehicular Technology* 71.4 (2022), pp. 3900–3916. DOI: https://doi.org/10.1109/TVT.2022.3146003.

[6] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. "The Quickhull Algorithm for Convex Hulls". In: *ACM Trans. Math. Softw.* 22.4 (Dec. 1996), pp. 469–483. ISSN: 0098-3500. DOI: 10.1145/235815.235821. URL: https://doi.org/10.1145/235815.235821.

[7] Keenan Burnett et al. "Are We Ready for Radar to Replace Lidar in All-Weather Mapping and Localization?" In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10328–10335. DOI: https://doi.org/10.1109/LRA.2022.3192885.

[8] Sarah H. Cen and Paul Newman. "Precise Ego-Motion Estimation with Millimeter-Wave Radar Under Diverse and Challenging Conditions". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6045–6052. DOI: 10.1109/ICRA.2018.8460687. URL: https://doi.org/10.1109/ICRA.2018.8460687.

[9] Yuwei Cheng et al. "A Novel Radar Point Cloud Generation Method for Robot Environment Perception". In: *IEEE Transactions on Robotics* 38.6 (Dec. 2022), pp. 3754–3773. DOI: 10.1109/tro.2022.3185831. URL: https://doi.org/10.1109/tro.2022.3185831.

[10] Sebastiano Chiodini, Marco Pertile, and Stefano Debei. "Occupancy grid mapping for rover navigation based on semantic segmentation". In: *ACTA IMEKO* 10.4 (2021), pp. 155–161. DOI: 10.21014/acta_imeko.v10i4.1144. URL: https://doi.org/10.21014/acta_imeko.v10i4.1144.

[11] Howie Choset et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. English. MIT Press, May 2005. ISBN: 0262033275. DOI: 10.1017/s0269888907218016. URL: https://doi.org/10.1017/s0269888907218016.

[12] S. Clark and G. Dissanayake. "Simultaneous localisation and map building using millimetre wave radar to extract natural features". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 2. 1999, 1316–1321 vol.2. DOI: https://doi.org/10.1109/ROBOT.1999.772543.

[13] M.W.M.G. Dissanayake et al. "A solution to the simultaneous localization and map building (SLAM) problem". In: *IEEE Transactions on Robotics and Automation* 17.3 (June 2001), pp. 229–241. DOI: 10.1109/70.938381. URL: https://doi.org/10.1109/70.938381.

[14] D. Fox, W. Burgard, and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics and Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977. URL: https://doi.org/10.1109/100.580977.

[15] Brian P Gerkey and Kurt Konolige. "Planning and control in unstructured terrain". In: *ICRA workshop on path planning on costmaps*. Citeseer. 2008. URL: https://citeseerx.ist.psu.edu/document?doi=dabdbb636f02d3cff3d546bd1bdae96a058ba4bc.

[16] *GitHub - radar-lab/ti_mmwave_rospkg: TI mmWave radar ROS driver (with sensor fusion and hybrid) — github.com*. https://github.com/radar-lab/ti_mmwave_rospkg. [Accessed 16-Jan-2023].

[17] Alessandro Giua and Carla Seatzu. *Analisi dei sistemi dinamici*. Springer Milan, 2009. ISBN: 9788847014831. DOI: 10.1007/978-88-470-1484-8. URL: https://doi.org/10.1007/978-88-470-1484-8.

[18] Bruno Gouveia et al. "Computation Sharing in Distributed Robotic Systems: a Case Study on SLAM". In: *IEEE Transactions on Automation Science and Engineering* 12 (Dec. 2014). DOI: 10.1109/TASE.2014.2357216. URL: https://doi.org/10.1109/TASE.2014.2357216.

[19] G. Grisetti et al. "A Tutorial on Graph-Based SLAM (vol 2, pg 31, 2010)". In: *IEEE Intelligent Transportation Systems Magazine* 7 (Dec. 2010), pp. 4–4. DOI: 10.1109/MITS.2015.2475018. URL: https://doi.org/10.1109/MITS.2015.2475018.

