IMPLEMENTING SLAM FOR PRACTICAL SCENARIOS

LIM ZHI JIAN

UNIVERSITI TUNKU ABDUL RAHMAN

IMPLEMENTING SLAM FOR PRACTICAL SCENARIOS

LIM ZHI JIAN

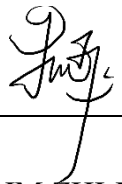
A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : LIM ZHI JIAN

ID No. : 17UEB01932

Date : 22 APRIL 2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"IMPLEMENTING SLAM FOR PRACTICAL SCENARIOS"** was prepared by **LIM ZHI JIAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr Ng Oon-Ee

Date : 25 April 2022

Signature : 

Co-Supervisor : Ir. Dr Danny Ng Wee Kiat

Date : 25/4/22

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisors, Dr Ng Oon-Ee and Ir. Dr Danny Ng Wee Kiat for their invaluable advice, guidance, and their enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my academic advisor, Dr Hau Lee Cheun, as well as my loving parents and siblings who had helped and encouraged me during the conduct of this research.

ABSTRACT

Mobile robots, especially service robots nowadays are equipped with LiDAR or RGBD cameras for 2D Simultaneous Localisation and Mapping (SLAM) and navigation purposes. However, the state-of-the-art 2D SLAM packages that are available in Robot Operating System (ROS) are prone to environmental factors, such as the presence of noise, the presence of repetitive structures, and the lack of features in an environment. The types of sensors used for mapping would affect the scan matching and loop closure abilities of the 2D SLAM packages. Therefore, this project aims to provide an in-depth understanding of the capabilities, performances, and limitations of the 2D SLAM packages, so that more insights could be provided for successful SLAM implementations. In this project, mapping procedures will be carried out on a service robot in the different scenarios of venues, sensors, and 2D SLAM packages, and the results will be compared for further evaluation. From the quality of mapping, this project would provide insights into the choice of SLAM package, the tuning of SLAM parameters, and the choice of different sensors, based on the nature of the surroundings, to obtain the best configuration that results in the best mapping quality.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

CAD	Computer-Aided Design
DOF	Dimension of Freedom
EKF	Extended Kalman Filter
FLIRT	Fast Laser Interest Region Transform
FOV	Field of view
GLC	Generic Linear Constraint
IMU	Inertial Measurement Unit
KF	Kalman Filter
LTS	Long Term Support
ORB	Oriented FAST and rotated BRIEF
RANSAC	Random Sample Consensus
RBPF	Rao-Blackwellised particle filter
ROS	Robot Operating System
SDF	Simulation Description Format
SLAM	Simultaneous Localisation and Mapping
SPA	Sparse Pose Adjustment
URDF	Unified Robotic Description Format
VO	Visual Odometry

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Robots are machines that could perceive the environment and make decisions to manipulate the physical world. Over the years, industrial robots have been successful in various domains, such as the manufacturing sectors and medical sectors, as they are being programmed to move within a controlled environment, replacing humans in completing dangerous, dirty, and dull jobs. Mobile robots, on the other hand, have been implemented in different areas such as indoor service robots, autonomous flight vehicles and robots for sea exploration. These applications are more challenging, as the robots are required to move around an area without any prior knowledge.

To deploy robots in various fields, an open-sourced framework, Robot Operating System (ROS), is applied to provide the necessary tools, libraries, and packages that could suit the software developments. As the scope of robotics domains is expanding continually, ROS has been developed to handle the software complexity, by offering communications between different processes and algorithms (Quigley, et al., 2009). ROS also allows different researchers to collaborate by compiling their codes into packages which further facilitate the development of robotics programmes.

Differs from industrial robots, a mobile robot is required to map the unknown environment while keeping track of its location, which brings up the Simultaneous Localisation and Mapping (SLAM) problem (Durrant-Whyte & Bailey, 2006). Currently, there are three main paradigms to solve the SLAM problem: Kalman filter-based, particle filter-based, and optimisation-based methods, which will be further discussed in Chapter 2. Although there are a wide variety of ROS packages for easy implementation of SLAM, however, they are prone to dynamic changes in the environment, which may cause the systems to fail. To understand the underlying problems of these conditions, the limitations of the state-of-the-art SLAM algorithms must be well-studied, so that better solutions could be developed for the successful implementation of SLAM in the real-world situations.

1.2 Importance of the Study

This study may provide insights towards the architecture and theory behind different 2D SLAM approaches, the features and performance of 2D SLAM packages available in ROS, the limitations of the state-of-the-art SLAM algorithms in real-world situations, as well as the knowledge of the current trend of research for improving the capabilities of SLAM algorithms.

1.3 Problem Statement

With the availability of various ROS packages that are based on different SLAM approaches, it is not easy to select the most suitable package for a specific mobile robotic application. An in-depth understanding of the features and working principles of some main SLAM packages are therefore necessary to aid in the decision making.

Even though current SLAM algorithms have implemented probabilistic approaches that deal with uncertainties, however, the functionality of the algorithm may be affected if the surroundings are highly dynamic. Therefore, the limitations of SLAM in practical scenarios must be identified, so that a better solution could be proposed to improve the capabilities of SLAM algorithms in real-world situations.

1.4 Aim and Objectives

This study aimed to implement SLAM for a mobile robot to map an environment. The specific objectives of this research were to:

i) Review and evaluate the capabilities of current SLAM methods,
ii) Implement SLAM for a mobile robot in different scenarios, and
iii) Identify the limitations that affect SLAM usage in practical conditions.

1.5 Scope and Limitation of the Study

The scope of this project focus on the software components, including the implementation of ROS and 2D SLAM methods, as well as the implementation of LiDAR and Realsense D435 Camera. In this study, only four state-of-the-art 2D SLAM methods available in ROS will be implemented and evaluated, including the in-depth review of three main paradigms of SLAM solutions, their comparisons, and the identification of the limitations of

SLAM. A few possible improvements to the 2D SLAM algorithm to overcome their limitations in the real-world application will be mentioned in this study, however, the details will not be included as they will be another area of research, which may require knowledge in computer vision.

1.6 Contribution of the Study

This project reviews the existing 2D SLAM methods and provides insights to ease the future implementation of these methods. Each 2D SLAM package is tuned and evaluated based on its performance and quality of mapping. The 2D SLAM packages will then be compared to identify their capabilities and limitations.

1.7 Outline of the Report

This report is divided into five chapters. Chapter 1 provides the general introduction to robotics and the SLAM problem, as well as the problem statement and the aim and objectives. Next, Chapter 2 provides the literature review of the three main paradigms of SLAM solutions, the comparison of a few 2D SLAM packages, and the identification of the limitations of 2D SLAM in practical situations. In Chapter 3, the methodology for the implementation of different 2D SLAM packages, the tuning the parameters, and the evaluation of the mapping results are provided. Then, Chapter 4 provides the evaluation of results and discussion for sensors and SLAM packages in different scenarios. Finally, Chapter 5 concludes the project and provides some recommendations for future project development.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction to the SLAM Problem

The SLAM problem is the problem of mapping the environment while determining the robot's pose, given the robot's observations of the environment and the controls sent to the robot. To formulate the SLAM problem, consider a robot that is traversing through the world, as shown in Figure 2.1 below, which is observing the landmarks through the sensor mounted on its body.

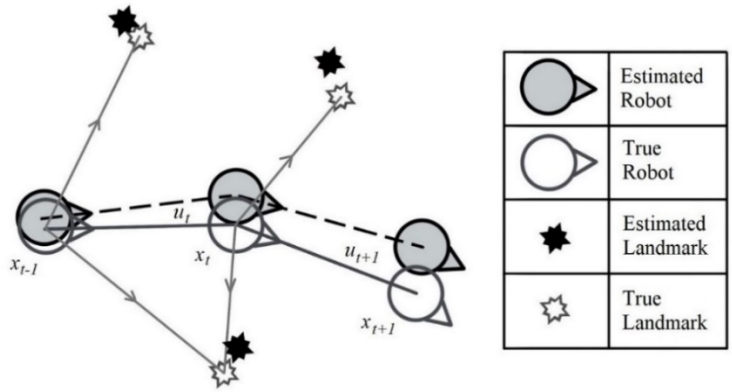


Figure 2.1: True and Estimated Locations of Robot and Landmarks.

Since the sensors are prone to cumulative errors and the motion of the robot increases the uncertainty of the system (Thrun, Burgard and Fox, 2005), a probabilistic approach is used to formulate the SLAM problem:

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T}) \tag{2.1}$$

where

$x_{0:T}$ = poses of the robot from period *0* to *T*

m = locations of all the landmarks $\{m_1, m_2, \ldots m_n\}$

$z_{1:T}$ = observations on the landmarks from period 1 to *T*

$u_{1:T}$ = control inputs to drive the robot from period 1 to *T*

The full SLAM formulation in (2.1) addresses the probability of a robot pose and the locations of the landmarks, given all the observations and all the input controls, where the entire path taken by the robot is estimated. In an online SLAM formulation, the previous poses of the robot are marginalised out, thus the probability distribution to describe the online SLAM is:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \tag{2.2}$$

There are a variety of solutions to the SLAM problems formulated above, which could be classified into filter-based approaches such as Kalman filters and particle filters or optimisation-based approaches. In this chapter, Section 2.2 reviews the general solutions to the SLAM problem, as well as their capabilities and limitations. In Section 2.3, the comparisons between a few 2D SLAM packages will be discussed. In Section 2.4, the limitations of SLAM in practical implementations will be studied, and a brief review of possible improvements will be provided.

2.2 Solutions to the SLAM Problem

2.2.1 Overview of the Filtering-Based Approach

Through the implementations of Bayes' rule and Markov assumption, the solution to the online SLAM problem in (2.2) could be achieved by estimating the state *x* through a two-step Bayes filter, which are the prediction step (2.3) and the correction step (2.4):

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1} \tag{2.3}$$

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t) \tag{2.4}$$

The prediction step considers the executed command u_t that moves the robot from its original state x_{t-1} to the predicted current belief of the robot pose x_t, whereas the correction step considers the sensor observations z_t to update the current belief of the pose of the robot x_t. The realisation of different recursive Bayes filters could be achieved using the Kalman filter and particle filter, which will be discussed in the next two subsections.

