Localization System for a Medical Service Robot in an Indoor Environment

A thesis paper submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Computer Science and Engineering.

Submitted by

Md. Shafiqul Islam
Student ID: 2016105050002
Batch: 3rd

Department of Computer Science and Engineering (CSE)
School of Science and Engineering
Khwaja Yunus Ali University
Enayetpur, Sirajganj-6751, Bangladesh

Examination Year- 2020

Localization System for a Medical Service Robot in an Indoor Environment

A thesis paper submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Computer Science and Engineering.

Submitted by

Md. Shafiqul Islam
Student ID: 2016105050002
Batch: 3rd

Supervised by

.....

Md. Tarequl Islam
Senior Lecturer, Dept. of CSE
Khwaja Yunus Ali University
Enayetpur, Sirajganj-6751

.....

Quamrul Hasan Mallik
Subject Expert,
Innovation Project Group 7,
SEW-Eurodrive, Bruchsal, Germany

Department of Computer Science and Engineering (CSE)
School of Science and Engineering
Khwaja Yunus Ali University
Enayetpur, Sirajganj-6751, Bangladesh

Examination Year- 2020

Declaration

I, hereby, declare that the thesis work entitled "**Localization System for a Medical Service Robot in an Indoor Environment**" has been carrid out by **Md. Shafiqul Islam** is the outcome of the investigation under the supervision of **Md. Tarequl Islam**, Senior Lectrurer, Department of Computer Science and Engineering (CSE), Khwaja Yunus Ali University, Enayetpur, Sirajganj-6751, Bangladesh. I also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

…….	…….	…….
Signature of the Candidate	Signature of the Supervisor	Signature of the Co-Supervisor
Md. Shafiqul Islam	**Md. Tarequl Islam**	**Quamrul Hasan Mallik**

Approval

Thesis/Research on **"Localization System for a Medical Service Robot in an Indoor Environment."** Submitted by Md. Shafiqul Islam, ID No: 2016105050002 to the Department of Computer Science and Engineering (CSE), Khwaja Yunus Ali University (KYAU), Enayetpur, Sirajganj-6751, Bangladesh, has been accepted by the Examination Committe as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents in December.

1 **…….....** **Supervisor**

Md. Tarequl Islam
Senior Lecturer
Department of Computer Science & Engineering
Khwaja Yunus Ali University, Enayetpur,
Sirajganj-6751

2 **…….....** **Co-supervisor**

Quamrul Hasan Mallik
Subject Expert,
Innovation Project Group 7,
SEW-Eurodrive, Bruchsal, Germany

3 **…….....……..** **Head of the Department**

Dr Mir Mohammad Azad
Professor
Department of Computer Science & Engineering
Khwaja Yunus Ali University, Enayetpur,
Sirajganj-6751

Abstract

Simultaneous Localization and Mapping (SLAM) problem has been the solution for mobile robots for a while. Simultaneous Localization and Mapping (SLAM) is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map. The SLAM process consists of a number of steps. The goal of this process is to use the environment to update the real-time position of the mobile robot. Since the odometry of the mobile robot is often erroneous we cannot depend directly on the odometry data. For this problem, we can use laser scans of the environment to correct the position of the mobile robot. This method is accomplished by extracting features from the environment and re-observing when the robot moves from one place to another. There are many different types of SLAM technique is used over the years like EFF based SLAM, FastSLAM, Gridmap based SLAM, GraphSLAM, etc. In EKF (Extended Kalmen Filter) based SLAM the EKF is responsible for updating where the robot thinks it is based on some features. These features are commonly called landmarks. In FastSLAM each observation only concerns a small amount of the state variables. GraphSLAM is to address the SLAM problem by using a graph. In recent days Rao-Blackwellized Particle Filters (RBPF) has been at the highest priority for its accuracy. This approach depends on a particle filter where each particle is an individual map of the environment. The most recent sensor observation is taken into account for creating the next generation of particles. The map is more accurate since the current observation is incorporated into the individual maps after considering its effect on the pose of the robot. This thesis aims to provide a review of implementing RBPF based SLAM with a reduced number of particles in a mobile robot in an indoor environment.

Keywords: SLAM, Particle filter, RBPF, adaptive resampling, hospital robots, EKF, LiDAR, Mapping.

Acknowledgment

All the thanks and gratitude go to Almighty ALLAH, who is the creator of this vast universe and the source of all kinds of knowledge and intelligence, the most merciful, gracious, who gave me the strength, guidance, patience, and ability to complete the thesis.

I would like to take the opportunity to express my sincere appreciation to my respected supervisor Md. Tarequl Islam, Senior Lecturer, Department of Computer Science and Engineering (CSE), Khwaja Yunus Ali University, Enayetpur, Sirajganj, Bangladesh, for his generous guidance and continuous support and inspiration throughout the work. Without his help and assistance, I won't be able to finish my research successfully. My heartfelt gratitude to my thesis Co-supervisor Quamrul Hasan Mallik, Subject Expert, Innovation Project Group 7, SEW-Eurodrive, Bruchsal, Germany, provided me valuable guidance and suggestions in the stages of my work. His support has enabled me to go against all odds. I also like to thank my honorable teachers of the CSE department, KYAU for their suggestions and comments. Besides this, I extremely thankful to my parents, family members, and friends for their support and encouragement. The journey would be much harder if they weren't present for me at every moment whenever I need them. Finally, I would like to acknowledge Khwaja Yunus Ali University for allowing me to use their resources and complete my thesis.

Dedication

Dedicated to My respected Parents and Teachers

Contents

List of Figures

List of Tables

Abbreviations

SLAM — Simultaneous Localization and Mapping

CML — Concurrent Mapping and Localization

EKF — Extended Kalman Filter

AD — Autonomous Driving

ACC — Adaptive Cruise Control

CS — Computer Science

EE — Electrical Engineering

IT — Information Technology

RBPF — Rao-Blackwellized Particle Filter

LiDAR — Light Detection and Ranging

ToF — Time of Flight

SoM — System on Module

KF — Kalman Filter

PF — Particle Filter

OS — Operating System

CPU — Central Processing Unit

GPU — Graphical Processing Unit

DRAM — Dynamic Random Access Memory

HDMI — High-Definition Multimedia Interface

USB — Universal Serial Bus

SSD — Solid State Drive

GPIO — General Purpose Input/Output

GUI — Graphical User Interface

HTTP— HyperText Transfer protocol

TCP — Transmission Control Protocol

GPS — Global Positioning System

UDP — User Datagram Protocol

1

Introduction

In this chapter, a brief introduction to the Simultaneous Localization and Mapping (SLAM) for a medical service robot is presented. This chapter also provides information on the approaches considered and ultimately, how they led to the final objective.

1.1 Background

In recent times, many companies in the automotive industry have spent a lot of time and funds on researching the possibility of producing autonomous service robots. The research has moved forward to the point that there are actually semi-autonomous service robots available in the market today, and it is widely believed that at some point in time, almost all vehicles will be autonomous. To achieve this, there is a need for advanced sensors and functions. One of the major issues is the fact that the vehicle needs to be aware of its current position within an unknown environment. This is known as the Simultaneous Localization and Mapping problem, commonly abbreviated as SLAM [1]. TUG is such a kind of autonomous service robot by Aethon [2]. The goal of this project is to implement the SLAM algorithm and build a real system that will create a map of the environment and also Localize the vehicle pose within that environment. As implementing a SLAM algorithm is a vital part of achieving this, this thesis will focus on finding the best SLAM algorithm for this purpose and implement it in the vehicle.

1.2 Motivation

The autonomous driving vehicle is useable and profitable almost anywhere in the world in this 21st century. We can see such vehicles in industries, hospitals, restaurants, and also in home-application. These vehicles can produce better service for people within a low cost of time [3]. In industries, an autonomous vehicle can take different products, tools to the destination. In hospitals, a service robot can peak medicines, foods, clothes, and other accessories for the patient as well as for the doctors. In a restaurant, an autonomous robot can peak foods for the customer in a very short time. At home, we can see different

robotic Vacuums that can clean the house by autonomously driving at short times.

For those above reasons, our motivation is to develop a localization system for the medical service robot as it could help the patient and make life easier for us.

1.3 Problem Statement

One of the fundamental challenges of robotics is Simultaneous Localization and Mapping (SLAM), also known as Concurrent Mapping and Localization (CML), which addresses the need to construct an environmental map while deciding the position of the robot within this map at the same time [3]. There are various types of SLAM technique which can be implemented in our system. Like EKF based SLAM, Gridmap based SLAM, Visual SLAM, RBPF based SLAM with grid maps representation. The problem is that we have to select the best fit for our system as we have to consider how effectively, accurately, and smoothly it works.

We want to build a service robot that can drive autonomously inside Khwaja Yunus Ali Medical College and Hospital (KYMAC). This vehicle can transport medicine, samples, food, hospital equipment, etc. For this, we need an Autonomous Driving system that can,

- Generate an accurate map of the hospital (+/- 5cm accuracy).
- Real-time position and orientation (x,y, and theta) of the robot in the hospital.

However, the following questions would arrive to achieve the goal of this thesis:

- Will the SLAM algorithm work properly in the vehicle?
- Will the system produce the accuracy which is acceptable as a real system?
- Can we reduce the probability of system failure?

1.4 Objectives

The primary objective of this thesis is to review different types of SLAM techniques and then choose the best one and implement that into a real mobile service robot. Our focus is to create the best localization system which can localize the position of the robot in a very smooth way and hence the robot can move and provide its services to the people. Other important intensions for building such a system are the followings:

- Can generate an accurate 2d map of the environment from the laser scan data.
- Using this preprocessed map of the mobile robot can determine the real-time pose with higher accuracy and navigate through that environment.
- The proposed system is favorable and effective which achieves the best performance of accuracy and helps the human being.

1.5 Thesis Scope

The purpose of this thesis is to implement a SLAM algorithm in a real running vehicle. A theoretical investigation would help which SLAM algorithm to choose. Extended Kalman filter-based SLAM (EKF), particle filter-based SLAM (FastSLAM), graph-based SLAM (GraphSlam), and RBPF-based SLAM are four related algorithms that will be investigated.