[20] Bernhard Großwindhager. "The Potential of Low-cost Millimeter-Wave Sensors for Mobile Radar Applications". In: *Tc* 100.2B (2019), 2B. URL: https://grosswindhager.com/pubs/grosswindhager2019mmwave.pdf.

[21] Ziyang Hong et al. "RadarSLAM: A robust simultaneous localization and mapping system for all weather conditions". In: *The International Journal of Robotics Research* 41.5 (Apr. 2022), pp. 519–542. DOI: 10.1177/02783649221080483. URL: https://doi.org/10.1177/02783649221080483.

[22] Jui-Te Huang et al. "Cross-Modal Contrastive Learning of Representations for Navigation using Lightweight, Low-Cost Millimeter Wave Radar for Adverse Environmental Conditions". In: (2021). DOI: 10.48550/ARXIV.2101.03525. URL: https://arxiv.org/abs/2101.03525.

[23] Cesar Iovescu and Sandeep Rao. "The fundamentals of millimeter wave radar sensors". In: *Texas Instruments* (2020). URL: https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf.

[24] Alonzo Kelly. *An intelligent predictive controller for autonomous vehicles*. Tech. rep. Carnegie-Mellon Univ. Pittsburg PA Robotics Inst., 1994. URL: https://www.ri.cmu.edu/pub_files/pub1/kelly_alonzo_1994_7/kelly_alonzo_1994_7.pdf.

[25] Stefan Kohlbrecher et al. "A flexible and scalable SLAM system with full 3D motion estimation". In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. 2011, pp. 155–160. DOI: 10.1109/SSRR.2011.6106777. URL: https://doi.org/10.1109/SSRR.2011.6106777.

[26] Andrew Kramer et al. "ColoRadar: The direct 3D millimeter wave radar dataset". In: *The International Journal of Robotics Research* 41.4 (Mar. 2022), pp. 351–360. DOI: 10.1177/02783649211068535. URL: https://doi.org/10.1177/02783649211068535.

[27] Jinho Lee et al. "GAN-Based LiDAR Translation between Sunny and Adverse Weather for Autonomous Driving and Driving Simulation". In: *Sensors* 22.14 (2022), p. 5287. DOI: 10.3390/s22145287. URL: https://doi.org/10.3390/s22145287.

[28] Yang Li et al. "The Millimeter-Wave Radar SLAM Assisted by the RCS Feature of the Target and IMU". In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185421. URL: https://www.mdpi.com/1424-8220/20/18/5421.

[29] Sohee Lim et al. "Radar-Based Ego-Motion Estimation of Autonomous Robot for Simultaneous Localization and Mapping". In: *IEEE Sensors Journal* 21.19 (2021), pp. 21791–21797. DOI: https://doi.org/10.1109/JSEN.2021.3101893.

[30] Chris Xiaoxuan Lu et al. "MilliEgo: Single-Chip MmWave Radar Aided Egomotion Estimation via Deep Sensor Fusion". In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. SenSys '20. Virtual Event, Japan: Association for Computing Machinery, 2020, pp. 109–122. ISBN: 9781450375900. DOI: 10.1145/3384419.3430776. URL: https://doi.org/10.1145/3384419.3430776.

[31] Chris Xiaoxuan Lu et al. "See through smoke". In: *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. ACM, June 2020. DOI: 10.1145/3386901.3388945. URL: https://doi.org/10.1145/3386901.3388945.

[32] P. C. Mahalanobis. "On the Generalized Distance in Statistics." In: *Proceedings of National Institute of Sciences (India)* 2.1 (1936), pp. 49–55.

[33] Federico Nardi et al. "Generation of Laser-Quality 2D Navigation Maps from RGB-D Sensors". In: *RoboCup 2018: Robot World Cup XXII*. Montréal, QC, Canada: Springer-Verlag, 2018, pp. 238–250. ISBN: 978-3-030-27543-3. DOI: 10.1007/978-3-030-27544-0_20. URL: https://doi.org/10.1007/978-3-030-27544-0_20.

