Urban and Indoor Vehicular Navigation using IMU, GNSS, LiDAR, and Radar

by

Alan Zhang

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical Engineering

Carleton University
Ottawa, Ontario

Abstract

Vehicles must be able to localize themselves in all environments (unmapped and mapped) including urban and indoor areas where Global Navigation Satellite Systems (GNSS) performance may degrade. The research and development in this thesis cover three major localization techniques that use an assortment of sensors to achieve this. In urban environments, an Inertial Measurement Unit (IMU) and GNSS fusion using the Extended Kalman Filter (EKF) is developed. For indoor environments, Light Detection and Ranging (LiDAR) Simultaneous Localization and Mapping (SLAM) and Radio Detection and Ranging (radar) SLAM systems are devised. Novel techniques are developed to tune EKF parameters using a Genetic Algorithm (GA) approach and to apply radar in a Rao-Blackwellized particle filter. The thesis presents in-depth explanations of experimental approaches as well as results that demonstrate a variety of localization systems performing high accuracy estimations in several experiments.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Background

The ability to self-localize is essential for many mobile systems, including autonomous vehicles (Figure 1.1), virtual reality setups, and domestic robots [1]. Such systems have many applications including entertainment, delivery services, and emergency response. These systems use knowledge of their location to perform tasks like avoidance of dangerous situations, navigation to objectives, and mapping of unknown areas. A Global Navigation Satellite System (GNSS) is a powerful tool for localization involving communication between a receiver and satellites in Medium Earth orbit (MEO) [2]. In urban environments where GNSS is mostly available, GNSS, an exteroceptive system, is often fused with Inertial Navigation Systems (INS), which use proprioceptive sensors that measure linear and angular motion of a platform. In indoor navigation where GNSS is unavailable, common sensor setups that are used include Light Detection and Ranging (LiDAR) and/or cameras as exteroceptive components and Inertial Measurement Units (IMUs) and/or wheel odometry as proprioceptive components [3]. Radio Detection and Ranging (radar) sensors as exteroceptive point cloud scanners are a relatively new development with unique characteristics compared to LiDAR technology.

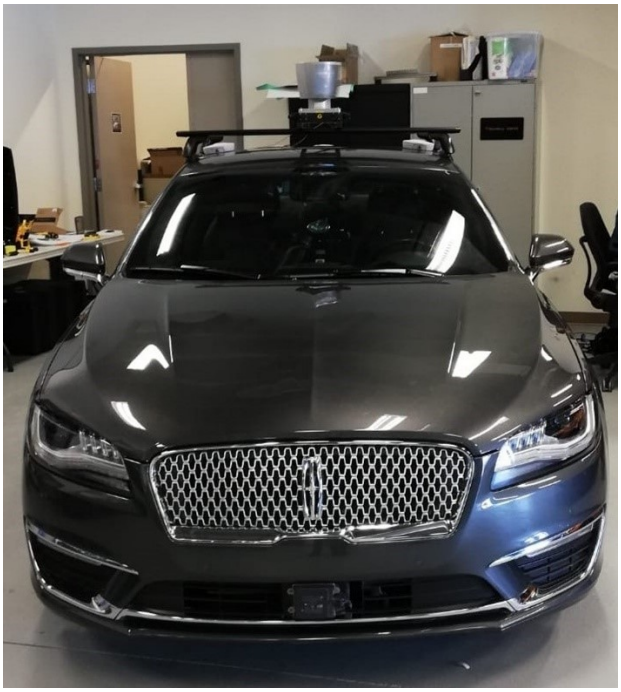


Figure 1.1 An autonomous vehicle used in the outdoor urban experiments

To address the urban and indoor localization problem, fusion of both exteroceptive and proprioceptive sensors is necessary and effective for several reasons. Exteroceptive sensors receive information from the environment to perform localization using targets with known poses, resulting in low drift estimations in many situations. They can also be used to recognize areas that have been seen before and use this information to improve the pose trajectory estimate. This use of exteroceptive sensors to learn and recognize the environment is known as Simultaneous Localization and Mapping (SLAM) which is a very effective method for localization. Proprioceptive sensors do not depend on the environment and, in the case of IMUs (Figure 1.2), can provide pose data in any situation. These sensors use dead reckoning to determine pose, which results in drift over time.

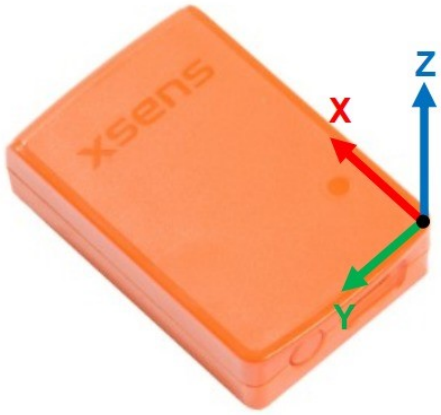


Figure 1.2 An Xsens IMU

Therefore, proprioceptive sensors are best used for short-term localization data, and exteroceptive sensors are best used for periodic accurate localization and mapping corrections. The complementary effect of both types of sensors allows for a well performing localization system in favorable conditions. Current state of the art autonomous vehicles emphasize on the reliance on a fusion of imperfect sensors since there is no single ideal sensor for every situation, and therefore primarily use a combination of LiDAR, radar, GNSS, IMU, and vision [4].

1.2 Motivation

1.2.1 Urban Localization

In urban settings, GNSS receivers provide global localization, but may fail or even provide misleading data [5] in areas with obstructions between the antennas of the receivers and satellites. Examples of these situations include roofed indoor areas or urban canyons [6]. One method of overcoming temporary outages in GNSS signal is to use an Extended Kalman Filter (EKF) to fuse the GNSS with an IMU while simultaneously estimating IMU

error characteristics so that the IMU can provide more accurate estimations during short GNSS outages. However, this fusion scheme involves tuning parameters with unknown relationships which make a successful implementation difficult and tedious to achieve.

1.2.2 Indoor Localization

In indoor settings, LiDAR scanners are common sensors used in localization without GNSS. 2D LiDAR scanners are commonly used for ground vehicles due to their reduced cost and complexity. However, 2D LiDAR systems also perform poorly under certain circumstances, such as in environments where limited distinguishable geometry exists like in long corridors (see Figure 1.3).

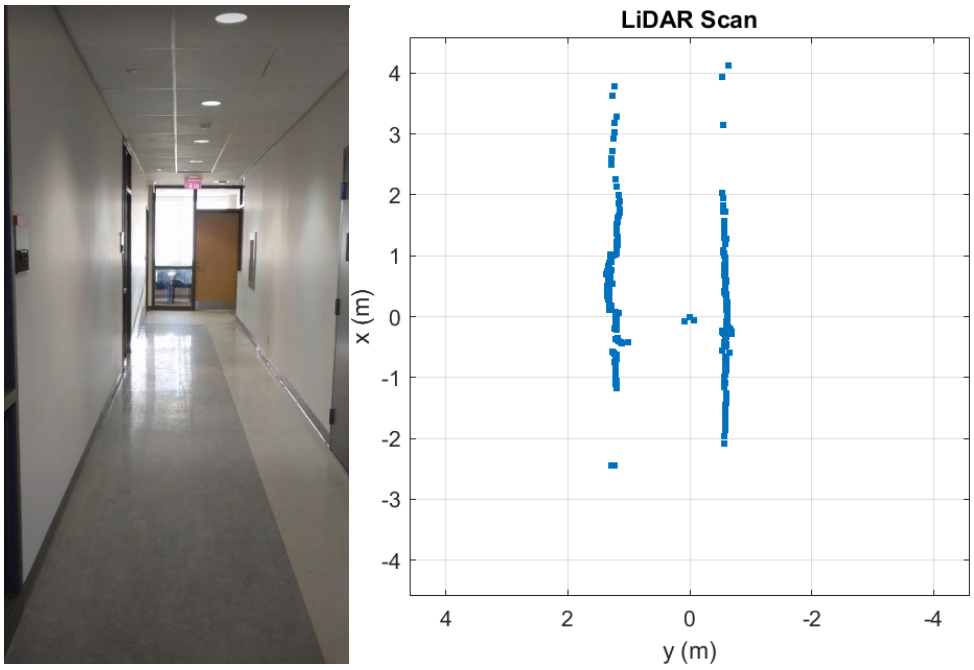


Figure 1.3 Long hallway where short range LiDAR detects parallel straight lines

Because 2D LiDAR scans detect most solid objects on a horizontal plane, a hallway appears as points arranged in two parallel lines. Scan matching with such features results in poor performance [7]. The matching results from using this geometry only provides localization confidence in the direction perpendicular to the hallway. Wheel odometry or Inertial Measurement Unit (IMU) dead reckoning is often used to aid in these systems when the exteroceptive sensors fail, and therefore can provide estimates along the uncertain dimensions. However, such dead reckoning systems have their own sources of uncertainty that accumulate the longer the traversal of the environment continues without updates from external measurements.

1.3 Objectives

The objectives of this thesis are to develop accurate localization techniques that can be applied in urban and indoor environments.

1.3.1 Urban Localization

In urban areas where GNSS is generally available, an IMU and GNSS fused with an EKF can be an effective localization method. To overcome the parameter tuning difficulty, a tuning framework using Genetic Algorithms (GA) is applied.

1.3.2 Indoor Localization

In indoor areas where GNSS signals are completely obstructed, a LiDAR and wheel odometry system fused using a graph-based SLAM technique [8] is developed. To avoid LiDAR performance deterioration in long hallways and large open areas, the use of radar

is explored. As such, a fusion of a radar sensor and two proprioceptive sensors, IMU and wheel odometry, are integrated into a SLAM system that aims to match and potentially surpass LiDAR systems in long hallways and large open areas. Specifically, a millimeter wave (mm-wave) Frequency-Modulated Continuous Wave (FMCW) fixed antenna array radar sensors is applied as SLAM sensor. Mm-wave FMCW fixed antenna array radar sensors (Figure 1.4) are gaining prevalence for their low cost, penetration ability, small size, and high accuracy, and are often integrated into commercial systems to provide autonomous capabilities [9] [10].

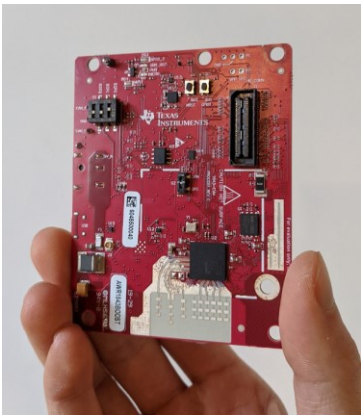


Figure 1.4 A Texas Instruments mm-wave FMCW fixed antenna array radar device used in the indoor experiments

To summarize, the objective is to develop an optimized GNSS/IMU EKF system for urban areas, and a LiDAR or radar-based system for indoor areas. Radar sensors may have superior attributes in certain areas compared to LiDAR sensors, and for this reason a radar SLAM system is created that overcomes some of the issues existing within conventional LiDAR systems.

1.4 Awards and Published Contributions

A list of awards and published/submitted work about the thesis work is given below:

[1] **A. Zhang** and M.A. Atia, "An Efficient Tuning Framework for Kalman Filter Parameter Optimization using Design of Experiments and Genetic Algorithms," Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), Miami, Florida, September 2019, pp. 1641-1652.

[2] **A. Zhang**: Best Presentation Award in ION GNSS+ 2019 Conference, session C4: Autonomous Applications, presentation # 2, Thursday, September 19, 2019.

[3] **A. Zhang** and M.A. Atia "An Efficient Tuning Framework for Kalman Filter Parameter Optimization using Design of Experiments and Genetic Algorithms", under second review round in ION NAVIGATION Journal, NAVI-2019-119.R1, March 2020.

[4] **A. Zhang** and M.A. Atia, "Comparison of 2D Localization Using RADAR and LiDAR in Long Corridors", submitted to IEEE Sensors 2020, Virtual Conference, Oct 25-28, 2020.

[5] S. Kourabbaslou, **A. Zhang** ; M. M. Atia "A Novel Design Framework for Tightly coupled IMU/GNSS Sensor Fusion using Inverse-Kinematics, Symbolic Engines and Genetic Algorithms", IEEE Sensors Journal, Vol(19), no(23), pp: 11424 – 11436, Dec. 2019.

1.5 Organization

The thesis is split into the following chapters. Chapter 2 provides background knowledge and related work on a variety of sensors, methods of representing the environment, and navigation algorithms. Chapter 3 presents a GNSS/IMU EKF system tuned with a GA. Chapter 4 presents the implementation of graph SLAM using LiDAR and wheel odometry. It also introduces the hardware platform used to develop the experiments in Chapter 4 and Chapter 5. Chapter 5 presents work in radar Rao-Blackwellized particle filtering, and includes an experiment comparing the scan matching ability of LiDAR and radar. An explanation of the developed radar SLAM system is provided along with a demonstration of the radar system localizing in multiple trajectories. Chapter 6 concludes the thesis, reiterates the contributions, and notes future works.

Chapter 2 Background and Related Work

The process of localization using sensor fusion is often divided into two components: the back end, which includes algorithmic approaches and optimization theory, and the front-end, which involves sensor interfacing, signal processing, and data manipulation. In this chapter, the relevant front-end technologies and back-end algorithms are summarized.

2.1 Navigation Technologies

The sensors used in the experiments consist of the exteroceptive sensors, GNSS, LiDAR, and radar, and the proprioceptive sensors, IMU and wheel odometry. The basic mechanisms and use of these sensors are introduced.

2.1.1 GNSS

GNSS allows small devices to determine their position on the planet to a high degree of precision within meters to centimeters in good conditions [11]. This widely used technology is enormously effective for a variety of applications including navigation, astronomy, cartography, fleet tracking, and many others [12].

GNSS uses signals transmitted along a line of sight from satellites to receivers that encodes information that can be used to calculate current time and position. A complete GNSS system consists of multiple satellites arranged in a constellation that ensures large coverage of the globe, and receivers that use multiple satellite transmissions to determine position and time with high accuracy. The basic operation involves receivers receiving time data, range and range rate measurements, which determine the line of sight distance and

velocity respectively, and satellite ephemeris data, which allows the precise position of the satellite to be found. By using the known locations of at least 4 satellites, the full 3D position of the receiver can be estimated using a form of trilateration, and the receiver clock errors can be estimated to improve the accuracy of future calculations.

The use of GNSS have a variety of complications. Atmospheric delays, satellite orbital errors, and signal multipath cause offsets in time sensitive measurements, which may result in hundreds of meters of error in the range estimates. Building obstructions and indoor environments often cause GNSS to completely fail. In order to reliably perform localization in such an urban environment, GNSS must be supplemented with additional sensors.

2.1.2 IMU

Micro-Electromechanical Systems (MEMS) IMUs are a widely used motion sensor due to their low cost, low power, and small form factor. Most importantly, the fact that IMUs are completely proprioceptive and do not need any contact or transmission from the external world make them an attractive technology. They are often used in most mobile systems for their ability to calculate attitude and pose with low resource cost.

IMUs measure accelerations and rates of rotations using accelerometers and gyroscopes, respectively. Sensing these motions is done using Newton's first law of motion, which states that an object's acceleration is proportional to the applied external forces on the object. In MEMS IMU sensors, an accelerometer uses a mass connected to a method of suspension with the displacement of the mass indicating the force (which is proportional to acceleration) applied in the direction along the suspension. Gyroscopes

measure rates of rotations using the same concept applied to vibrating masses affected by Coriolis forces [13].

MEMS IMUs suffer from a high degree of error, including biases, scale factors, cross-coupling, and random noises. Without accurately characterizing these errors, the use of IMUs to estimate pose is often extremely limited in effectiveness due to the double integration necessary to calculate position from acceleration. Methods for characterizing IMU errors have been developed to correct for this, including use of Gauss-Markov (GM) random processes and Allan Variance (AV) techniques. The current approach of using these techniques is to generate long records of static measurements to obtain modelling parameters [14] [15]. However, these methods do not handle disturbances or changes in the environment and may not result in accurate tuning for practical dynamic motions. In general, pose estimates are accurate within a few seconds, but errors quickly grow unbounded to hundreds of meters in a few subsequent seconds. Therefore, IMUs are often good with estimating pose in the short term but cannot be relied upon in the long term.

2.1.3 Wheel odometry

Wheel odometry involves counting wheel revolutions using rotary encoders. Rotary encoders are cheap and reliable, and due to this are often included into wheeled systems as a source of pose information. Rotary encoders are electro-mechanical devices that use a pattern attached to the rotating component and a sensor that detects changes in the pattern. In doing so, the angular velocity of the rotating component, and sometimes the angle, can be calculated. Through knowledge of the arrangement of wheels and their dimensions, the linear and angular velocity, position, and heading of a wheeled system can be calculated.

Wheel odometry can provide fairly accurate pose estimates in ideal conditions but are susceptible to large errors caused by bumps in the ground, wheel slippage, or changes in surface. Additionally, unavoidable errors in providing the exact dimensions of the wheel diameter and arrangements, including dynamic changes to the materials, result in drift over time. This makes wheel odometry similar to IMUs in that they are useful in determining pose over the short term but must rely on other sensors or complicated processes to detect and overcome errors in the system to sustain localization performance.

2.1.4 LiDAR

LiDAR sensors send out beams of light and measure reflections to create 2D or 3D scans. They are often used to make high-resolution 3D maps for land surveying, geology, forestry, and navigation. LiDAR sensors are often known for being powerful technology with a large price tag and complex mechanical structure. However, recent developments have produced cheaper designs with similar characteristics to high end systems, making them more accessible to a wider audience.

LiDAR emits ultraviolet, visible, or near infrared light in a beam commonly steered by rotating mirrors. These beams can detect most material at very high resolutions, resulting in a system that is able to map physical structures at a centimeter level resolution. By measuring the difference in time between transmitting and receiving a beam, the distance travelled by the beam can be determined. By measuring the angle of the rotating mirrors with rotary encoders at the time of reception, the direction of the beam can be recovered with high angular resolution. The output of LiDAR scanners is typically point

cloud data, which represents 3D or 2D points for every beam emitted in one cycle of the mirror rotations.

LiDAR as a navigation tool suffers from issues such as being obstructed by rain, aerosols, and dust. Furthermore, moving components pose more risk for mechanical failure and wear and tear. In summary, LiDAR sensors have high potential in being an accurate source of localization data in many environments yet cannot be relied upon for every situation.

2.1.5 Radar

The properties of radar technology make it a unique sensor with interesting characteristics compared with LiDAR and other exteroceptive devices. Radar localization can rectify the deficiencies of the previously discussed localization systems, but certainly has its own challenges. In this section, explanations about the strengths and weaknesses and the background of FMCW fixed antenna array radar mechanisms are covered in detail to provide a good understanding of radar as a localization device. Table 1 provides a summary of major advantages of using radar sensors compared to other commonly used sensors.

Table 1 Radar sensor advantages

Property	Radar	LiDAR	Camera
Cost [16] [17]	Tens of dollars	Hundreds to hundreds of thousands	Tens of dollars
Adverse environmental conditions [18]	None: penetrates most airborne particles, rain, fog	Poor performance in dusty, foggy, or rainy conditions. Lenses need to be clean	Poor performance in dusty, foggy, or rainy conditions. Lenses need to be clean. Needs to have good lighting conditions
Sturdiness	Solid state, can be covered with hard shell materials	Most use rotating mechanisms, subject to mechanical failure over time, and require transparent lenses that present a point of weakness	Solid state, but requires transparent lens that presents a point of weakness
Range [19] [20] [21] [22]	Low cost devices up to 150m	Low cost devices up to 25m, high end devices up to 220m	For stereo cameras, up to 40m for 3D sensing
Instantaneous velocity detection [10] [23]	Doppler effect from radio waves	Only niche types of LiDAR provide doppler effect readings	No direct velocity data
Primary detected features	Radar reflective materials, including that which may be buried behind walls and other obstacles	Solid objects	Surfaces with contrasting colors
Failure Modes	Environments with sparse radar reflective features	Adverse environmental conditions, locations with sparse features, locations with simple geometry (ie. long hallways, one straight wall, etc)	Adverse environmental conditions, locations with uniform color or patterns

The major issues in using radar are that detections are very noisy, sparse, and often present in unexpected locations. This is due to many factors, including penetration of radar-transparent material, multipath reflections, off-altitude reflections, and noise introduced in the signal processing electronics [24, 25, 26]. Such characteristics make precise scan matching and map building with radar a challenging task.

2.1.5.1 Radar Fundamentals

A millimeter wave radar system transmits electromagnetic waves and measures the incoming waves that have reflected off the environment. By measuring the time of outgoing transmissions and incoming reflections and the differences in received signals from different antennae, both the distance and the direction of the reflected wave can be recovered [10].

The AWR1843, a Texas Instrument radar sensor used for the experiments, uses FMCW signals to achieve the measurements. FMCW involves transmitting a chirp signal, which is a continuous wave that linearly increases in frequency with time. Figure 2.1 shows the chirp in a time vs magnitude plot and a time vs frequency plot.

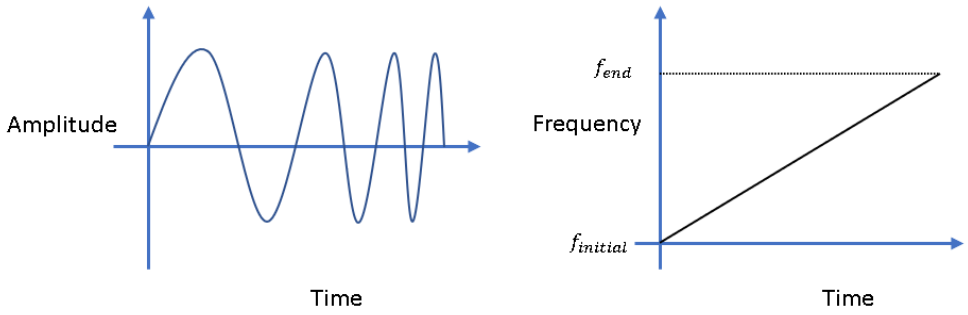


Figure 2.1 Chirp signal in an amplitude vs time plot and a frequency vs time plot [10]

An object detected would reflect the continuous wave resulting in the receiver receiving a delayed chirp. Using a mixer, the difference in the instantaneous frequency of the transmitted and received chirp is generated as a new constant signal, shown in Figure 2.2.

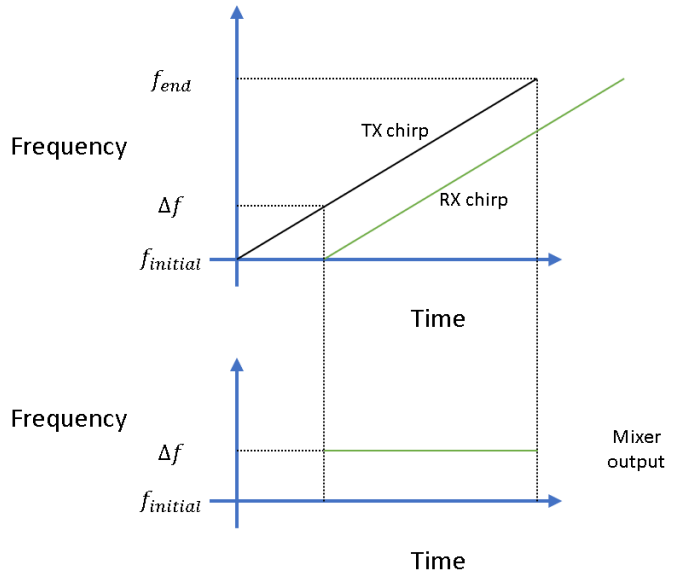


Figure 2.2 Subtraction between transmitted and received chirps [10]

The frequency of this signal is proportional to the distance to the object. The further away a detected object is, the longer the delay, the larger the difference in measured frequency, and thus the higher the frequency. Multiple detections of different objects would be seen as multiple tones in the mixer output, which is shown in Figure 2.3. The slope of the chirp and the minimum and maximum frequencies determine the maximum distance that can be measured with the radar.