The SLAM algorithm is intended to construct a map of the environment of the vehicle simultaneously, as well as to determine the location of the vehicle within this map [17]. This data is to be transmitted to NX Devkit, which will display in a GUI the map and the location of the vehicle in real-time. The priority is to introduce a SLAM based on particle filters.

To reach our goals, the following needs to be investigated:

- A theoretical analysis that compares various SLAM algorithms to make a valid decision on which one to use
- How to use limited resources to run a SLAM algorithm. First and foremost, this means choosing a SLAM algorithm that uses very few resources. However, modifications to the algorithm (or the implementation of one may be required if this is not sufficient).
- Theoretical studies and simulations considering complex structures

1.6 Limitations

As the common rules from nature there isn't such a system or module that is 100% perfect, our systems have some limitations. The following limitations are set for this project:

- Ethical aspects of any kind will not be discussed in this thesis.
- Our system may not work in an outdoor environment.
- Delayed processing of laser scans may estimate the current pose wrong.
- Too high errors in odometry (wheel slip, uneven ground) may result in a wrong initial estimate of the pose.

- In absence of distinguishing features in laser scan or map, scan-matching in local regions provides multiple maxima. (e.g. in a corridor). In this case, the estimated pose x_t may be wrong.
- In a dynamic environment, the known features of the map may be covered by unknown objects – resulting in failed scan-matching.

1.7 Risk analysis

There are several risks that certain parts of this thesis may be delayed or fail. Damaging any of the hardware will stop our project because the hardware we use is too costly and also most of them are not available in Bangladesh.

2

Detailed Literature Review

This chapter discusses Autonomous Driving (AD), how mobile robot localization works, different kinds of filtering algorithms for the SLAM technique, and particle filters. We will also provide information about Laser Scan Data and Odometry Data.

2.1 Autonomous Driving

The ability to navigate and execute motion maneuvers without human assistance is called Autonomous Driving (AD). It's been a while where semi or fully autonomous industrial and manufacturing robots are standard products. ACC, or adaptive cruise control, is one of the aspects of automotive technology used in automated vehicles. This device will automatically change the speed of the vehicle to ensure that it maintains a secure distance from the vehicles in front of it. This feature relies on data obtained from the vehicle's sensors and enables the vehicle to perform tasks such as braking when it detects that any vehicles ahead are approaching.

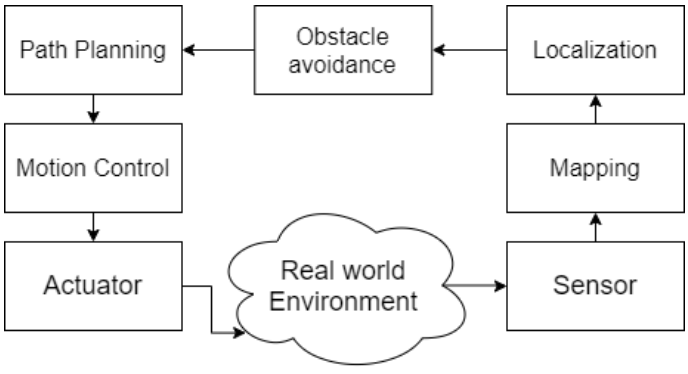

Fig 2.1: General scheme for autonomous driving.

As we can see there are different tools and techniques used to achieve autonomous driving. In AD sensors are used to sense the surrounding environment and with the help of motion control or any other actuator, it acts to that particular environment. There are also some important parts of AD, like path planning, obstacle avoidance. SLAM that is mapping and localization is another most important technique of AD. AD is one of the hottest topics in recent years as it encompasses CS, Mechatronics, EE, IT, Mechanical

Engineering. As we know, industries are investing a lot of money and effort to make self-driving autonomous cars.

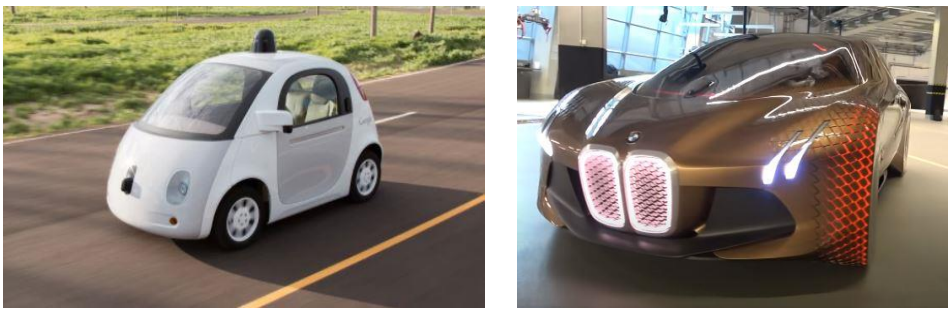

Fig 2.2: Self-driving cars from google (left), Vision Next 100 from BMW (right).

In fig 2.2 we can see the self-driving cars from google (left) and Vision Next 100 (right) from BMW which are testing and running in the road autonomously without any driver.

2.2 Mobile Robot Localization

The problem of deciding the pose of a robot relative to a given map of the environment is mobile robot localization. It is also called estimation of position or monitoring of position. Mobile robot localization is an instance of the general problem of localization, which in robotics is the most fundamental perceptual problem. This is because almost all robotics tasks involve knowledge of the robots' location and the objects being tampered with.

Localization can be seen as a concern in the transformation of coordinates. Maps are described in a global coordinate system, which is independent of a robot's pose. Localization is the process of establishing cooperation between the coordination system of the map and the local coordination system of the robot. Knowing this transformation of coordinates helps the robot to communicate within its own coordinate frame the position of objects of interest, an essential prerequisite for robot navigation.

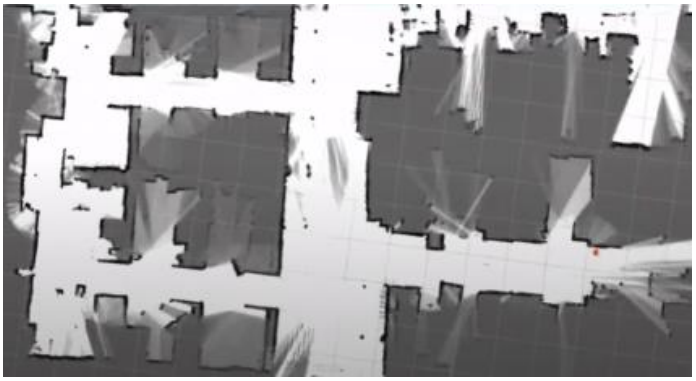

Fig. 2.3: Example maps used for robot localization.

Fig 2.3 shows the example of a typical 2D map used for mobile robot localization.

2.3 Kalman Filter

The Kalman filter, which is used in linear Gaussian systems, is a technique that implements a Bayes filter. Basically, for a continuous state, it measures a belief [7]. This state represents vehicle details, such as odometry, and its surrounding environment, such as landmarks [7]. The State's assumption can be represented by its context and its covariance at a particular time instance t bel(x_t) [7]. The Kalman filter has three specific prerequisites which are as follows [7]:

- The current state x_t must be represented by linear arguments, which implies that the current state

  $$x_t = A_t(x_{t-1}) + B_t u_t + \varepsilon_t$$

  where A_t and B_t are constants. The current state x_t that depends on previous steps x_{t-1}, the control u_t, and the noise ε_t .

- The current measurement z_t requires its arguments to be of linear nature meaning that

  $$z_t = C_t x_t + \delta_t$$

  where C_t is a constant and the noise source is Gaussian. The current measurement z_t depends on the current state x_t and the noise source δ_t

- The last prerequisite is that the start belief bel(x_0) has to be Gaussian distributed.

Algorithm 2.1 Kalman Filter algorithm

1: function Kalman_filter(μ_{t-1}, Σ_{t-1}, u_t , z_t)
2: $\mu_t = A_t\mu_{t-1} + B_tu_t$
3: $\Sigma_t = A_t\Sigma_{t-1}A_t^T+R_t$
4: $K_t = \Sigma tC_t^T(C_t\Sigma_tC_t^T +Q_t)^{-1}$
5: $\mu_t = \mu_t + K_t(z_t - C_t\mu_t)$
6: $\Sigma_t = (I - K_tC_t)\Sigma_t$
7: return μ_t , Σ_t
8: end function

In Algorithm 2.1 from the above, the Kalman Filter algorithm is shown. Where x_t is the current position at different times t. Here the output of the algorithm, the bel(x_t) is expressed by the mean μ_t and the covariance Σ_t [7].

2.4 Extended Kalman Filter

The Extended Kalman Filter is an approach used to deal with nonlinear models that cannot be treated by the conventional normal Kalman filter. Measurements of the real world and state transitions are often not linear, so the assumption of linear data by a standard Kalman filter is not reliable. By defining the probability of state change and the probability of measurement with nonlinear functions, the approach of how EKF processes this is below [5],

$$x(t) = g(u(t), x(t-1)) + \varepsilon(t)$$

$$z(t) = h(x(t)) + \delta(t)$$

The EKF uses nonlinear functions, as mentioned above. The algorithm's performance is an approximation with an estimated mean and covariance, in the form of a Gaussian distribution. The algorithm's approximation section is due to the fact that it uses nonlinear functions. These functions need to go through a process of linearization. This approach linearizes the functions by using a first-order expansion of Taylor around a linearization point, for example.

Algorithm 2.2 Extended Kalman Filter algorithm

1: function extended_Kalman_filter(μ_{t-1}, Σ_{t-1}, u_t , z_t)
2: $\mu_t = g(u_t, \mu_{t-1})$
3: $\Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t$
4: $K_t = \Sigma t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \mu_t + K_t(z_t - h(\mu_t))$
6: $\Sigma_t = (I - K_t H_t)\Sigma_t$
7: return μ_t , Σ_t
8: end function

We can see that the above Algorithm 2.2 is a little different from Algorithm 2.1. This is because the EKF uses the nonlinear functions to determine the mean μ_t and the covariance Σ_t.

2.5 Particle Filter

An alternative nonparametric implementation of the Bayes filter is the particle filter. Particle filters approximate the posterior by a small number of parameters, much like histogram filters. However, they differ in the manner in which these parameters are created and in which the state space is populated [5].