2.2.2 Kalman Filter-Based Approach

2.2.2.1 Extended Kalman Filter

The Extended Kalman Filter (EKF) is developed based on the recursive Bayes filters, replacing Kalman Filter (KF) which only assumes the linear motion model and linear observation model (Thrun, Burgard and Fox, 2005). The ability of EKF to describe these two models as non-linear functions:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \tag{2.5}$$

$$z_t = h(x_t) + \delta_t \tag{2.6}$$

allows it to be applied in non-Gaussian situations. In the prediction step, local linearisation is applied through the First Order Taylor expansion to estimate the robot's current state:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1}) \tag{2.7}$$

The correction step then updates the estimated robot's state based on the current sensory data in which the result is known as the posterior:

$$h(x_t) \approx h(\bar{u}_t) + \frac{\partial h(\bar{u}_t)}{\partial x_t}(x_t - \bar{\mu}_t) \tag{2.8}$$

2.2.2.2 EKF-SLAM

The application of EKF to SLAM estimates the state, μ and the covariance matrix Σ. The state represents the robot's pose, *x,* and the locations of the landmarks in an environment, *m*. The covariance matrix, on the other hand, updates the correlations between each pose and landmark.

$$\mu = \begin{pmatrix} x \\ m \end{pmatrix} \tag{2.9}$$

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix} \tag{2.10}$$

In EKF-SLAM, the prediction step predicts the new state μ and the covariance matrix Σ. Then, the correction step updates both terms by considering the uncertainties that are present in the sensors.

2.2.2.3 The Capabilities and Limitations of EKF-SLAM

Although EKF can maintain the Gaussian assumptions to solve the SLAM problem in non-linear situations, when compared to other SLAM solutions, it is relatively less robust in handling conditions where the non-linearity is large, especially in outdoor SLAM implementations. This is because the error in maintaining the Gaussian assumption made by EKF in its prediction step grows larger with the greater non-linear conditions, which is due to the increasing uncertainty. This, in turn, will cause the EKF solutions to diverge.

In EKF-SLAM, when the robot moves through an environment, the covariance matrix will be constantly updated, making the landmarks to be more correlated, thus allowing the algorithm to obtain a much more accurate relative map of the environment. However, due to this nature, the EKF-SLAM is not applicable for large-scale applications. This is because all the correlated robot's pose and landmarks saved in the huge covariance matrix need to be updated whenever another new observation is obtained, causing a quadratic increase in memory consumption.

2.2.3 Particle Filter-Based Approach

2.2.3.1 Particle Filter

The particle filter is a type of recursive Bayes filter that utilises a non-parametric approach. Differing from the KF which only models Gaussian distributions, the particle filters could deal with arbitrary distributions by representing the posterior probability with multiple weighted samples.

The particle filter algorithm could be summarised into three steps. First, since an arbitrary distribution is hard to model, a set of particles is sampled from a proposal distribution representing a set of state hypotheses. Second, the samples are individually weighted using the importance sampling principle, taking care of the differences between the target (arbitrary) distribution and the proposal distribution, obtaining a set of weighted particles as below:

$$weighted\ samples: \mathcal{X} = \left\{\langle x^{[j]}, w^{[j]}\rangle\right\}_{j=1,\dots,J} \tag{2.11}$$

Third, in the resampling step, the samples with lower weights are replaced by samples with high likeliness to reduce the representation of unlikely states, using the Roulette wheel or stochastic universal sampling methods.

The particle filter, however, only works well for low-dimensional applications. This is because a huge number of samples are required for it to cover a high dimensional region, making it to be computationally inefficient.

2.2.3.2 Rao-Blackwellisation

The implementation of the particle filter in SLAM directly is infeasible because of the high dimensionality problem:

$$x = \left(x_{1:t}, m_{1,x}, m_{1,y}, \dots, m_{M,x}, m_{M,y}\right)^T \tag{2.12}$$

As the number of landmarks becomes higher, it would be computationally inefficient to implement the particle filter in SLAM. To solve the problem, Rao-Blackwellisation is applied to reduce the sample space, factorising the SLAM posterior into a path posterior and a map posterior:

$$p(x_{0:t}, m_{1:M}|z_{1:t}, u_{1:t}) = p(x_{0:t}|z_{1:t}, u_{1:t})\, p(m_{1:M}|x_{0:t}, z_{1:t}) \tag{2.13}$$

The factorisation allows the particle filter to only represent the path posterior using weighted particles, which has lower dimension. The map posterior will then be analytically calculated from the set of particles.

Given the trajectory of the robot is known and each observation made is independent, the landmarks will also be independent of each other. Therefore, the map posterior for each particle in (2.13) could be represented as independent Gaussian distributions:

$$p(m_{1:M}|x_{0:t}, z_{1:t}) = \prod_{i=1}^{M} p(m_i|x_{0:t}, z_{1:t}) \tag{2.14}$$

This shows that the map posterior could be represented by multiple 2-dimensional EKFs, which are less complex than a single high-dimensional EKF used in EKF-SLAM. Therefore, a recursive estimation could be performed effectively by 1) calculating the path posterior using particle filtering, and then 2) calculating the map posterior from the weighted particles using the 2-dimensional EKFs.

Equation (2.14) is the key property of Rao-Blackwellised particle filter-based SLAM because all the landmarks could be represented independently using Gaussian distributions, reducing the complexity into linear form, as a contrast to the quadratic complexity in EKF-SLAM which uses a covariance matrix to represent all the landmarks.

2.2.3.3 The Capabilities and Limitations of FastSLAM

Proposed by Montemerlo, et al. (2002), the FastSLAM algorithm uses the Rao-Blackwellised particle filter (RBPF) method to model the posterior. This is achieved by sampling and computing the weighted particles, followed by the update of belief as in EKF. The implementation of FastSLAM is greatly influenced by the sample size and the number of particles sampled, which will impact the mapping accuracy and the computational complexity.

In conditions where the number of landmarks is high (big sample size), the FastSLAM algorithm has lower computational complexity as compared to EKF-SLAM. This is because the particles in FastSLAM represent each landmark with multiple low-dimensional EKFs, in contrast to the EKF-SLAM which uses a single high-dimensional covariance matrix. On the other hand, in cases where the sample size is small, the FastSLAM is less preferable than EKF-SLAM because the decreased sample size reduces the number of particles that can be sampled, thus reducing its accuracy.

Even though FastSLAM is more efficient than EKF-SLAM in high dimensional spaces, if the number of particles sampled by the RBPF is high, a high computational effort is still required. Besides, the RBPF also suffers from the particle-depletion problem, due to the implementation of the resampling step (Grisetti, Stachniss and Burgard, 2007). These two issues could be minimised while implementing the GMapping package, which will be further discussed in Section 2.3.

2.2.4 Optimisation-Based Approach

2.2.4.1 Pose-graph

In the graph-based SLAM, a graph consisting of nodes connected by edges is constructed to represent the SLAM problem, where the nodes represent the full trajectory of robot poses or landmarks, and the edges represent the spatial constraints between nodes, as illustrated in Figure 2.2. These edges are either created from the odometry data between sequential robot poses, or through the alignment of observations of the same environment, i.e., during loop closure.

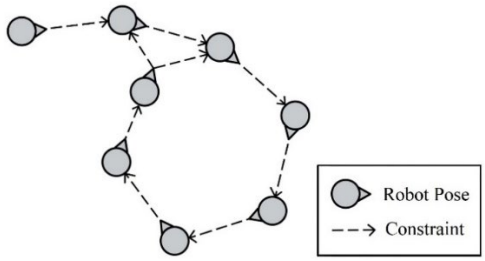


Figure 2.2: Pose Graph Example.

Due to the noise in the sensor, there are uncertainties in the created constraints. Therefore, upon building the graph, the graph-based SLAM finds the nodes' configurations that best fit the constraints (Grisetti, et al., 2010) thus optimising the graph and the map.

2.2.4.2 Graph-based SLAM

The graph-based SLAM is implemented in two parts. The front end builds the graph using the raw sensor measurements (graph construction), creating the nodes and constraints through data associations. Using the given edges, the back end maximises the consistency between the node's configurations and the measurements (graph optimisation), through the minimisation of errors between the predicted and real observations.

Figure 2.3 below shows the expected and the real measurement of pose x_j as seen from pose x_i, represented in homogeneous coordinates. Let the vector of parameters representing the pose of nodes in Figure 2.2 above be: $x = (x_1, \dots, x_T)^T$, when the robot is at position x_i observing a previously seen environment when it was at position x_j, a virtual measurement z_{ij} is created about the position of x_j as seen from position x_i.

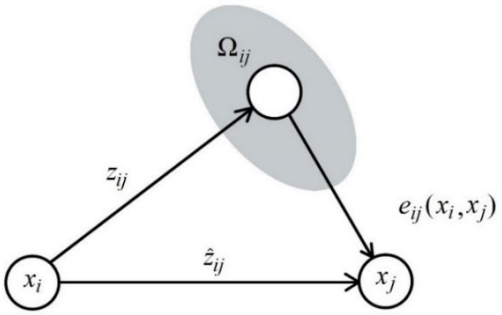


Figure 2.3: Expected Pose and Real Pose of x_j.

The predicted virtual measurement, $\hat{z}_{ij}(x_i, x_j)$ is a transformation that predicts the virtual measurement given the configurations of the nodes x_i and x_j respectively, whereas the error function $e_{ij}(x_i, x_j)$ is a function that calculates the errors between the predicted observation $\hat{z}_{ij}$ and the real observation z_{ij}. The optimisation of the graph is performed when the least-squares approach is applied to find the new state x* that best represents the optimised graph:

$$x^* = \underset{x}{\operatorname{argmin}} \sum_k e_k^T(x)\Omega_k e_k(x) \tag{2.15}$$

where

$e_k(x)$ = error function $e_{ij}(x_i, x_j)$

Ω_k = information matrix representing the uncertainties of the constraints

The equation (2.15) could be solved iteratively using the Gauss-Newton algorithm to find the state x* which represents the most likely nodes configurations that have the minimum squared error (Grisetti, et al., 2010).

2.2.4.3 The Capabilities and Limitations of Graph-Based SLAM

Since the error term $e_{ij}(x)$ depends only on the variables x_i and x_j, the resulting Jacobian in the linearised error function is a sparse matrix. This sparse matrix structure allows the graph-based SLAM to solve a huge linear system efficiently, making it suitable to map a large area efficiently.

However, according to Takleh, et al. (2018), since the graph-based SLAM must consider all the poses and landmarks information, it consumes a high computational cost, making it expensive to be implemented in a large

area. This issue, however, is not critical, because the graph-based SLAM, according to Santos, Portugal and Rocha (2013), is usually more efficient as compared to other approaches in mapping a large area.

2.2.5 Summary - SLAM Approaches

In summary, there are three main paradigms for solving the SLAM problem, which are the Kalman-filtering methods, particle-filtering methods, and graph-based-optimisation methods. The comparison of strengths and weaknesses of each approach could be summarised in Table 2.1 below.

Table 2.1: The Comparisons of Different SLAM Approaches.

Approaches	Strengths	Weaknesses
EKF-SLAM	Deal with moderate non-linearities.	Diverge if non-linearity is large.
	Accurate relative map.	Computationally difficult for large maps.
FastSLAM (RBPF)	Deal with the arbitrary distribution.	Particle depletion issues.
	Less memory is required.	Computationally inefficient in high-dimension applications.
	More computationally efficient than EKF in large maps.	Much more inaccurate than EKF-SLAM in the case of the small sample size.
Graph-based SLAM	Ability to process large maps.	High computational cost in large maps (but still much more efficient than other approaches).