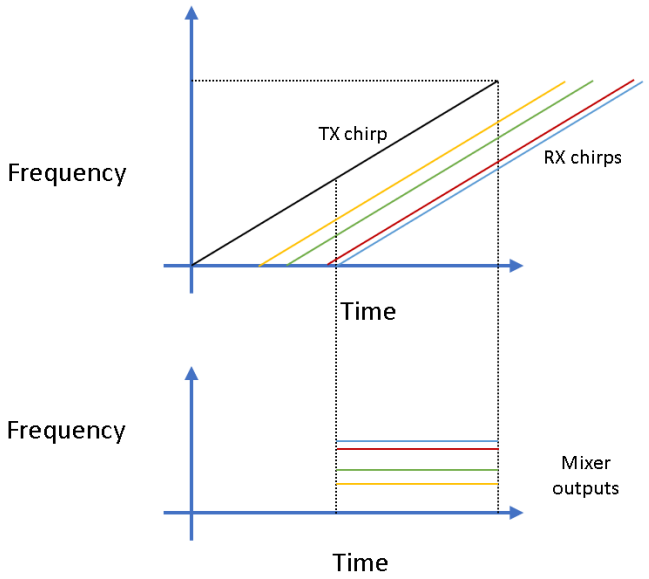


Figure 2.3 Detections from multiple distances [10]

The angle of the reflected wave is calculated using the differences in the signals received by the different antennae. The AWR1843 has 4 receive antennae arranged in linear pattern with each antenna separated by half a wavelength [27] as seen in Figure 2.4.

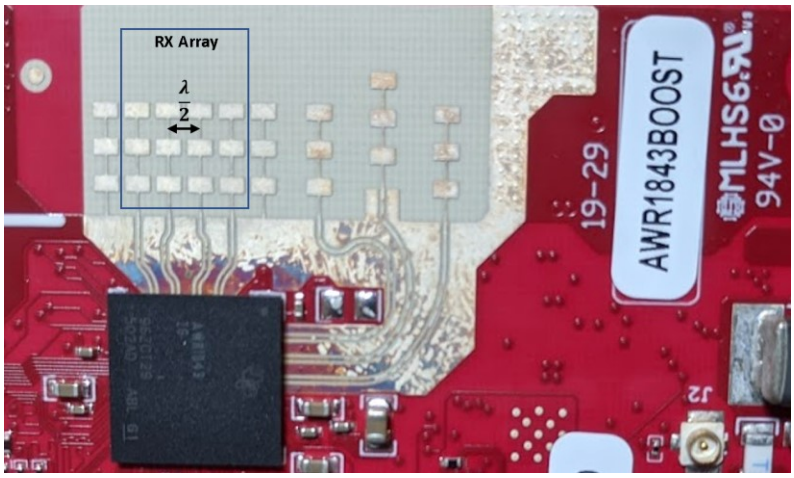


Figure 2.4 AWR1843 antenna array

As a detected object is moved from the front, 0 degrees, to the side, 90 degrees, the phase difference of the received signal will range from 0 to 180 degrees out of phase for two adjacent antennae. The lagging signal determines the sign of the direction. Figure 2.5 demonstrates the difference in distance from each antenna, where $0 < \Delta d < \lambda/2$ for detections from one side.

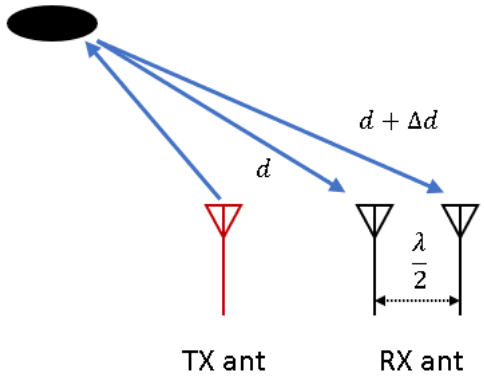


Figure 2.5 Different distances between obstacle and antennas [10]

An unambiguous measurement requires the received signal to originate from the front half of the radar, meaning this sensor has an angular field of view of 180 degrees.

2.1.5.2 Related Work on Radar Localization

The radar sensor as a SLAM component has been increasing in interest over the past few years due to the development of the combination of fixed antenna arrays, allowing for a solid state system, single systems on chip, allowing for low cost and low power devices, and mm-wave technology, which allows for small form factor [10]. Autonomous localization is also currently a popular research topic due to the rise of self-driving vehicles.

[28] and [29] presents Graph SLAM approaches using an automotive grade radar mounted on a vehicle. These authors' experiments were performed outdoors where longer range radar systems may detect many trackable features in environments with rich radar reflective materials. In radar systems for indoor localization, the sensor must be either configured to limit the range or otherwise must filter out high levels of multipath. One aspect in [28] is of loop closure using computer vision techniques. Their method of using a series of computer vision techniques to find loop closures is effective for large search spaces in the long outdoor routes over several km. In the experiments done in this thesis, a particle filter was sufficient in performing loop closure in indoor environments. In [30], authors use an Unscented Kalman Filter to fuse a Recurrent Neural Network, used to replace the IMU mechanization model, with radar scan matching implemented through a normal distributions transform method. Their experiments were done within the confines of a room, while in contrast the work in this thesis explore the performance of a radar localization system in long hallways. In [31], a particle filter is implemented with 4 radar sensors mounted on a vehicle. The large noise associated with radar measurements are the focus of [25]. The researchers use a deep neural network to learn the noise characteristics of the radar and achieve a more realistic inverse sensor model that segments free space and occupied space. Their system outperforms standard filtering approaches. In this thesis, a simple sensor model was used that only considered the area around radar detection points as useful information. This was suitable for the length and environment of the experiments. Many papers show SLAM with radar results using rotating antenna radar [32] [33] [34]. In general, these radar sensors are more accurate, however, are more complex due to mechanical parts and incorporating rotations into account.

2.2 Environment Representation

Exteroceptive sensors are useful for localization by being able to track the external environment, which often requires building a map. There are two standard environment representation techniques in localization systems: feature-based and grid occupancy based. Representing the environment through features involves extracting high level information from raw data, which introduces additional sources of error. In grid occupancy maps, a map is filled with cells that are either occupied or free, which allows a more straightforward integration with point cloud data such as those provided by a LiDAR or radar scanner [35]. This section explores the properties of using occupancy grid-based techniques considering the point cloud output of LiDAR and radar sensors.

2D Occupancy grid maps can be used to represent a map using 2D point cloud data. Grid maps divide the world into cells, with each cell representing a binary random variable and the cell value being the probability of the cell being occupied by an obstacle [36]. The map uses the assumption that each cell is completely empty or occupied, meaning that the larger the cell size, the less detail the map will hold. It is also assumed that the world is static, and that each cell is independent of each other. These assumptions allow the simple computation of the probability of the map

$$p(m) = \prod_i p(m_i) \tag{2.1}$$

where m_i is the ith cell in the map. Point cloud data, once positioned correctly, are used to update the map. Given the sensor data and poses, the map is estimated as

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{2.2}$$

Using Bayes rule and the Markov assumption, which is used to remove dependence on past states given the known current state, a static state binary Bayes filter is developed and the following derivations in (2.3)-(2.6) show the development of the occupancy map update equation in log odds notation. This is done by creating a ratio of opposites $p(m|z,x)$ and $p(\neg m|z,x)$ such that the output is also a ratio of opposites in binary states that can be computed easily.

$$\begin{aligned} p(m|z_{1:t}, x_{1:t}) &= \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})} \\ p(\neg m|z_{1:t}, x_{1:t}) &= \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})} \end{aligned} \tag{2.3}$$

$$\begin{aligned} \frac{p(m|z_{1:t}, x_{1:t})}{p(\neg m|z_{1:t}, x_{1:t})} &= \frac{p(m|z_{1:t}, x_{1:t})}{1 - p(m|z_{1:t}, x_{1:t})} \\ &= \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)} \end{aligned} \tag{2.4}$$

From here, the log odds ratio defined as

$$l(x) = \frac{p(x)}{1 - p(x)} \tag{2.5}$$

The log odds notation simplifies the map update so that products are turned into sums

$$l(m|z_{1:t}, x_{1:t}) = l(m_i|z_t, x_t) + l(m_i|z_{1:t-1}, x_{1:t-1}) - l(m_i) \qquad (2.6)$$

where $l(m_i|z_t, x_t)$ is the inverse sensor model, $l(m_i|z_{1:t-1}, x_{1:t-1})$ is the recursive term or previous cell probability, and $l(m_i)$ is the prior, or initial cell probability in log odds notation. The inverse sensor model defines the probability of occupancy given the sensor beam characteristics. The sensor emits independent beams at different angles that reflect off obstacles. For each beam, the inverse sensor model that determines the probability of the map given the detected point and robot position is shown in Figure 2.6 for a LiDAR sensor and a radar sensor, for example.

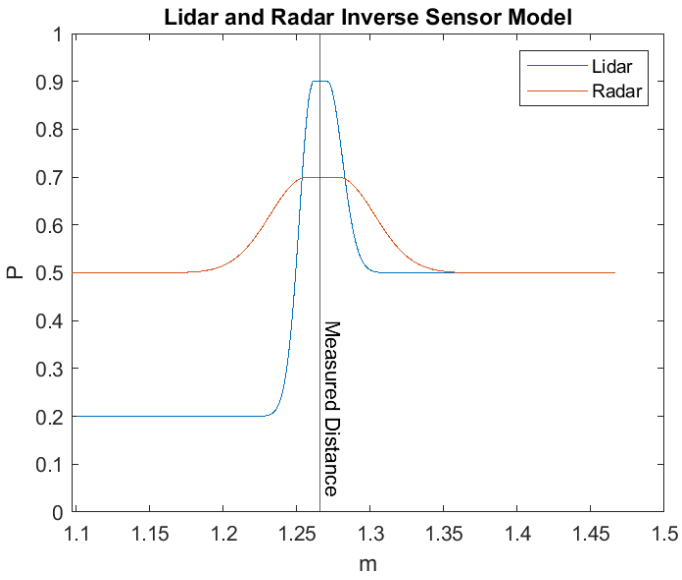


Figure 2.6 Inverse sensor models for LiDAR and radar

To implement this mapping process, a digital differential analyzer algorithm is used to approximate the laser line of sight at the specified angle on a grid [37]. For a LiDAR,

the plateau in the graph at the example measured distance has a fixed radius of 1 cm, with each cell in the radius assigned to $p_{occupied} = 0.9$. Every cell on the line past the plateau until the edge of the map is assigned to $p_{prior} = 0.5$, denoting the beam contributing no information on the occupancy status of cells past the detection, and every cell on the line before the plateau is assigned $p_{unoccupied} = 0.2$, since an absence of reflections suggests that the space is unoccupied. A gaussian blur is applied to the resulting measurement map to approximate uncertainty and to smooth edges. To incorporate the measurements to the global map, the map and computed map measurement is transformed to log odds through (2.5) and added together in the occupancy grid mapping scheme described in (2.6). Figure 2.7 shows the example of a grid map of a 6m by 10m room with a grid cell size of 3cm^2, where the brightness of a cell represents the likelihood of it being free space.

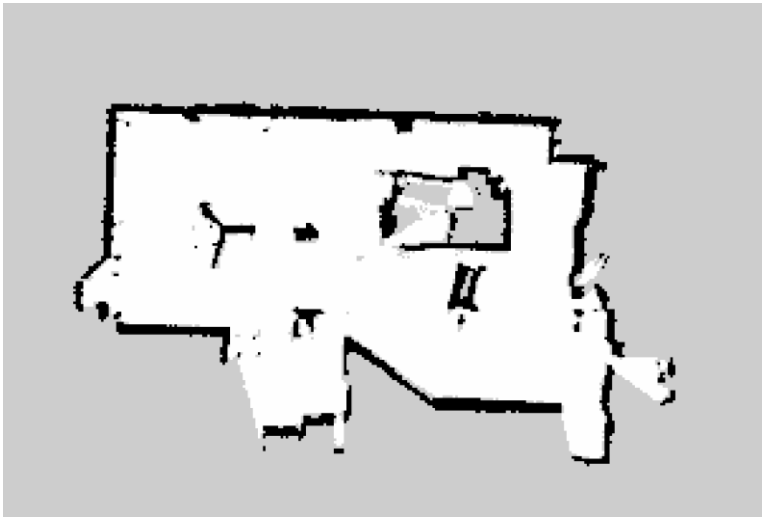


Figure 2.7 Example of an occupancy grid map

The developed radar SLAM algorithm uses occupancy grid maps since they do not require feature detectors, which allows for a more robust system. However, grid maps do

not scale well, and setting the size of the cell requires a balance between map detail and map memory requirements.

2.3 Navigation Algorithms

The navigation algorithms, often called the back-end, are responsible for using sensor model outputs to calculate and optimize pose estimations. Methods often involve using sensor statistics to estimate pose as a probability distribution due to the errors associated with each sensor.

2.3.1 EKF

In the EKF, a set of state variables are estimated using multiple sensors, and state errors and sensor noise are all considered to be Gaussian [38]. The EKF uses a system model and a measurement model that may be non-linear, represented in the general form:

$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u(t)\big) + w(t) \\ y(t) &= h\big(x(t)\big) + v(t) \end{aligned} \tag{2.7}$$

where $f(.)$ is a nonlinear system model, $u(t) \in \mathbb{R}^d$ is a control signal vector of d sensors that trigger the evolvement of the n-element state vector $x(t) \in \mathbb{R}^n$, and $w(t) \in \mathbb{R}^n$ is the stochastic noise of covariance Q. A measurement vector $y(t) \in \mathbb{R}^m$ represents m observable states of the systems; these are the states that can be directly measured by an external observer. The measurement model is commonly a nonlinear function *h(.)* that maps the system's state $x(t)$ into the system's observables $y(t)$. As measurement

instruments are not perfect, $v(t) \in \mathbb{R}^m$ is used to model measurement noise with covariance R. Using first-order Taylor series approximation, (2.7) can be linearized as follows:

$$\begin{gathered} \delta\dot{x}(t) = F(t)\delta x(t) + w(t) \\ \delta\dot{y}(t) = H(t)\delta x + v(t) \\ F(t) = \frac{\partial f(x(t), u(t))}{\partial x} \\ H(t) = \frac{\partial h(x(t))}{\partial x} \end{gathered} \tag{2.8}$$

EKF optimal state estimation can be performed through a prediction step, and an update step described as follows:

Prediction

$$\begin{gathered} x_{k+1} = f(x_k, u_k) \\ P_{k+1} = (I + F_{k+1}T)P_k(I + F_{k+1}T)^T + Q_{k+1} \\ K_{k+1} = P_{k+1}\, H_{k+1}^T (\, H_{k+1}\, P_{k+1}\, H_{k+1}^T + R_{k+1})^{-1} \end{gathered} \tag{2.9}$$

Update

$$\begin{gathered} x_{k+1} = x_{k+1} + K_{k+1}[y_{k+1} - h(x_{k+1})] \\ P_{k+1} = (I - K_{k+1}H_{k+1})P_{k+1} \end{gathered} \tag{2.10}$$

Where P_k is the state covariance matrix and K_k is the Kalman gain at timestep k. The urban pose estimation scheme developed in this thesis uses GNSS and INS in a tightly coupled scheme fused with an EKF [39].

2.3.2 Scan Matching

Scan matching is a method of determining the transform between two scans, two maps, or a scan and a map that contains matching environments. In doing so, scan matching also provides an estimation of mapping unknown areas using mapped areas. Several techniques exist to perform this task, including Iterative Closest Point [40], which seeks to minimize the pointwise distance between two inputs, feature based scan matching [41], which uses details in the data to perform point associations between scans, and correlative methods [42], which generates sample points in a uniform distribution and computes a probability distribution of scan matching likelihood over the samples. In this work, the correlative method is used for its reliability in finding global maxima and non-reliance on correct data associations.

The correlative scan matching process for a generic sensor is as follows. The inputs to scan matching in the 2D case are two 2D point clouds. One scan is rasterized into a grid map. This is done by discretizing each point to the nearest cell. Each cell containing a point is assigned a high probability of being occupied. A gaussian blur is applied to the map to approximate sensor noise. This map is considered an occupancy grid map m. The second scan, corresponding to measurement z, is evaluated as a scan emitted from position x_i where i represents one x and y position on the grid at a θ heading. The probability of the scan matching the map at i is

$$\mathrm{p}(\mathrm{z}|x_i, m) \tag{2.11}$$

This can be computed by considering each scan to be a separate measurement, and thus the probability becomes

$$\mathrm{p}(\mathrm{z}|\mathrm{x}_i, \mathrm{m}) = \prod_j \mathrm{p}(\mathrm{z}_j|\mathrm{x}_i, \mathrm{m}_j) \tag{2.12}$$

where j corresponds to each cell containing a measurement point. Evaluating this probability at every ith position for a range of x and y positions and θ headings generates a probability distribution where the maximum probability at cell and heading i can be taken as the most likely transformation that matches the second scan to the first scan.

2.3.3 Graph SLAM

Graph SLAM is an established technique that involves using a series of nodes to represent poses and edges to represent measured transformations, creating a graph such as the one shown in Figure 2.8, where red nodes are poses and black lines are edge measurements. Measurement edges are provided by sensor data, such as wheel odometry measurements or LiDAR scan matching. In two independent processes, the graph is expanded and is optimized to minimize errors, self-correcting the trajectory and map together.

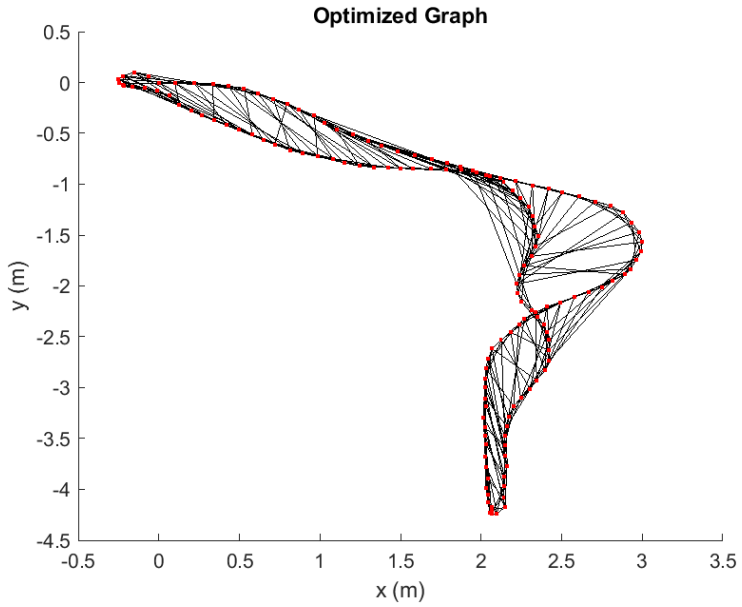


Figure 2.8 Example of graph SLAM visualization

The process begins with the first node being created where the trajectory starts. As the vehicle moves, periodically, nodes are created. Every adjacent node is connected with one edge that corresponds to the odometry estimate. In a LiDAR based system, a scan measurement is also associated with every node. At every new node, the new scan is matched to scans of surrounding nodes, with area determined by the uncertainty associated with the pose calculations. If a scan observation observes the same part of the environment from two different nodes, in other words, if scan matching produces a successful match, an extra edge is built between the two nodes supplying the scans using the scan match to determine the edge transform. Through this method, both recent nodes behind the current position and long past nodes from any direction can be connected, which elegantly combines loop closure and the use of scans for incremental odometry. Nodes consist of pose and an index, and edges consist of connected nodes, a direction, and a transformation usually expressed as a homogenous transformation matrix. Edge observations have

associated noise, and therefore also contain an information matrix for each edge. The information matrix is the inverse of a covariance matrix, where the larger the values of the information matrix, the more confident the edge measurement is. As the graph starts to grow, the location of nodes according to edges will start to conflict, in that some edges will place a node at one location, and other edges connected to the same node would place the node at another location. Figure 2.9 displays a situation where measurement edges conflict with node locations, creating instances of error in the graph.

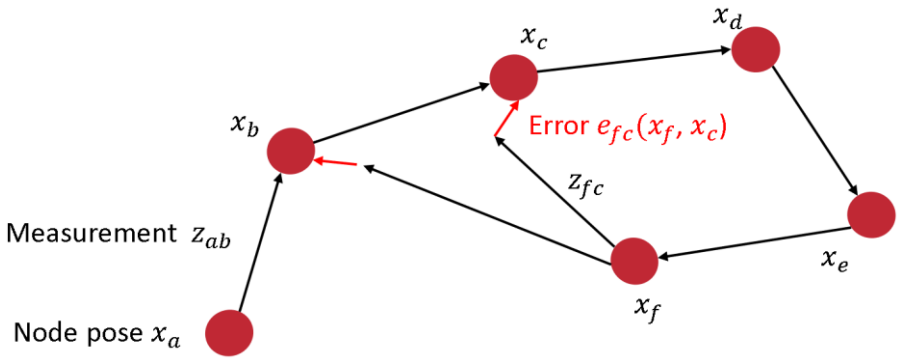


Figure 2.9 Error created by conflicting measurements and node locations

This results into an overdetermined system, which contains more equations than unknowns. The goal of graph optimization is to minimize the error conflicts while considering the uncertainty of each edge measurement.

A least squares approach is used to solve the set of equations in graph SLAM. Specifically, the Gauss Newton iterative least squares method is effective for overdetermined nonlinear systems with large sets of equations, such as in this graph optimization case. The general process is as follows.

Given a vector of states x, a set of measurements z_i, and a set of functions $f_i(x) = \hat{z}_i$, where $\hat{z}$ are estimated measurement, the optimal x are $\bar{x}$ such that $f_i(\bar{x}) = z_i$. The error function is defined as the difference between the estimated and actual measurement as shown:

$$e_i(x) = z_i - f_i(x) \tag{2.13}$$

This error term is assumed to have associated noise with an information matrix of Ω_i. The square error is calculated as

$$E_i(x) = e_i(x)^T \Omega_i e_i(x) \tag{2.14}$$

And the goal is to find the state positions x^* that minimize the global error as

$$x^* = argmin_x \sum_i e_i(x)^T \Omega_i e_i(x) \tag{2.15}$$

One iteration of the Gauss Newton approach involves solving for an incremental change in state Δx that moves the state towards the minimum of the global error. The solution is

$$\Delta x = -(\sum_i J_i^T \Omega_i J_i)^{-1} \sum_i J_i{}^T \Omega_i e_i \tag{2.16}$$

where J_i is the Jacobian that evaluates the slope of the error function e_i at the current position x. This can be simplified by making the following substitutions

$$b = \sum_i {J_i}^T \Omega_i e_i$$
$$H = \sum_i J_i^T \Omega_i J_i \tag{2.17}$$

Such that the solution can be found from

$$\Delta x = -H^{-1} b \tag{2.18}$$

The state change Δx is a step that is added to the states to bring them closer to the optimal location.

$$x \leftarrow x + \Delta x \tag{2.19}$$

This is iterated until the change in states are negligible.

As seen from the derivations above, the procedure for finding Δx involves defining the error function and the square error function, determining the Jacobian of the error function, computing the b and H terms using the error function, Jacobian, and information matrix, and solving for Δx with (2.18). The algorithm is presented in [8] applied to a 2D LiDAR sensor on a wheeled robot. Google's Cartographer [43] is a version of graph SLAM

involving matching scans to submaps implemented as a real-time system using LiDAR and wheel odometry.