The main idea of the particle filter is to represent the bel(x_t) posterior by a collection of samples of random states drawn from this posterior [5]. Instead of representing the distribution by a parametric shape (an exponential function that determines the normal distribution density), the particle filters represent the distribution of a collection of samples taken from that distribution. Such a representation is approximate, but it is non-parametric and can therefore represent a far wider distribution space than Gaussians, for instance.

The samples with a posterior distribution are called particles in particle filters and thus are denoted [5],

$$X_t = x_t^{[1]} , x_t^{[2]} , \ldots , x_t^{[M]}$$

Each particle $x_t^{[m]}$ (with $1 \leq m \leq M$) is a concrete instantiation of the state at time t, that is, a hypothesis as to what the true world state may be at time t. Here M denotes the number of particles in the particle set X_t.

Algorithm 2.3 Particle Filter algorithm

1: function particle_filter(X_{t-1}, u_t , z_t)
2: $\tilde{X}_t = X_t = \emptyset$
3: for m = 1 to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t , x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\tilde{X}_t = \tilde{X}_t + (x_t^{[m]} , w_t^{[m]})$
7: end for
8: for m = 1 to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to X_t
11: end for
12: return X_t
13: end function

In Algorithm 2.3 the particle filter is described, where the $\tilde{X}_t$ is the prediction position vector, X_t is the corrected position vector.

2.5.1 Sampling

New values for every particle in the filter are produced during sampling. These values are only temporary, and not until the resampling stage will the particles themselves be changed. This new state for each particle is due to the control u_t and the preceding state of the particle $x_t^{[m]}$. After this the value factor $w_t^{[m]}$ is determined. The significant factor is due to the z_t calculation and is used as a probability metric of the particle used in the next resampling of the particle filter [7]

2.5.2 Resampling

The actual particles in the filter are modified during resampling. This is achieved by selecting a particle produced in the sampling process randomly, where the likelihood of selection for each particle is due to its factor of importance. Resampling is achieved with substitution, so it is possible to draw each particle many times [7].

2.6 Rao-Blackwellized Particle Filter

Rao-Blackwellized Particle Filter (RBPF) is another extension of a regular particle filter where some state variables are represented by different particles and some with Gaussians [7]. The RBPF splits the state vector x_t, into two parts where KFs are used, one-part x_t^p, which is calculated using the PF, and another

part x_t^k. Basically, the joint probability density function is given by Bayes' rule as a denotation of the measurements and states up to time t with $y_t = \{y_j\}^t_{j=0}$ and $x_t = \{x_j\}^t_{j=0}$, respectively [8].

$$P(x_t^p, x_t^k \mid y_t) = p(x_t^k \mid x_t^p, y_t)\, p(x_t^p \mid y_t)$$

KF PF

If the model is linear Gaussian conditionally, that is, if the concept $p(x_t^k \mid x_t^p, y_t)$ is linear Gaussian, a KF can be used to estimate it optimally. To obtain the second factor, in all cases where there is at least one nonlinear state relationship or one non-Gaussian noise component, nonlinear filtering techniques must be applied (the PF will be used here). The interpretation is that each particle in the PF is associated with a KF. This gives a mixed state-space representation, as shown in the above equation, with x^p represented by particles and x^k represented with a Kalman filter for each particle [8].

2.7 Odometry

Odometry-distance measurement is a basic technique used for navigation by robots. Using the internal clock of the embedded device, measuring time is easy. It is more complicated to measure speed. Wheel encoders are used to count the rotations of some educational robots, while speed is determined from the properties of the motors in others. The new position of the robot can be computed from the distance moved s = vt [9]. The calculation is trivial in one dimension, but when the movement includes turns, it becomes a little more complicated. The distance computation by odometry is presented in this section, first for a robot moving linearly and then for a robot making a turn.

Given the distance traveled by each wheel, we calculate the change in the robot's distance and orientation. If a robot starts from a position Xt, and the right and left wheels move respective distances Δu_{Tr} and Δu_{Tl} [9],

$$\textit{Distance, } \Delta u_T = (\Delta u_{Tr} + \Delta u_{Tl})/2$$

$$\textit{Orientation, } \Delta\theta = (\Delta u_{Tr} - \Delta u_{Tl})/2L$$

Now we are able to update the position/orientation in global coordinates,

$$\Delta x = \Delta u_T \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta u_T \sin(\theta + \Delta\theta/2)$$

2.8 Laser Scan

In the SLAM method, the first step is to obtain data about the robot's environment [10]. We get laser data as we use a laser scanner. The SICK laser scanner we're using will calculate the output range from a 270° angle. It has a vertical resolution of 0.33°, which means that the area diameter measured by the laser beams is 0.33° wide. The standard output of a laser scanner would look like this:

2.98, 2.99, 3.00, 3.01, 3.00, 3.49, 3.50, ..., 2.20, 8.17, 2.21

In terms of meters, the output of the laser scanner shows the range from right to left. Finally, it is worth noting that laser scanners are very fast. They can be retrieved at about 11 Hz using a serial port.

The challenging thing about odometry and laser data is getting the timing correct. If the odometry data is retrieved later, the laser data will be obsolete at some point [10]. To ensure that they are accurate at the same time the data can be extracted. Since the controls are known, it is easiest to extrapolate odometry results. It can be very difficult to predict the measurements of the laser scanner. If one has control of when the measurements are returned it is easiest to ask for both the laser scanner values and the odometry data at the same time.

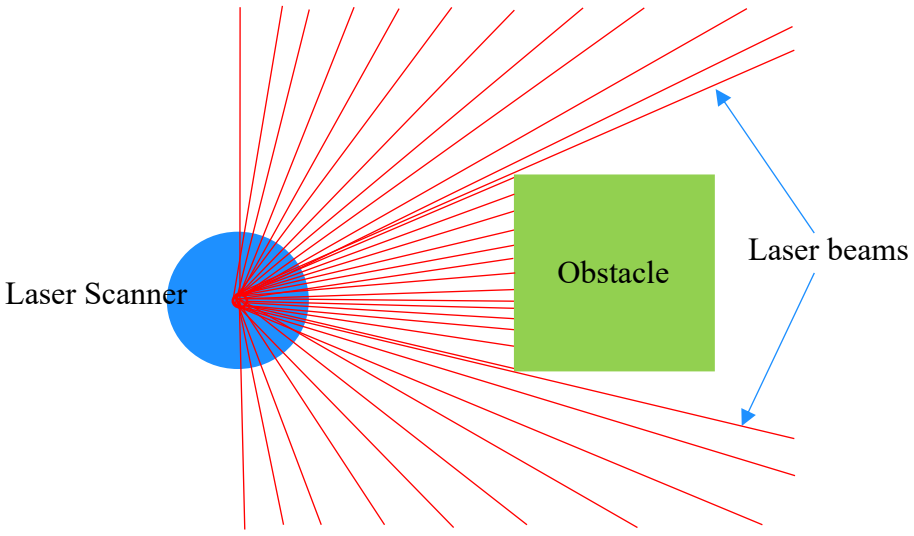

Fig 2.4: Typical view of laser scan.

Fig 2.4 shows the typical laser scan view. Here the blue circle is the laser scanner, green square is the obstacle or object and the red lines are the laser beams reflected from the scanner.

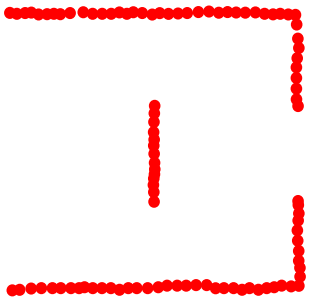

Fig 2.5: Typical map generated from laser scan.

Fig 2.5 shows the typical map of an environment from laser scanner. The small red dots are the particles that strikes the laser beams and the free space are those places where laser beams can not enter.

3

Proposed System

In this chapter, a brief description of our proposed system will be shown. We discussed the architecture of the proposed system with the necessary hardware and software we used. We also discussed the working procedure, system algorithm, the flowchart of the algorithm, and so on.

3.1 Proposed System Architecture

The implementation of the proposed system is based on different hardware components that will communicate with each other with the valid software we used. For the hardware components, we used a 2D LiDAR laser scanner to sense the environment and Jetson Xavier NX Devkit for the processing platform. In our proposed system we have also ESP32 for the motion with two motors connected with wheels, an order management system, Graphical User Interface (GUI) to interact with humans (Figure 3.1).

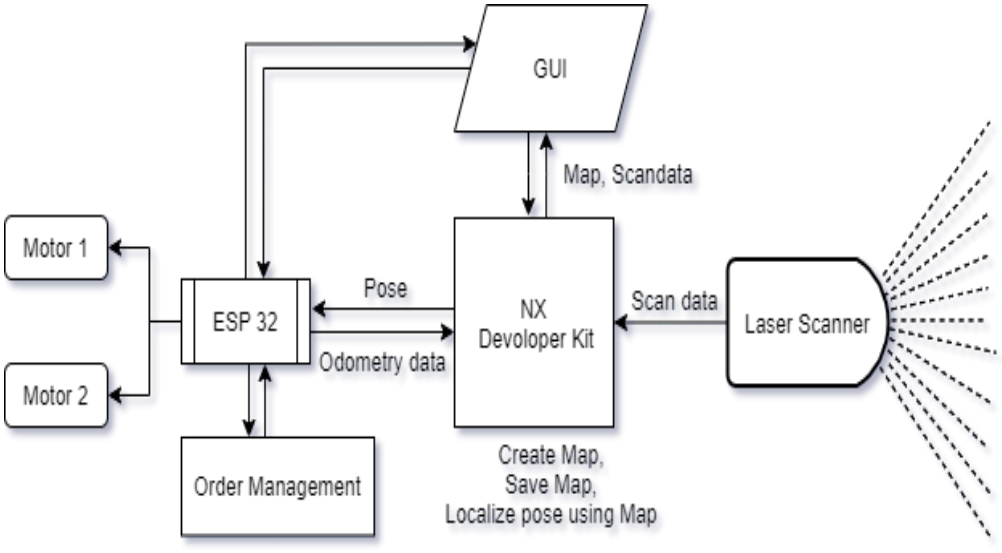

Fig 3.1: Architecture of the proposed system.