2.3 2D SLAM Packages

Although most of the 2D SLAM packages are based on the three general SLAM approaches reviewed in the previous section, the number of SLAM packages can be overwhelming. In this section, a few ROS SLAM packages are discussed and compared, to ease the choice of SLAM package for a specific robotics application.

2.3.1 Hector SLAM

The Hector SLAM is a package that fuses the 2D SLAM and 3D navigation system, which are based on the LiDAR scan data and the Inertial Measurement Unit (IMU) data respectively (Kohlbrecher, et al, 2011). The high update rate of LiDAR sensors is used in this system to achieve the real-time estimation of the robot's movement. Since the Hector SLAM does not utilise odometric information, it can be implemented in 6 DOF applications such as aerial robots.

The Hector SLAM is implemented in two stages: Fast scan matching for 2D pose estimation in the front end; and slow 3D state estimation in the back end. In the front end, the 2D pose estimation is obtained using the scan matching of the beam endpoints within a range of z-coordinates, which could be solved by using the Gauss-Newton equation. In the back end, the 2D pose is updated to 3D state estimation using the navigation filter based on EKF, by incorporating the inertial measurements available in aerial robots.

However, according to Santos, Portugal and Rocha (2013), the Hector SLAM might exhibit problems when the scan rates are low, as it depends on the high update rate of LiDAR sensors to perform the real-time estimation. This is because the Hector SLAM could not make use of the odometry information in wheeled robots even though they are accurate.

2.3.2 GMapping

GMapping is a SLAM package that is based on the improvised RBPF approach proposed by Grisetti, Stachniss and Burgard (2007). This approach solved the particle-depletion issue and high computational complexity issue in conditions where the number of particles sampled is high.

To solve the particle-depletion issue, GMapping deploys an adaptive resampling technique, which allows the algorithm to do resampling only when

necessary, keeping the diversity while minimising the depletion problem. On the other hand, to solve the computational complexity issue, the GMapping package uses a proposal distribution that instead of considering only odometry information, it fuses that information with the current sensory observations through the scan matching procedure. This increases the mapping accuracy, decreases the estimation error, and therefore, requires fewer particles to represent the posterior, which in turn lowers the computational effort.

2.3.3 Cartographer

The Cartographer is a package that is based on the graph-based SLAM approach, proposed by Hess, et al (2016). The overview of the Cartographer system is shown in Figure 2.4 below.

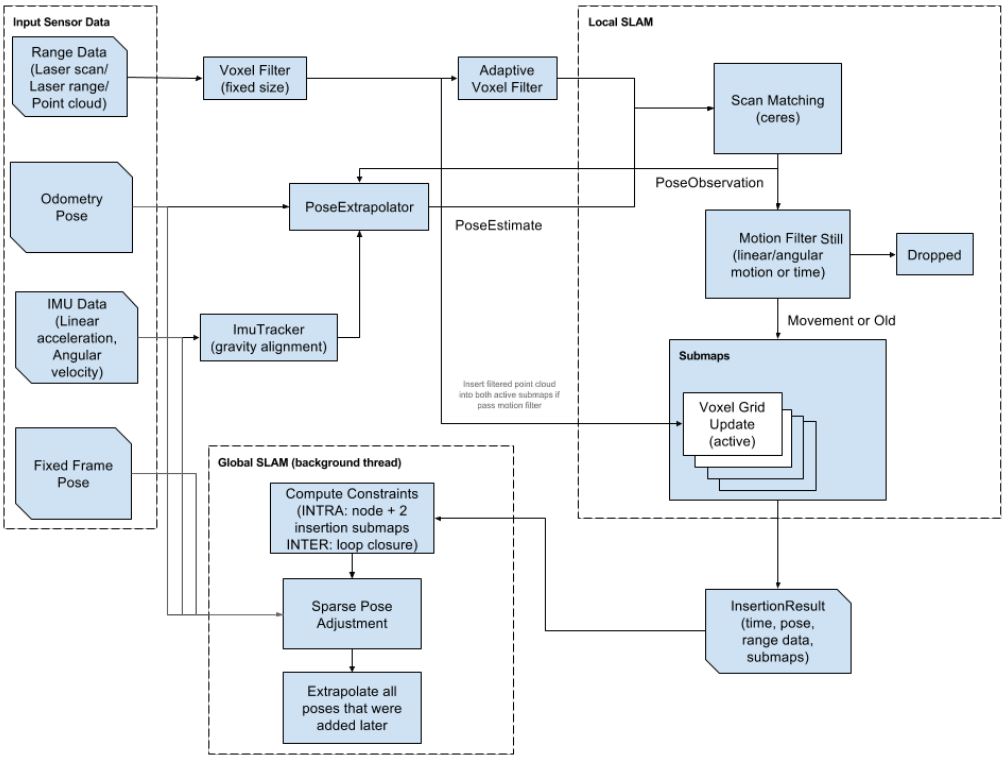


Figure 2.4: Overview of Cartographer (Cartographer ROS, 2019).

In Cartographer, the extracted sensory data is first downsampled by the voxel filter to decrease the computational resource required. Then, the pose estimator ensures that each of the scan data is matched with odometry data before they are fed into the Local SLAM for scan matching.

In the Local SLAM, the current pose is estimated by matching the scan data with the sub-maps, using the Google Ceres scan matcher which is based on a non-linear optimisation method. These scans are then placed into the current submap; however, they contain an accumulation of drifting errors, which will be minimised in the Global SLAM through pose optimisation. After that, the Global SLAM creates intra sub-maps constraints, which are obtained from the scan matching between collected scans with the completed sub-maps. As a result, all sub-maps could be linked to obtain an overall map.

2.3.4 SLAM Toolbox

The SLAM Toolbox is a ROS SLAM package introduced by Macenski in ROSCon 2019. It is built upon the Open Karto (Konolige et al., 2010), which is a pose-graph optimisation-based SLAM, thus it can handle the robot's resources effectively while mapping a large environment. According to Luknanto (2020), the SLAM Toolbox uses local and global approaches, similar to Cartographer. In the local approach, consistent pose estimation is performed to match the new scan with a few recent scans. In the global approach, the most recent scan is matched against the map to perform loop closure.

According to Macenski and Jambrecic (2021), the SLAM Toolbox can store (serialise) and retrieve (deserialise) the raw pose-graph data, therefore allowing the users to modify them or to assist a loop closure. The users are also allowed to choose the different modes of operations, which are: the synchronous mode which focuses on the quality of mapping, or the asynchronous mode which focuses on the quality of real-time localisation. Besides, since the Sparse Pose Adjustment (SPA) in KartoSLAM is replaced with Google Ceres for scan matching and loop closures procedures in SLAM Toolbox, much adaptable optimisation settings could be provided.

2.3.5 Summary - ROS SLAM Packages

In summary, the SLAM packages reviewed in this section are based on different approaches. The strengths and weaknesses of these packages are generally affected by the underlying approaches used. A simple comparison between the ROS SLAM packages reviewed in this section could be

summarised in Table 2.2 below, however, this table does not summarise the real performances of these packages, as they will be reviewed again later in this study. Included in the table are also some of the weaknesses of the SLAM packages pointed out by Macenski and Jambrecic (2021).

Table 2.2: The Comparisons of Different ROS SLAM Packages.

Package	Strengths	Weaknesses
Hector SLAM (EKF for 3D estimation)	Real-time estimation.	Inaccurate pose and map estimation at low scan frequency.
	Applicable for 6 DOF applications.	Do not utilise odometry information.
GMapping (RBPF)	An improvised RBPF method.	Loop closure problem in large space.
Cartographer (Optimisation-based SLAM)	Can build an accurate map in real-time.	Challenging to modify the complex software.
	Provide data serialisation.	
SLAM Toolbox (Optimisation-based SLAM)	Can build an accurate map in real-time	Newly developed package.
	Provide multiple modes of mapping.	
	Provide data serialisation.	

2.4 Factors Affecting SLAM in Practical Scenarios

The current SLAM methods have generally solved the SLAM problem and have been successfully applied using LiDAR-based SLAM such as LOAM, and Visual SLAM such as ORB-SLAM. However, these approaches assume a static environment, which is not the case for most real-life applications, such as search-and-rescue operations, sea exploration, and lunar exploration that require the robot to function long-termly in a dynamic and unstructured environment.

The implementation of the state-of-the-art SLAM in practical situations is prone to 1) the limitation of the current SLAM algorithm in handling the dynamic environments, 2) the sensor's limitations and inaccuracy in observing harsh environments, as well as 3) the actuator's imprecision caused by the actuator degradation and the influences of unstructured environments. These limitations will be discussed in the next two subsections.

2.4.1 Limitations of current SLAM Algorithms

As the SLAM applications are transitioning into the large-scale environment, much of the recent research is focusing on graph-based SLAM because of its efficiency in mapping a large area. The summary of a typical SLAM system, which consists of the front end and back end, is shown in Figure 2.5 below.

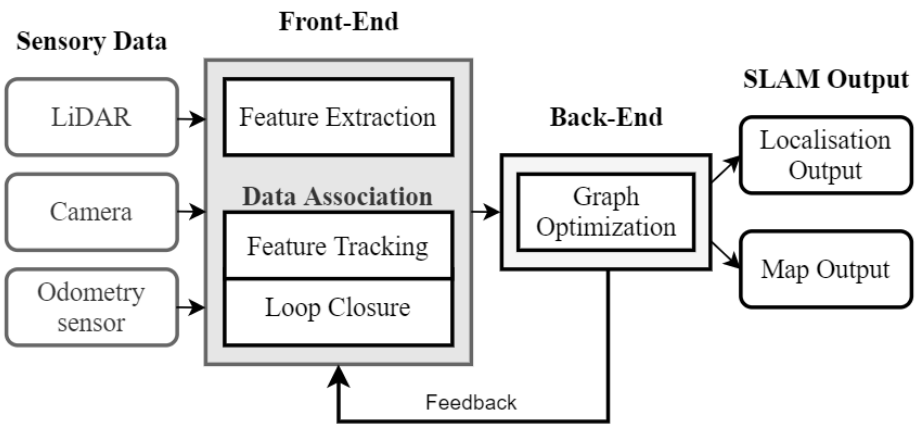


Figure 2.5: Typical SLAM System.

The front end uses the sensory data to represent the environment into models applicable for estimations, while the back end processes the data to provide the localisation and mapping as output. In this subsection, the robustness and the scalability of the SLAM algorithm will be reviewed, which are: 1) how the SLAM front end (feature extraction and data association) could be influenced by a high dynamics environment, and 2) how SLAM back end (graph optimisation process) could be influenced by the increasing complexity.