2.3.4 Rao Blackwellized Particle Filter

This chapter outlines the theory of Monte Carlo Localization using particle filters [44] and explains the Rao-Blackwellized particle filter enhancement. The generic algorithm described comes from FASTSLAM, which is designed for a LiDAR and odometry fusion system.

2.3.4.1 Monte Carlo Localization

The particle filter represents robot pose using numerous samples, which allows it to model an arbitrary pose distribution. The particle filter aims to model the pose distribution using the Importance Sampling Principle [36]. This principle states that a distribution can be modeled by drawing samples from a known distribution, called the proposal distribution, and computing a weight w for each sample j by evaluating the sample using the target distribution as shown.

$$w_j = \frac{target(x_j)}{proposal(x_j)} \tag{2.20}$$

Figure 2.10 is a graphic of the distributions shown for a 1D case to illustrate the process.

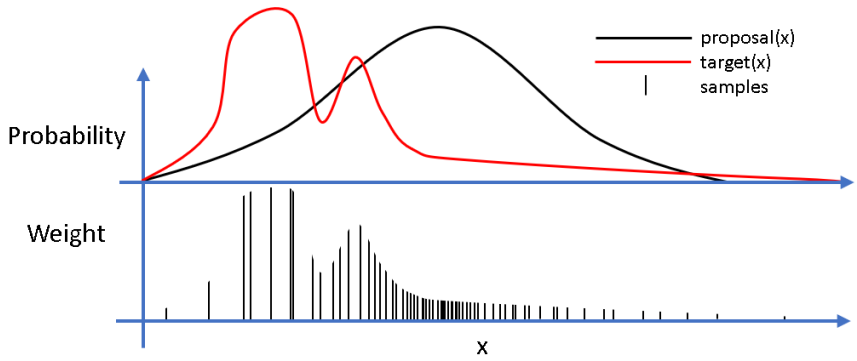


Figure 2.10 Importance Sampling Principle

Resampling with probability proportional to the weights approximates sampling from the target distribution. Hence, the particle filter involves three steps. First, the samples are drawn from the proposal distribution. Then, the importance weights are calculated by dividing the target, which can be directly evaluated, by the proposal probability density function, which results in the samples being weighted based on the likelihood of the target distribution and distributed according to the proposal function. Finally, resampling is done where new particles are chosen from the samples with probability proportional to their weighting, resulting in a distribution of particles more closely representing the target distribution. Monte Carlo Localization is a standard robot localization approach using particle filtering [45]. Each particle represents a possible pose hypothesis of the vehicle. The proposal distribution is the motion model, and the weighting is based on the ratio between the target distribution and the motion model, which is evaluated with the observation model.

2.3.4.2 Grid-FastSLAM

Grid-FastSLAM is an improved implementation of the Monte-Carlo localization technique applied to occupancy grid maps [46]. It uses Rao-Blackwellization to marginalize the joint probability of estimating the map and trajectory to a product of the two using the factorization shown in the equation.

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t})p(m|x_{1:t}, z_{1:t}) \tag{2.21}$$

The factorization allows for the separate computation of the path posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$ and map posterior $p(m_{1:M}|x_{1:t}, z_{1:t})$. With this framework, a particle filter is used to estimate the path posterior only, and for each particle, the straightforward process of mapping with known poses is applied. Incorporating the measurements and inputs together in the proposal distribution leads to an accurate posterior since the input error alone is relatively high. This allows the samples to be more concentrated in the likely areas, creates more accurate maps, and reduces the number of particles required. The proposal uses measurements and inputs and is expanded with Bayes rule as shown in the equation.

$$p(x_t|x_{t-1}, m, z_t, u_t) = \frac{p(z_t|x_t, m)p(x_t|x_{t-1}, u_t)}{p(z_t|x_{t-1}, m, u_t)} \tag{2.22}$$

Scan matching distributions are often very peaked in local estimation but may have multiple maxima in the vicinity. The motion model has a large uncertainty but only has one maxima. The product of these two distributions, $p(z_t|x_t, m)p(x_t|x_{t-1}, u_t)$, allows the optimal peak in the scan matching distribution to be chosen using the motion model as a

limit in the global position and scan matching as a local peak, which results in a tight distribution. The proposal distribution is simplified as shown in (2.23) using the law of total probability.

$$\begin{aligned} \tau(x_t) &= p(z_t|x_t, m)p(x_t|x_{t-1}, u_t) \\ p(z_t|x_{t-1}, m, u_t) &= \int p(z_t|x_t, m)p(x_t|x_{t-1}, u_t)\, dx_t \\ p(x_t|x_{t-1}, m, z_t, u_t) &= \frac{\tau(x_t)}{\int \tau\,(x_t)dx_t} \end{aligned} \tag{2.23}$$

Next, samples are drawn from the proposal distribution, labelled π as an approximation of an unknown probability distribution, then weighted with (2.20). The target distribution, which is the path posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$ from the Rao-Blackwellized factorization in (2.21), is divided by the proposal distribution leading to the derivation of the weight for particle i to be

$$\begin{aligned} w_t^i &= \frac{p\left(x_{1:t}^i \middle| z_{1:t}, u_{1:t}\right)}{\pi\left(x_{1:t}^i \middle| z_{1:t}, u_{1:t}\right)} \\ &= \frac{\eta p\left(z_t \middle| x_{1:t}^i, m^i\right) p\left(x_t^i \middle| x_{t-1}^i, u_t\right) p\left(x_{1:t-1}^i \middle| z_{1:t-1}, u_{1:t-1}\right)}{\pi\left(x_t^i \middle| x_{t-1}^i, m^i, z_t, u_t\right) \pi\left(x_{1:t-1}^i \middle| z_{1:t-1}, u_{1:t-1}\right)} \\ &= \frac{\eta p\left(z_t \middle| x_t^i, m^i\right) p\left(x_t^i \middle| x_{t-1}^i, u_t\right)}{\frac{p\left(z_t \middle| x_t^i, m^i\right) p\left(x_t^i \middle| x_{t-1}^i, u_t\right)}{\int p\left(z_t \middle| x_t^i, m^i\right) p\left(x_t^i \middle| x_{t-1}^i, u_t\right)}}\, w_{t-1}^i \\ &\propto w_{t-1}^i \int \tau(x_t^i) dx_t^i \end{aligned} \tag{2.24}$$

Where η is the normalization factor equal across all particles. The larger the weight value, the more the scan matching aligns with the motion model and the more confident the chosen scan matching peak. Once this is done, the resampling step is performed to redistribute samples according to the target distribution. The work in [47] presents the underlying SLAM system described in this section, developed for a 2D 360-degree LiDAR scanner.

Chapter 3 Urban Localization using GA tuned IMU/GNSS EKF

In this chapter, a vehicle localization system that fuses an IMU and GNSS with a tightly coupled scheme using the EKF is explained. Parameter tuning of EKF systems is a known difficulty, and a framework to systematically optimize parameter tuning using GA is the focused of this chapter. This fusion scheme is suitable for urban areas where GNSS is mostly available with intermittent interruptions.

3.1 Tightly Coupled IMU and GNSS EKF

A 3D tightly coupled IMU/GNSS EKF system [48] consists of determining the range and range rate measurements of transmissions between satellites and a receiver using IMU-calculated position and velocity and the satellite ephemeris data. The residual is then computed as the difference between the predicted range and those measured by the receiver. The states of the filter consist of the position, velocity, orientation, biases/scale factors of the IMU sensor, and the GNSS receiver clock bias and drift. The IMU sensor errors are modeled by random walk components, time-varying biases, and scale-factors modeled by Gauss-Markov (GM) [39] random processes. The GNSS receiver clock bias and drifts are modelled with random walk components. The pseudorange measurement noises are characterized with standard deviation estimated by the GNSS receiver and scaled by a tuned parameter value tuned by the proposed framework. A block diagram of a tightly coupled IMU/GNSS EKF is included in Figure 3.1.

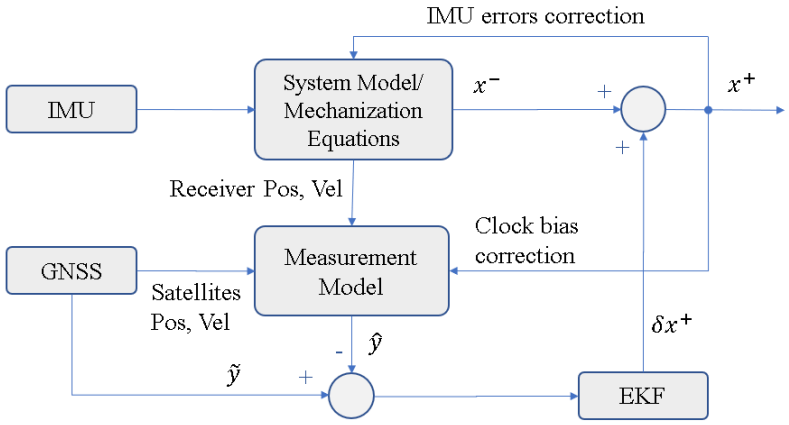


Figure 3.1 EKF Block Diagram [49]

To initialize the filter in this experiment, the position, velocity, and orientation states are set to the ground truth solution provided by a separate high-end system (to be described in 3.4).

3.1.1 System Model

The INS mechanization equations are an approximate method of determining 3D position, velocity, and orientation using a 3-axis gyroscope and 3-axis accelerometer. The rate of change of a quaternion vector given the gyroscope measurements can be calculated by [50]

$$\dot{q}_B^L = \frac{1}{2}\begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix}\begin{bmatrix} 0 \\ \left(\frac{\hat{\omega}_{LB}^B - b_g}{I_{3\times 3} + s_g}\right)_{3x3} \end{bmatrix} \tag{3.1}$$

where $\widehat{\omega}_{LB}^{B}$ are the compensated gyroscope measurements that measure the rotation rates of the vehicle with respect to the local level frame (L frame), which is considered as the ground plane with axes oriented North, East, and Down, and b_g, s_g are gyroscope biases and scale factors respectively. $q_B^L = [a\ b\ c\ d]^T$ is the quaternion that defines the orientation of the vehicle with respect to the ground plane, and $[.]_\times$ is the skew matrix operator. To calculate $\dot{v}^L$, the velocity in the L frame, the accelerometer values $\hat{a}_{IB}^{B}$ are, after being corrected with scale factor s_a and bias b_a, rotated by the orientation in direct cosine matrix form C_B^L. Additional acceleration forces are compensated to account for gravity, transport rate ω_{EL}^L, and earth rotation rate ω_{IE}^L.

$$\dot{v}^L = C_B^L\left(\frac{\hat{a}_{IB}^{B} - b_a}{I_{3\times3} + s_a}\right) + g^L - (2\omega_{IE}^L + \omega_{EL}^L) \times v^L \tag{3.2}$$

Finally, the position model $\dot{P}^L$ is the current velocity with the transportation rate dependent on the current position.

$$\dot{P}^L = -\omega_{EL}^L \times P^L + v^L \tag{3.3}$$

The bias and scale factor errors are important to estimate accurately, since well characterized errors allow the IMU to be far more accurate. To achieve this, the 6 bias and 6 scale factors for the accelerometer and gyroscope axes are modelled as GM processes [51]. The GM model is shown in (3.4), where n is the noise characteristic being modelled,

β is the inverse correlation time constant, σ is the standard deviation of the noise, and $w_u(t)$ is zero mean Gaussian noisy with unity variance.

$$\dot{n}(t) = -\beta n(t) + \sqrt{2\beta\sigma^2} w_u(t) \tag{3.4}$$

Combining the models together yields the state vector:

$$[(P^L)_{1\times3}, (v^L)_{1\times3}, [b, c, d], \quad (b_g)_{1\times3}, (b_a)_{1\times3}, (s_g)_{1\times3}, (s_a)_{1\times3}\ ]_{21\times1}$$

3.1.2 Measurement Model

In the measurement model, the pseudorange and pseudorange rates provided by the GNSS receiver are estimated and compared to the IMU predicted values. The pseudorange ρ^m measures the distance between a satellite and receiver using the time delay between transmission and reception. It is contaminated with errors including atmospheric delays and clock errors, and thus the pseudorange can be calculated as done in (3.5), where r^m is the true range, $c\delta t_r$ is the receiver clock bias, e_ρ^m is random noise, and A^m is atmospheric delays.

$$\rho^m = r^m + c\delta t_r + e_\rho^m + A^m \tag{3.5}$$

The true range is calculated as

$$r^m = \sqrt{(x - x^m)^2 + (y - y^m)^2 + (z - z^m)^2} \tag{3.6}$$

and x, y, z and x^m, y^m, z^m are receiver position coordinates and satellite position coordinates, respectively.

The pseudorange and pseudorange rate models are shown in (3.7), where $c\delta\dot{t_u}$ is the receiver clock drift, v is the receiver velocity vector, and v^m is the mth satellite velocity vector.

$$\begin{aligned} \delta\rho^m &= (x^m - x).\mathbf{1}^m + c\delta t_r + e_\rho^m \\ \delta\dot{\rho}^m &= -(v^m - v).\mathbf{1}^m + c\delta\dot{t_r} + e_\rho^m \\ \mathbf{1}^m &= \frac{[(x-x^m),\quad (y-y^m),\quad (z-z^m)]^T}{\sqrt{(x-x^m)^2 + (y-y^m)^2 + (z-z^m)^2}} \end{aligned} \tag{3.7}$$

This leads to the linearization (3.8).

$$\begin{bmatrix} \delta\rho^1 \\ \vdots \\ \delta\rho^M \\ \delta\dot{\rho}^1 \\ \vdots \\ \delta\dot{\rho}^M \end{bmatrix} = \begin{bmatrix} (\mathbf{1}^1)^T & 0_{3\times1} & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ (\mathbf{1}^M)^T & 0_{3\times1} & 1 & 0 \\ 0_{3\times1} & (\mathbf{1}^1)^T & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0_{3\times1} & (\mathbf{1}^M)^T & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta v_x \\ \delta v_y \\ \delta v_z \\ \delta b_c \\ \delta d_c \end{bmatrix} + \begin{bmatrix} v_\rho^1 \\ \vdots \\ v_\rho^M \\ v_{\dot{\rho}}^1 \\ \vdots \\ v_{\dot{\rho}}^M \end{bmatrix} \tag{3.8}$$

Which includes two new states $b_c = ct_r$, the clock bias, and $d_c = c\dot{t_r}$, the clock drift, resulting in the full state being

$$x = [(P^L)_{1\times3}, (v^L)_{1\times3}, [b, c, d], (b_g)_{1\times3}, (b_a)_{1\times3}, (s_g)_{1\times3}, (s_{ga})_{1\times3}, b_c, d_c\,]_{23\times1}$$

3.2 EKF Parameter Tuning

The INS/GNSS EKF system contains both parameters that are approximately measurable and parameters that are completely unknown. The process covariance matrix for instance, commonly denoted Q, contains the random walk error of the IMU sensors, which can be estimated through certain modelling techniques but cannot be calculated optimally. The measurement noise covariance matrix, denoted R, is unknown and can often only be tuned in a manual trial and error method. These matrices are of high importance since they directly impact the degree to which the filter favors one sensor reading over another, and small variations in some values may result in a complete divergence in the estimation. Therefore, optimal tuning of these matrices is strongly desired.

GM processes are used to model the bias and scale factor values of the INS and comprises the standard deviation and time constant of both the bias and scale factors [51]. Applied to every axis of every sensor in an INS, this leads to 24 GM parameters. AV is used to determine the random walk noise component of each INS reading, leading to 6 values [52]. The receiver clock bias and drift random walk (Qd matrix), and a scale factor for the variance of all satellite range readings (R matrix) are the tunable parameters relating to the GNSS, leading to a total of 33 tuning parameters as shown in Table 2.

Table 2 EKF Tuning Parameters

					X	Y	Z
INS	Gauss Markov constants	Gyro	Bias	Standard Deviation (G matrix)	**1**	**2**	**3**
				Time Constant (F matrix)	**4**	**5**	**6**
			Scale Factor	Standard Deviation (G)	**7**	**8**	**9**
				Time Constant (F)	**10**	**11**	**12**
		Accel	Bias	Standard Deviation (G)	**13**	**14**	**15**
				Time Constant (F)	**16**	**17**	**18**
			Scale Factor	Standard Deviation (G)	**19**	**20**	**21**
				Time Constant (F)	**22**	**23**	**24**
	Gyro			System Noise (Q matrix)	**25**	**26**	**27**
	Accel			System Noise (Q)	**28**	**29**	**30**
GNSS	Receiver clock			Bias Random Walk (Q)	**31**		
				Drift Random Walk (Q)	**32**		
	Range readings			Standard Deviation Scale (R matrix)	**33**		

3.3 Genetic Algorithms

GA are heuristic search techniques that are based on the theory of evolution and natural selection. GA have been proven to outperform grid-search methods in terms of computation. In addition, GA techniques avoid local minima problems by keeping diverse sets of solutions. Another benefit of GA is that they can be efficiently implemented on parallel computing platforms [53]. GA are well suited for the large search space and complex relationships of the EKF parameters. This GA is used to optimize the performance of the EKF, which is represented by minimizing a fitness function shown in Figure 3.2. The fitness function takes an input in the form of a parameter set and uses the parameters in the EKF to estimate a trajectory. The corresponding position, velocity, and attitude

ground truth solution for this trajectory is provided by a separate high-end INS/GNSS system, further described below. After the collection of both raw data and ground truth, the EKF tuning process is done offline. The difference between the EKF estimation and the known ground truth trajectory is used to produce the Root Mean Square Error (RMSE) of each state estimation. The RMSE of each state is then weighted and summed according to the equation in the Figure 3.2, where the state errors are shown as east, north, and altitude in meters; and roll, pitch, and heading in degrees. The fitness function takes into account the scale differences between meters and radians by a separate weighting factor for each RMSE. The relative impact of the heading error is then compensated by a larger weight relative to the weights of the other errors. These weights are chosen through trial and error to achieve the best pose RMSE as determined by the users.

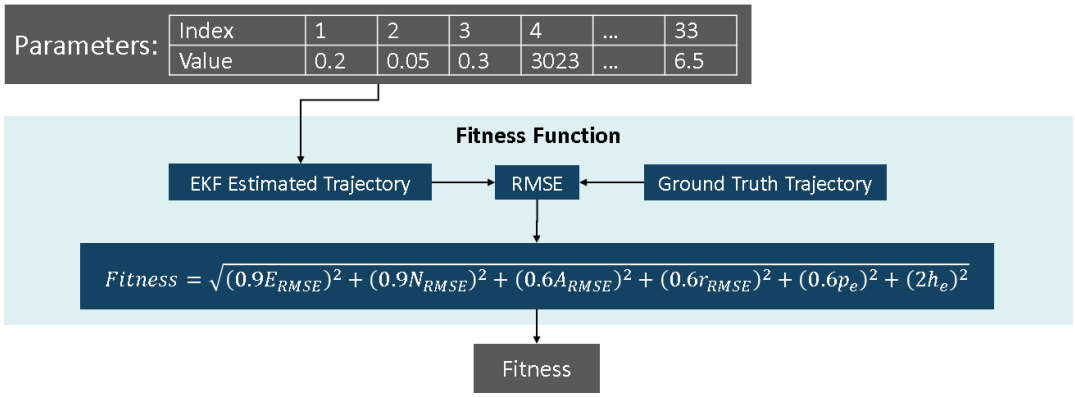


Figure 3.2 Block diagram of fitness function

The process of the GA starts with generating a population of parameter sets uniformly distributed between bounds. The upper and lower bounds are empirically selected to leniently encompass all feasible values of the parameters, and the population is set to 1000, chosen through trial and error. Each parameter set is evaluated by the fitness

function and ranked according to fitness. Next, two parents are selected with a probability based on their fitness value and reproduce offspring by the two-point crossover function. The two-point crossover technique selects two random locations in the parameter list for the parent delivering the parameters to switch. Some parents have their parameters randomly mutated by a small amount to increase diversity, allowing the algorithm to increase the search space. Crossover happens until the new generation is of the desired population size, from which the cycle repeats until the convergence of the population and best fitness is achieved. A standard GA run over 25 generations, and an example of the results is shown in Figure 3.3.

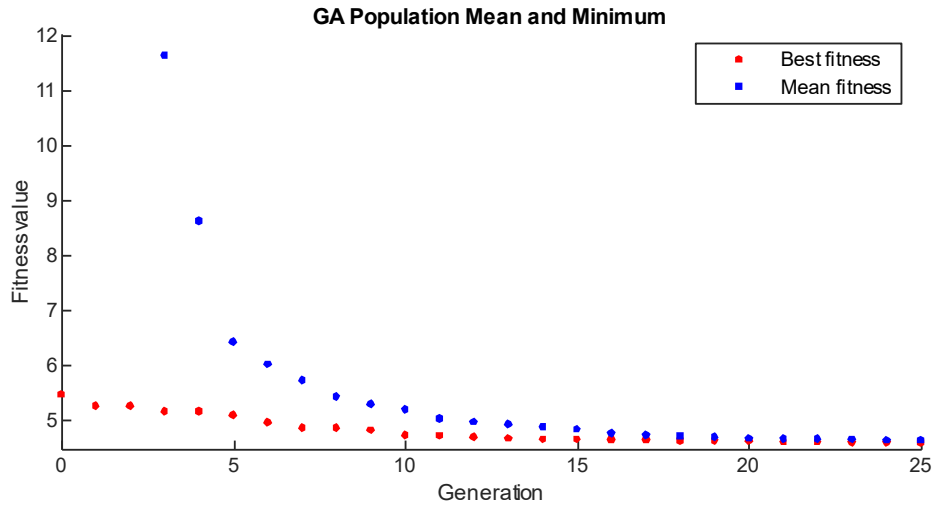


Figure 3.3 Standard GA results

3.4 Experimental Setup

To record IMU and GNSS data, a logger system developed in the Embedded Multisensor Systems Lab (EMSLab) was used. The EMSLab is a research group developing sensor fusion methods for robust and accurate positioning and navigation at

Carleton University under Professor Mohamed Atia. This logger contains the MPU 9250, a low-cost MEMS IMU from Invensense [54], and a ProPak6 GNSS receiver from Novatel [55] connected to a NVIDIA Jetson TX computer board. The logger is mounted on an automobile, shown in Figure 3.4. This experimental vehicle is the self-driving car platform operated by the QNX company (Ottawa-based).

Figure 3.4 Hardware setup

For data collection, the vehicle drove in a 17.5 km route with several stops, turns, and straight sections. The ground truth data was recorded with the Novatel Propak6 equipped with an OEM6 GNSS receiver and a high-end Fiber Optics Gyro (FOGs)-IMU KVH1750 IMU [56], utilizing the RTK technology of the Propak6. A portion of the route, shown in Figure 3.5, is used for the GA tuning. In addition to the vehicle test, a 7-hour stationary dataset for the MEMS MPU 9250 IMU was collected for noise modelling.