3.2 Working procedure of the Proposed System

As we know for SLAM implementation, we need information from the environment. To achieve our goal, we have chosen a 2D laser scanner. The laser scanner is directly connected with the NX Devkit over the ethernet cable. The UDP protocol is used for transferring data from the laser scanner to the NX board. The NX board is the computation unit of our system. By running the RBPF based SLAM software we can create a map, save the map of the environment from the scanner data. Then we can localize the vehicle pose from the saving map with the matching of the newly generated map. Then we have a GUI connect to our project to interact with the vehicle. The connection between NX and GUI is duplex. Then we have an ESP32 for the motion model. NX board receives data of the odometry from the ESP32 and then calculates the position of the robot by the previously saved map. And ESP32 receives the pose of the vehicle. Then we have an order management system to give the robot some order to pick something from one source to drop that in the destination path.

3.3 Hardware used for the proposed system

In this section, we have described our main hardware for the system we used to obtain our final result. We have chosen our hardware best fit for the system for higher performance.

LiDAR

- By the Time of Flight(ToF) process, LiDAR obtains the object distance from the middle of the laser scanner, to obtain the object coordinates in the LiDAR coordinate system, which has the benefits of high frequency, high accuracy, long detection range, and negligible effects of external illumination [11]. In large-scale, light-dark environments with LiDARs, SLAM approaches can be used. The knowledge from the point cloud gathered by LiDAR may be used to approximate the pose of the vehicle in an iterative optimization process, since the initial value of the pose transformation is known or guessed.

Fig 3.2: Sick TIM781s LiDAR

We used sick TIM781s LiDAR [12] for the implementation. To determine the distance from the center point that is the scanning point to the LiDAR base, the LiDAR uses the ToF method. The ToF solution means that the LiDAR transmitter emits an ultra-short laser pulse and the laser is transmitted onto the target to reflect sparsely. The LiDAR receiver absorbs the diffuse reflection light and the scan point is determined as per the moment the laser beam flies in the atmosphere. The distance from the scanning point of the LiDAR is computed as [12],

$$d = c \times (\Delta t/2)$$

In LiDAR light is emitted and by the light the distance from the center of the LiDAR to the object's scanning point is calculated, and this is achieved by the transmit time and receive time from the scanner.

Table 1: TIM781s LiDAR working parameters.

Parameter Names	Numerical Value
Light source	Infrared (850 nm)
Working angle (Horizontal)	270°
Frequency	15 Hz
Resolution	0.33°
Working range	0.05 m ... 25 m (> 90% remission)
Blind spot	0 m ... 0.05 m
Maximum scanning distance	8m

- Jetson Xavier NX Devkit
  - We used the embedded Xavier NX System-on-Module (SoM), Development Kit [13]. A CPU, DRAM, a GPU, and flash storage are included. SoM is an embedded single board that has only a row of different connector pins for different I/O. All the pins on the embedded board are attached to different types of ports like HDMI, USB, and Ethernet. A Jetson NX Development kit is an identical board to a Raspberry Pi or any other Single Board Computers which are paired with a development board.

Fig 3.3: Jetson Xavier NX Devkit

We get Nvidia's custom Carmel ARM-based cores with the board. We also get a 384-core Volta-based GPU, and a Hexa-core Processor, and 8 GB LPDDR4x RAM. DisplayPort, HDMI, 4x USB 3.1 ports, Gigabit Ethernet, Bluetooth, Wi-Fi, 2x camera connectors, an SSD M.2 slot, and 40 GPIO pins added to the development board.

3.4 Software used for the proposed system

From the software perspective, we used an open-source RBPF based SLAM software package and developed the approach that is available as a library and easily used as a black box [14]. Making changes to the algorithm was done using C++. Other than that we have used Linux Operating System (OS) to run all the software packages. To implement the GUI, we used ReactJS and for transferring the map data to GUI we used WebSocket. WebSocket is a computer communication protocol that provides full-duplex communication channels over a single HTTP TCP connection. For request or reply based communication we used HTTP communication.

3.5 System Algorithm

We used RBPF based SLAM technique for our system because of the higher accuracy and low computational complexity. The improved RBPF algorithm which we are used is stated as below,

I. For each particle, do,
II. Using previous pose x_{t-1}, sampling a new pose x_t from the odometry data,
III. Use the sampled pose as the beginning of the estimate, perform a local search in the map by maximizing the scan-matching score (match laser scan to map),
IV. Update weight of the particle using the likelihood best-found pose,
V. Calculate the pose xt-1 of the particle to best-found pose x_t,
VI. Find the number of effective particles Neff from total weights.
VII. If Neff < 0.5 * Ntotal; do a weighted random resampling of all particles.

3.6 Flowchart of the System Algorithm

We have discussed our system algorithm using a flowchart diagram below in fig 3.4.

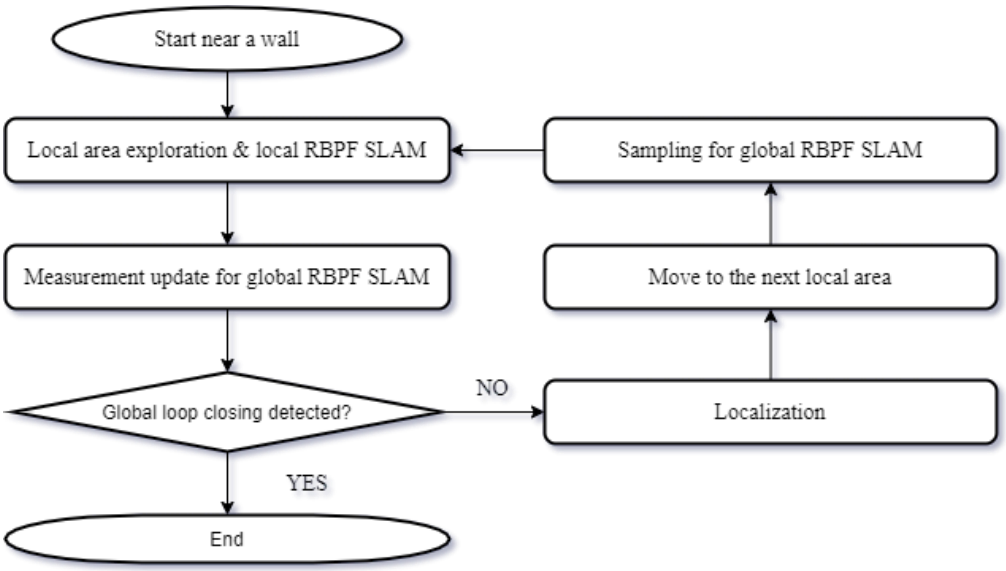

Fig 3.4: Flowchart of the improved RBPF algorithm.

The system starts near a wall that means the physical obstacles that are static in the environment and it completes generating maps for the local area that means one room of the whole model and does a local RBPF SLAM. After that, the system updates all the necessary measurements for the global RBPF SLAM which means for the whole model of the environment. Then it makes a decision

if the global loop closing is detected that means the whole environment exploration is complete the algorithm stops generating the map. If the global loop closing is not detected it continues localization and finds the next local area to explore and then sampling data for the global RBPF SLAM and then go back to the loop to explore the local area again. The loop is continuously going on if it doesn't reach the global closing loop.

4

Methodology

In this chapter, a brief discussion of various types of SLAM techniques will be accomplished. We will also discuss the algorithm and computational complexity for each technique.

4.1 Introduction to SLAM

Simultaneous localization and mapping (SLAM) is an issue in which a moving object has to construct a map of an unknown area while calculating its location within this map at the same time [15]. There are many fields that could benefit from the introduction of autonomous vehicles with SLAM algorithms. The mining industry, marine exploration, and planetary exploration should be examples. Using a probability density function denoted $p(x_t, m|z_{1:t}, u_{1:t})$, where x_t is the vehicle's location, m is the map, and $z_{1:t}$ is a vector of all measurements, the SLAM problem can generally be formulated [24]. Depending on the function, $u_{1:t}$ is a vector of the control signals of the vehicle, which is either the control commands themselves or odometry [15].

SLAM consists of multiple parts; Landmark extraction, data association, state estimation, state update and landmark update. There are many ways to solve each of the smaller parts [10].

Several study groups and researchers have worked and are currently working on SLAM, and laser-based, sonar-based and vision-based systems can be grouped into the most widely used sensors. To better interpret robot state data and the outside world, additional sensory sources are used, such as compasses, infrared technology and the Global Positioning System (GPS) [11]. All these sensors, however, bear some errors, also referred to as measurement noise, and often have several range limitations that make it difficult to navigate through the environment, such as walls that can’t be penetrated by light and sound.

4.1.1 Full SLAM and online SLAM

Two different forms of problems with SLAM exist. The online SLAM problem of which, given the control input u1:t and measurements z1:t, only the current pose x_t and the map m is expressed. As well as the total issue of SLAM that

expresses the entire robot trajectory. The complete SLAM problem's PDF is referred to as $p(x_{0:t}, m|z_{1:t}, u_{1:t})$, where all the robot poses are taken into account, including its initial pose x_0 [16].

4.2 EKF SLAM

In an Extended Kalman Filter (EKF) based method the SLAM problem is solved using sensor data obtained from the robot's motion and rotation. For example, by using wheel encoders, and gyroscopes, this can be done. In addition, it is often important to collect environmental information by using, for example, a Laser Range Finder (LiDAR) or camera. With this knowledge, SLAM preserves track of where the robot is probable to be located on a map and keeps track of unique landmarks observed [10].

Because the robot odometry (which gives the location of the robots) is always inaccurate, we cannot rely on the odometry directly. To correct the location of the robot, we can use laser scans of the surroundings. This is done by removing and re-observing features from the environment as the robot travels some distance. Such attributes are generally called landmarks. The heart of the EKF based SLAM method is an EKF (Extended Kalman Filter). It is responsible for adding new landmarks to the model, updating old landmarks, and also updating the current trajectory. The EKF keeps track of an estimation of the uncertainty in the position of the robot and also of the uncertainty in the environment of the landmark positions [10].