2.4.1.1 Limitations of Feature Extraction

The feature extraction algorithm in the front end provides the data for the feature matching and pose estimation of the SLAM algorithm. These features are represented by planes, lines, or points, which could work well in static environments. In real-world conditions, however, the presence of moving objects could reduce the reliability of the mapping process of Visual SLAM, as many erroneous features extracted from dynamic objects would lead to wrong data associations, incorrect camera ego-motion estimation and drifting of pose estimations (Liu and Miura, 2021).

With the advancements in deep learning and image recognition, many recent works have proposed semantic-based methods to detect the presence of dynamic objects, making Visual SLAM more robust to be deployed in real-world situations. Among them is the RDS-SLAM by Liu and Miura (2021) based on ORB-SLAM3 which uses semantic information to optimise camera pose, and SaD-SLAM by Yuan and Chen (2020) based on ORB-SLAM2 which detects static and dynamic feature points to improvise camera pose estimation.

2.4.1.2 Limitations of Data Association

The data association in the SLAM front end performs feature tracking and loop closure detection. The feature tracking algorithm associates the pixel measurements within two consecutive frames as the same point, whereas the loop closure detection associates a new observation to an old feature, minimising the reliability towards dead reckoning.

In a practical situation such as when the robot is traversing through a long corridor or a parking lot, since there are a lot of similar features, the data association algorithm is prone to the perceptual aliasing phenomenon (Cadena, et al., 2016), whereby the algorithm would wrongly perceive different features of the environment as the same. On the other hand, when the robot is required to function long-termly in a dynamic environment, the feature tracking and loop closure detection algorithm might fail to associate the new observation of a previously visited area as the same feature, due to the change in viewpoint, the difference in illumination of a scene, and the shifting of objects during the deployment period (Shi, et al., 2020).

According to Cadena, et al. (2016), the robustness of the feature tracking algorithm could be improved if the framerate of the sensor is significantly higher than the robot's dynamics, as the sensor's position does not vary much as the time increases from *t* to *t+1*. The loop closure, on the other hand, could potentially be detected using the FLIRT features jointly with Random sample consensus (RANSAC), proposed by Tipaldi and Arras (2010). Cadena, et al. (2016) also mentioned that loop closure quality could be made certain through loop closure validation, which could be achieved using RANSAC in vision-based applications, and scan matching process in laser-based applications.

2.4.1.3 Limitations of Graph Optimisation Process

For the SLAM back end that uses the graph-based optimisation method, as previously mentioned in Section 2.2.4, the computational complexity will increase with the increasing map, which is the case for outdoor practical applications that requires the robot to operate over a long time. As the pose-graph increase indefinitely with the increasing size of the explored map, the resource of the robot will no longer be able to support the execution of the system.

To reduce the complexity of the system, a sparsification method could be implemented by decreasing the addition of new nodes or by removing the current nodes that contain less information, such as through the node removal method using generic linear constraint (GLC) proposed by Carlevaris-Bianco (2015). Besides, the complexity issue could also be solved through the implementation of sub-mapping algorithms by dividing the computational load of the system into multiple processors, or the utilisation of multiple robots to map a large environment (Cadena, et al., 2016)

2.4.2 Limitations of Sensors and Actuators

Other factors that will impact the implementation of the SLAM system include the limitations of sensors and actuators in harsh environments. In a survey conducted by Wang, Zhang and An (2017), the presence of a large number of dust particles and the lack of feature points in the surroundings could reduce

the certainty of the information perceived, leading to a non-convergence problem that affects the robustness of SLAM.

Besides, in conditions where the robot is required to move through uneven terrain, the wheel odometry information becomes unreliable because the wheel could sink, skid, and slip. To tackle the problem, Marks, et al (2009) proposed the use of visual odometry (VO) in estimating the vehicle motion, replacing the wheel odometry which is shown unreliable through their experiment.

2.4.3 Summary - Limitations of SLAM

An understanding of the limitations of the state-of-the-art SLAM algorithms is important for the successful implementations of future robotics applications in an outdoor environment.

In summary, the SLAM front end could be influenced by the highly dynamic and unstructured environments, such as the incorrect feature extractions caused by the moving objects, the perceptual aliasing phenomenon in repetitive environments, and the failure in detecting loop closure due to changes in the environment during the long deployment period, which in turns impacts the robustness of the SLAM algorithm. While deploying the robot for an extended period on a large map, the complexity of the system could be increased unboundedly, which affects the scalability of the SLAM algorithms. On the other hand, the sensors and actuators of the robotic applications could be influenced by the harsh conditions of the environment, resulting in high uncertainties of perceived information. Nevertheless, successful implementations of SLAM would be eased by increasing the robustness and scalability, such as the use of deep learning and computer vision techniques.

2.5 Overall Summary

In summary, the three main paradigms to solve the SLAM problem are the Kalman filter-based, particle filter-based, and optimisation-based approaches. The KF and RBPF are developed based on Bayesian filters, whereas the graph-based SLAM uses the least-square method to optimise the SLAM estimation model. Using SLAM packages available in ROS such as Hector SLAM, GMapping, Cartographer, and SLAM Toolbox, these SLAM

algorithms could be easily adapted to different applications by tweaking the parameters offered in the packages.

However, as the mobile robotics applications had been transitioning into a wider and more dynamic environment, the robustness and scalability of the state-of-the-art SLAM are challenged. The failure of feature extraction algorithms and loop closure algorithms in a dynamic environment, the unbounded complexities of the SLAM system in large environments, and the performance issues of sensors and actuators in harsh environments are affecting the implementation of state-of-the-art SLAM. These issues are currently being attempted to be solved through the implementation of Computer Vision techniques, which is another area of research.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Overview of Project Work Plan

In this chapter, the methodology used to evaluate the capabilities and limitations of the state-of-the-art 2D SLAM during implementations will be explained. In general, a robot equipped with a LiDAR and an RGBD camera would be teleoperated in different real-world environments, where the data from these sensors were recorded and fed into different 2D SLAM packages for mapping. The tuning of parameters and the quality of mapping of each package will be further evaluated to compare their capabilities and limitations. The project flow chart is shown in Figure 3.1 below and will be discussed in detail in the next few subsections.

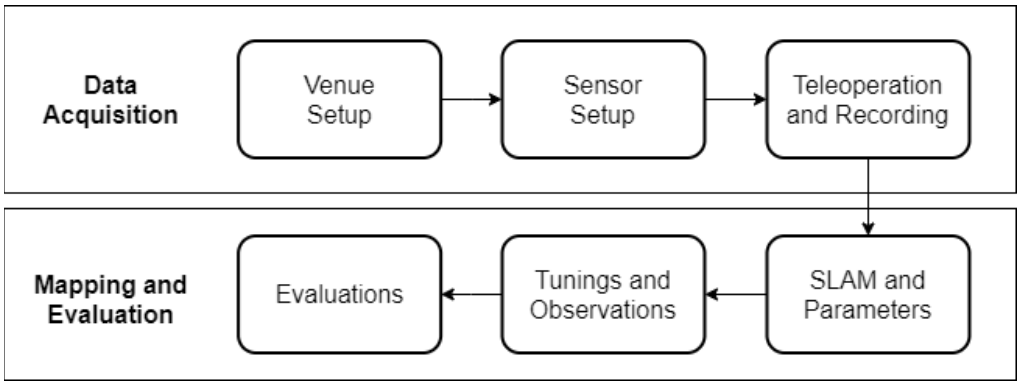


Figure 3.1: Project Flow Chart.

3.2 Hardware Details

The project was carried out using a differential drive robot, equipped with a LiDAR sensor, a depth camera, and a tracking camera. The robot was controlled using a computer with ROS Melodic installed. The specifications of the robot are listed in Table 3.1 below:

Table 3.1: Robot Specifications.

Sensors	Hokuyo 4-metre LiDAR (URG-04LX-UG01) Realsense RGBD Camera (D435) Realsense Tracking Camera (T265)
Operating System	Ubuntu 18.04 LTS (ROS Melodic installed)
CPU	Intel i7-8550U @ 1.80 GHz
RAM	12.0 GB
Graphics Card	NVIDIA GeForce MX150

3.3 Data Acquisition

3.3.1 Venue Setup

The 2D SLAM nodes were run based on the data collected from KB613 Lab and the 6th Floor Corridor, as shown in Figure 3.2 below.

(a) (b)

Figure 3.2: Venues for Mapping: (a) KB613 Lab, and (b) 6th Floor Corridor.

The KB613 Lab contained repetitive structures of tables in the middle area, and it contained cluttered scenes such as stools and treadmills. The 6th Floor Corridor was 27.88 metres long and it was featureless. The performance of 2D SLAM packages would be studied through their mapping results, based on these characteristics of the scenes, as summarised in Table 3.2 below.

Table 3.2: Characteristics of Different Venues

Venue	Lab	Corridor
Characteristics	Repetitive structures	Long (27.88 metres)
	Cluttered scenes	Featureless

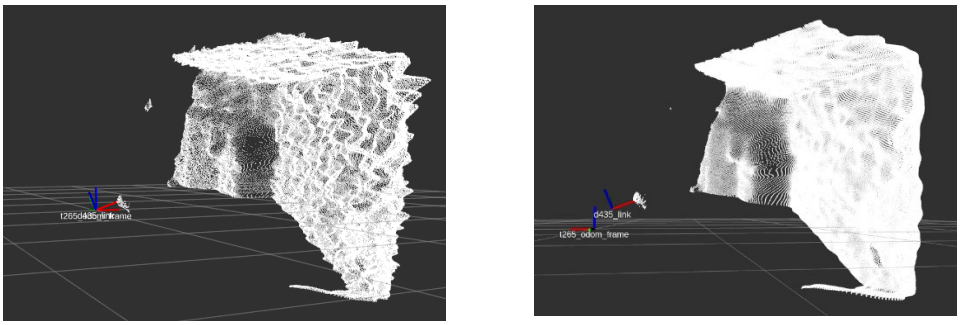
3.3.2 Sensor Setup

The input scan data for 2D SLAM were based on two sources: the scan data from Hokuyo LiDAR, and the scan data extracted from the depth image of the Realsense D435 camera, by applying the *depthimage_to_laserscan* ROS package. The odometry data was provided by the Realsense T265 Tracking Camera. The specifications of the sensors that provide the scan data were tabulated in Table 3.3 below.

Table 3.3: Specifications of Sensors.