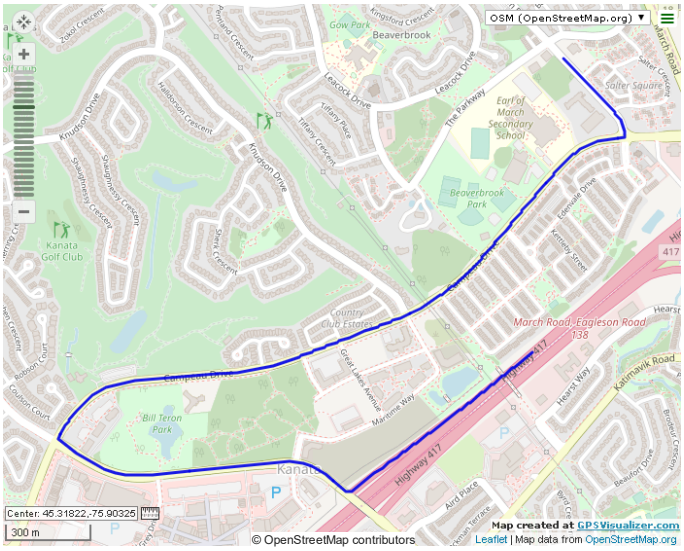


Figure 3.5 Trajectory

The data is processed, and the experiment is run using the following steps:

1) Gauss-Markov and Allan Variance methods are used on the stationary dataset to obtain an initial set of noise parameters called the "nominal design point".
2) The nominal design point is evaluated by estimating a trajectory using the parameters and comparing the estimated trajectory with the known ground truth.
3) The GA is then run with the fitness function to produce a GA-tuned set of parameters.
4) The GA-tuned parameters are evaluated in the same process as the nominal parameters.

3.5 Results

3.5.1 Nominal Design Point

First, GM and AV methods are applied to find the nominal design point. From the autocorrelation function of a 7 hour stationary IMU data recording, the GM bias time

constants and standard deviations are calculated and displayed in Table 3. The AV random walk is also displayed in Table 3. For the rest of the parameters, values were tuned manually by trial and error over a few iterations.

Table 3 Nominal parameters

	GM Standard Deviation			GM Time Constant (s)			AV Random Walk		
	X	Y	Z	X	Y	Z	X	Y	Z
Gyro (rad/s)	0.034	0.028	0.025	5194	4062	7439	0.13	0.14	0.14
Accel (m/s^2)	0.007	0.034	0.003	3619	10232	6319	0.02	0.02	0.03

3.5.2 Nominal and GA Results

The above nominal values are then used with the EKF to estimate the trajectory. The performance results in RMSE are displayed in Table 4. The GA is then run to tune all parameters, where individuals are defined as sets of 33 EKF parameters and genes are individual parameters. Figure 3.6 displays over 10 different runs of the GA on the system to display the variance of the results.

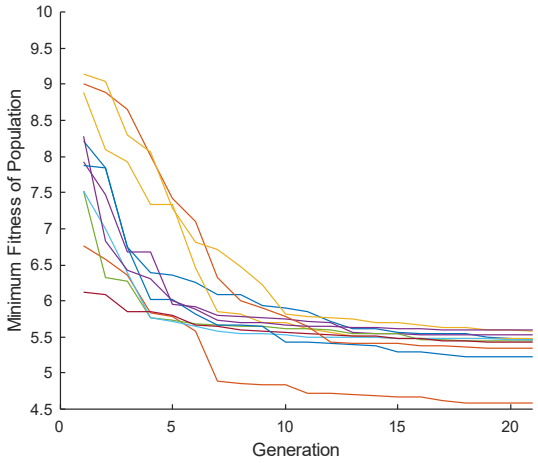


Figure 3.6 Multiple GA runs

Using the MATLAB GA function, with the population set to 1000 and crossover function set to two-point crossover, the GA tuned EKF performed significantly better than that of the nominal EKF. In the results shown in Table 4, GA particularly improved the heading divergence that was produced by the nominally tuned EKF.

Table 4 Training dataset RMSE results

RMSE	Nominal Tuned Parameters			GA Tuned Parameters		
	Position (m)	Velocity (m/s)	Orientation (deg)	Position (m)	Velocity (m/s)	Orientation (deg)
East/Roll	0.86	0.32	0.59	0.11	0.10	0.50
North/Pitch	2.20	0.66	0.51	0.29	0.18	0.33
Altitude/Yaw	0.55	0.11	7.88	0.14	0.08	2.66

3.5.3 Filter Validation

EKF validation is done to confirm that the tuned parameters allow the EKF to properly model sensor characteristics and converge to accurate estimations for multiple trajectories that have not been used in the tuning and have not been seen by the filter before to check for overfitting. Using a GA tuned parameter set, the performance results of an EKF estimation for two other trajectories are shown in this section. To distinguish the datasets, the dataset used to tune the parameters with the GA in the previous section will be called the “training dataset”, and the two new datasets will be called “test set #1” and “test set #2”. Information about these datasets are displayed in Table 5 and trajectories are shown in Figure 3.7 and Figure 3.8, which overlays trajectories on Google Maps.

Table 5 Datasets used

	Duration of data collection (s)	Distance driven (m)
Training dataset	310	4541.93
Test set #1	200	1905.83
Test set #2	170	1748.10

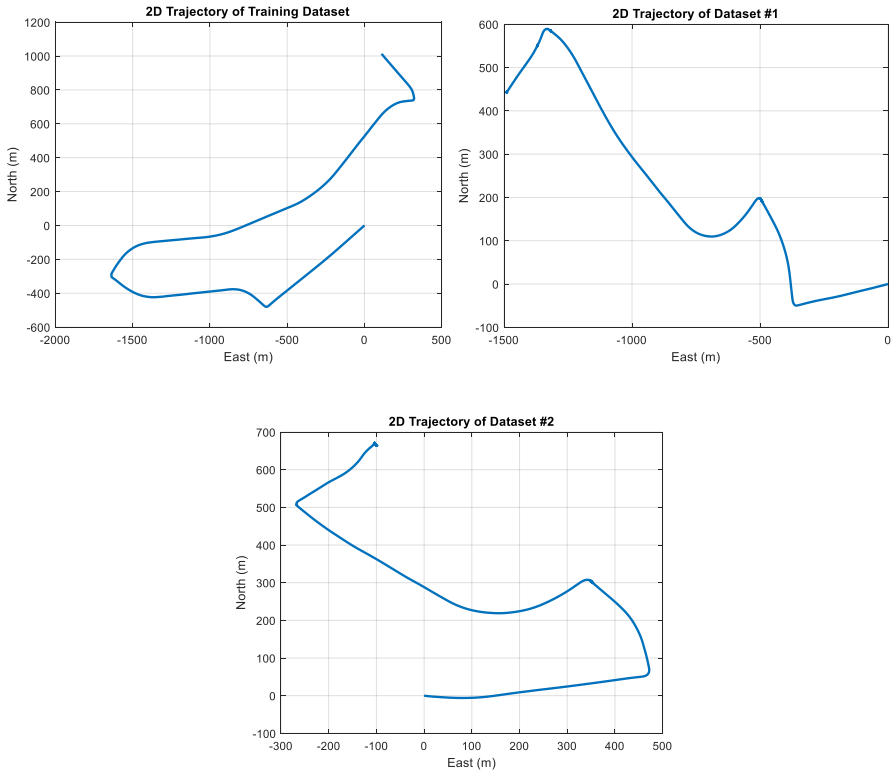


Figure 3.7: Trajectories used

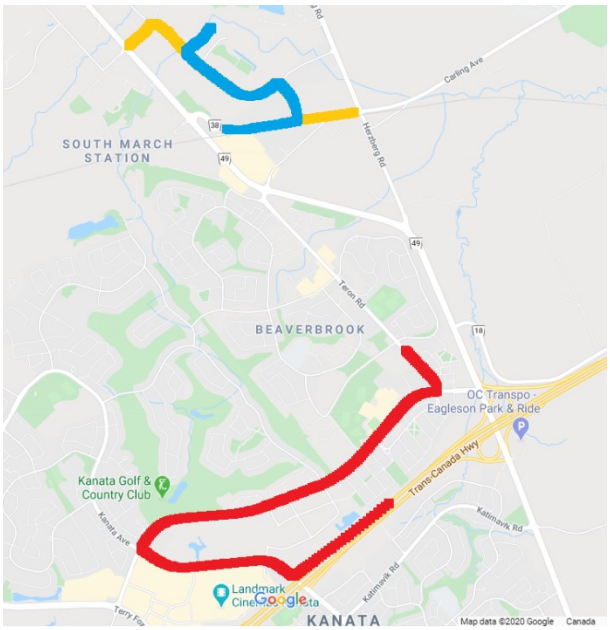


Figure 3.8: Training dataset (red), Test set #1 (yellow), and Test set #2 (blue) [57]

Table 6 show the RMSE results of using tuned parameters in the EKF for test sets #1 and #2. The parameters used have been tuned by the GA on the tuning dataset only. The low error shows that the tuned EKF performs well for a variety of trajectories in addition to the dataset used to tune the parameters which verifies that there is no overfitting and the tuned filter can be used for other data sets that have not been used in the tuning process.

Table 6 RMSE results for test sets

RMSE	Test set #1			Test set #2		
	Position (m)	Velocity (m/s)	Orientation (deg)	Position (m)	Velocity (m/s)	Orientation (deg)
East/Roll	0.09	0.09	0.35	0.10	0.10	0.48
North/Pitch	0.16	0.14	0.38	0.12	0.12	0.72
Altitude/Yaw	0.12	0.08	2.35	0.12	0.08	2.90

To further evaluate the proposed tuning technique, horizontal position RMSE under partial or complete outage with respect to the number of visible satellites, traveled distance,

and outage duration are shown in Figure 3.9. Outage scenarios were designed with different durations (10 to 30 seconds with 5 seconds increment steps) and different numbers of visible satellites (0 to 4). Each outage scenario was repeated ten times on different regions of the dataset and the mean was recorded. As expected, compared to complete outage, observing even a single satellite considerably contributes in bounding the error growth. Only under complete outage, non-holonomic constraints of the vehicle were applied to the velocity state updates. Figure 3.10 shows the EKF solution against the ground truth solution during a 30 second GNSS artificial outage.

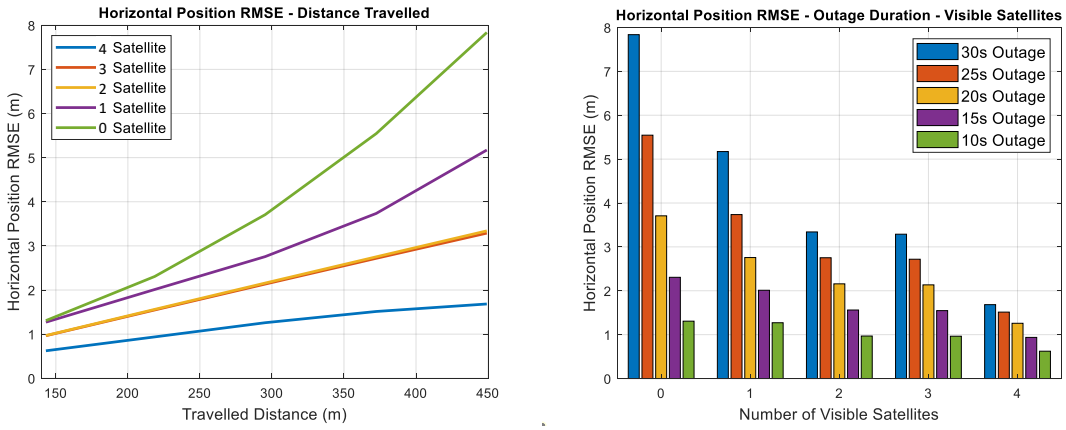
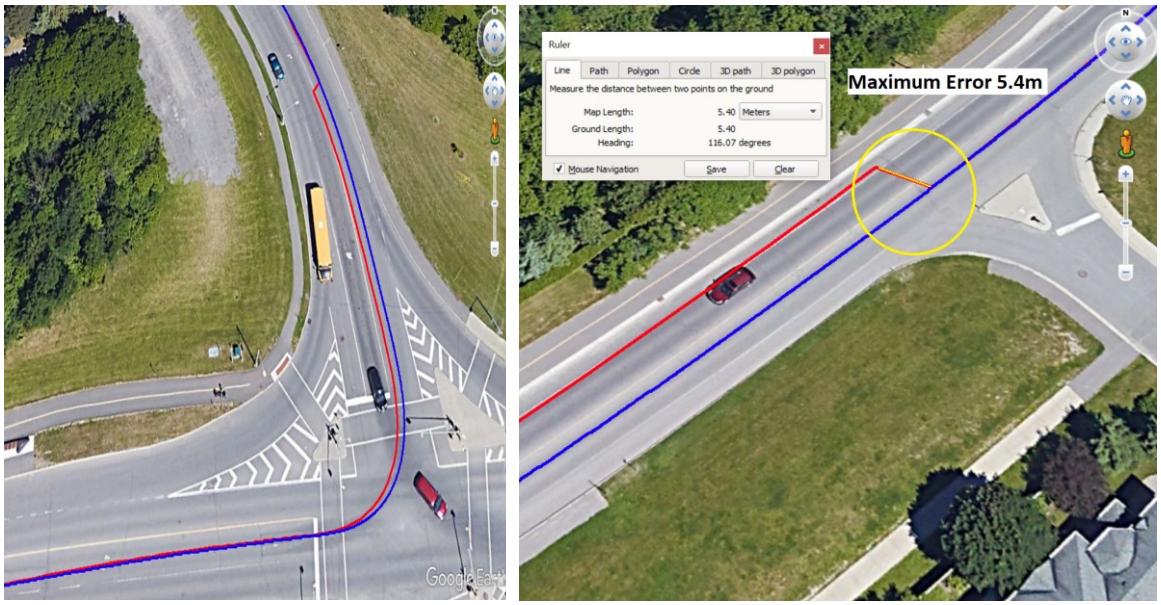


Figure 3.9 Outage Test for Test set #1



(a) (b)

Figure 3.10 (a) Outage during a turn (b) Max. error at end of an outage [58]

3.6 Outcome

This work demonstrates the localization ability of a genetically tuned IMU/GNSS EKF fusion system. The work binds conventional stochastic modeling techniques with GA in an efficient manner to obtain a more accurate EKF. In the experimental results, the nominal parameters performed satisfactorily, but the GA tuned results significantly improved upon the nominal parameters. The filter was validated through observing estimations on new datasets and GNSS outage performance.

Chapter 4 LiDAR Graph SLAM for Indoor Localization and Mapping

SLAM as a method of localization was introduced in Chapter 2. In this chapter, an experimental study of graph SLAM using a 360-degree 2D LiDAR with wheel odometry is explained, with results demonstrating its effectiveness in geometrically rich areas and deficiencies in a long hallway trajectory. Graph SLAM is an efficient SLAM implementation due to the possibility of parallelizing graph building and graph optimization. Furthermore, it continuously optimizes current and past trajectories based on new measurements and loop closures, leading to an overall elegant SLAM system.

4.1 Graph Optimization using Least Squares Formulation

To optimize the node locations in a pose graph, the optimization procedure introduced in Chapter 2 is adopted. The error function is defined as taking the homogenous transformation matrix between two node poses in homogenous transformation matrix form X_i and X_j and multiplying it with the transformation matrix associated with the edge between the nodes Z_{ij}, as shown in (4.1), converted into vector form using the function $t2v$. The function $t2v$ maps a 2D homogenous transformation matrix $\begin{bmatrix} r_{11} & r_{12} & d_x \\ r_{21} & r_{22} & d_y \\ 0 & 0 & 1 \end{bmatrix}$ to a vector pose $[d_x \quad d_y \quad \text{atan2}\,(r_{21}, r_{11})]$ corresponding to the x translation, y translation, and θ rotation.

$$e_{ij}(x) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \tag{4.1}$$

In a fully optimized graph, if the initial position is node i, the resulting transformation moves from node i to node j back to node i, with the net pose change being zero. In the case of a discrepancy between the transformation between nodes and the transformation determined by an edge, there will be a small net transformation. This error transformation matrix is converted to vector form, which ensures that the error evaluates to zero in the optimal case. The goal is to minimize the squared error term

$$x^* = argmin_x \sum_i e_i(x)^T \Omega_i e_i(x) \tag{4.2}$$

The initial locations of the nodes are the odometry estimates or the results of the last optimization performed. The error function is linearized, and the Jacobian of the error function is found through numerical methods.

$$J_{ij} = \frac{\partial e_{ij}(x)}{\partial x} \tag{4.3}$$

Note that J_{ij} only depends on the nodes i and j, since error e_{ij} only depends on nodes i and j. The minimum error is calculated according to the following least square formula:

$$\Delta x^* = -H^{-1}b \tag{4.4}$$

where the b and H terms are found using the Jacobian and the error function for every edge and summed as described in 2.3.3. Using a linear systems solver such as Lower Upper

or Cholesky solver provides the step towards the minima. This is repeated until the change in error over iterations is less than a small threshold $\varepsilon = 0.00001$.

4.2 Hardware and Experimental Setup

The hardware used in the experiments consist of a Turtlebot 3 Waffle Pi [59]. In this thesis, the platform was customized by adding a Texas Instrument AWR1843BOOST radar unit [16] as shown in Figure 4.1. The Turtlebot system is used to perform localization in an indoor environment using graph SLAM and particle filtering approaches with LiDAR and radar sensors.

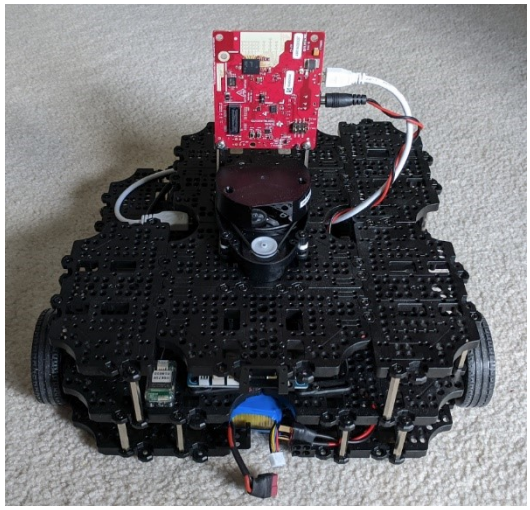


Figure 4.1 Turtlebot3 with AWR1843BOOST

4.2.1 Turtlebot Hardware

The Turtlebot 3 Waffle Pi is a robotic platform that includes a 360-degree 2D LiDAR, two Dynamixel driven wheels in a differential drive configuration, a 6 axis IMU, and a Raspberry Pi Camera as sensors. The robot is controlled with a Raspberry Pi 2B and

an OpenCR controller board for distributing power and controlling the motors. Power comes from a 3 cell 11.1V 1800mah lithium polymer battery. The Turtlebot and its components are shown in Figure 4.2. The Turtlebot Raspberry Pi runs the Raspberry Pi OS, a Linux operating system designed for the specifications of the Raspberry Pi hardware [60]. The pre-installed software includes the Robot Operating System (ROS) and ROS packages [61] that connect and control the sensors and motors. ROS also includes packages that provide localization and mapping solutions using sensor measurements.

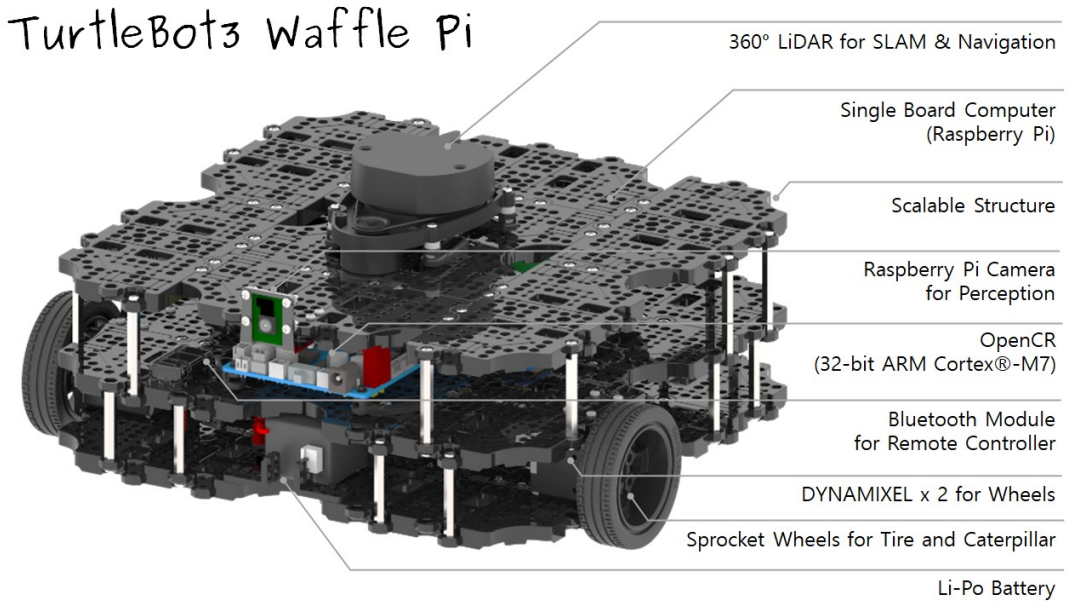


Figure 4.2 Turtlebot3 components [59]

4.2.2 Turtlebot Device Interfaces, Data Collection, and ROS

In ROS, sensor data is published to ROS topics, which are bus type systems unique to the type of data it holds. Any program can view and retrieve the data by subscribing to the desired topics. Sensor data is formatted into ROS messages, which specify the structure of the data contained.

Data is recorded into a txt file through a subscriber that subscribes to every sensor topic chosen, which includes topics for the wheel odometry, IMU, LiDAR, radar, and the transforms message, which contains the transformations to reference points including SLAM and localization generated reference points used for the ground truth. The angular rotations and linear accelerations from the IMU are recorded at 200 Hz, the scans from LiDAR are recorded at 5 Hz, and the wheel rotational positions and velocities, the radar scans, and transform messages that provide ground truth pose data are recorded at 30 Hz independently.

In this work, the robot is used to collect data and provide ground truth solutions using Adaptive Monte Carlo Localization (AMCL) [62], a localization system requiring a known map, and Cartographer [43], a robust SLAM algorithm developed by Google for 2D LiDAR and wheel odometry. Cartographer is a 2D LiDAR and wheel odometry developed SLAM solution that uses grid maps and graph SLAM. Figure 4.3 shows a screen capture of Cartographer mapping an area.

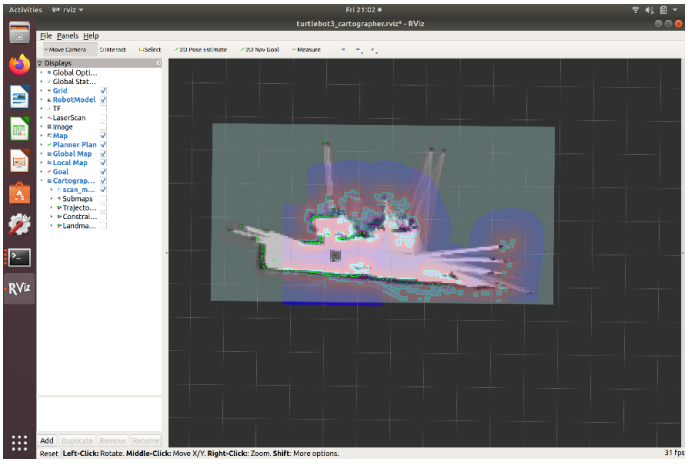


Figure 4.3 View of real time Cartographer

LiDAR SLAM is an established as accurate and robust in locations with rich geometry, and thus is usually a good reference solution to compare with the developed systems.

4.2.3 Data Collection Process

The ROS platform allows for distributed computing. As such, a stationary laptop computer is used remotely for online SLAM work, robot control, and data interfaces. A complete data collection session involves the following: first, the ROS core program is started on the host machine, a Microsoft Surface Pro 4 in these experiments, which begins a collection of programs that are prerequisites of a ROS system. Then, the sensors and actuator interfaces and the ground truth navigation programs are started. All required data are now published and able to be saved, so the data collection program is run to record sensor data and ground truth poses. Lastly, the teleoperation command is run, and the robot is driven manually through a course.