4.2.1 Algorithm

When the robot is switched on the sensors on the robot, such as the wheel encoders and the gyroscope, capture robot position information. Moreover, based on new findings from the LiDAR that is mounted on the robot, landmarks from the area are also collected. These new findings are related to previous observations and are modified as part of the EKF algorithm. However, if an observation of a landmark cannot be related to a prior observation, the observation itself is introduced as a new observation to the EKF algorithm. [5] This mechanism is seen in Fig 4.1.

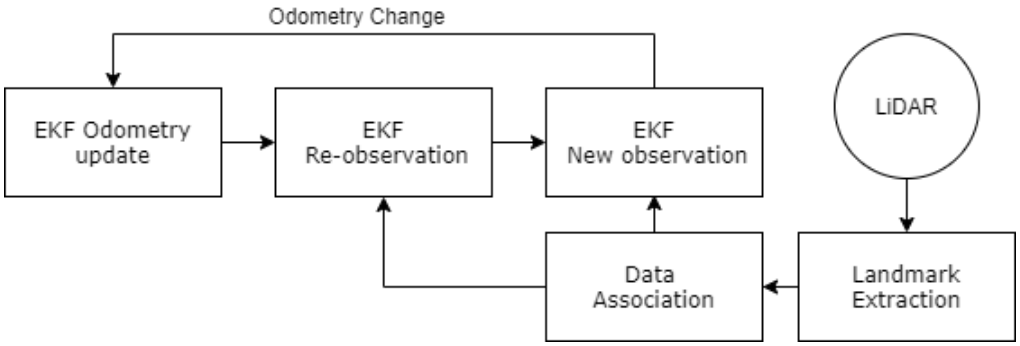

Fig 4.1: EKF Based SLAM process

First and foremost, the robot uses observed landmarks and information from the sensors to localize itself within the map being generated. Preferably, landmarks should be visible from various angles and not isolated from other landmarks. These prerequisites provide a good opportunity for the EKF algorithm to differentiate between landmarks at a later date. In addition, the landmarks used should also be stationary [10]. An EKF SLAM process consists of two stages in shorts. The process is as follow,

- Prediction step: In this step, it predicts the current position from the predicted mean and the predicted covariance. The calculation is based on control input, matrices, the previous covariance, and the previous mean [4].
- Correction step: In this step, the idea is the data association. In this data association process, the idea is to differentiate between an earlier observed landmark and a newly observed one [4]. If a new measurement has occurred then the procedure is the following,
  - Save the location of the newly observed landmark [4].
  - Associate old observed landmarks with newly observed landmarks [4].

4.2.2 Computational Complexity

Compared to other SLAM algorithms, the cost of computing the EKF SLAM algorithm is quite expensive. As new landmarks are observed, they are applied to the filter's state vector, and the map of landmarks marked with N is linearly increased [20]. The use of memory in terms of numbers is $O(N^2)$. The cost of calculating one step of the algorithm is nearly $O(N^2)$ [19], the complete cost of calculating the entire algorithm is $O(N^3)$, which depends heavily on the map size [19].

4.3 FastSLAM

The FastSLAM algorithm, which allows the use of the Rao-Blackwellized particle filter, is another way to solve the SLAM problem. FastSLAM takes full advantage of something which many other SLAM algorithms do not, notably that only a limited number of state variables are associated with each observation. The robot's position is likely based on its previous position, while the landmark positions are likely dependent on the robot's location. [21] The EKF SLAM algorithm, for instance, which has to recalculate the covariance matrix with each change, does not take this into account, resulting in weak scaling on large maps. [21] In FastSLAM, this is not a concern, since it factors the equations into simpler subproblems

$$P(s^t , \theta \mid z^t , u^t , n^t) = P(s^t \mid z^t , u^t , n^t) \prod_{n=1}^{N} P(\theta_n \mid s^t , z^t , u^t , n^t)$$

Contributing to the robot direction s^t and each of the N landmarks θ_n. Here, one important distinction between FastSLAM and EKF SLAM, for instance, is that the entire direction s^t is measured, as compared to only the current pose x^t. This implies that a complete SLAM algorithm is FastSLAM and an online SLAM algorithm is EKF SLAM.

4.3.1 Algorithm

The FastSLAM algorithm can be summarized into four steps. These are as follows,

1. For each of the M particles, sample a new robot path s_t from $p(s_t \mid s_{t-1}^{[m]} , u^t)$
2. With the new calculation, update the landmark filters
3. Give each particle an importance factor
4. Resample the particles, with particles with a higher importance factor having a higher draw probability

In step 1, for each particle, a new robot route is sampled. This path, with the robot control as feedback, is taken from the motion model. After this because of the current calculation, the posteriors of the landmarks are changed. After that, a significance factor is assigned to each particle, which is a measure of how practical the variables in the particle are. Later, to get rid of those that do not have a high chance, the particles are resampled. [21]

4.3.2 Computational Complexity

If the FastSLAM as mentioned above is explicitly applied, it has the computation cost of O(M*N), where M is the total number of particles and N is the total number of landmarks. [21] It wouldn't be a smart idea to incorporate an algorithm like this because it would scale linearly with the number of landmarks. For large maps, this will make the algorithm unsuitable. FastSLAM is introduced using binary trees instead of the array structure. The algorithm will then have O(M(log(N))) complexity [5].

4.4 Graph-based SLAM

Another solution to addressing the SLAM problem is GraphSLAM. Using a graph, the idea behind GraphSLAM is to answer the SLAM problem. The graph includes nodes representing the robot $x_0....x_t$ poses as well as landmarks on the map denoted as $m_0.....m_t$. However, to resolve the SLAM problem, this is not enough as there is also dependency between the robot's positions x_t, x_{t-1}, $x_{t-2}....x_{t-n}$ and the landmarks $m_0....m_t$ [22]. The dependency is the distance between neighboring locations such as x_{t-1}, x_t, as well as the distance to the landmarks $m_0.....m_t$ across locations. Due to data from the odometry source u_t, these dependencies can be constructed [22] There is a possibility to convert the graph described earlier to a form of the sparse matrix. Which can be used more efficiently, whereas dense matrices [16].

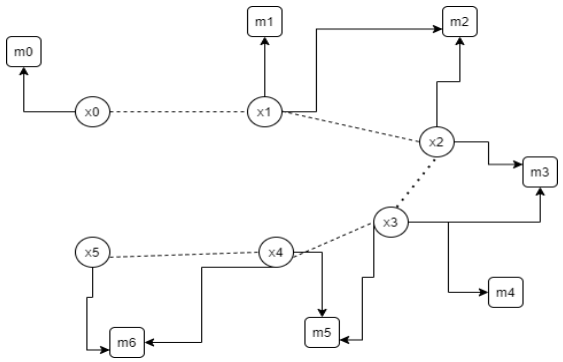

Fig 4.2: Graph-based SLAM

In fig-4.2, we can see how the landmarks $m_0....m_t$ and the locations that are $x_0...x_t$ and the constraints are linked together. The square circles are the robot's pose in different aspects of time. The squares are the landmarks in the environmental model. The dotted line represents the robot’s motion based on the Robot-specified commands. And lastly, the clear lines represent the estimation of the distance from the landmarks.

4.4.1 Algorithm

$P(x_{0:t}, m|z_{1:t}, u_{1:t})$ is represented as the full SLAM problem. All the $x_{0:t}$ poses and the m landmarks can be represented by a state-space vector y for simplicity.

Then an optimization algorithm must be computed in order to obtain the most probable location of the robot in the map and the coordinates of the landmarks on the map. [16] In addition, $p(y|z_{1:t}, u_{1:t})$ includes the likely nonlinear $g(u_t, x_{t-1})$ and $h(x_t, m_i)$ functions, while mi is the i-th landmark observed at time t [16]. Linear functions are predicted by the optimization algorithm. An essential step, therefore, is to linearize them. This may be achieved by performing a Taylor linearization algorithm, for instance. [16]

The GraphSLAM approach to solving the SLAM problem is by adding a graph, as described earlier. It is possible to transform this graph to a sparse matrix. This sparse matrix essentially comprises all the data that the robot acquires when traveling through an area. The robots travel through the map and landmarks found at various sites, for example. Therefore, the sparse matrix will grow in size and produce still more details as the robot travels around the environment [16].

4.4.2 Computational Complexity

Computational load and memory consumption are some interesting figures when describing the GraphSLAM algorithm. Compared to the EKF SLAM algorithm, which increases memory use by $O(N^2)$ when new landmarks N are added and computational costs by $O(N^3)$, when new landmarks N are introduced, the memory use of GraphSLAMs gradually increases. As with the computation complexity of the GraphSLAM algorithm, it is time-dependent, so if the vehicle's journey is very long, the computational cost will be expensive. In general, however, the properties of the GraphSLAM algorithm are very attractive when there are a lot of landmarks in the setting, which appear to fail with the EKF algorithm.[22][5]

4.5 Grid-map based SLAM

In a grid-map based method, the model is expressed as a regular pattern of cells arranged in a grid-like manner. Each cell in the model corresponds to a unit physical space. A regular grid-like organization in 2D is known as a grid map; while in 3D the same is expressed as a voxel-map. Laser scans from a scanning laser range finder are collected and integrated online into a grid representation. During this integration, the continuous laser scans are discretized into the cell representation. The integration of new laser scans is only possible when a reliable estimate of the current pose (orientation and position) is available. Odometry as a second input is (generally) required. Grisetti G developed a

factored particle filter-based approach and tested it on a wide range of data sets - providing an efficient and superior solution to the SLAM problem using grid maps [8]. His methodology has also recently been integrated into the software suite of the Robot Operating System (ROS) [23], where a module based on the algorithm of Adaptive Monte Carlo Localization (AMCL) [5] uses the maps generated by RBPF based SLAM. Within ROS, the interface is transparent and well documented. Although RBPF based SLAM itself is in the research domain; we are aware of many groups that have successfully used it and we believe it is the most realistic method for navigating indoor robots or similar applications.