	Hokuyo 4-metre LiDAR	D435 Realsense Camera
Field of View	240˚	87˚
Range	4 metres	10 metres
Accuracy	3%	Varies

The Hokuyo URG LiDAR had a scan area of 240˚ with a maximum distance of 4 metres and had 3% of guaranteed accuracy. The results were accurate thus no further processing was required. The Realsense D435 Camera, on the other hand, had a Field of View (FOV) of 87˚ with a maximum distance of 10 metres, but the accuracy was greatly affected by the environmental conditions such as lighting noise. The scan result on a flat surface was wavy, where its amplitude increased with an increasing distance. Therefore, the point cloud was filtered using a combination of spatial, temporal, disparity, and decimation filters. Referring to Figure 3.3 below, the filtered results of the point cloud provided a good representation of flat surfaces.



(a) (b)

Figure 3.3: Point Clouds: (a) before Filtering, and (b) after Filtering.

3.3.3 Teleoperation and Recording

Figure 3.4 and Figure 3.5 below show the simplified floor plan of KB613 Lab and 6th Floor Corridor, respectively. The robot was teleoperated in KB613 Lab with the trajectory of A, B, C, D, A, E, F, G, H, A, whereas it was teleoperated in the 6th Floor Corridor with the trajectory of A, B, A. All the *rostopics* were recorded into the *rosbag* files.

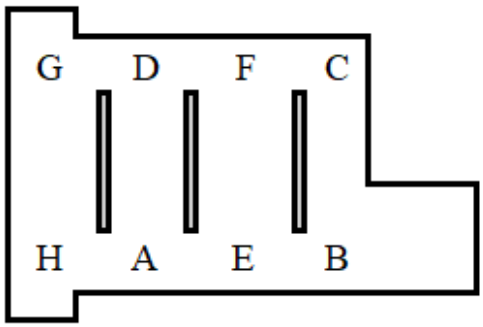


Figure 3.4: Simplified Floor Plan of KB613 Lab.

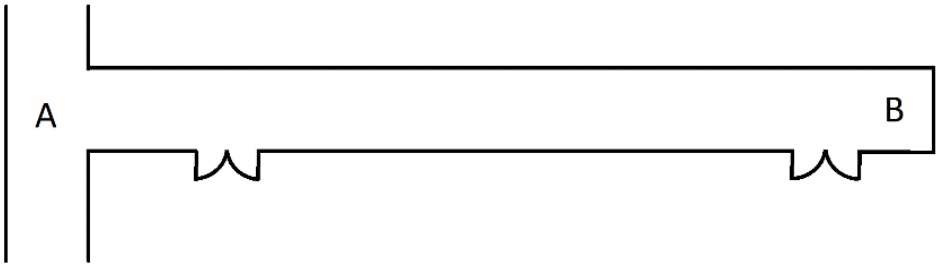


Figure 3.5: Simplified Floor Plan of 6th Floor Corridor.

3.4 Mapping and Evaluation

3.4.1 SLAM and Parameters

In this project, the four SLAM packages: Hector SLAM, GMapping, Cartographer, and SLAM Toolbox would be analysed for their performance, capabilities, and limitations, in the different scenarios described in the previous subsections. All the SLAM packages would be run based on the same *rosbag* file, but with different configurations of parameters specific to each package and scenario. The tuning aimed to obtain the best configuration that would result in the best mapping quality for evaluation.

In this section, the effect and process of tuning the SLAM packages will be discussed. The general parameters that were similar to all four packages were: 1-second map update interval, 4-metre-long maximum usable scan data for mapping, and 0.05 map resolution. To obtain the best mapping result, the tuning of important parameters that were specific to the GMapping, Cartographer, and SLAM Toolbox were listed and described in Table 3.4, Table 3.5, and Table 3.6 below. There were no specific parameters to be tuned for Hector SLAM.

Table 3.4: Specific Parameters for GMapping.

Specific Parameters	Description
particles	The number of particles for state estimation.
lstep, astep	Linear and angular optimization steps.

Table 3.5: Specific Parameters for Cartographer.

Specific Parameters	Description
submaps_num_range_data	Size of the submaps.
translation_weight, rotation_weight	Greater value means scan matching had to generate greater value for its result to be accepted.
optimize_every_n_nodes	Optimize the graph after several *n* batches of nodes were inserted.

Table 3.6: Specific Parameters for SLAM Toolbox.

Specific Parameters	Description
loop_search_space_dimension	Size of search grid for loop closure detection.
loop_match_minimum_chain_size	Minimum chain length required to detect loop closure.
minimum_travel_distance, minimum_travel_heading	Linear and angular update steps.
distance_variance_penalty, angle_variance_penalty	Penalty to apply as the matched scan differs from an odometry pose.

The parameters specific to each SLAM package had to be tuned according to different scenarios and conditions. The wrong settings of parameters would adversely affect the performance of SLAM. For instance, one of the most frequent errors was the perceptual aliasing phenomenon. Given an environment, a lower linear update step would allow the scan matching to be performed more frequently, however, this would also allow the SLAM to be more easily associated two consecutive scans as similar, thus causing a section of the map to appear shorter. Besides, a lower minimum number of nodes required for loop closure optimization allowed a frequent loop closure detection, however, this would also allow the SLAM to be more easily matched two different areas as similar, thus causing the map to be distorted as two different areas were merged.

3.4.2 Tunings and Observations

Based on the different sensors and venues, trials and errors methods would be performed for the tuning of SLAM parameters through visual inspection, whereby the quality of loop closure detection and scan matching were observed for tuning purposes. The best configuration, as tabulated in Table 3.7, Table 3.8, and Table 3.9 below, that results in the best mapping result would be used for evaluation and analysis of the SLAM packages in the next step.

Table 3.7: Best Specific Parameters for GMapping.

	Default	Lab		Corridor	
		LiDAR	D435	LiDAR	D435
lstep (m)	0.05	0.01	0.01	0.01	0.01
astep (m)	0.05	0.01	0.01	0.01	0.02
particles	30	100	100	100	100

Table 3.8: Best Specific Parameters for Cartographer.

	Default	Lab		Corridor	
		LiDAR	D435	LiDAR	D435
submaps_num_range_data	90	150	50	200	50
translation_weight	10	10	50	10	50
rotation_weight	1	1	10	1	10
optimize_every_n_nodes	90	20	20	20	20

Table 3.9: Best Specific Parameters for SLAM Toolbox.

	Default	Lab		Corridor	
		LiDAR	D435	LiDAR	D435
loop_search_space_dimension	8	15	8	20	20
loop_match_minimum_chain size	10	7	10	3	5
minimum_travel_distance	0.5	0.5	0.5	0.5	0.5
minimum_travel_heading	0.5	0.5	0.5	0.5	0.5
distance_variance_penalty	0.5	0.5	0.005	0.5	0.5
angle_variance_penalty	1.0	1.0	0.001	1.0	10

3.4.3 Evaluations

3.4.3.1 Evaluation of Lab Mapping Results

To evaluate the accuracy of the map at KB613 Lab, the ground truth and the resulting maps were binarized and then aligned using the Image Processing Toolbox in MATLAB. The evaluation was then carried out using the K-Nearest Based Normalised Error (k=1) metric, introduced by Santos, Portugal and Rocha (2013). This normalised error was calculated by averaging the sum of distances among each occupied cell on the resulting map, $K(k,l)$, and the corresponding cell on the ground-truth map, $I(i,j)$, where $K_{k,l} = knn_{i,j}^{k=1}(I,K)$.

$$NE = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sqrt{(k-1)^2+(l-j)^2}[I(i,j)=0]}{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)=0]} \quad (3.1)$$

The normalised error, NE in Equation (3.1) is always greater than or equal to 0, where 0 indicates that both maps of size $n \times m$ are the same. The sample output image was shown in Figure 3.6 below, where the green pixels indicated the resulting map, pink pixels indicated the ground truth map, and black pixels indicated the overlapping pixels of both maps. The value of NE was the value of normalized error between both maps after the registration process.

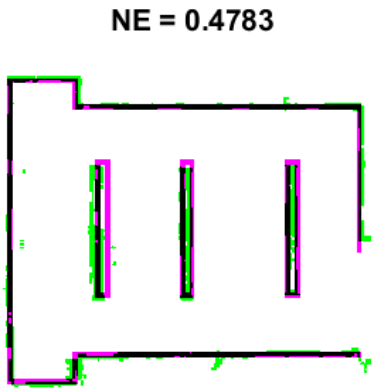


Figure 3.6: Result of Image Registration in MATLAB.

3.4.3.2 Evaluation of Corridor

To evaluate the resulting map of the 6th Floor Corridor, two aspects of the resulting map were observed and calculated: first, the angular deviation of the map of the corridor from its horizontal position, and second, the linear deviation of the map from the true length of the corridor.

To find the angular deviation, linear regression was performed to find the best fit line that suits the image data, using MATLAB. Then, the degree of deviation was calculated from its gradient, which would be in the clockwise or anticlockwise direction.

To find the linear deviation, first, the two points that were intersecting both the map and the best fit line were found by taking the mean of the points that had a low squared error value. Their distance was multiplied by 0.05(m/pixel) to obtain the length in metres. The sample output image was shown in Figure 3.7 below, where the black pixels indicated the corridor, the red line indicated the best fit line, and the blue and black dots each indicated the mean point that intersects the best fit line.

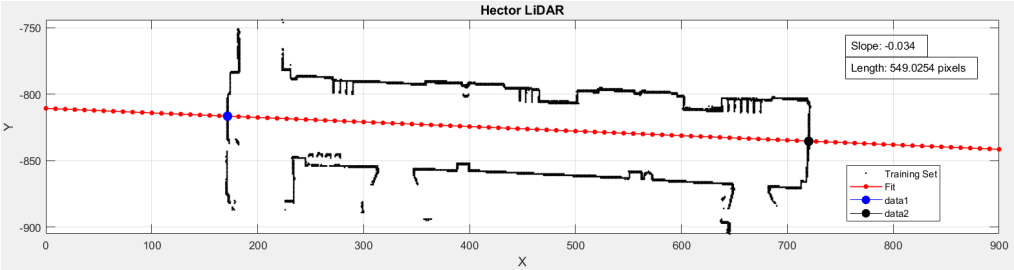


Figure 3.7: Result of Linear Regression in MATLAB.

3.5 Resource Allocations and Project Planning

The project was carried out using the service robot available in UTAR, and utilized packages available in ROS, therefore no extra cost or budget was required to carry out the project. All the planned tasks were achieved successfully for FYP Part 2 in this trimester.

3.6 Summary

In summary, the project could be carried out in two stages: the Data acquisition stage, and the mapping and evaluation stage. In the data acquisition stage, when the UTAR service robot was teleoperated in both KB613 Lab and 6th Floor Corridor, the scan data from the Hokuyo 4-metre LiDAR and the filtered scan data from Realsense D435 Camera were captured and recorded into *rosbag* files. In the mapping and evaluation stage, the *rosbag* files were replayed for four 2D SLAM methods: Hector SLAM, GMapping, Cartographer, and SLAM Toolbox. Then the output maps from each SLAM in each scenario were evaluated using image registration and K-Nearest Method for the map of Lab, and linear regression for the map of the Corridor.