4.3 Experimental Results

This graph SLAM implementation was developed for a LiDAR and wheel odometry system with an offline MATLAB program. Data was collected by the Turtlebot system in two different trajectories. The first trajectory involves a traversal of a small environment with rich geometry. The second trajectory involves long hallways with minimal features suitable for LiDAR scan matching. The trajectory estimations in these experiments are evaluated and compared with two sources of ground truth, which will be explained in the following sections.

4.3.1 LiDAR Graph SLAM Performance in Feature-Rich Environment

For the first trajectory, Figure 4.4 shows the results on using this system compared to Cartographer SLAM.

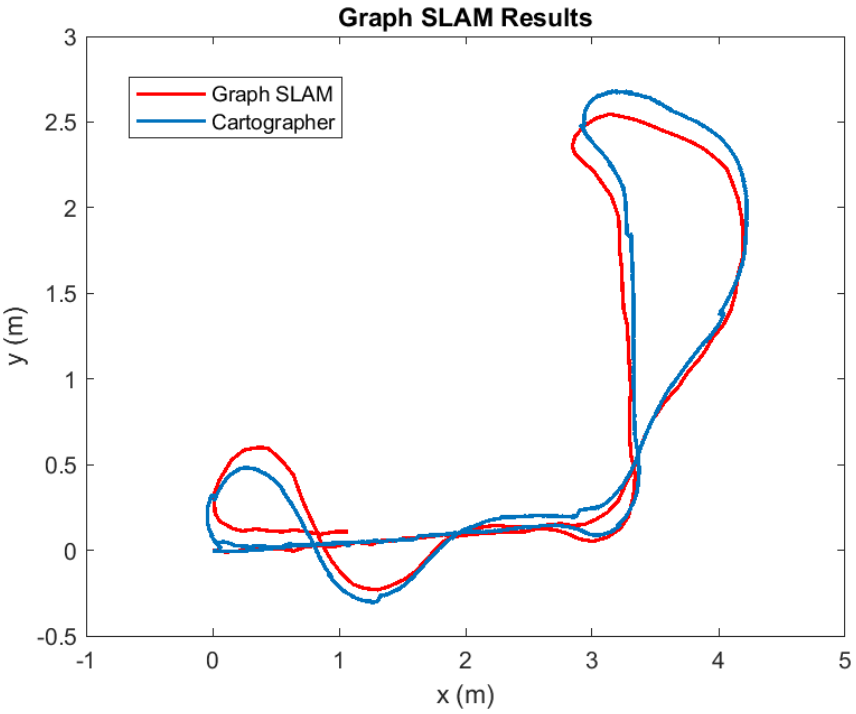


Figure 4.4 Graph SLAM results compared with Cartographer

In this experiment, the Cartographer performance does not appear to be a good ground truth estimate. The map created by projecting LiDAR scans from pose trajectories reveals that the Cartographer SLAM misaligns the scans to the top left of the map, while the graph SLAM implementation results in LiDAR scans correctly overlapping for most of the map (Figure 4.5).

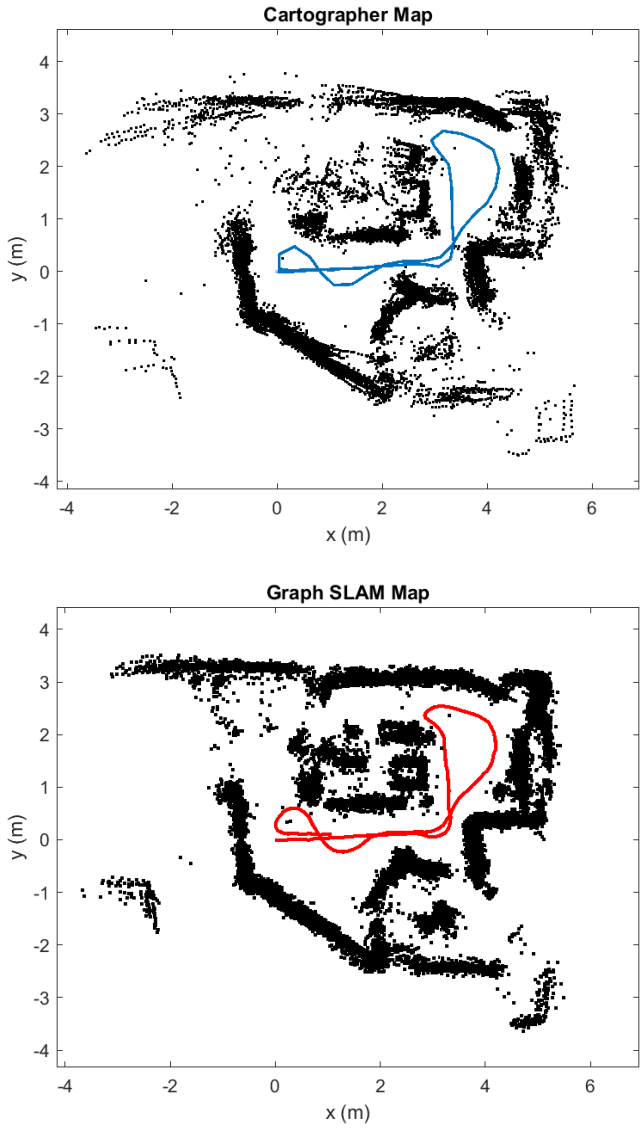


Figure 4.5 Scans projected by Cartographer and graph SLAM

The implemented graph SLAM approach performs well in this environment due to the rich geometry of the area seen by the LiDAR and the constant revisiting of previously travelled areas. This both provides many opportunities for adding edges and further constraining the graph as well as good scan candidates for well performing scan matching. Figure 4.6 shows all the node poses and edges that correspond to scan matching and odometry measurements.

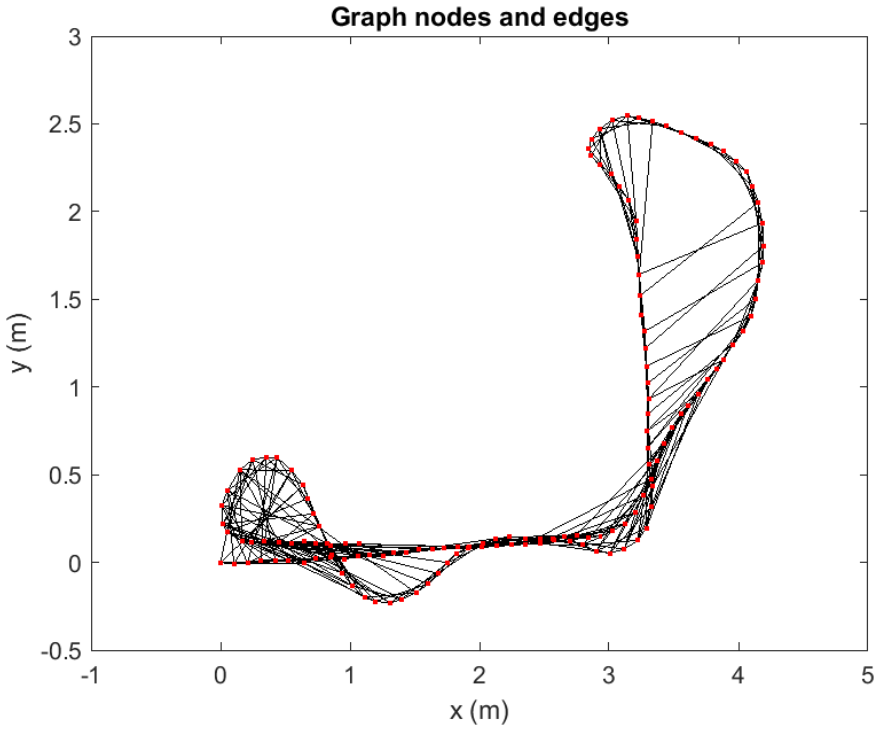


Figure 4.6 Graph SLAM results showing nodes and edges

4.3.2 LiDAR Graph SLAM Performance in Long Hallways

Applying the LiDAR graph SLAM to a long hallway type environment leads to a very poor estimation, seen in Figure 4.7. Long hallways are difficult for LiDAR scan matching due to the environment lacking features to match against. The ground truth in this trajectory is created by first mapping the area with Cartographer, then performing localization using a known map with AMCL. By driving the Turtlebot in a trajectory deliberately for the purpose of map building [63], a better map can be created and the AMCL, which relies on a prebuilt map, can provide a better ground truth.

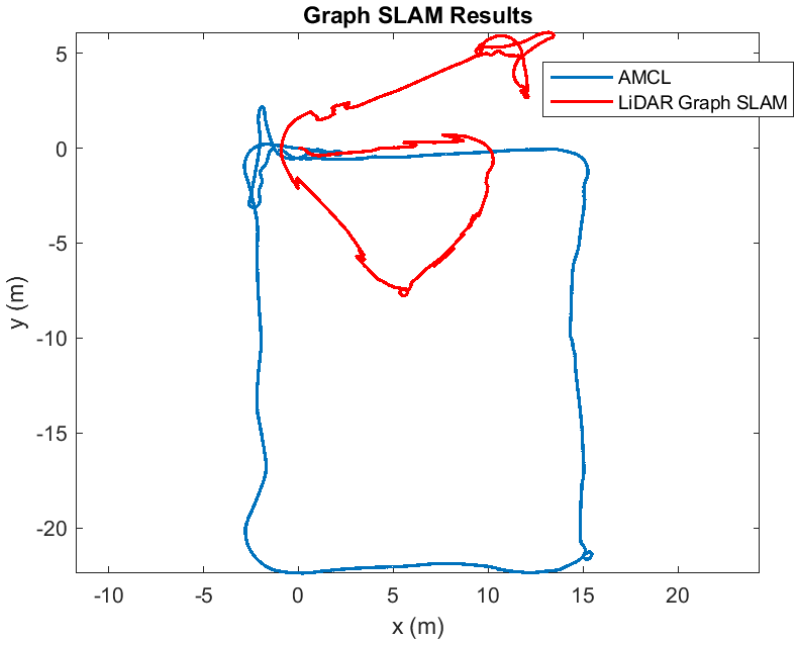


Figure 4.7 Graph SLAM results vs AMCL in long hallways

4.4 Radar Limitations in Graph SLAM

To address LiDAR limitations in long hallways, the use of radar sensors is considered. In implementing graph SLAM with radar, the main difficulty lies in loop closure. Matching radar scans mostly involve matching clusters to each other rather than lines and curves as seen from LiDAR scans. This results in a much larger ambiguity in correct associations and results in most scan matches to be false positives. Radar scan matching works well in local areas, such as in incremental scan matching. But every long edge seen in Figure 4.6 requires a high global certainty in scan matching over large areas, which radar cannot provide reliably. This is one issue in uncertainty representation that is solved by using a particle filter. In particle filters, the particles are each considered a candidate for the best trajectory, and do not need to undergo scan matching over large search areas to close loops. The implementation of the particle filter with a radar sensor is discussed in Chapter 5.

Chapter 5 Radar Particle Filter for Indoor Localization

As seen from experimental results in Chapter 4, LiDAR SLAM performs poorly in environments with simple geometry. As hallways are common in indoor settings, this is a significant drawback. In 2.1.5, several assets of using radar sensors are listed, including capability of penetrating airborne obstructions, physical sturdiness, and large range. Accordingly, the motivation is set for creating a radar SLAM system that takes advantages of radar characteristics and overcomes flaws inherent in using LiDAR sensors.

In this work, a radar SLAM system is implemented using the particle filtering approach described in 2.3.4. This particle filter algorithm is chosen for its strength in modelling non-parametric distributions, which is suitable for a radar SLAM application because of the irregularities in radar measurements resulting in multimodal distributions in some situations. See Figure 5.1 for an example of a situation with multimodal distributions from radar scan matching due to a map with crowded occupied cells.

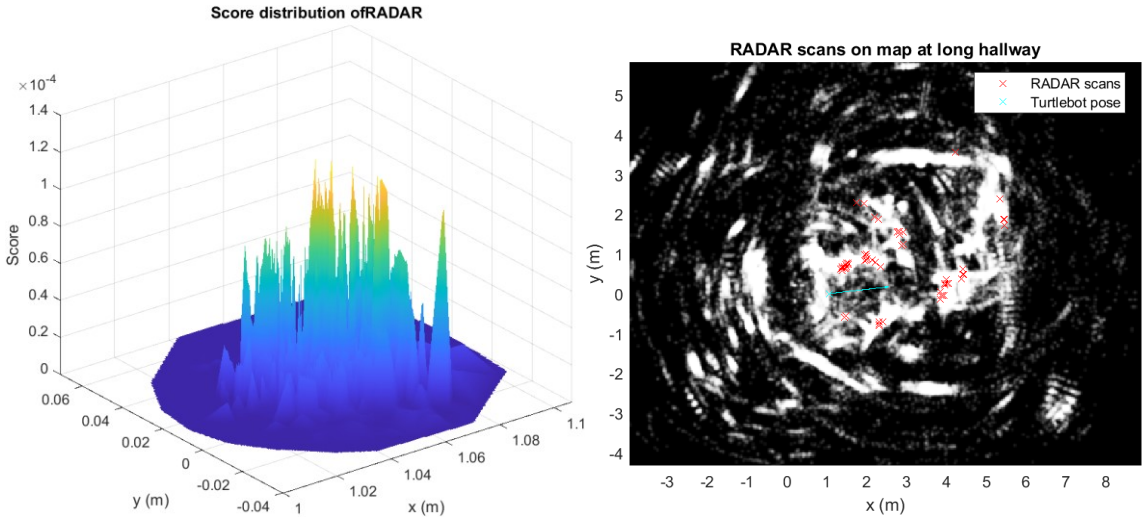


Figure 5.1 Radar Scan matching visualization and score distribution in crowded map

Furthermore, a grid map based technique is used since the irregularities of the cluster density and shape of radar scans, the difficulty in creating descriptors for a cluster feature, and the sparsity of clusters in some areas constitutes major difficulties in creating a robust feature detector. Finally, loop closure using ordinary radar scan-matching is challenging due to the ambiguity of matching clusters, and the approach that particle filtering uses to close loops does not require scan matching with a large search space. When particles arrive in previously mapped areas, simply the particle map with the best fitting overlap is weighted the most, ensuring a high probability of the particle being chosen through resampling. A high level block diagram of the radar SLAM system is shown in Figure 5.2, where gyroscope, wheel odometry, and radar scans are fused using a particle filter to estimate pose and maps.

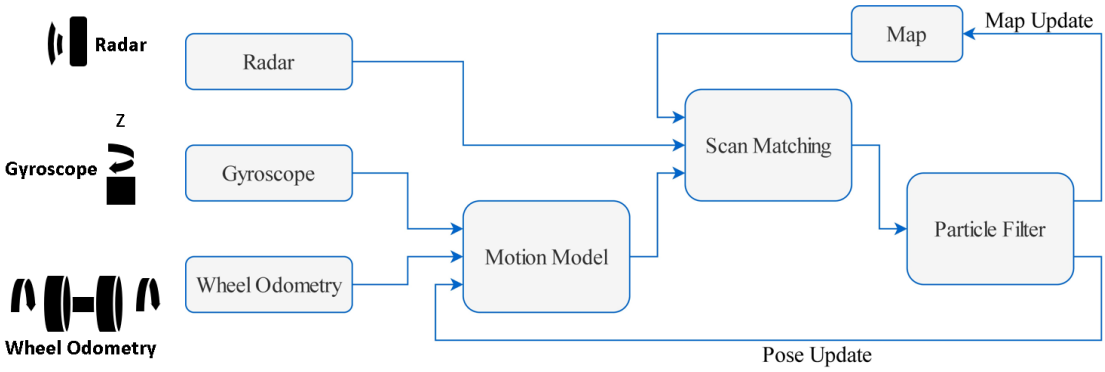


Figure 5.2 Block diagram of radar SLAM

5.1 Radar Sensor Information and Analysis

The Texas Instrument AWR1843BOOST is a radar sensor evaluation module for the TI AWR1843 radar sensor. The module operates in the 76-81 GHz range and uses an

array of antennas (3 transmitters and 4 receivers) to emit and detect radar waves from a range of directions. With the wavelengths being roughly 4mm, these types of sensors are known as mm-wave radar [10]. The AWR1843 sensor includes an ARM Cortex microprocessor and C674x DSP hardware to perform all low-level radio control systems and signal processing in addition to high level features like object tracking and point cloud processing. The evaluation module contains everything necessary to power and interface with to the radar sensor. With a USB connection, high level output data including object categorization, object localization, and point cloud data can be recorded, and configuration data can be sent to the sensor to adjust output data characteristics. In this work, the 2D point cloud data is used for all processing.

5.1.1 Radar Sensor Configuration

The TI AWR1843 chirp configuration [64] can be modified to suit different design specifications. Figure 5.3 displays the many parameters that can be modified to change sensor characteristics on a time vs frequency plot.

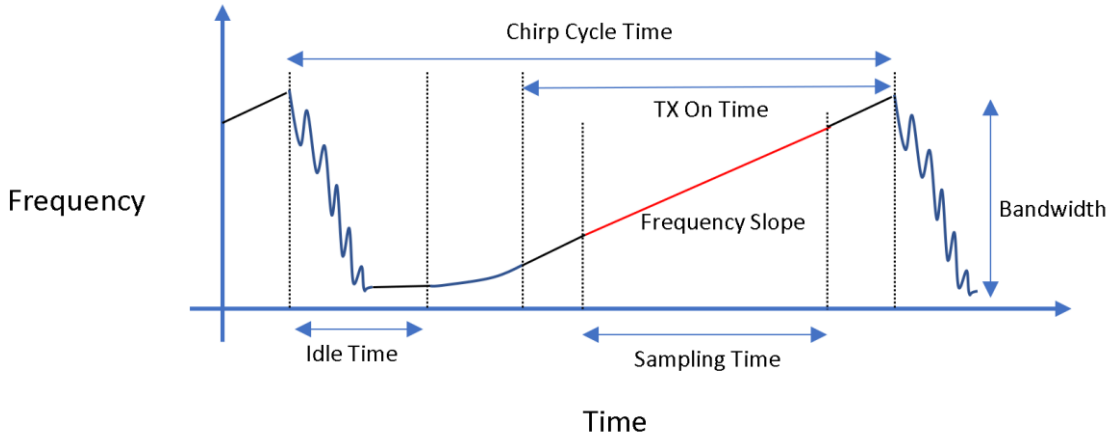


Figure 5.3 Chirp configurable parameters [64]

Configurations are set by modifying a .config text file with the keywords and values chosen by the user. Quantities including maximum range, range resolution, angular resolution, velocity resolution, maximum velocity, and Constant False Alarm (CFAR) threshold, which is a method for removing noise [65], are all adjustable by changing configuration parameters. The configuration of the chirp heavily impacts the quality of the point clouds. The configuration used in all experimental results presented in this thesis was chosen to detect high densities of points at ranges of 4-6 m, and was arranged to have a 6 m maximum range, sub-cm precision, 0.3 deg angular resolution, and 180 deg angular range.

5.1.2 Device Interface

The AWR1843 radar sensor is connected to the Turtlebot through USB and is powered by a separate 5V barrel connector from the power distribution and motor control board. Texas Instruments provides a ROS package to configure and stream radar output data from their devices [66]. The AWR1843 radar sensor must be flashed with the appropriate firmware prior to using the ROS package. Texas Instruments provides different firmware setups for different applications, including tracking vehicles on a highway, proximity automated parking, and monitoring driver vital signals.

The default radar ROS message had several issues including missing data and a bit-encoded format. To make reading the radar data easier, a message type “Radar_scan” was created in a similar format to the existing LaserScan message, where point cloud data is stored as 32-bit floating-point arrays. The text file that determines the msg format is shown as shown in Figure 5.4:

```
Header header
int32 width
float32[] x
float32[] y
float32[] z
float32[] intensities
float32[] velocities
```

Figure 5.4 Radar point cloud scan ROS message format

Note that the header stores time stamp data. Next, the radar ROS publisher was modified to output this message type. In the radar ROS package, the "DataHandlerClass.cpp" program reads UART data that the radar streams, converts this to the desired ROS message, and publishes the message to the desired topic. This program was modified to store radar data into floating-point arrays and publish to the radar scan topic.

5.1.3 Radar Data

The radar outputs a point cloud with between 0-100 points at 30Hz. While the point cloud may contain detections of objects and their location similar to a LiDAR, the radar output cannot directly replace a Lidar scan in most applications. The sparsity and the noise of the point cloud, along with the widely varying number of points, results in scan matching efforts failing consistently. Figure 5.5 shows the sparsity of a single radar scan and noisy data in 20 overlapped radar scans compared to a typical LiDAR scan, with the corresponding environment pictured.

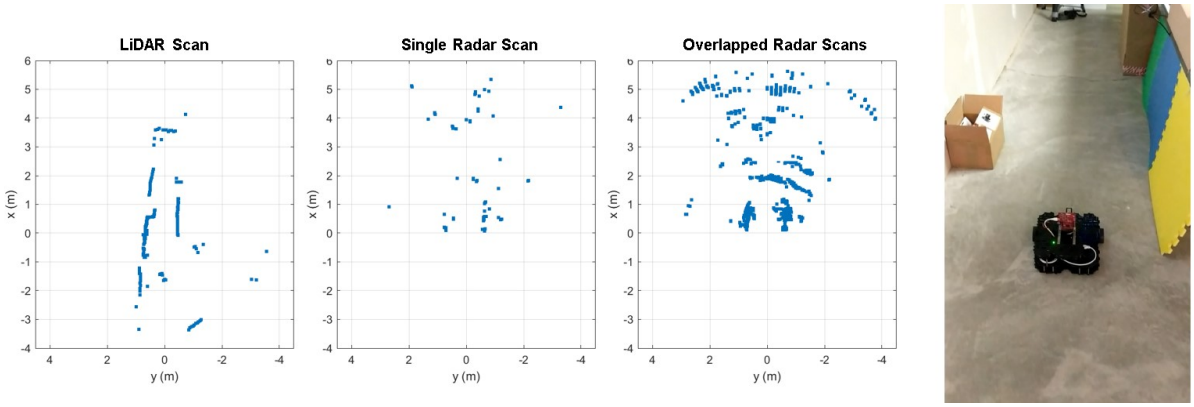


Figure 5.5 Radar scans compared to LiDAR scan

To provide more insight on the mechanisms of the radar point cloud, Figure 5.6 shows radar scans projected onto a ground truth trajectory in a long hallway environment. Note that the ground truth trajectory is not perfect since the LiDAR scans do not show perfectly straight hallways. The environment in Figure 5.6 consists of a drive around the Mackenzie Building 5th floor hallways.