4.6 Comparison between different SLAM

It is possible to compare SLAM algorithms using different qualitative and quantitative metrics. We decided to compare the interesting approaches based on applicative features such as robustness, computational requirement, and failure modes. The comparison is as follows,

- EKF SLAM
  - Robustness: Faulty data association results in incorrect results, so the EKF SLAM's robustness is poor [16].
  - Computational Efficiency: The complexity of the algorithm is $O(N^2)$. Here N is the number of landmarks observed; or contained in the map. In case of a single landmark being observed, the whole map needs to be updated. Therefore, it is suitable for minor maps with fewer landmarks due to substantial computation [15]. But localization and data association will be problematic when the landmarks are few in the environment [16].
- Gridmap based RBPF SLAM
  - Robustness: Gridmap based SLAM is significantly more robust than EKF because of the data association problem. Instead of landmarks, using a particle filter reduces the impact of erroneous data association.
  - Computational Efficiency: The computational complexity is stated as O(M(log(N))). Here N is the number of landmarks and M is the number of particles used to compute the complexity.
- Graph-based SLAM
  - Robustness: The GraphSLAM algorithm is more robust than that of the EKF algorithm, partially because it is possible to re-examine older data associations if an incorrect data association

has been created. This effectively decreases the likelihood of creating erroneous future data correlations, thus increasing the robustness of the data.

- Computational Efficiency: In contrast to the EKF algorithm that uses $O(N^2)$ where N is the number of landmarks, the use of memory is linearly dependent on the number of landmarks, O(N). In addition, the computational load is period dependent, so if the path of the surrounding is larger, the algorithm cost can be very expensive.

From the above comparison, we can see the EKF SLAM is less robust in comparison to Gridmap based RBPF SLAM and Graph-based SLAM. We also see that the EKF SLAM algorithm is more costly in terms of efficiency. So, we have chosen to implement Gridmap based SLAM with a 2d LiDAR sensor.

5

Real-World Experiment

In this chapter, we have discussed the overall overview of the thesis. We also show how the thesis obtains its final results from the data set we used and at the end we have a discussion about our system.

5.1 Overview

Chapter 3 introduced the implementation of RBPF based SLAM algorithms with necessary hardware and software components. From the implementation, it was decided that RBPF based SLAM was the appropriate algorithm to use with the physical vehicle. This section describes a real-world experiment featuring a map generated by the physical vehicle while running the RBPF based SLAM algorithm. The real-world experiment of the RBPF based SLAM algorithm was performed with the data from the 2D LiDAR. The data set we used is shown in the below section.

5.1.1 Dataset Used

In Fig 5.1, 5.2, and 5.3 the data set from the laser scan is shown. Each of the data set there is the Odometry data that is the ODOM which contains the value of x, y, and theta. The FLASER is the data from the laser scanner and these contain 360 data from the polar coordinates. And these are the values of theta and the angular resolution for the scanner is 0.33 which is the value of r.

ODOM 0.000485382 6.20016e-05 -0.001534 0 0 0 0.227623 pippo 0.227623
FLASER 360 1.65 1.66 1.63 1.63 1.63 1.63 1.68 1.75 1.74 6.94 6.96 6.98 6.98 6.87 6.87 7.3 6.89 6.85 6.83 6.85 6.9 7.41 7.41 7.44 7.45 81.91 4.28 4.14 4.11 4 3.84 3.85 3.8 3.79 3.79 3.79 3.84 3.85 3.83 3.8 3.83 3.97 4.59 4.6 4.62 4.65 4.66 4.68 4.7 4.72 4.74 4.05 3.73 3.69 3.65 3.66 3.67 2.09 2.01 1.97 1.97 1.98 2 2.06 2.08 2.09 2.11 2.13 2.15 2.15 2.17 2.19 2.2 2.22 2.24 2.25 2.27 2.29 2.3 2.32 2.35 2.37 2.39 2.41 2.43 2.46 2.47 2.5 2.53 2.56 2.59 2.6 2.63 2.67 2.7 2.72 2.75 2.78 2.82 2.85 2.88 2.92 2.92 2.91 2.95 2.96 2.94 2.93 2.91 2.89 2.91 2.95 2.99 3.04 3.1 3.71 3.77 3.85 3.92 3.99 4.05 4.13 4.21 4.29 11.03 11.01 4.54 4.51 4.49 4.56 4.83 4.94 5.07 5.21 5.34 5.49 5.63 5.83 5.86 5.93 6.28 6.49 6.7 6.94 7.18 7.29 8.2 8.18 8.26 8.62 8.63 8.71 10.13 10.17 10.16 10.18 10.17 10.16 10.16 10.15 10.12 10.1 12.33 12.29 12.29 12.28 12.27 12.27 12.27 12.27 12.4 12.41 10.13 10.08 10.1 10.11 10.11 10.12 10.12 10.1 10.09 8.69 8.55 8.54 8.16

8.09 7.19 6.92 6.49 5.9 5.73 5.68 5.4 5.13 4.92 4.54 4.32 4.27 4.14 4 3.87 3.71 3.59 2.85 2.73 2.65 2.63 2.63 2.64 2.65 2.65 2.66 2.65 2.61 2.6 2.6 2.52 2.47 2.42 2.38 2.32 2.28 2.24 2.2 2.17 2.13 2.09 2.06 2.02 1.99 1.97 1.94 1.9 1.88 1.85 1.82 1.8 1.77 1.75 1.74 1.72 1.69 1.67 1.65 1.63 1.61 1.6 1.59 1.57 1.55 1.53 1.51 1.51 1.48 1.44 1.41 1.38 1.37 1.37 1.38 1.39 1.4 1.4 1.39 1.36 1.35 1.35 1.33 1.33 1.32 1.3 1.29 1.29 1.28 1.26 1.26 1.26 1.25 1.24 1.23 1.22 1.21 1.21 1.2 1.19 1.18 1.18 1.18 1.17 1.17 1.16 1.15 1.15 1.15 1.15 1.14 1.13 1.12 1.11 1.11 1.11 1.11 1.11 1.1 1.09 1.09 1.08 1.08 1.07 1.06 1.06 1.06 1.06 1.06 1.05 1.05 1.05 1.05 1.05 1.04 1.05 1.04 1.04 1.03 1.03 1.03 1.03 1.02 1.02 1.02 1.02 1.02 1.02 1.01 1.02 1.01 1.01 1.01 1.01 1.01 1.01 1 1 1 1 1 1 1 1 1 0.99 1 1 1 1 1 0.99 0.99 1 0.00123601 -0.00106807 2.85e-05 0.00123601 -0.00106807 2.85e-05 0.227623 pippo 0.227623
ODOM 0.00123601 -0.00106807 2.85e-05 0 0 0 0 pippo 0
NEFF 10

Fig 5.1: Laser scanner dataset one

ODOM -0.000363369 -0.000317486 2.85e-05 0 0 0 0.700156 pippo 0.700156
FLASER 360 1.66 1.65 1.63 1.63 1.63 1.64 1.68 1.75 1.74 6.94 6.95 6.97 6.96 6.86 6.87 7.3 6.89 6.84 6.83 6.84 6.9 7.4 7.41 7.45 7.44 81.91 4.28 4.14 4.1 4 3.84 3.84 3.78 3.78 3.79 3.78 3.84 3.85 3.84 3.8 3.83 3.97 4.59 4.6 4.62 4.64 4.65 4.68 4.7 4.72 4.74 4.03 3.72 3.69 3.64 3.66 3.68 2.08 2.01 1.96 1.97 1.99 2 2.06 2.07 2.09 2.11 2.13 2.14 2.15 2.16 2.19 2.2 2.22 2.24 2.25 2.27 2.29 2.3 2.32 2.35 2.36 2.39 2.4 2.43 2.46 2.47 2.51 2.53 2.57 2.58 2.61 2.63 2.67 2.69 2.72 2.76 2.78 2.81 2.85 2.88 2.91 2.92 2.9 2.96 2.96 2.94 2.93 2.91 2.9 2.91 2.95 2.99 3.05 3.1 3.7 3.77 3.84 3.91 3.99 4.05 4.13 4.21 4.29 11.02 11.01 4.55 4.51 4.49 4.57 4.83 4.95 5.08 5.21 5.35 5.5 5.63 5.83 5.86 5.92 6.27 6.49 6.71 6.95 7.18 7.29 8.19 8.17 8.27 8.62 8.62 8.72 10.13 10.17 10.16 10.17 10.18 10.16 10.15 10.14 10.12 10.11 12.33 12.28 12.28 12.27 12.26 12.26 12.26 12.26 12.4 12.4 10.12 10.07 10.1 10.11 10.11 10.12 10.13 10.1 10.08 8.68 8.54 8.54 8.16 8.08 7.18 6.92 6.48 5.85 5.73 5.68 5.39 5.13 4.91 4.51 4.31 4.27 4.13 3.99 3.86 3.71 3.58 2.85 2.72 2.65 2.63 2.63 2.63 2.64 2.65 2.66 2.65 2.61 2.6 2.6 2.52 2.46 2.41 2.37 2.32 2.28 2.24 2.2 2.16 2.13 2.09 2.06 2.02 1.99 1.95 1.93 1.9 1.88 1.85 1.82 1.8 1.77 1.75 1.73 1.72 1.69 1.67 1.65 1.63 1.61 1.6 1.59 1.57 1.55 1.53 1.51 1.49 1.48 1.43 1.41 1.38 1.37 1.37 1.38 1.39 1.4 1.39 1.37 1.36 1.36 1.35 1.34 1.33 1.32 1.3 1.3 1.28 1.27 1.27 1.25 1.25 1.25 1.24 1.23 1.22 1.21 1.21 1.2 1.19 1.18 1.17 1.18 1.17 1.17 1.16 1.16 1.15 1.15 1.15 1.14 1.13 1.12 1.11 1.1 1.1 1.1 1.1 1.09 1.09 1.09 1.08 1.08 1.07 1.07 1.06 1.06 1.06 1.05 1.05 1.05 1.04 1.04 1.04 1.04 1.04 1.04 1.04 1.03 1.03 1.03 1.03 1.02 1.02 1.02 1.02 1.02 1.01 1.02 1.01 1.01 1.01 1.01 1.01 1.01 1.01 1 1 1 1 0.99 0.99 1 0.99 0.99 0.99 0.99 0.99 1 1 1 0.99 1 0.99 0.99 0.000247391 -0.00217191 2.85e-05 0.000247391 -0.00217191 2.85e-05 0.700156 pippo 0.700156
ODOM 0.000247391 -0.00217191 2.85e-05 0 0 0 0 pippo 0
NEFF 9.99998