CHAPTER 4

RESULTS AND DISCUSSION

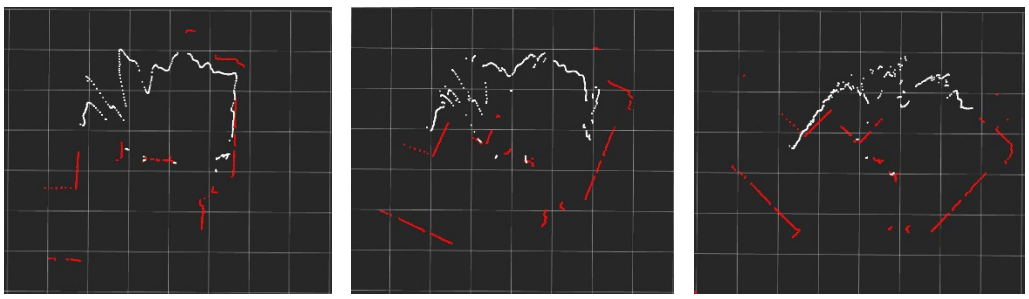
4.1 Introduction

In this chapter, the results obtained from the different sensors and SLAM packages will be compared and discussed. The first section evaluates the scan results obtained from LiDAR and D435 camera, as they influence the performance of SLAM. The second section evaluates the performance of different SLAM packages using different sensors and at different venues. Finally, the results will be compared to evaluate the capabilities and limitations of the SLAM packages.

4.2 Evaluations of Sensors

4.2.1 Change of Viewpoint

Figure 4.1 below shows the scan data from LiDAR (red-coloured) and D435 camera (white-coloured) as the robot rotated and changed its viewpoint towards the treadmills in Figure 4.2 located at the corner around point B in the KB613 Lab.



(a) (b) (c)

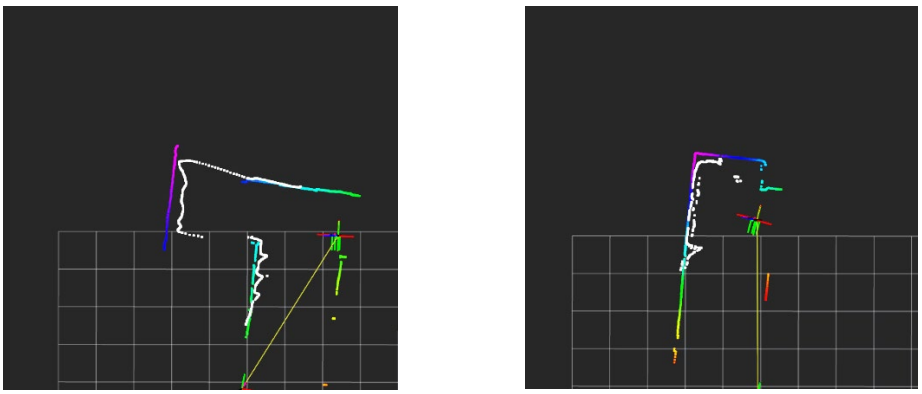
Figure 4.1: The Effect of Change of Viewpoint towards LiDAR Data and D435 Data as the Robot Rotated.

Figure 4.2: Treadmills at Point G.

The scan data from LiDAR remained constant as the robot moved, which was an accurate representation of the scene. On the other hand, the filtered scan data from the D435 camera contained fillets that joined the treadmills and their background together, where the scan data would be varied as the robot was rotated and changed its viewpoint. This effect was due to the noise and filtering effect on the point cloud data explained in Section 3.3.2. This effect was undesirable as the change in scan data would affect the scan matching capabilities of the SLAM packages.

4.2.2 Field of View (FOV)

Figure 4.3 shows the scan data from LiDAR (colourful) and D435 camera (white) when mapping the corner around Point G circled in Figure 4.4 below.



(a) (b)

Figure 4.3: Scan Data of the Corner Near Point G from: (a) D435 (white), and (b) LiDAR (colourful).

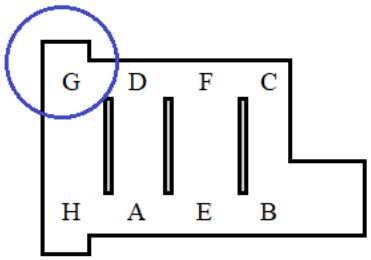
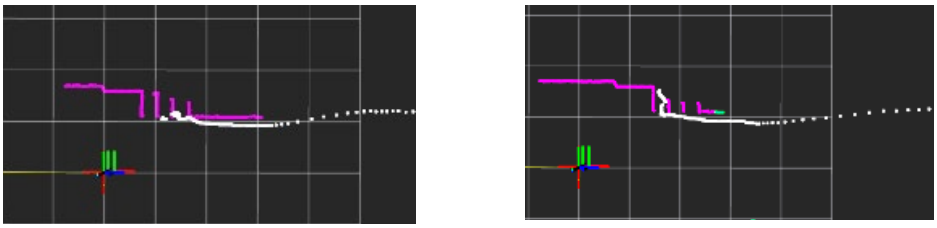


Figure 4.4: Map of the KB613 Lab.

The LiDAR had a wider FOV, thus it was able to scan the corner correctly (Figure 4.3(b)). On the other hand, the D435 camera had a smaller FOV, thus there was missing of information as it did not observe the corner. Additionally, the filtering of point cloud data input from the D435 camera had joined the corners of the two non-consecutive walls together, resulting in a misrepresentation of the corner (Figure 4.3(a)).

4.2.3 Features Captured

Figure 4.5 shows the scan data from LiDAR (colourful) and D435 camera (white) of the compartments of cabinets at 6th Floor Corridor, as in Figure 4.6.



(a) (b)

Figure 4.5: Scan Data of the Cabinet Compartments: (a) LiDAR (colourful), and (b) D435 (white).

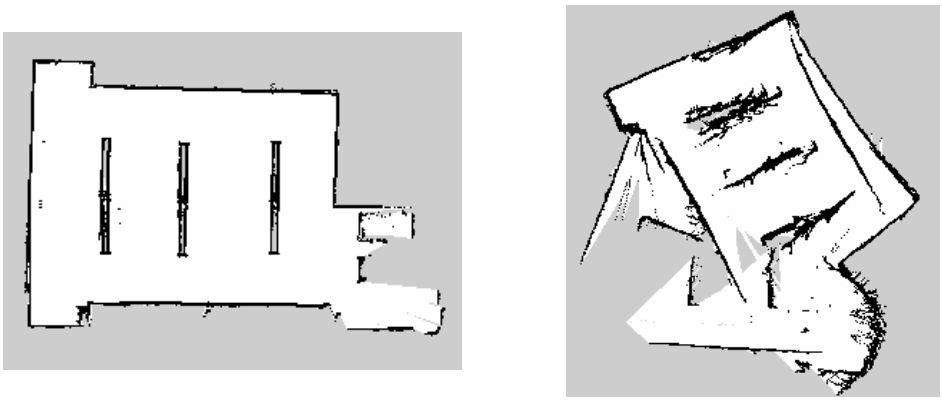
Figure 4.6: Cabinets at 6th Floor Corridor.

When using D435 camera, since the point cloud was filtered, the walls of the corridor were smoothened, therefore some features were filtered out. Figure 4.5(a) above shows that the features of the cabinets compartment, which were captured by the LiDAR as small protruding edges, were filtered out for the case of D435 camera, which represented them as a curvy line (Figure 4.5(b)). This would affect the performance of SLAM as the number of features in the corridor that was available for scan matching was reduced.

4.3 Evaluations of Lab Mapping

4.3.1 Hector SLAM

Figure 4.7 below shows the maps generated by Hector SLAM in Lab using LiDAR and D435 Camera, and Figure 4.8 shows the normalized error value. The normalized error was not calculated for the case of D435 camera as input because the map was not significant for comparison.



(a) (b)

Figure 4.7: Hector SLAM Result at Lab: (a) using LiDAR as Sensor, and (b) using D435 camera as Sensor.

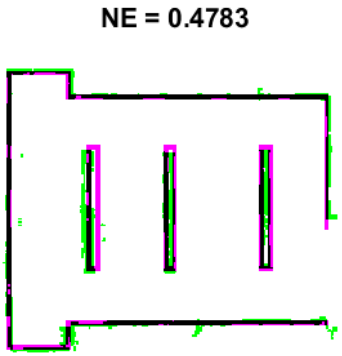


Figure 4.8: Normalized Error Results after Image Registration (LiDAR).

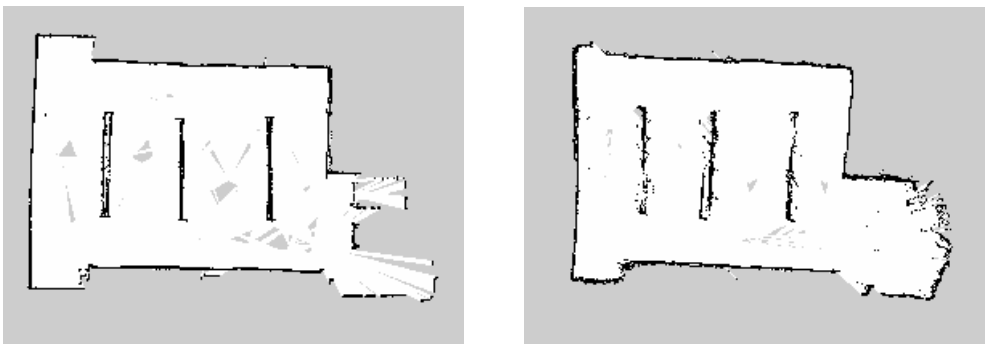
Based on Figure 4.7(a), using LiDAR, the walls and tables in the resulting map were well-aligned between each consecutive scan, with minor misalignment on the left side, between points G and H. The low normalized error of 0.4783 showed that Hector SLAM was able to do scan matching for mapping correctly without relying on odometry data or loop closure detection capabilities, given that an accurate input scan data was available.

Based on Figure 4.7(b), using D435 camera, the resulting map was highly distorted, where several duplicated representations of walls and tables were present on the map. The Hector SLAM was able to map correctly when it was travelling in a straight line, however, errors appeared when the robot was rotating, for instance, as the robot travelled from point A to B, rotated while observing the treadmills, and travelled to point C, the map get distorted.

The main reason for this distorted map was due to the change in the viewpoint of the robot as it was rotating, which greatly affect the representation of point cloud data from D435 camera, as explained in Section 4.2.1. This caused the failure of the scan matching function, as the data from the consecutive frames were different. Additionally, since Hector SLAM did not utilise the odometry data to optimise its scan matching result, the errors from the scan matching were not corrected.