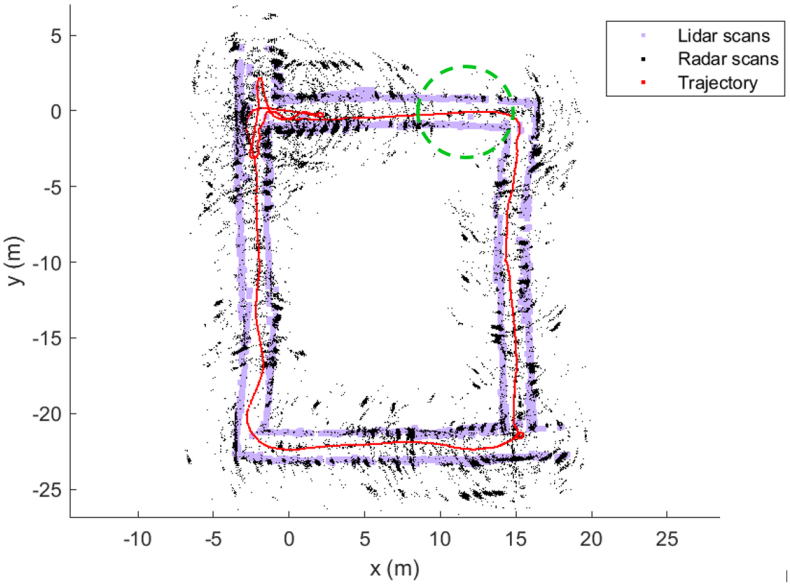


Figure 5.6 Radar and LiDAR data projected on AMCL ground truth solution at the 5th floor of the Mackenzie Building at Carleton University

Figure 5.6 effectively shows the extent of how radar point clouds can be used. Over many consecutive scans, dense clusters of points are revealed in key locations. These clusters likely correspond to metal struts in the walls, doorframes, windowsills, and other large metal structures. In other locations, such as inside the green circle, there are very few detections. Additionally, since the robot drives clockwise in this trajectory, it can be seen that the radar provides more detections on non-metallic objects when obstacles are directly in front of the radar, seen at every corner at the wall in front of the robot before the turn has ben made. At these locations, many more detections are visible on those walls compared to walls anywhere else. This point assumes that in these areas there is not more metal behind walls than usual. Many detections also do not lie on observable obstacles, such as clusters behind walls and smaller clusters on the floor. Most of these clusters are the effects of multipath and floor reflections from radar waves emitted downwards. Large clusters behind the wall are multipath reflections since they are often symmetrical to large clusters on the actual walls, with the opposing wall being the line of symmetry. This is apparent at (3,3) and (14,-25).

The clusters of detections are of interest since they can potentially be parameterized and used to distinguish strongly and weakly trackable obstacles. The cluster shapes are irregular, with many appearing to have tapered curves with the curvature pointing to the sensor as the center. Since the radar only provides 180 degrees of angular range, the map appears to have semi-circles that point in the direction of travel. This suggests that the radar has high noise associated with the angular component compared with the distance

component of a detection. Processing of clusters is a possible direction of study but is not implemented in this work due to their highly irregular and inconsistent shapes.

One critical observation is that, while radar detections may be used to track sensor location, radar maps may not fully represent occupied and unoccupied space for the purpose of navigating through free space. This is a limitation that may necessitate an additional sensor to provide reliable sensing of free space in certain situations where navigation decisions are made using only sensor data.

5.2 Comparisons of Radar vs Lidar Scan Matching

This experiment compares the performance of using LiDAR and radar scans for localization. Localization is performed using a scan matching algorithm that is developed through a correlative method. Experiments have been performed in a long hallway scenario. As the results will show, radar scan matching can maintain better pose confidence in the direction of a straight hallway where a similar LiDAR system commonly fails. The main reason for this is that radar emissions reflect off different materials with different intensities and can penetrate through some materials. Therefore, radar scans often can find features to track in places where a LiDAR may find straight walls, in situations like long hallways. The non-homogeneous makeup of the walls, wall mounted objects, and metal doorframes provide clusters that can be tracked despite appearing as smooth surfaces to a LiDAR. In the sections with long hallways, LiDAR scans show 2 parallel lines. However, radar scans consistently show clusters along the walls. The clusters may be doorframes, pictures, or support structure within the walls.

5.2.1 Radar Scan Matching with Clustered Scans in Long Hallways

As single radar scans are sparse and noisy, 40 scans are projected using wheel odometry and combined to a single update to maintain a usable density of points. The short distances covered through the combined radar scans produces minimal odometry error. Figure 5.7 displays one such combined scan overlaid on a partially mapped radar occupancy map.

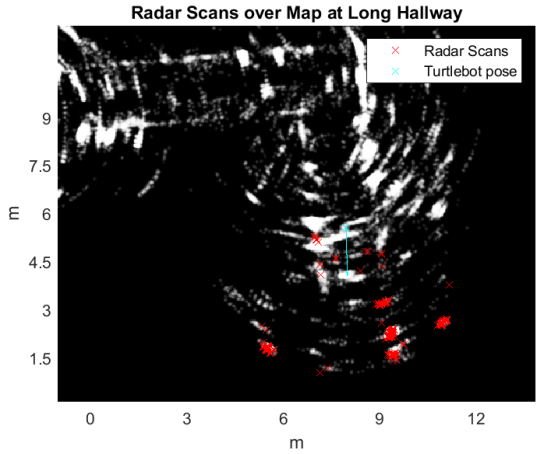


Figure 5.7 Combined radar scans overlaid on radar occupancy map

By combining scans, several visible clusters are revealed and can be used as reliable mapping features. Cluster outliers are not ruled out as noise because they may be detections for clusters in a future scan, and do not strongly affect the scan matching process. Scan matching results are shown for a long hallway in Figure 5.7. Figure 5.8 also shows the score distribution of scans matching over a small area around the best score.

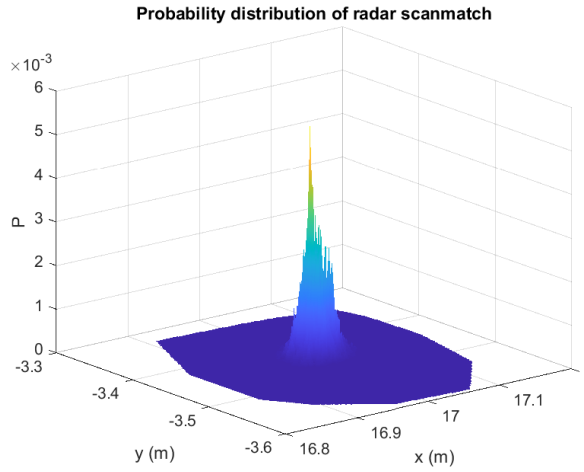


Figure 5.8 Score distribution of radar scan matching

The distribution shows that the probability of a match is peaked, which suggests high confidence in all dimensions of the position. One major flaw of the radar is that at certain locations even the combined scans are too sparse. These locations are brief but result in the trajectory diverging. Odometry can be used in these scenarios to take over, and since the instances of poor scan density are brief, using odometry estimates can maintain high certainty.

5.2.2 LiDAR Scan Matching Ambiguity in Long Hallways

LiDAR scans reveal the shape of the objects around the sensor accurately. In situations where the geometry is varied, LiDAR scan matching results are very accurate. However, when the LiDAR advances through a long featureless hallway, the scan matching process produces large uncertainty along the direction of the corridor. Figure 5.9 shows the scenario of scan matching in a long corridor.

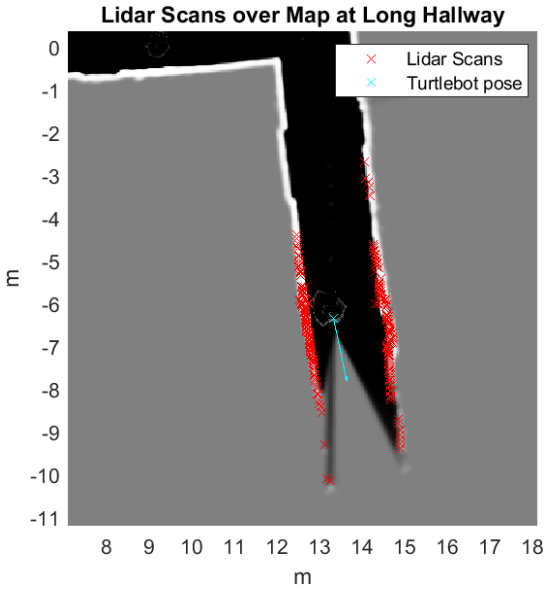


Figure 5.9 LiDAR scans on LiDAR map

The red scan measurements may be translated up the hallway with no large reduction in score. Furthermore, if the true location is down the hall, the matching process will prefer to remain in place since the unmapped areas produce smaller scores. This is the case in Figure 5.9 where the odometry mean estimate places the y location at -6.46 m, while the highest peak of the LiDAR scan matching is located at -6.37 m. The corresponding distribution of scan matching scores over a small area is shown in Figure 5.10, where a ridge and multiple peaks are seen along the direction of the hallway.

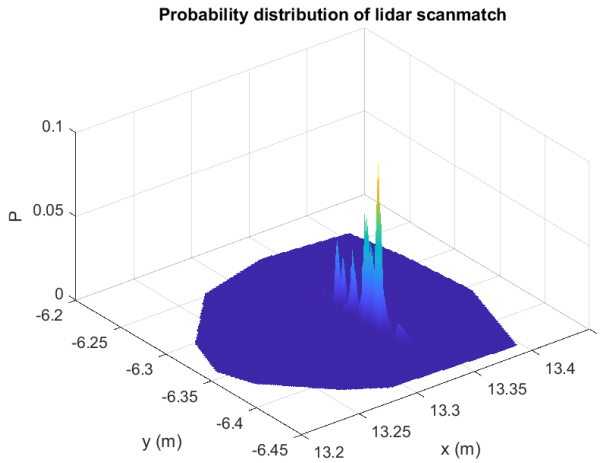


Figure 5.10 Score distribution of LiDAR scan matching

5.2.3 Scan Matching Localization Accuracy, Radar vs. LiDAR

Figure 5.11 shows the full odometry aided scan matching trajectories for LiDAR and radar scans, where radar scans with insufficient richness are ignored. The root-mean-square-error (RMSE) for each trajectory is given in Table 7.

Table 7 RMSE of scan matching trajectories

	X (m)	Y (m)	Heading (deg)
Wheel Odometry	4.7467	2.9070	28.8035
Lidar	2.8415	8.3401	6.6853
Radar	0.4603	1.4835	6.5133

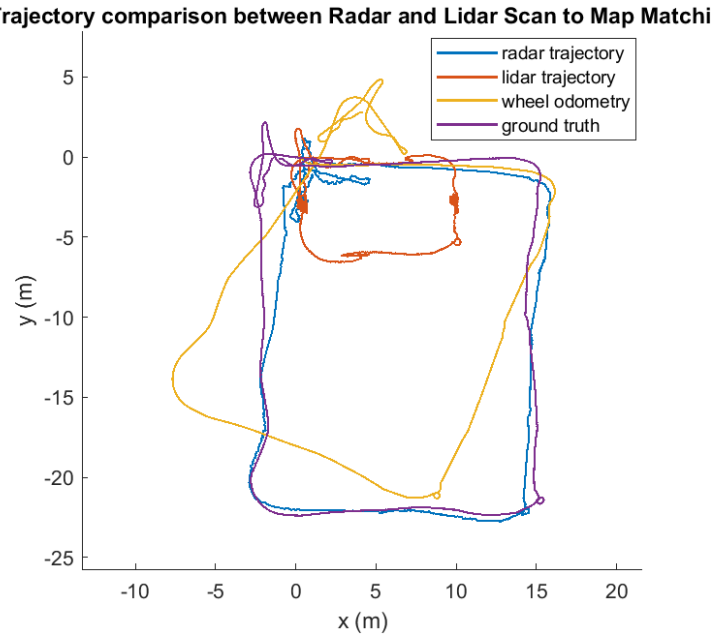


Figure 5.11 Trajectory comparisons

The LiDAR trajectory tracks heading and rotations well, however, collapses the long corridors into short zigzags, while the radar trajectory is able to maintain localization along the hallways and remain much closer to the correct path. In this experiment, a radar scan matching system demonstrates superior performance in areas where a LiDAR system would struggle. In long hallways where a LiDAR sensor would detect straight parallel lines, a radar sensor would see periodic clusters, providing a means of localizing in the direction of the hallway. While most odometry fusion techniques involve some weighted averaging between scan matching results and odometry estimates in the final pose calculations, this experiment does not use odometry except as an aid for initializing the scan matching process. This reveals the effectiveness and issues of scan matching radar and LiDAR scans alone.

5.3 Radar SLAM Implementation

The developed system to perform radar SLAM using a particle filter is described in more detail in this section. The pipeline of the implementation is provided as a flowchart in Figure 5.12, where subsections 5.3.1 to 5.3.7 are dedicated to explaining each major block in the order following the diagram.

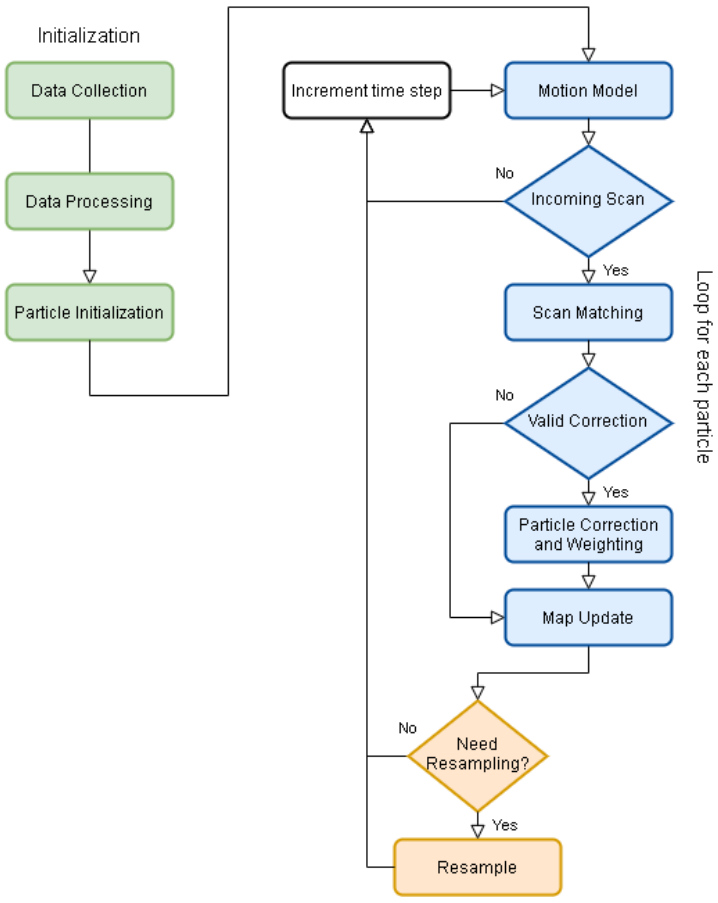


Figure 5.12 Flowchart of radar SLAM process

5.3.1 Data Preprocessing and Time Synchronization

Data is first collected from the Turtlebot as explained in 4.2.2. Then, a MATLAB program processes the data and estimates the trajectory using the developed SLAM

algorithm. The recorded data is first processed to produce augmented radar scans and synchronized wheel rotational velocities, gyroscope angular rate, and forward velocity calculated from wheel velocities.

Augmented scans are radar scans combined by projecting their locations according to wheel odometry trajectory estimations. By combining a small number of scans, the odometry contributes little error to the system. Figure 5.13 shows a comparison of a single radar scan, 8 radar scans, and 40 radar scans overlaid on an example radar map. The number of scans to consider is an arbitrary tuning parameter that can affect the performance of the SLAM system. It is found that between 10 to 40 scans provide good results.

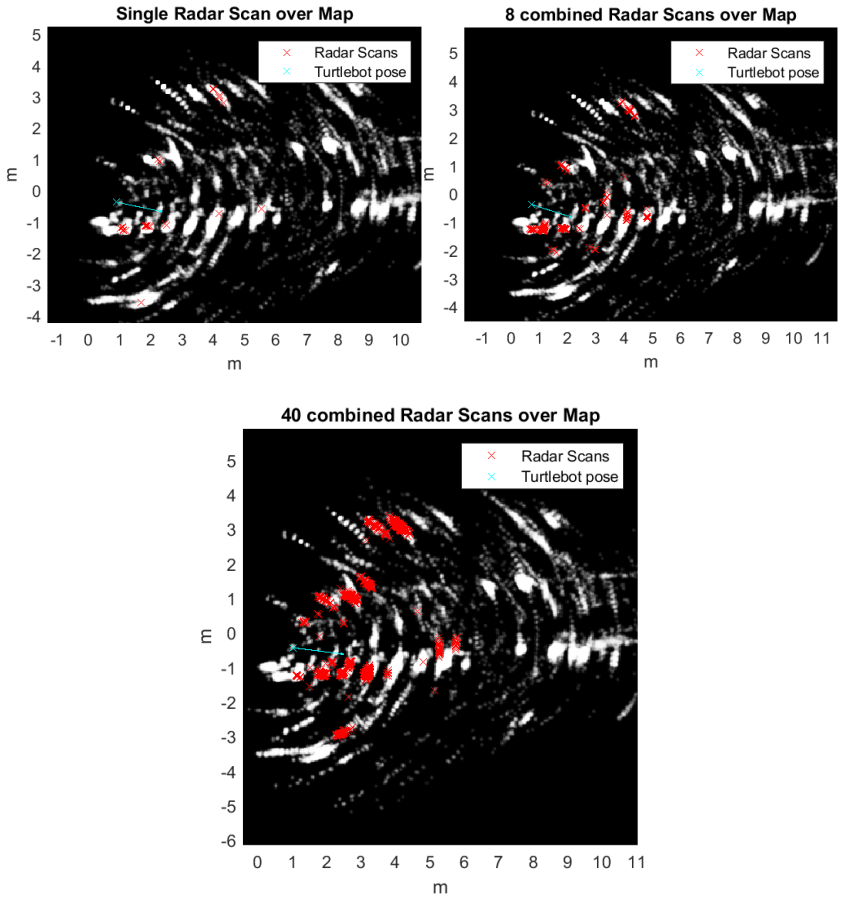


Figure 5.13 Different number of combined scans matched to a radar map

The augmented scans are considered as scan measurements with the scan time being the time of the first single radar scan in the augmented scan. Notably, the augmented scans contain several clusters of points, which allow the process of scan matching to a map updated with previous scans to be much more effective compared using the single scan.

Wheel angular velocities and gyroscope angular velocities are directly provided by the recorded file. Forward velocity is calculated from the wheel rotational velocities as follows:

$$v = \frac{W}{4}(\omega_r + \omega_l) \tag{5.1}$$

where W is the Turtlebot wheel diameter in meters and ω_l and ω_r are left and right wheel angular velocities in radians per second.

Wheel velocities, forward velocity, gyroscope angular rate, and the ground truth pose trajectory are interpolated at predetermined timesteps to synchronize the data across a common time vector. The predetermined timesteps increment at 20 Hz with augmented scan times inserted when they occur. Figure 5.14 shows a diagram of this synchronization process, where the black vertical lines represent the times at which continuous data is interpolated.

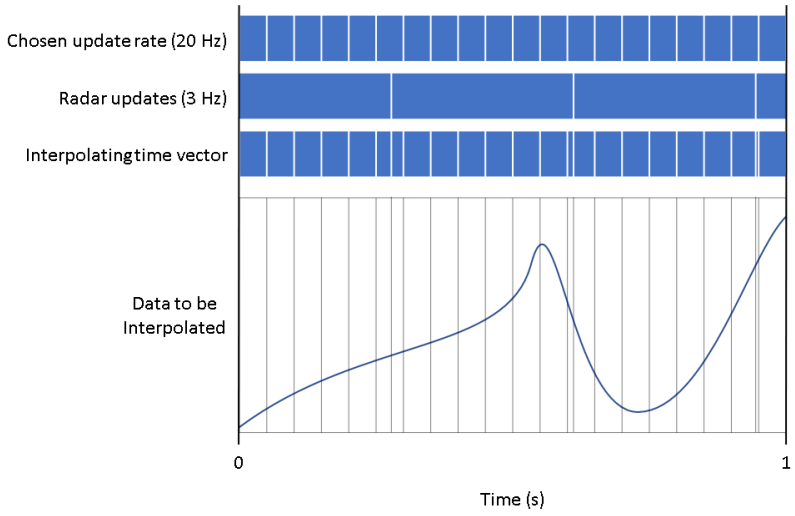


Figure 5.14 Time synchronization process

5.3.2 Particle Initialization

The initial pose of the robot is assumed to be known with full confidence. 10 particles are initialized at the origin with a heading of 0 degrees and given an initial map that incorporates the first scan at the origin using the log odds map update equations described in 2.2. The map is implemented as a 2D array with the size of the cells set to represent 0.03 m^2 areas, which is found to be a balance between computation time and map precision. The origin of the map, which is the initial position of the Turtlebot, lies at an array coordinate dependant on the size and formation of the map. Because of this, the grid origin as an array coordinate must be stored and updated when the map changes size to accommodate new scans. Figure 5.15 shows an initial map with the axes labelling the position coordinates in meters and the data tip labelling the row and column array coordinates of the origin.

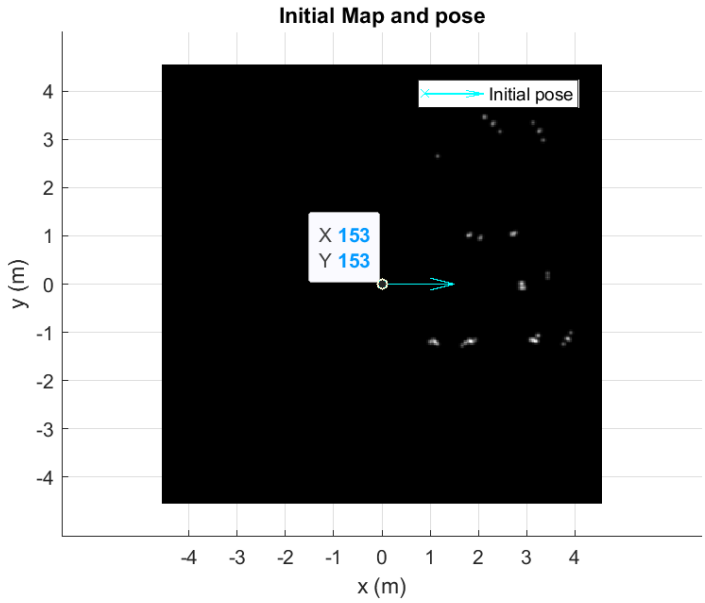


Figure 5.15 Particles initial map

Particles are stored as structures that consists of the following data: pose, map, pose covariance, weight, and grid origin coordinate. Particle weights are initially equal and normalized.

5.3.3 Motion Model

At every timestep, every particle is advanced using the motion model. The radar SLAM motion model consists of two motion models fused together to provide a strong short-term estimate of position, which is useful for the scan matching step. One motion model uses wheel angular velocities sent in radians/second as inputs and estimates pose and covariance, shown in the following equations.

$$x_{final} = x + \begin{bmatrix} \frac{W}{4}(\omega_r + \omega_l)\Delta t \cos(x_3) \\ \frac{W}{4}(\omega_r + \omega_l)\Delta t \sin(x_3) \\ \frac{W}{2D}(\omega_r - \omega_l)\Delta t \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & -\frac{W}{4}(\omega_r + \omega_l)\Delta t \sin(x_3) \\ 0 & 1 & \frac{W}{4}(\omega_r + \omega_l)\Delta t \cos(x_3) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$$\Omega_{final} = G\Omega G^T + Q_w$$

where W is the Turtlebot wheel diameter in meters, G is the linearized motion model matrix, D is the distance between the two wheels, ω_r and ω_l are left and right wheel angular velocities, x is the pose state for x-position, y-position, and heading angle, Q_w is the process noise associated with the model, Ω is the covariance matrix for the pose uncertainty, and x_{final} and Ω_{final} are the resulting pose state and covariance matrix, respectively.