[34] Hesham Ibrahim Mohamed Ahmed Omara and Khairul Salleh Mohamed Sahari. "Indoor mapping using kinect and ROS". In: *2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*. 2015, pp. 110–116. DOI: 10.1109/ISAMSR.2015.7379780. URL: https://doi.org/10.1109/ISAMSR.2015.7379780.

[35] *OpenSLAM.org: The OpenSLAM Team*. URL: http://openslam.org/.

[36] Yeong Sang Park et al. "3D ego-Motion Estimation Using low-Cost mmWave Radars via Radar Velocity Factor for Pose-Graph SLAM". In: *IEEE Robotics and Automation Letters* 6.4 (Oct. 2021), pp. 7691–7698. DOI: 10.1109/lra.2021.3099365. URL: https://doi.org/10.1109/lra.2021.3099365.

[37] *Point Cloud Library - pointclouds.org*. https://pointclouds.org/. [Accessed 16-Jan-2023].

[38] Akarsh Prabhakara et al. "High Resolution Point Clouds from mmWave Radar". In: (2022). DOI: 10.48550/ARXIV.2206.09273. URL: https://arxiv.org/abs/2206.09273.

[39] Robotis. *TurtleBot3 Overview*. 2021. URL: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

[40] R. Rouveure, P. Faure, and M.O. Monod. "Radar-based SLAM without odometric sensor". In: *ROBOTICS 2010 : International workshop of Mobile Robotics for environment/agriculture*. Actes du colloque. Clermont Ferrand, France, Sept. 2010, 9 p. URL: https://hal.science/hal-00583427.

[41] Joao Machado Santos et al. "A Sensor Fusion Layer to Cope with Reduced Visibility in SLAM". In: *Journal of Intelligent Robotic Systems* 80.3-4 (Jan. 2015), pp. 401–422. DOI: 10.1007/s10846-015-0180-8. URL: https://doi.org/10.1007/s10846-015-0180-8.

[42] Lorenzo Sciavicco et al. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010. ISBN: 1849966346. DOI: 10.1007/978-1-84628-642-1. URL: https://doi.org/10.1007/978-1-84628-642-1.

[43] Anish Shastri et al. "A Review of Indoor Millimeter Wave Device-based Localization and Device-free Sensing Technologies". In: *arXiv preprint* (2021). DOI: 10.48550/arXiv.2112.05593. URL: https://arxiv.org/abs/2112.05593.

[44] Scott Wilson, Johan Potgieter, and Khalid Mahmood Arif. "Robot-assisted floor surface profiling using low-cost sensors". In: *Remote Sensing* 11.22 (2019), p. 2626. DOI: 10.3390/rs11222626. URL: https://doi.org/10.3390/rs11222626.

[45] Jie Yang et al. *3-D Positioning and Environment Mapping for mmWave Communication Systems*. 2019. DOI: 10.48550/ARXIV.1908.04142. URL: https://arxiv.org/abs/1908.04142.

[46] Ali Yassin et al. "MOSAIC: Simultaneous Localization and Environment Mapping Using mmWave Without A-Priori Knowledge". In: *IEEE Access* 6 (2018), pp. 68932–68947. DOI: 10.1109/ACCESS.2018.2879436. URL: https://doi.org/10.1109/ACCESS.2018.2879436.

[47] Barry Loh Tze Yuen, Khairul Salleh Mohamed Sahari, and Zubaidi Faiesal Mohamad Rafaai. "Improved Map Generation by Addition of Gaussian Noise for Indoor SLAM using ROS". In: *Journal of Robotics, Networking and Artificial Life* 4 (2 2017), pp. 118–123. ISSN: 2352-6386. DOI: 10.2991/jrnal.2017.4.2.3. URL: https://doi.org/10.2991/jrnal.2017.4.2.3.

[48] Boyu Zhang et al. "Grid Map Guided Indoor 3D Reconstruction for Mobile Robots with RGB-D Sensors". In: *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2018, pp. 498–503. DOI: 10.1109/AIM.2018.8452296. URL: https://doi.org/10.1109/AIM.2018.8452296.