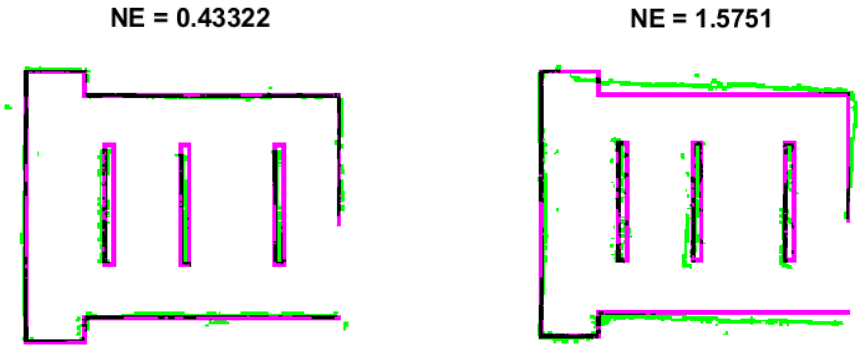
4.3.2 GMapping

Figure 4.9 and Figure 4.10 show the maps generated by GMapping in Lab using LiDAR and D435 Camera, and their calculated normalized error value.



(a) (b)

Figure 4.9: GMapping SLAM Result at Lab: (a) using LiDAR as Sensor, and (b) using D435 as Sensor.



(a) (b)

Figure 4.10: Image Registration and Normalized Error Results: (a) with LiDAR as Sensor, and (b) with D435 as Sensor.

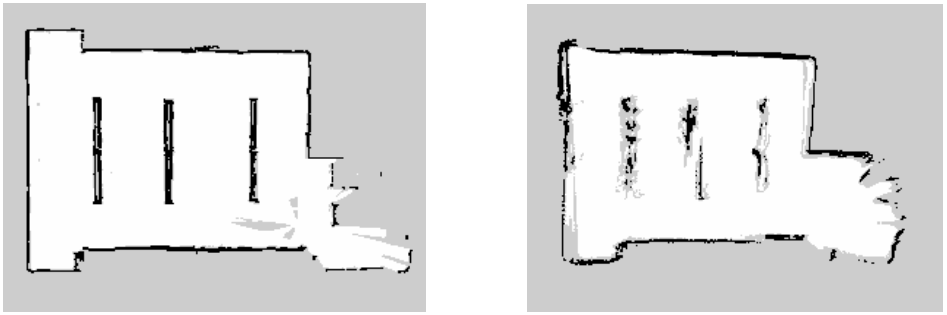
Based on Figure 4.9(a), using LiDAR, the walls in the map were well-aligned between each consecutive scan, with minor misalignment on the bottom side, between points A and B. For the mapping of the tables in the middle section, there are some misalignments, as both sides of the tables were matched as a single line during the scan matching process. Even though, the low normalized error of 0.43322 showed that GMapping was able to represent the map correctly.

Based on Figure 4.9(b), the map was slightly deviated, which increases the normalized error of the map to 1.5751. The main reason was due to the noisy scan data from D435 camera, which were still slightly wavy even though they were filtered. When travelling in straight lines, although the scan matching algorithms functioned properly to align the scan data, the wavy data causes them to slightly deviate as they were being aligned.

When rotating at the corners, in contrast to the behaviour shown by Hector SLAM, GMapping utilized odometry data to correct the predictions, thus the map did not have large distortion. This was shown when the robot observed the cluttered area at point B (treadmills). Although the scan data was not significant for scan matching, GMapping was able to do loop closing and utilized the odometry information to track the location of the robot, resulting in a correct representation of the corner.

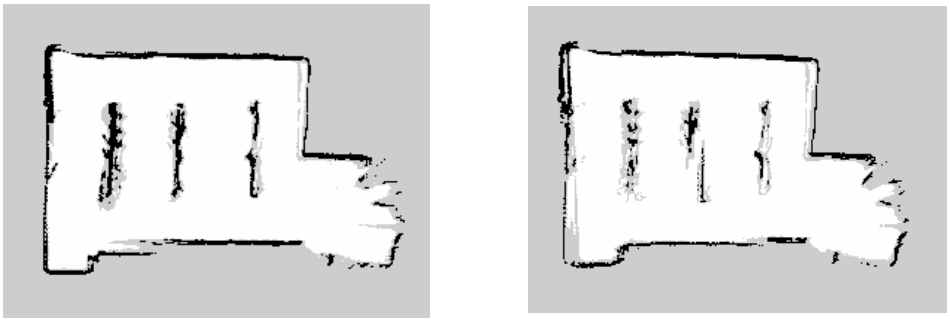
4.3.3 Cartographer

Figure 4.11 and Figure 4.13 show the maps by Cartographer in Lab using LiDAR and D435 Camera, and their normalized error value. Figure 4.12 shows the effect of final loop closure on the resulting map when using D435.



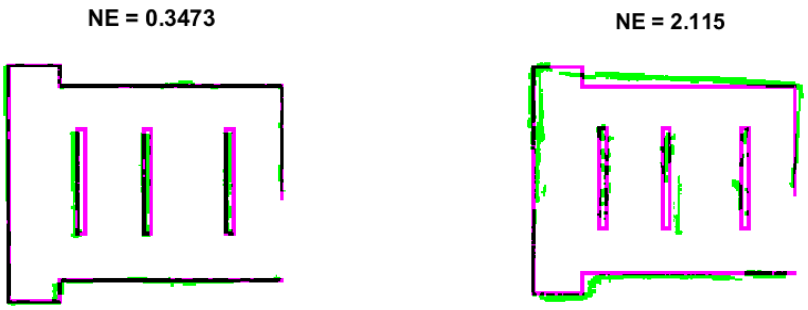
(a) (b)

Figure 4.11: Cartographer Result at Lab: (a) using LiDAR as Sensor, and (b) using D435 as Sensor.



(a) (b)

Figure 4.12: Effect of Final Loop Closure on Cartographer when using D435: (a) before Loop Closure, and (b) after Loop Closure.



(a) (b)

Figure 4.13: Image Registration and Normalized Error Results: (a) with LiDAR as Sensor, and (b) with D435 as Sensor.

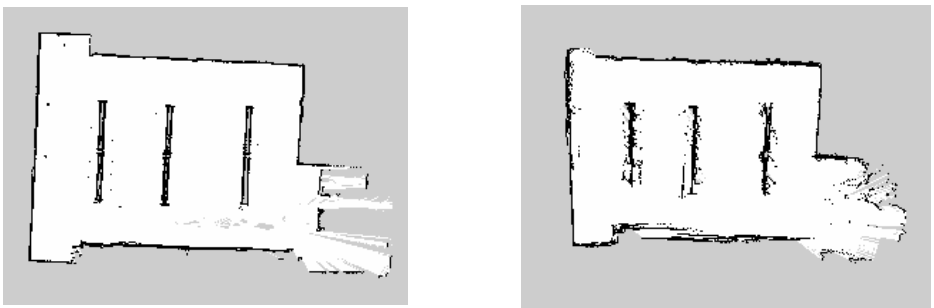
Based on Figure 4.11(a), using LiDAR, the walls and tables were well aligned. The low normalized error of 0.3473 shows that Cartographer could perform scan matching accurately within the submaps themselves, and between each of the submaps throughout the whole operation using both inputs.

Based on Figure 4.11(b), using D435 camera, there was slight distortion, especially in the middle area of the tables. This was caused by the final loop closure when the robot travelled from point H to point A. Referring to Figure 4.12, although Cartographer could perform scan matching and local optimizations correctly, the map representation (Figure 4.12(a)) had become inaccurate after the final loop closure (Figure 4.12(b)). The reason was the misrepresentation at Point G, due to the smaller FOV of D435 camera, as explained in Section 4.2.2, which had caused Cartographer to localize the robot further to the negative x-direction. Although the loop closure could close the loop and align the wall between Point H and point A, all the submaps were optimized to an inaccurate position, causing the whole map to be distorted.

On the other hand, when the robot observed the cluttered scene at point B (treadmills), even though the scan data from D435 Camera was noisy, Cartographer was still able to map the section correctly when travelling from point A to B, and C, because it utilized the odometry data for optimization.

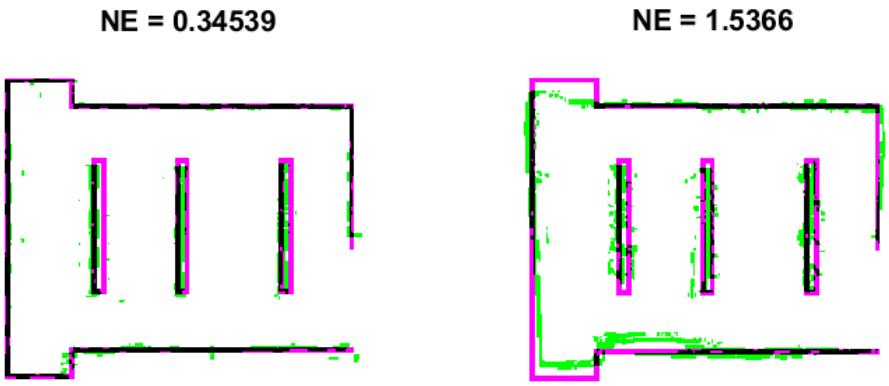
4.3.4 SLAM Toolbox

Figure 4.14 and Figure 4.15 below show the maps by SLAM Toolbox in Lab using LiDAR and D435 Camera, and their normalized error value.



(a) (b)

Figure 4.14: SLAM Toolbox Result at KB613 Lab: (a) using LiDAR as Sensor, and (b) using D435 as Sensor.



(a) (b)

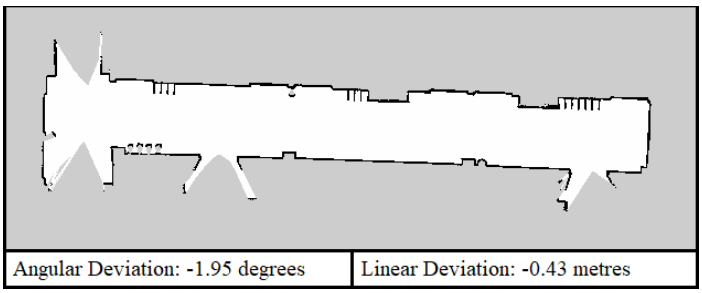
Figure 4.15: Image Registration and Normalized Error Results: (a) with LiDAR as Sensor, and (b) with D435 as Sensor.

Based on Figure 4.14(a), using LiDAR, the walls and tables were well aligned. The low normalized error of 0.34539 shows that SLAM Toolbox was able to perform scan matching accurately. Based on Figure 4.14(b), using D435 camera, similar to Cartographer, SLAM Toolbox could perform scan matching and local optimizations correctly, but the corner at Point G was not being mapped correctly due to the smaller FOV of the D435 camera. This in turn caused the walls around Point A to misalign. In contrast to Cartographer, SLAM Toolbox did not perform the loop closure optimization for this section. Overall, the mapping using D435 camera had a normalized error of 1.5366.