Fig 5.2: Laser scanner dataset two

ODOM -19.7585 0.896626 -1.75272 0 0 0 241.585 pippo 241.585

FLASER 360 4.54 4.55 4.53 4.49 4.47 4.48 4.51 4.55 4.54 4.5 4.5 4.49 4.49 4.54 4.54 4.52 4.52 4.64 4.65 4.65 4.66 4.58 4.58 4.59 4.6 4.52 4.54 4.11 4.11 4 4 4.74 4.72 3.16 3.16 3.16 3.16 4.67 4.68 4.69 4.69 4.56 4.57 4.33 4.36 2.94 2.95 2.72 2.74 2.51 2.53 1.94 1.94 1.86 1.87 1.81 1.81 1.74 1.74 1.75 1.76 1.78 1.78 1.79 1.79 1.75 1.75 1.69 1.69 1.64 1.65 1.6 1.59 1.55 1.56 1.52 1.52 1.49 1.48 1.46 1.45 1.42 1.42 1.39 1.39 1.36 1.36 1.33 1.33 1.32 1.31 1.29 1.28 1.26 1.25 1.24 1.24 1.21 1.22 1.2 1.19 1.18 1.17 1.15 1.15 1.14 1.14 1.12 1.12 1.11 1.11 1.09 1.09 1.08 1.08 1.07 1.07 1.06 1.07 1.05 1.04 1.03 1.03 1.02 1.02 1.02 1.02 1 1 0.99 0.99 0.98 1 1.02 1.02 1.07 1.07 1.09 1.09 0.87 0.88 0.84 0.83 0.81 0.83 0.81 0.82 0.82 0.81 0.85 0.84 3.04 3.03 3.08 3.07 81.91 81.91 3.8 3.81 3.71 3.91 4.9 4.91 4.45 4.46 4.39 4.39 4.34 4.34 4.38 4.38 4.38 4.38 4.46 4.45 6.3 6.31 2.96 2.96 2.88 2.89 6.35 6.35 2.84 2.84 2.77 2.77 2.76 2.76 2.82 2.81 3 3 3.01 3.01 5.5 5.5 5.5 5.5 5.5 5.5 5.4 5.4 5.43 5.42 5.44 5.44 4.47 4.5 4.45 4.45 3.44 3.44 3.26 3.27 3.04 3.05 2.87 2.87 2.71 2.71 2.56 2.57 2.38 2.39 2.27 2.27 1.89 1.91 1.8 1.8 1.79 1.8 1.68 1.68 1.65 1.65 1.52 1.53 1.48 1.49 1.44 1.44 1.34 1.34 1.31 1.31 1.21 1.22 1.15 1.15 1.11 1.11 1.07 1.07 1.11 1.1 1.09 1.09 1.07 1.08 1.05 1.04 1.02 1.02 1 1.01 1.02 1.02 1.05 1.05 1.37 1.37 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 1.39 1.39 1.35 1.36 1.32 1.33 1.32 1.32 1.36 1.36 1.39 1.39 1.36 1.37 1.35 1.35 1.33 1.33 1.31 1.31 1.3 1.3 1.3 1.3 1.34 1.34 1.39 1.38 1.44 1.44 1.5 1.49 6.2 6.2 6.01 6.08 5.68 5.68 3.36 3.35 3.29 3.29 3.28 3.29 2.92 2.92 2.9 2.9 3.24 3.22 3.47 3.47 3.76 3.75 4.13 4.12 4.26 4.26 5.2 5.2 5.21 5.21 5.66 5.65 5.69 5.69 8.5 5.92 9.67 9.67 10.15 10.15 11.8 -19.7521 0.894171 -1.71437 -19.7521 0.894171 -1.71437 241.585 pippo 241.585
ODOM -19.7521 0.894171 -1.71437 0 0 0 0 pippo 0
NEFF 5.93671

Fig 5.3: Laser scanner dataset three

5.2 Results

The results of the implementation of RBPF based SLAM are shown in the below section. This section represents real-life experiments of the algorithm. Each figure from below shows 2D maps of the model or environments. All the maps generated are highly accurate.

Maps from the simulation

This section below shows the simulated maps of the algorithm. For this simulation, we used the recorded data from online. We were able to see this simulation by the GUI of the QT creator.

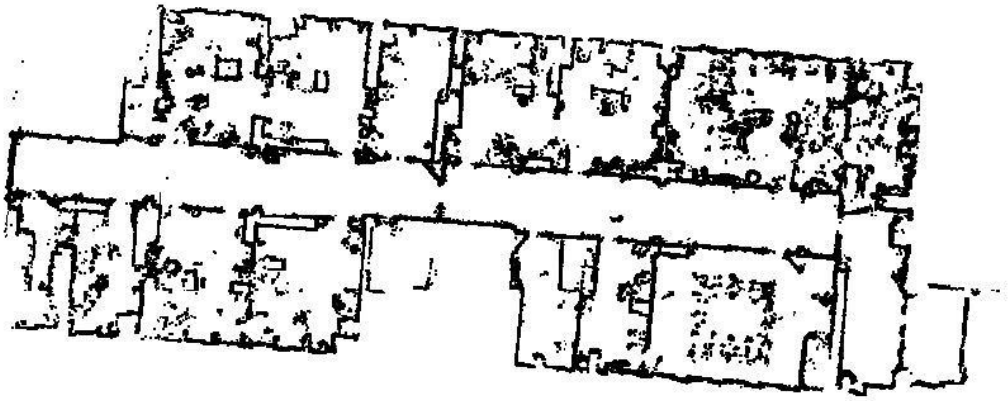

Fig 5.4: 2D maps generated from the simulation

Fig-5.4, shows 2D maps from the simulation from the recorded data with higher accuracy. This is basically a grid-based map. Each map consists of various particles.

Maps from 2D LiDAR Scan data

This section below shows the generated maps from the implemented algorithm. For this, we used Jetson NX Devkit as the computation unit. For the data collection, we used a 2D LiDAR rangefinder.

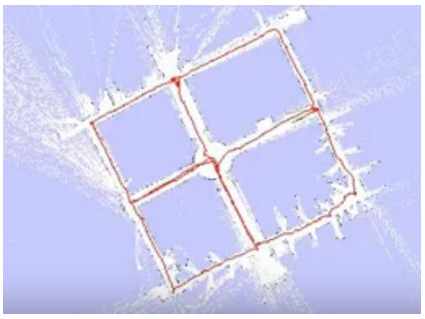

Fig 5.5: 2D maps of the whole environment from LiDAR data.

Fig-5.5, shows 2D maps for the whole environment from the LiDAR data with higher accuracy.

Fig 5.6: 2D maps of a single room from the LiDAR data.

Fig-5.6, shows 2D maps of a room from the whole environment which is generated from the LiDAR scanned data.

Fig 5.7: Detailed view of the portion of a map generated from the LiDAR data.

Fig-5.7, shows zoomed 2D maps of a small portion of the map from the whole environment which is generated from the LiDAR scanned data. These are the particles of a bigger portion of the maps.

The accurate position of the vehicle is calculated form the RBPF based SLAM algorithm. The data of the position of the vehicle from the log is shown in the below section.

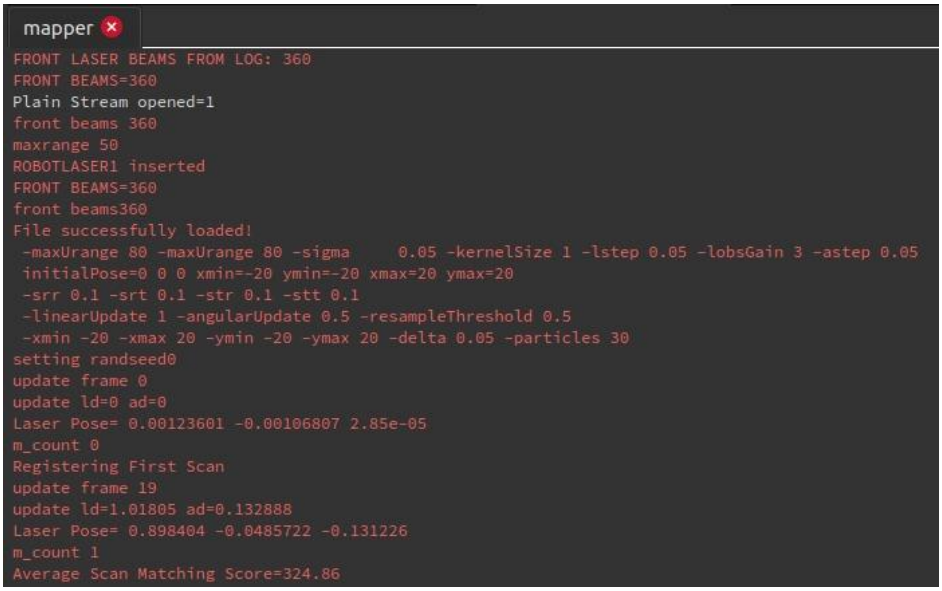

Fig 5.8: Position data of the vehicle at startup.

Fig 2.8, shows the position of the startup of the vehicle. This is the real-time position of the robot at that time in the environmental model.

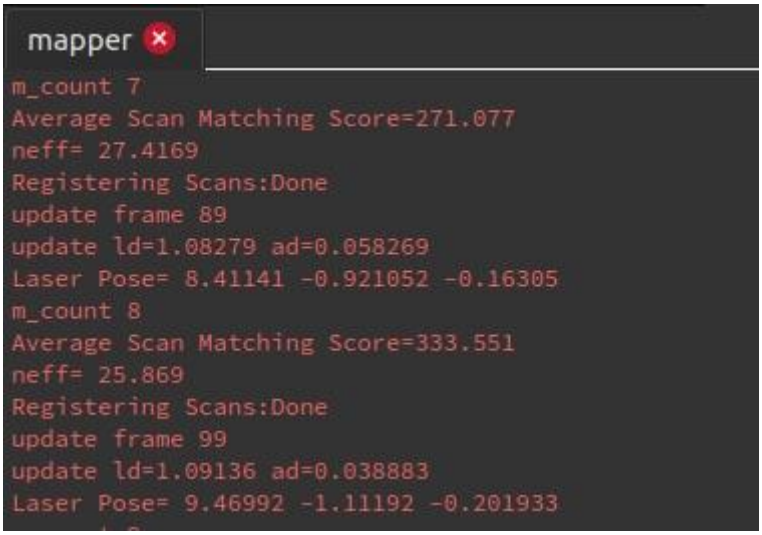

Fig 5.9: Position data of the vehicle while moving

Fig 2.9, shows the position of the moving vehicle. This is the real-time position of the robot at that time in the environmental model.