The second motion model, a velocity model, uses a combination of wheel angular velocities and gyroscope rate of rotation to also estimate the pose and covariance. Initially, a 2D IMU model was developed as the second motion model. However, the low speeds of the Turtlebot, which drives around 0.1-0.2 m/s, result in very short and low acceleration values. The noise floor of the accelerometer exceeded the acceleration values, and even with the use of low pass filters, any meaningful velocity and position data were imperceptible. The gyroscope had a far lower noise floor and produced good sensor data to estimate heading. The velocity motion model equations are shown below, using forward velocity v and angular velocity ω as inputs.

$$x_{final} = x + \begin{bmatrix} -\frac{v}{\omega}\sin(x_3) + \frac{v}{\omega}\sin(x_3 + \omega\Delta t) \\ \frac{v}{\omega}\cos(x_3) - \frac{v}{\omega}\cos(x_3 + \omega\Delta t) \\ x_3 + \omega\Delta t \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & -\frac{v}{\omega}\cos(x_3) + \frac{v}{\omega}\cos(x_3 + \omega\Delta t) \\ 0 & 1 & -\frac{v}{\omega}\sin(x_3) + \frac{v}{\omega}\sin(x_3 + \omega\Delta t) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$\Omega_{final} = G\Omega G^T + Q_v$$

where x_3 is the pose heading.

In the motion step, two separate predictions for Turtlebot pose μ and covariance Σ are calculated by the two models. They are combined through a product of Gaussians, which uses the covariance matrices and mean values to produce a weighted overall mean and covariance, shown in (5.4). The product of Gaussians is a simple and effective way of combining two Gaussian sources of information and is mathematically equivalent to the Kalman filter [67].

$$\begin{aligned} \Sigma &= \left(\Sigma_1{}^{-1} + \Sigma_2{}^{-1}\right)^{-1} \\ \mu &= \Sigma\Sigma_1^{-1}\mu_1 + \Sigma\Sigma_2^{-1}\mu_2 \end{aligned} \tag{5.4}$$

For every timestep that does not contain a radar scan, the dual motion models are used to advance the position and covariance matrix for each particle.

The input data for each motion model, consisting of the wheel velocities and gyroscope z axis, are coupled with random variables drawn from a normal distribution with covariance Q_w for the wheel model and Q_v for the velocity model. These are tunable

parameters that impact the development of the distributions of particles. This step is necessary since the particle filter uses the particles to model the uncertainty of the trajectory.

5.3.4 Radar Scan Matching

Once a timestep is reached where an associated augmented radar scan exists, scan matching is performed using the incoming scan, existing map, grid origin, current pose estimate, and covariance matrix associated with the particle. Scan matching is done in the correlative method where the size of the search space is scaled according to the current particle covariance with the center being the current pose estimate. The search space is divided into a grid which is aligned with the occupancy grid map. The heading angular steps are set to the smallest angle that causes the furthest scan point to change cells. The grid step and angle that maximizes the scan matching score is the optimal pose correction as calculated by the scan matcher.

The scan matcher fails under certain conditions with sparse scans, which led to the development of a thresholding process to determine if a scan matching process is valid. Failed scan matches often present as pose corrections that have large heading changes or large lateral movement, and thus a threshold is set to ignore scan match results that differ greatly in heading estimates from the initial estimate or lateral motion with respect to the current heading. In these cases, the particle continues with the existing motion model estimation as the current pose and updates the map (see 5.3.6) from this pose before the system processes the next particle. If the scan matcher outputs an acceptable pose

correction, the particle continues to the importance weighting, which fuses the scan matching results with the current odometry estimations.

5.3.5 Importance Weighting

In this step, the particle pose is corrected with the scan match results and its weight is calculated according to the particle filter theory explained previously. The proposal distribution and weighting require the function τ, which is the product of the observation distribution (scan matching distribution) and the motion distribution (motion model distribution). To achieve this, 1000 new samples are drawn from a 3-dimensional Gaussian distribution with mean at the scan match corrected position, and covariance according to the particle state covariance. Each of these samples are evaluated in the same method as in scan matching, using (2.12). The same samples are also used to create the Gaussian approximation of the motion model. The Gaussian formula (5.5) is used to evaluate the probability distribution of the sample x given the mean pose $\bar{x}$ and covariance Σ, which are set to be the motion model pose estimate and covariance from the last motion model computation.

$$g(x) = \frac{e^{-\frac{1}{2}(x-\bar{x})\Sigma(x-\bar{x})}}{\sqrt{(2\pi)^3|\Sigma|}} \tag{5.5}$$

Figure 5.16 shows the resulting scan matching and motion model distributions. They are not normalized to maintain their weighting, since normalizing distributions discards overall score information and only retains shapes of distributions.

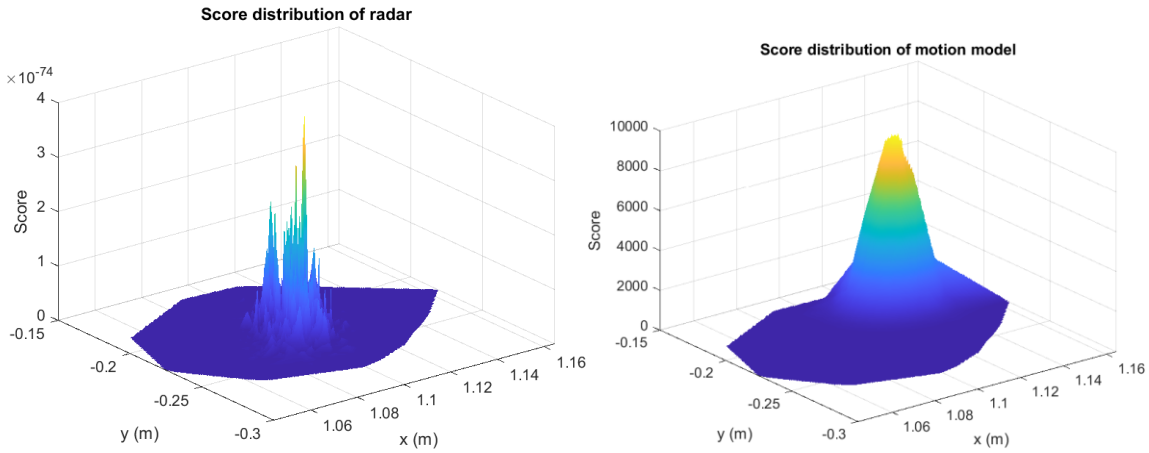


Figure 5.16 Score distributions for the scan matching and motion model

When not normalized, scan matching scores have very small values due to method of multiplying probabilities (numerical values between 0 to 1) for each point in a large point cloud. Quantization errors were not observed in these values but may need to be considered if the process is done on a different setup.

The distributions of the scan matcher and the motion model are then element-wised multiplied to find the τ distribution. The τ distribution is normalized through dividing by its sum. Through the multiplication and normalization, the small values of the scan matching distribution are eliminated, but the shape and overall score still properly impacts the resulting distribution. The final particle location is chosen by sampling from τ (Figure 5.17), using the MATLAB function randsample, using samples as the population, and tau as weightings.

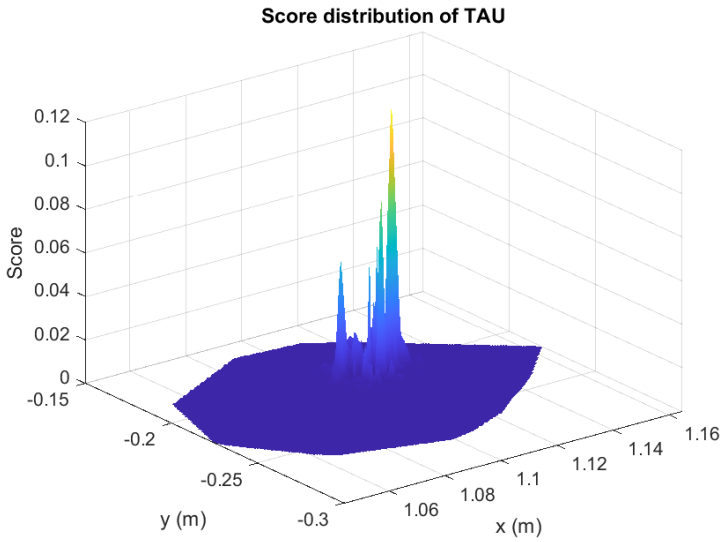


Figure 5.17 Normalized score distribution of TAU

The weight of the particle itself is the sum of tau. The weight describes how well the scan match scores and how closely the scan matching distribution agrees with the motion model distribution.

5.3.6 Map Update

Next, the map and grid origin are updated using the corrected particle pose and the scan. The radar scans are updated to the map in a similar way to the method in 2.2. Due to the large noise associated with every measurement and the fact that radar can penetrate objects, the radar does not reveal any information anywhere other than at the point of detection. Therefore, detection points are assigned to $p_{occupied} = 0.7$, a gaussian blur is applied, and then the measurement map is added to the radar map in log odds. The radar inverse sensor model uses a larger gaussian blur to model the large uncertainty of a radar detected point and does not update values using the beam model since no information is gained from cells behind or in front of the detections. Once the map is updated, the

covariance of a particle is reset to its initial value, and the loop is repeated for the next particle.

5.3.7 Resampling

After every particle has been processed in the above steps, the particle weights are normalized. The variance of the particle weights determines how well the target distribution is being approximated and if resampling is necessary. Resampling too often results in particle depletion, which is the elimination of viable particles. To remedy this, the inverse variance of the normalized particle weights is checked against a threshold to determine whether resampling needs to be done [47].

$$\frac{1}{\sum_i (w_t^{[i]})^2} < N/2 \tag{5.6}$$

The inequality evaluating to true means that for the N samples, weights w for particles i differ enough that the particles do not represent the target distribution well. This causes resampling steps to happen in situations where loops are closed or previously seen environments are recognized, since particles in accurate locations are likely to score a larger weight. Resampling is done using the MATLAB randsample function. At this point, one timestep has been processed and the loop begins again for all particles.

5.4 Experimental Results

In this section, analysis and results of the processing and trajectory estimations using the radar SLAM algorithm are displayed and discussed. The experiments start with

the Turtlebot driving through three trajectories while gathering data. The same hardware and data collection processes described in 4.2 are used in these experiments, with the addition of the radar sensor and collection processes. From the recorded data, the necessary inputs to the SLAM algorithm are generated. The SLAM algorithm is run, and the trajectories are created. The estimations are compared with the AMCL LiDAR/odometry localization solution or Cartographer SLAM solution and with the motion model alone.

5.4.1 Trajectory A

The first test involves a loop in a 6 x 8 m area. Figure 5.18 displays the AMCL estimated trajectory and both LiDAR and radar data projected using the AMCL estimate. Several radar clusters are present, which suggests that the environment has sufficient radar detections for a successful SLAM estimation. In some locations, there appears to be an overabundance of radar detections. This potentially causes the radar to drift in scan matching, since a larger uncertainty in the scan matching causes the trajectory to bias towards following the motion model. Additionally, a large amount of detections is seen behind walls.

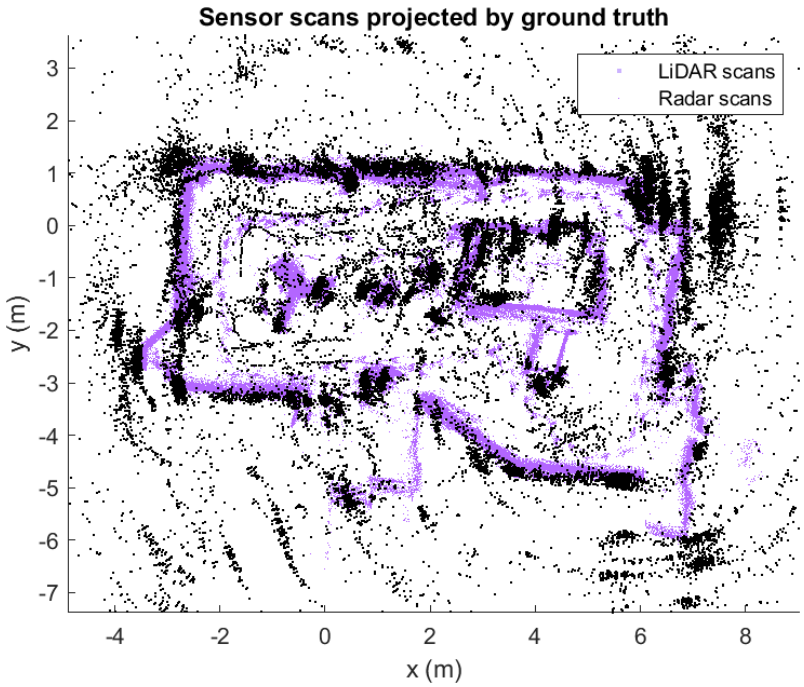


Figure 5.18 AMCL ground truth scans for trajectory A

Figure 5.19 displays the particle distribution without resampling and with resampling, which shows the highest weighted particle compared to the AMCL solution and the motion model. The bulk of the particles appear to follow a path slightly diverging from the ground truth. However, the high weight particle is the most accurate trajectory, and maintains an accurate estimation over the loop.

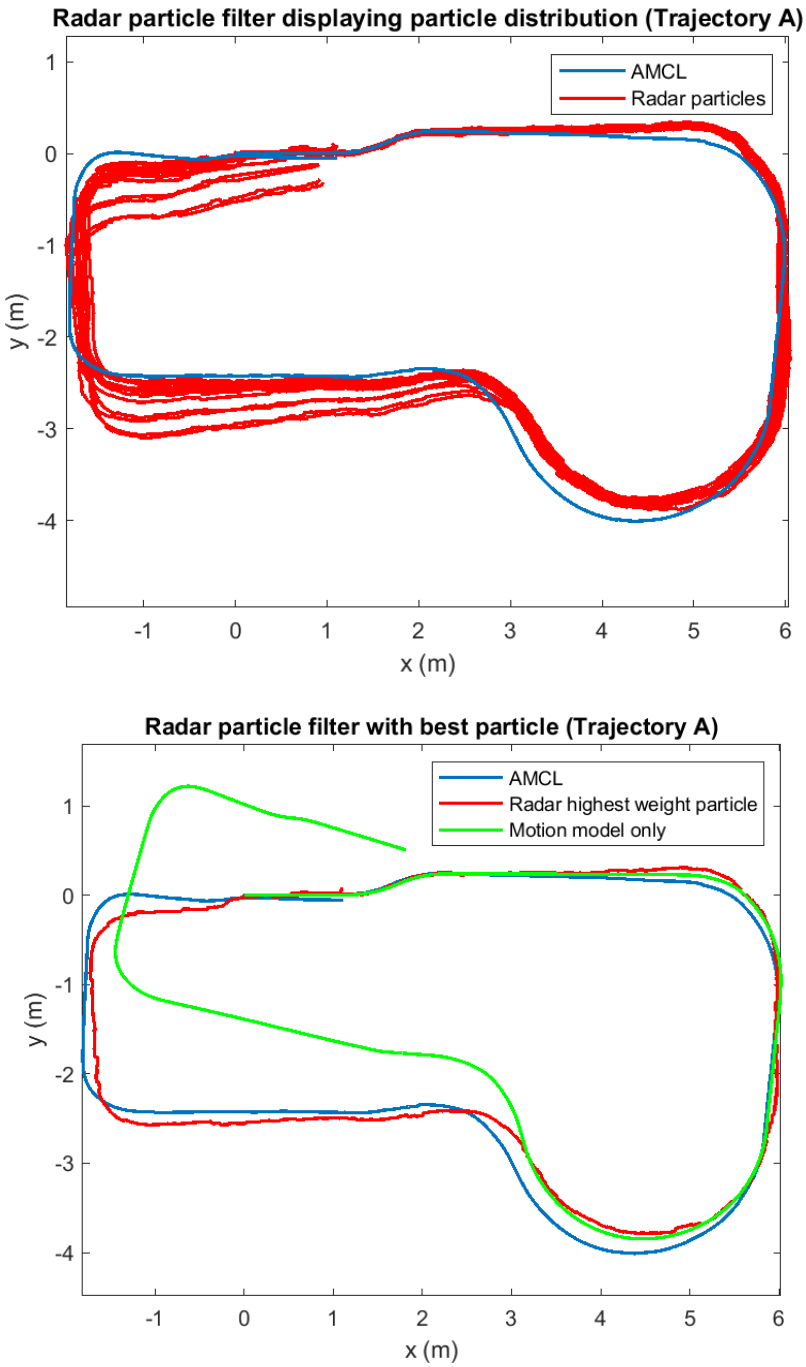


Figure 5.19 Particle filter trajectory estimations for trajectory A

A video of the particle filter estimating this trajectory is available at https://www.youtube.com/watch?v=qFAM-A_XItw. The Root Mean Square Error (RMSE) for each trajectory is shown in Table 8.

Table 8 RMSE for trajectory A

RMSE	X (m)	Y (m)	Heading (deg)
	0.1549	0.1880	6.2965

5.4.2 Trajectory B

This trajectory, which is also used in the graph SLAM experiment in Chapter 5, is in the same location as the previous experiment where obstacles have been shifted. The trajectory involves backtracking on previously traveled locations and is used to evaluate how much better the radar SLAM algorithm can estimate trajectory when using previously mapped areas. LiDAR and radar scans are projected and overlaid using the Cartographer solution in Figure 5.20 to obtain a good view of the environment.

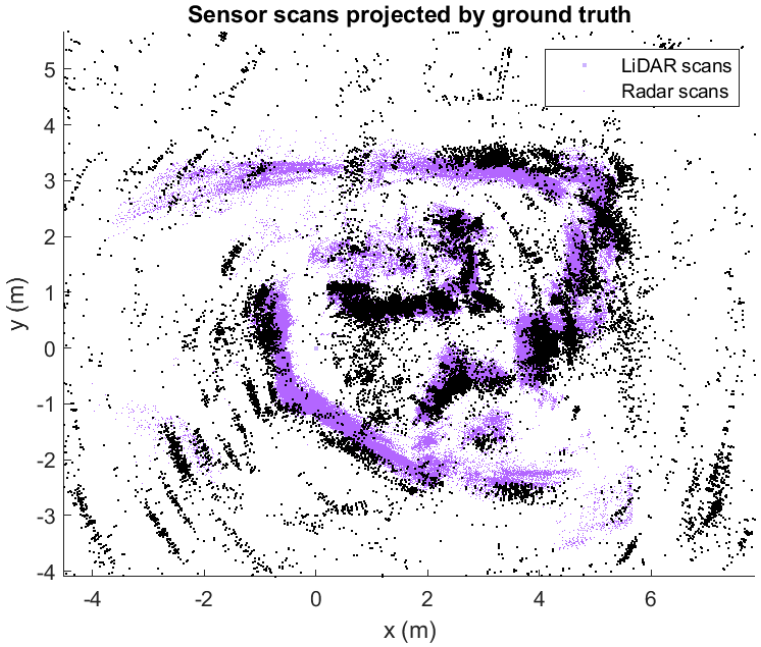


Figure 5.20 Cartographer ground truth scans for trajectory B

As noted previously, the Cartographer estimate does not perform particularly well in this environment, but still provides a reference to make comparisons. Radar scans are very dense and covers larger areas, which presents trackable features but may result in high uncertainty in scan matching results. Figure 5.21 displays the estimation results compared to the Cartographer and motion model.

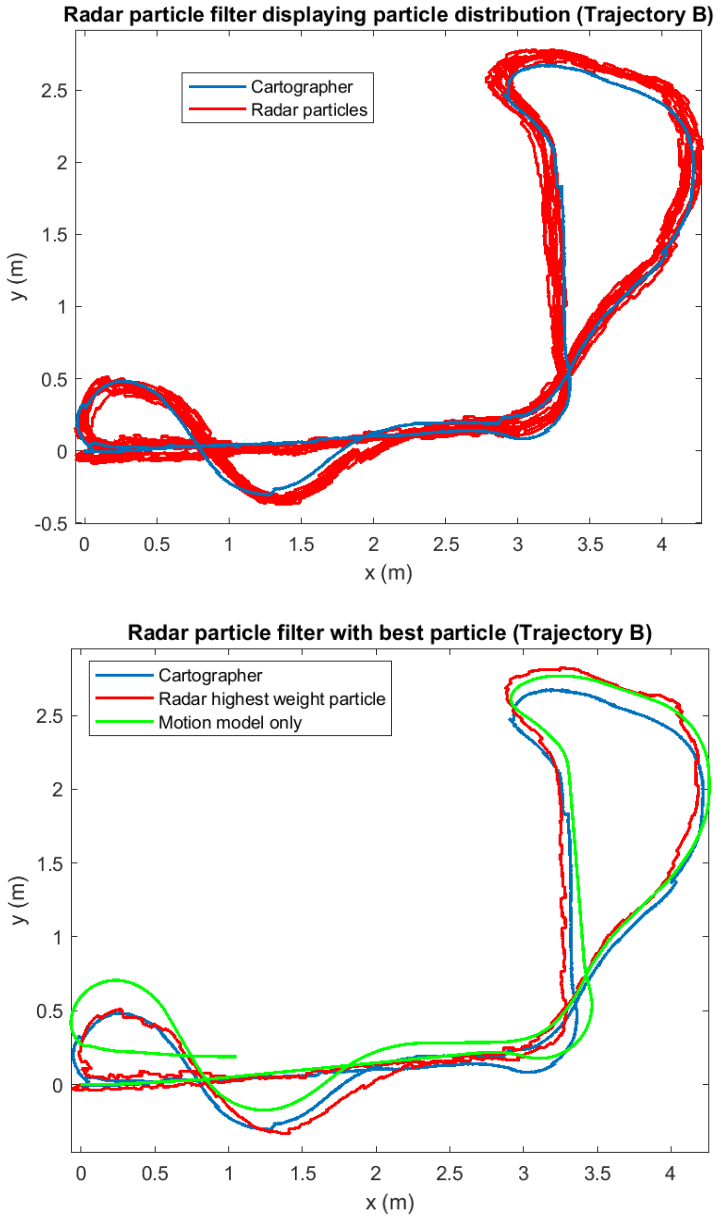


Figure 5.21 Particle filter trajectory estimates for trajectory B

The particles show a thick but consistent distribution and individual trajectories are not as smooth as they are in other environments, which agrees with the idea that the dense radar scans present trackable features but result in relatively large uncertainty during scan matching. The particle filter resamples often in this trajectory due to the multiple loop closures. The highest weight particle is able to correct the odometry drift, however, the trajectory is not very smooth and has many lateral jumps. This likely is caused by the multimodal distribution of the radar scan matching, which is reflected in the sampling of τ. Table 9 shows that the trajectory has very low errors in the position estimates.

Table 9 RMSE for trajectory B

RMSE	X (m)	Y (m)	Heading (deg)
	0.0747	0.0829	5.5127

5.4.3 Trajectory C

The final trajectory takes place in the hallways of the Mackenzie Building 5th floor. This is a much more challenging environment due to the long straight corridors and large distance covered before closing the loop. Such an environment is also challenging for a LiDAR system, and this can be seen from results in Chapter 4, and also from the AMCL solution showing some irregularities in the walls, which should be straight, in Figure 5.22.