5.3 Conclusion

The maps are shown in fig. 5.5, 5.6, and 5.7 show that the RBPF based SLAM algorithm can successfully create a map of the environment. All the objects in front of the vehicle are successfully depicted on the map. The positioning part of the algorithm from the log of the NX Devkit is shown in fig 5.8 and 5.9. We can see that the RBPF based SLAM can successfully calculate the vehicle position at different aspects of time. The accuracy of the vehicle map and the position was very good for an autonomous vehicle.

6

Discussion and Future work

In this chapter, we have discussed the achievement of the thesis. We also mentioned the difficulities throuthout the thesis. We also talked about how we can improve our system, or add some new features in next future.

6.1 Achievement

We have the following achievements from this thesis:

- A simulation model of the SLAM algorithm runs successfully.
- A small-scale vehicle that is working perfectly with the SLAM algorithm.
- A detailed study comparing different SLAM algorithms.

The model-based development approach was proven to be beneficial for this thesis. Comparing the estimated position with an actual position would have been significantly difficult if only the real-world experiment had been performed, as it would have required a highly sophisticated test apparatus. Without the model, the results section of this thesis would have been rather thin. The model was probably not necessary for the final implementation itself though, since the output of the algorithms could have been examined visually in order to evaluate which one should be implemented. The performance numbers could not have been obtained without it though, and the model-based approach can be really useful in situations where testing is difficult and/or expensive. The performance of the SLAM algorithm implemented was highly satisfactory. One of the main reasons for this project is to develop something which could be demonstrated at job fairs, the RBPF based SLAM algorithm works really well in this scenario. One of the reasons for this is that it works seamlessly without any special kind of environment, as long as too many moving objects are avoided. The results are also very visual, and could easily appeal to a large audience. The comparison of the SLAM algorithms was also successful. Several performance tests were devised, and clearly show how well the RPBF based SLAM algorithms perform in the indoor environment

6.2 Complications

During the thesis, work difficulties emerged over one of the stated questions which were to be investigated. "Collecting all hardware components, because this type of hardware is not available in Bangladesh and also very expensive" was the stated issue. As the modeling of the vehicle itself was too time-consuming, much of the time was spent on modeling the vehicle's sensors instead. This led to a rather simplified model of the vehicle, which is only roughly equivalent to the vehicle used in this thesis. As the study was about testing SLAM algorithms, it is not of high value to provide a practical motion model of the vehicle. What is relevant here is a detailed simulation of the sensors, which has been performed to a degree successfully.

6.3 Future work

Since the vehicle model has been already developed, any of the other SLAM algorithms with limited further implementation could be evaluated using it. A few newer algorithms, such as HectorSlam or Google Cartographer, that have not been discussed in this study may be included. For future purposes, we can add human leg detection from the laser scanner and that can help to interact with humans. We can also add some safety features in the system from the laser scanner for safety purposes.

References

1. M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. IEEE Transactions on Robotics and Automation, 17(3):229–241, 2001.
2. https://aethon.com/mobile-robots-for-healthcare/
3. http://eia.udg.es/~qsalvi/papers/2008-CCIAa.pdf
4. Cyrill Stachniss. Robot mapping EKF SLAM. http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam04-ekf-slam.pdf, 2012.
5. S. Thrun, D. Burgard and W. Fox. Probabilistic Robotics. MIT Press, 2005. ISBN-10:0-262-20162-3
6. H. Durrant-White and T. Bailey. Simultaneous localization and mapping. IEEE Robotics and Automation magazine, 13(2):99–108, 2006.
7. Wei Wang, Li Dongying, and Yu Wenxian. Simultaneous localization and mapping embedded with particle filter algorithm. 2016 10th European Conference on Antennas and Propagation (EuCAP), Antennas and Propagation (EuCAP), 2016 10th European Conference on, page 1, 2016.
8. Grisetti G, Stachniss C, Burgard W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. IEEE Trans Robot. 2007.
9. Odometry Kinematics, http://www.hmc.edu/lair/ARW/ARW-Lecture01-Odometry.pdf
10. EKF SLAM for dummies, https://dspace.mit.edu/bitstream/handle/1721.1/119149/16-412j-spring-2005/contents/projects/1aslam_blas_repo.pdf
11. A Survey of SLAM Research based on LiDAR Sensors, http://www.remedypublications.com/open-access/a-survey-of-slam-research-based-on-lidar-sensors-4870.pdf
12. https://www.sick.com/us/en/c/products
13. https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit
14. https://openslam-org.github.io/gmapping.html
15. M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. IEEE Transactions on Robotics and Automation, 17(3):229–241, 2001.
16. Curotto Molina and Franco Andreas. Graphslam algorithm implementation for solving simultaneous localization and mapping. Master's thesis, Universidad de Chile, 2016.

17. Implementation of SLAM algorithms in a small-scale vehicle using model-based development, https://liu.diva-portal.org/smash/get/diva2:1218791/FULLTEXT01.pdf

18. Fast large-scale SLAM with improved accuracy in mobile robot. 2010

19. IEEE International Conference on Robotics and Biomimetics, Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on, page 791, 2010.

20. S.B. Samsuri, H. Zamzuri, M.A. Abdul Rahman, and Ab Mazlan, S.A. Computational cost analysis of extended Kalman filter in simultaneous localization and mapping (EKF-SLAM) problem for autonomous vehicle.

21. Michael Montemerlo and Sebastian Thrun. FastSLAM - A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics. Springe Berlin Heildelberg New York, 1 edition, 2007.

22. Roland Siegwart, R. Illah NourBakhsh, and Davide Scaramuzza. Introduction to Autonomous Mobile Robots. Springer-Verlag London Limited 2003, 2 edition, 2011.

23. Quigley M, et al (2009) ROS: an open-source robot operating system. In: Proceedings of the ICRA-Workshop on Open-Source Robotics, software: ros.org

24. http://eia.udg.es/~qsalvi/papers/2008-CCIAa.pdf

Appendix

Sample Laser scanner dataset

ODOM -19.7585 0.896626 -1.75272 0 0 0 241.585 pippo 241.585
FLASER 360 4.54 4.55 4.53 4.49 4.47 4.48 4.51 4.55 4.54 4.5 4.5 4.49 4.49 4.54 4.54 4.52 4.52 4.64 4.65 4.65 4.66 4.58 4.58 4.59 4.6 4.52 4.54 4.11 4.11 4 4 4.74 4.72 3.16 3.16 3.16 3.16 4.67 4.68 4.69 4.69 4.56 4.57 4.33 4.36 2.94 2.95 2.72 2.74 2.51 2.53 1.94 1.94 1.86 1.87 1.81 1.81 1.74 1.74 1.75 1.76 1.78 1.78 1.79 1.79 1.75 1.75 1.69 1.69 1.64 1.65 1.6 1.59 1.55 1.56 1.52 1.52 1.49 1.48 1.46 1.45 1.42 1.42 1.39 1.39 1.36 1.36 1.33 1.33 1.32 1.31 1.29 1.28 1.26 1.25 1.24 1.24 1.21 1.22 1.2 1.19 1.18 1.17 1.15 1.15 1.14 1.14 1.12 1.12 1.11 1.11 1.09 1.09 1.08 1.08 1.07 1.07 1.06 1.07 1.05 1.04 1.03 1.03 1.02 1.02 1.02 1.02 1 1 0.99 0.99 0.98 1 1.02 1.02 1.07 1.07 1.09 1.09 0.87 0.88 0.84 0.83 0.81 0.83 0.81 0.82 0.82 0.81 0.85 0.84 3.04 3.03 3.08 3.07 81.91 81.91 3.8 3.81 3.71 3.91 4.9 4.91 4.45 4.46 4.39 4.39 4.34 4.34 4.38 4.38 4.38 4.38 4.46 4.45 6.3 6.31 2.96 2.96 2.88 2.89 6.35 6.35 2.84 2.84 2.77 2.77 2.76 2.76 2.82 2.81 3 3 3.01 3.01 5.5 5.5 5.5 5.5 5.5 5.5 5.4 5.4 5.43 5.42 5.44 5.44 4.47 4.5 4.45 4.45 3.44 3.44 3.26 3.27 3.04 3.05 2.87 2.87 2.71 2.71 2.56 2.57 2.38 2.39 2.27 2.27 1.89 1.91 1.8 1.8 1.79 1.8 1.68 1.68 1.65 1.65 1.52 1.53 1.48 1.49 1.44 1.44 1.34 1.34 1.31 1.31 1.21 1.22 1.15 1.15 1.11 1.11 1.07 1.07 1.11 1.1 1.09 1.09 1.07 1.08 1.05 1.04 1.02 1.02 1 1.01 1.02 1.02 1.05 1.05 1.37 1.37 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 81.91 1.39 1.39 1.35 1.36 1.32 1.33 1.32 1.32 1.36 1.36 1.39 1.39 1.36 1.37 1.35 1.35 1.33 1.33 1.31 1.31 1.3 1.3 1.3 1.3 1.34 1.34 1.39 1.38 1.44 1.44 1.5 1.49 6.2 6.2 6.01 6.08 5.68 5.68 3.36 3.35 3.29 3.29 3.28 3.29 2.92 2.92 2.9 2.9 3.24 3.22 3.47 3.47 3.76 3.75 4.13 4.12 4.26 4.26 5.2 5.2 5.21 5.21 5.66 5.65 5.69 5.69 8.5 5.92 9.67 9.67 10.15 10.15 11.8 -19.7521 0.894171 -1.71437 -19.7521 0.894171 -1.71437 241.585 pippo 241.585
ODOM -19.7521 0.894171 -1.71437 0 0 0 0 pippo 0
NEFF 5.93671

Fig A.1: Laser scanner dataset