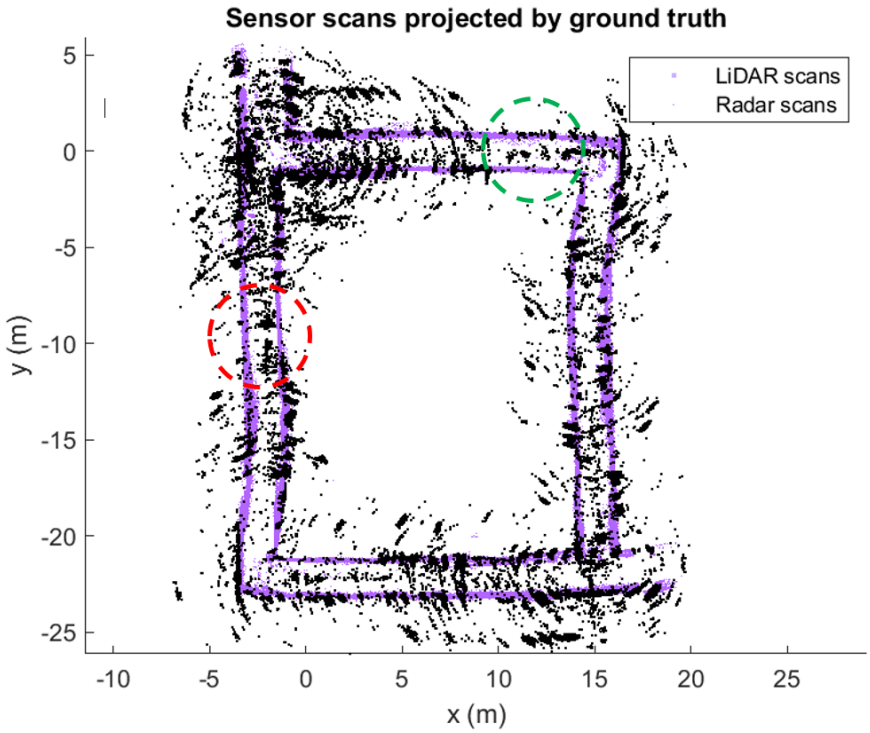


Figure 5.22 AMCL ground truth scans for trajectory C, at the 5th floor of the Mackenzie Building at Carleton University

To add to the difficulty, there are locations where radar scans are much sparser than in previous trajectories. To achieve good results, a different set of parameters must be used. One change is that the motion noise inputs are decreased and the motion distribution is more peaked. In this trajectory, a few particular areas consistently cause radar scan matching mismatches due to the lack of good clusters. In Figure 5.22, the green circled area has sparse detections that make scan matching difficult. The red circled area has a combination of sparse wall detections and dense floor reflections. Hence radar scans which produce low confidence should be taken with less weighting. Additionally, the floor is different in this trajectory. In previous trajectories, small bumps and a section of shallow carpet create larger error in the odometry measurements, where this environment has a

continuous smooth floor. Figure 5.23 shows the trajectory results compared with the AMCL and motion model estimation.

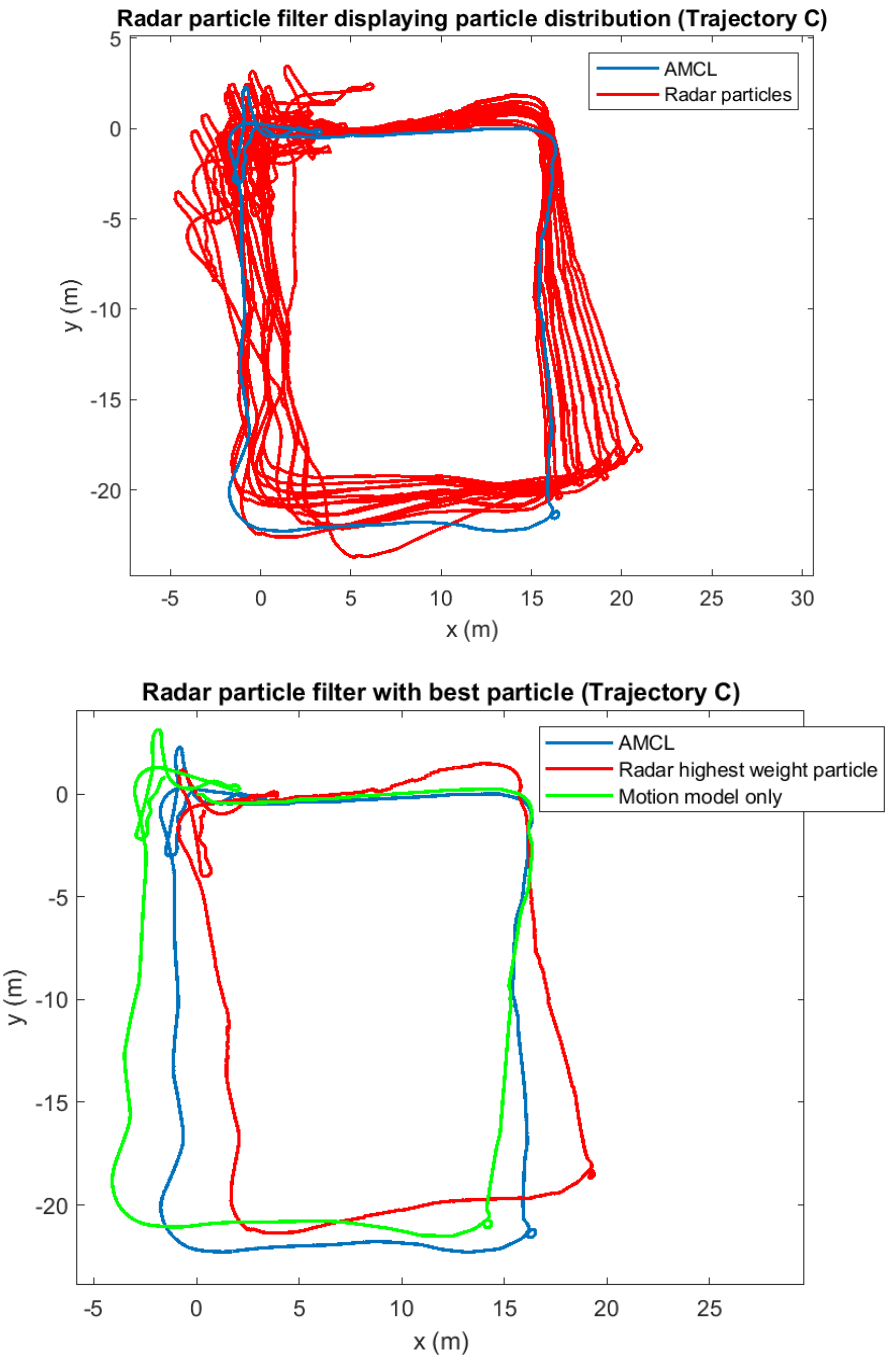


Figure 5.23 Particle filter estimations for trajectory C

Table 10 RMSE for trajectory C

RMSE	X (m)	Y (m)	Heading (deg)
	2.0104	1.6837	10.9367

The radar SLAM estimation is able to close the loop, however, ends up in a tilted path. The interesting result is that this suggests that the radar made a few erroneous estimates in the beginning of the trajectory, but accurately maintained the rest of the trajectory. Figure 5.24 demonstrates that, with a correction transformation of [0.1m 0.31m -0.14 rads] applied to the trajectory, it follows the AMCL estimation fairly well.

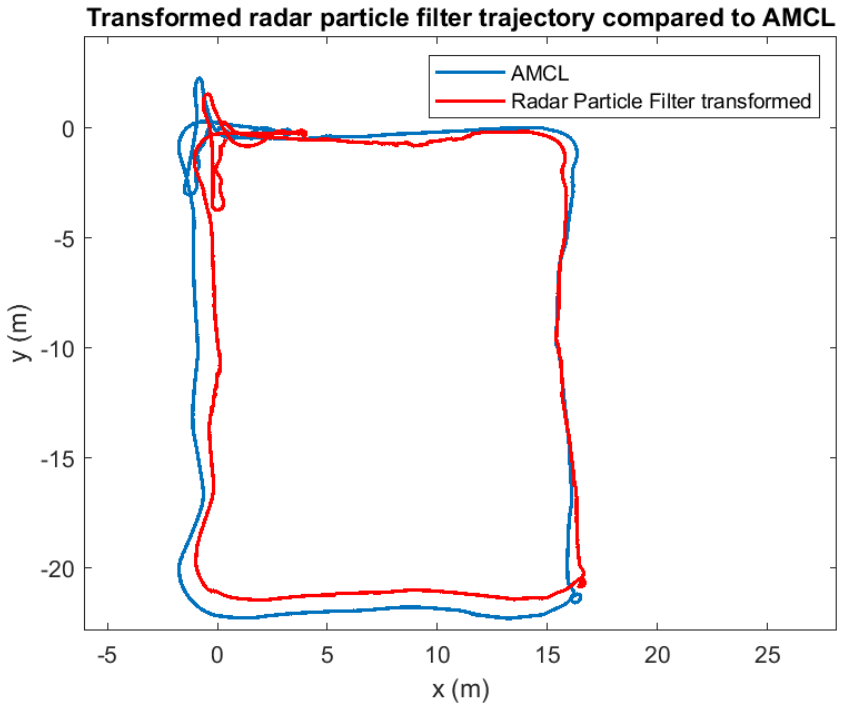


Figure 5.24 Best particle transformed to match AMCL

Table 11 and Figure 5.24 demonstrate a very accurate estimation over a long trajectory. From this experiment, it's made clear that radar scan matching based SLAM has

the potential to provide accurate estimations, yet, in specific areas in the environment, error prone scans may provide confident false positives. The solution to this issue is to develop robust detection of false matches or provide complementary measurements using different sensors such as LiDAR or camera. Despite these issues, radar SLAM provides a much better estimate compared to LiDAR graph SLAM.

Table 11 RMSE for transformed trajectory C

RMSE	X (m)	Y (m)	Heading (deg)
	0.5125	0.5499	6.6737

Chapter 6 Conclusion

This thesis demonstrated the development and the implementation of three main localization techniques and systems for urban and indoor environments. For each developed system, a thorough explanation of the components, background, experiments, and results was included. First, the motivation for the development of each system is discussed, where the difficulties of GNSS outages and IMU errors are explained and the unique advantages of the radar system and the common vulnerabilities of other standard SLAM systems are listed. Next, the background of the technologies used in this thesis were introduced. The methods of localization and SLAM are then described, including representations of the environment, EKF, graph SLAM, and particle filtering.

The INS/GNSS EKF system for urban navigation was efficiently tuned with a GA, which resulting in significant improvements in localization accuracy and improved localization under GNSS temporary interruptions. For indoor locations, a LiDAR graph SLAM implementation was developed which worked well in geometrically rich locations, however, performed poorly in long hallway environments. To address this limitation, an innovative radar SLAM using particle filtering was introduced. The thesis work showed that radar SLAM using a Rao-Blackwellized particle filter is an effective solution for indoor localization in a variety of locations. In summary, the contents of this thesis provide knowledge on a variety of options, implementation details, and innovative ideas supported by both experimental testing and results analysis in technologies and methodologies in the field of urban and indoor localization.

6.1 Contributions

The contributions achieved in this thesis can be summarized as follows:

1. Development of a genetic algorithm tuning scheme for tightly coupled IMU/GNSS EKF systems suitable for urban localization.
2. Implementation and analysis of LiDAR scan matching compared to radar scan matching, including analysis of radar generated point cloud characteristics.
3. Development of a radar SLAM system including the following goals:
    a. Design and implementation of ROS programs to support the radar system including data streaming and logging for all required sensors.
    b. Development of a particle filter using augmented radar scans to create dense and clustered scans.
    c. Design of a new motion model that fuses wheel odometry and gyroscope readings as a simple cost effective way to increase motion prediction accuracy.
    d. Proposal of various modifications to the particle filter including thresholding to reject invalid scan matching results and sampling directly from the sample-based tau distribution due to the radar scan matching distribution having strongly multimodal distributions.
4. The following papers have been published from the thesis work:
    a. **A. Zhang**, M.M Atia, "An Efficient Tuning Framework for Kalman Filter Parameter Optimization using Design of Experiments and Genetic Algorithms", Proceedings of the 32nd International Technical Meeting of

the Satellite Division of The Institute of Navigation (***ION GNSS+2019***), Miami, FL, USA, September 2019. (***Awarded Best Presentation Award***)

b. **A. Zhang**, M.M Atia, "An Efficient Tuning Framework for Kalman Filter Parameter Optimization using Design of Experiments and Genetic Algorithms", ***Journal of the Institute of Navigation***, July 2020.

c. S. S. Kourabbaslou, **A. Zhang** and M. M. Atia, "A Novel Design Framework for Tightly Coupled IMU/GNSS Sensor Fusion Using Inverse-Kinematics, Symbolic Engines, and Genetic Algorithms," in ***IEEE Sensors Journal***, vol. 19, no. 23, pp. 11424-11436, 1 Dec.1, 2019.

d. **A. Zhang**, M.M. Atia, "Comparison of 2D Localization Using RADAR and LiDAR in Long Corridors", submitted to ***IEEE Sensors Conference***, Rotterdam, The Netherlands, October 2020.

6.2 Future Work

Future works include incorporating both LiDAR and radar sensors into a particle filter where the accuracy of the heading and lateral location provided by LiDAR can be combined with radar's accuracy in estimating pose in the long direction of a hallway to form a robust indoor SLAM system. The long range of the radar also makes it a viable sensor for outdoor urban SLAM, where GNSS outages can be supported by radar maps in addition to IMUs. In addition, based on this work, radar SLAM with any dead reckoning fusion is an effective solution as a low cost and rugged system for ground vehicles, robots, and UAVs in challenging situations with poor visibility and destructive environments where LiDAR and camera based SLAM systems would not be well suited.

Bibliography

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics,* vol. 32, pp. 1309-1332, 2016.

[2] U.S. Air Force, "Space Segment," National Coordination Office for Space-Based Positioning, Navigation, and Timing, 30 June 2020. [Online]. Available: https://www.gps.gov/systems/gps/space/. [Accessed 04 July 2020].

[3] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," in *9th IEEE International Conference on Intelligent Systems*, Madeira, 2018.

[4] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access,* vol. 8, pp. 58443-58469, 2020.

[5] S. Khanafseh, B. Kujur, M. Joerger, T. Walter, S. Pullen, J. Blanch, K. Doherty, L. Norman, L. d. Groot and B. Pervan, "GNSS Multipath Error Modeling for Automotive Applications," in *31st International Technical Meeting of The Satellite Division of the Institute of Navigation*, Miami, 2018.

[6] Y. Cui and S. Ge, "Autonomous vehicle positioning with GPS in urban canyon environments," in *IEEE International Conference on Robotics and Automation*, Seoul, 2001.

[7] W. Shao, S. Vijayarangan, C. Li and G. Kantor, "Stereo Visual Inertial LiDAR Simultaneous Localization and Mapping," *arXiv preprint arXiv:1902.10741,* 2019.

[8] G. Grisetti, R. Kümmerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intelligent Transportation Systems Magazine,* vol. 2, no. 2, pp. 31-43, 2010.

[9] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications," *IEEE Internet of Things Journal,* vol. 5, no. 2, pp. 829-846, 2018.

[10] C. Lovescu and S. Rao, "The Fundamentals of Millimeter Wave Sensors," *Texas Instruments,* 2017.

[11] E. Rademakers, P. D. Bakker, C. Tiberius, K. Janssen, R. Kleihorst and N. E. Ghouti, "Obtaining real-time sub-meter accuracy using a low cost GNSS device," in *European Navigation Conference*, Helsinki, 2016.

[12] F. O. Abulude, A. Akinnusotu and A. Adeyemi, "GLOBAL POSITIONING SYSTEM AND IT'S WIDE APPLICATIONS," *Continental J. Information Technology,* 2015.

[13] A. P, S. Z, N. A and E.-S. N, MEMS-Based INTEGRATED NAVIGATION, Norwood, MA: Artech House, 2010.

[14] M. Miroslav and Š. Mikuláš, "Computation and Evaluation Allan Variance Results," in *2016 New Trends in Signal Processing (NTSP)*, Demanovska Dolina, Slovakia, 2016.

[15] M. . H. Elder, "MEMS IMU stochastic error modelling," *Systems Science & Control Engineering,* vol. 5, no. 1, pp. 1-8, 2016.

[16] T. Instruments, "Texas Instruments," Texas Instruments, 2020. [Online]. Available: https://www.ti.com/product/AWR1843. [Accessed 04 07 2020].

[17] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, 2019.

[18] P. Fritsche, S. Kueppers, G. Briese and B. Wagner, "Radar and LiDAR Sensorfusion in Low Visibility Environments," in *13th International Conference on Informatics in Control, Automation and Robotics*, Portugal, 2016.

[19] T. Instruments, "Traffic Monitoring Object Detection and Tracking Reference Design Using Single-Chip mmWave Radar," Texas Instruments, Texas, 2019.

[20] Slamtec, "RPLIDAR A3," Slamtec, 2020. [Online]. Available: https://www.slamtec.com/en/Lidar/A3. [Accessed 31 08 2020].

[21] Velodyne Lidar, "Alpha Prime," Velodyne Lidar, 2020. [Online]. Available: https://velodynelidar.com/products/alpha-prime/. [Accessed 31 08 2020].

[22] Stereolabs, "Depth Settings," Stereolabs, 2020. [Online]. Available: https://www.stereolabs.com/docs/depth-sensing/depth-settings/. [Accessed 31 08 2020].

[23] J. Hecht, "Lasers for Lidar: FMCW lidar: An alternative for self-driving cars," LaserFocusWorld, 2019.

[24] K. Thurn, R. Ebelt and M. Vossiek, "Noise in Homodyne FMCW radar systems and its effects on ranging precision," in *IEEE MTT-S International Microwave Symposium Digest*, Seattle, 2013.

[25] R. Weston, S. Cen, P. Newman and I. Posner, "Probably Unknown: Deep Inverse Sensor Modelling Radar," in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, 2019.

[26] S. H. Cen and P. Newman, "Precise Ego-Motion Estimation with Millimeter-Wave Radar Under Diverse and Challenging Conditions," in *IEEE International Conference on Robotics and Automation*, Brisbane, 2018.

[27] Texas Instruments, "xWR1843 Evaluation Module (xWR1843BOOST) Single-Chip mmWave Sensing Solution," 05 2020. [Online]. Available: https://www.ti.com/lit/ug/spruim4b/spruim4b.pdf. [Accessed 10 07 2020].

[28] M. Holder, S. Hellwig and H. Winner, "Real-Time Pose Graph SLAM based on Radar," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, 2019.

[29] F. Schuster, C. G. Keller, M. Rapp, M. Haueis and C. Curio, "Landmark based radar SLAM using graph optimization," in *19th International Conference on Intelligent Transportation Systems* , Rio de Janeiro, 2016.

[30] Y. Almalıoğlu, M. Turan, C. X. Lu, N. Trigoni and A. Markham, "Milli-RIO: Ego-Motion Estimation with Millimetre-Wave Radar and Inertial Measurement Unit Sensor," in *arXiv preprint*, 2019.

[31] M. Schön, M. Horn, M. Hahn and J. Dickmann, "Real-Time Radar SLAM," 2017.

[32] J. W. Marck, A. Mohamoud, E. v. Houwen and R. v. Heijster, "Indoor radar SLAM A radar application for vision and GPS denied environments," in *European Radar Conference*, Nuremberg, 2013.

[33] J. Callmer, D. Törnqvist, F. Gustafsson, H. Svensson and P. Carlbom, "Radar SLAM using visual features," *EURASIP Journal on Advances in Signal Processing volume,* no. 71, 2011.

[34] R. Rouveure, P. Faure and M. Monod, "Radar-based SLAM without odometric sensor," in *International workshop of Mobile Robotics for environment/agriculture*, Clermont Ferrand, 2010.

[35] J.-A. M. David Filliat, "Map-based navigation in mobile robots:: I. A review of localization strategies," *Cognitive Systems Research ,* pp. 243-282, 2003.

[36] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Cambridge: MIT Press, 2005.

[37] A. P. Godse, "Computer Graphics," Pune, Technical Publications Pune, 2009, pp. 1-17, 1-18, 1-19.

[38] M. Grewal and A. Andrews, Kalman Filtering Theory and Practice, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[39] J. Farrell, Aided Navigation, GPS with High Rate Sensors, 1 ed., New York: McGraw Hill, 2008.

[40] F. Wang and Z. Zhao, "A survey of iterative closest point algorithm," in *Chinese Automation Congress*, Jinan, 2017.

[41] J. Li, R. Zhong, Q. Hu and M. Ai, "Feature-Based Laser Scan Matching and Its Application for Indoor Mapping," *Sensors,* vol. 16, no. 8, p. 1265, 2016.

[42] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, Kobe, 2009.

[43] W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, 2016.

[44] A. Doucet, N. d. Freitas, K. Murphy and S. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks," in *Sequential Monte Carlo Methods in Practice*, Springer Verlag, 2001.

[45] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation*, Detroit, 1999.

[46] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics,* vol. 23, no. 1, pp. 34-46, 2007.

[47] G. Grisettiyz, C. Stachniss and W. Burgard, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective

Resampling," in *IEEE International Conference on Robotics and Automation*, Barcelona, 2005.

[48] M. M. Atia, Multi-Sensor Integrated Navigation in Urban and Indoor Environments: GNSS, Inertial, and Range Sensors Integration, K. Yu, Ed., Idea Group Inc, 2018, p. 46.

[49] S. S. Kourabbaslou, A. Zhang and M. M. Atia, "A Novel Design Framework for Tightly Coupled IMU/GNSS Sensor Fusion Using Inverse-Kinematics, Symbolic Engines, and Genetic Algorithms," *IEEE Sensors Journal,* vol. 19, no. 23, pp. 11424-11436, 2019.

[50] P. G. Savage, Strapdown Analytics - Second Edition, Strapdown Associates, Inc, 2007.

[51] U. D and D. K, "Estimation of deterministic and stochastic IMU error parameters," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, 2012.

[52] A. Quinchia, G. Falco, E. Falletti, F. Dovis and C. Ferrer, "A Comparison between Different Error Modeling of MEMS Applied to GPS/INS Integrated Systems," *Sensors,* vol. 13, no. 8, pp. 9549-9588, 2013.

[53] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, 1989.

[54] InvenSense Inc, 17 01 2014. [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdf. [Accessed 30 06 2019].

[55] NovAtel, "NovAtel ProPak6 Triple-Frequency GNSS receiver," [Online]. Available: https://www.novatel.com/products/gnss-receivers/enclosures/propak6/. [Accessed 25 July 2019].

[56] KVH, "KVH-1759 IMU," [Online]. Available: https://www.kvh.com/Military-and-Government/Gyros-and-Inertial-Systems-and-Compasses/Gyros-and-IMUs-and-INS/IMUs/1750-IMU.aspx. [Accessed 25 July 2019].

[57] Google, "Kanata," Google, 2020. [Online]. Available: https://www.google.ca/maps/@45.3363708,-75.9119784,14.75z. [Accessed 29 08 2020].

[58] Google Earth, "Satellite view of roads," Google, 2020. [Online]. Available: https://earth.google.com/web/. [Accessed 2020].

[59] robotpilot, routiful, LeonJung, ROBOTIS-Will, ooeygui, ROBOTIS-David and kkjong, "Turtlebot3," Robotis, 24 Jan 2020. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/. [Accessed 04 July 202].

[60] Raspberry Pi, "RaspberryPi," RaspberryPi, 2020. [Online]. Available: https://www.raspberrypi.org/. [Accessed 04 07 2020].

[61] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs and R. Wheeler, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[62] D. Fox, "KLD–Sampling: Adaptive Particle Filters," Department of Computer Science and Engineering, University of Washington, Washington, 2001.

[63] C. Stachniss, D. Hahnel and W. Burgard, "Exploration with active loop-closing for FastSLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, 2004.

[64] Dham Vivek, "Programming Chirp Parameters in TI Radar Devices," 02 2020. [Online]. Available: https://www.ti.com/lit/an/swra553a/swra553a.pdf. [Accessed 04 07 2020].

[65] M. Richards, Fundamentals of Radar Signal Processing, McGraw Hill, 2005.

[66] Texas Instruments, "Autonomous Robotics with ROS for mmWave," 2020. [Online]. Available: http://dev.ti.com/tirex/explore/node?node=AD79e41r4iMaKIGvWAXKjg__VLyFKFf__LATEST. [Accessed 04 07 2020].

[67] N. Freier, "Kalman Filter: Multiplying Normal Distributions," TU Dresden, 2013.