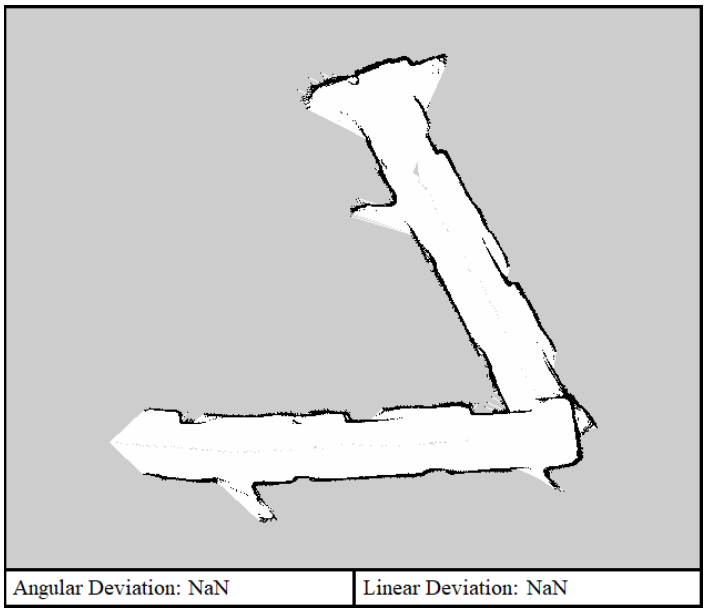
4.4 Evaluation of Corridor Mapping

4.4.1 Hector SLAM

Figure 4.16 below shows the maps generated by Hector SLAM at Corridor using LiDAR and D435, with their angular and linear deviation. The graphical result for the deviation calculations were in Appendix A.



(a)



(b)

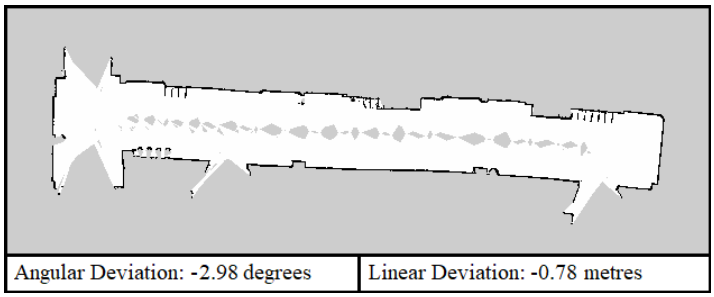
Figure 4.16: Maps Generated by Hector SLAM: (a) using LiDAR as Sensor, and (b) using D435 Camera as Sensor.

Based on Figure 4.16(a), using LiDAR, the map had a low angular deviation of -1.95 degrees and a low linear deviation of -0.43 metres. The map of the corridor was generally straight, while the compartments of the cabinets was clearly mapped as small protruding lines. This shows that the Hector SLAM was able to fully utilise the features captured by LiDAR to perform scan matching properly as it travels along a straight line.

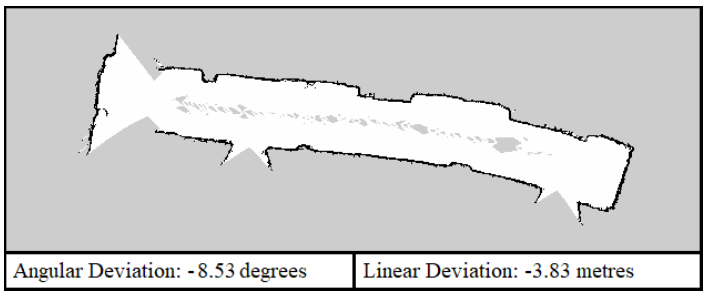
Based on Figure 4.16(b), using D435 camera, however, the map was duplicated at a different angle as the robot travelled along the corridor for the second time. This was due to the combined effect of noisy scan data from D435 camera, the lack of features captured in the scan data from D435 camera, and the nature of Hector SLAM which did not utilise the odometry data for optimization. Therefore, when the robot turned 180 degrees at the end of the corridor, due to the change of viewpoint and the change in scan data provided by D435 camera, the scan matching result was affected, therefore contributing to the error in mapping.

4.4.2 GMapping

Figure 4.17 below shows the maps generated by GMapping at Corridor using LiDAR and D435 Camera, with their angular and linear deviation. The graphical results for the deviation calculations were in Appendix A.



(a)



(b)

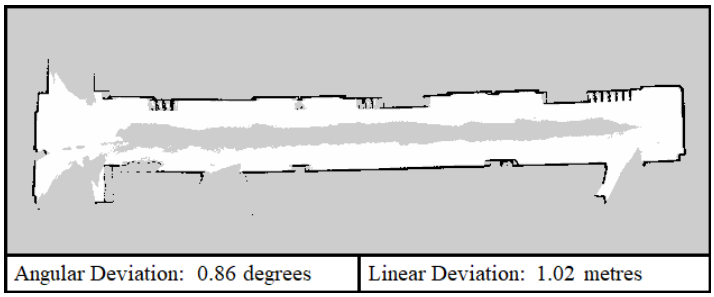
Figure 4.17: Maps Generated by GMapping: (a) using LiDAR as Sensor, and (b) using D435 Camera as Sensor.

Based on Figure 4.17(a), using LiDAR, the map had a small angular deviation of -2.98 degrees and a small linear deviation of -0.78 metres. Although GMapping performed scan matching and loop closure detection as it travelled along, the map of the long corridor was still slightly curved.

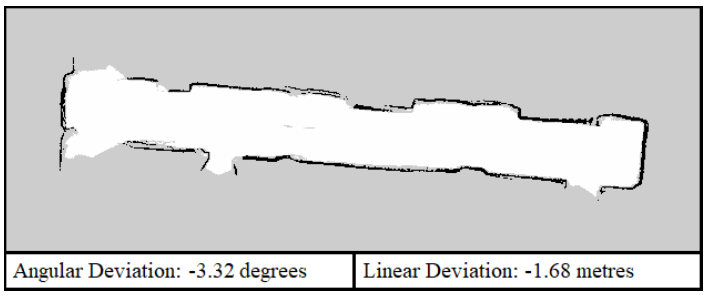
Based on Figure 4.17(b), using D435, the map had a large angular deviation of -8.53 degrees and a large linear deviation of -3.83 metres. This was due to the lesser features that were captured by the D435 camera, as explained in Section 4.2.3, which in turn caused the perceptual aliasing phenomenon. Besides, the noisy scan data from D435 camera also affected the scan matching process which caused the resulting map to curve at the end of the corridor. Although odometry data was used for optimization, the result was still greatly affected by the noisy scan data.

4.4.3 Cartographer

Figure 4.18 below shows the maps generated by Cartographer at Corridor using LiDAR and D435 Camera, with their angular and linear deviation. The graphical results for the deviation calculations were in Appendix A.



(a)



(b)

Figure 4.18: Maps Generated by Cartographer: (a) using LiDAR as Sensor, and (b) using D435 Camera as Sensor.

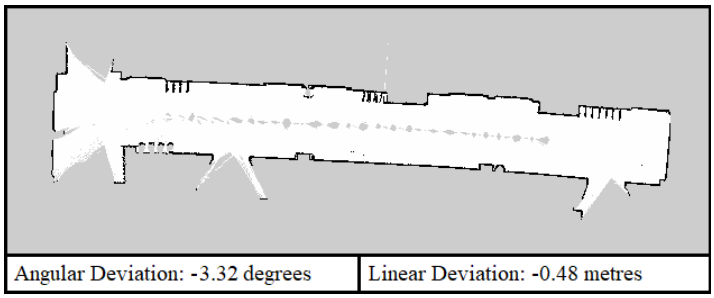
Based on Figure 4.18(a), using LiDAR, the map had a small angular deviation of 0.86 degrees and a small linear deviation of 1.02 metres. The map was slightly curved as the robot travelled along the corridor. During the returning trip from Point B to Point A, Cartographer was able to do loop closure optimization as it travelled, thus minimising the translational error in the y-direction. However, a minor translational error towards the negative x-direction was observed, which due to the perceptual aliasing phenomenon.

Based on Figure 4.18(b), using D435, the map had a large angular deviation of –3.32 degrees and a linear deviation of -1.68 metres. Since the scan data from D435 was noisy, scan matching errors were performed thus contributing to the deviation of the corridor as the error accumulates. Besides, the effect of the perceptual aliasing phenomenon was much greater in this case because the D435 captured much less scan information as compared to LiDAR,

as explained in Section 4.2.3. Thus, lesser useful features were able to be used for scan matching purposes. However, Cartographer was still able to utilize its odometry data for optimization, thus the result was still favourable as compared to GMapping.

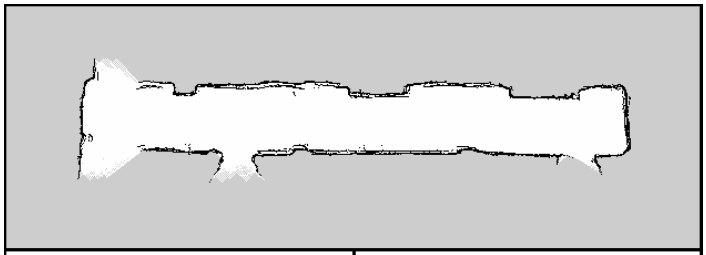
4.4.4 SLAM Toolbox

Figure 4.19 below shows the maps generated by SLAM Toolbox at Corridor using LiDAR and D435 Camera, with their angular and linear deviation. The graphical results for the deviation calculations were in Appendix A.



Angular Deviation: -3.32 degrees	Linear Deviation: -0.48 metres

(a)



Angular Deviation: -0.32 degrees	Linear Deviation: -3.33 metres

(b)

Figure 4.19: Maps Generated by SLAM Toolbox: (a) using LiDAR as Sensor, and (b) using D435 Camera as Sensor.

Based on Figure 4.19(a), when using LiDAR as input, the map had an angular deviation of –3.32 degrees and a small linear deviation of -0.48 metres. Similar to GMapping and Cartographer, the resulting map was slightly curved as the robot travelled along the corridor.

Based on Figure 4.19(b), when using D435 as input, the map had a small deviation of –0.0055, however, with a large linear deviation of -3.33 metres. SLAM Toolbox was able to utilise the odometry data for optimization, thus the noise from D435 camera did not affect the angular deviation much.

However, the resulting map was not good as it suffered from the perceptual aliasing phenomenon, which contributed to the high linear deviation due to the lack of features in D435 camera's scan data, as explained in Section 4.2.3.

4.5 Comparisons and Summary

Figure 4.20 and Figure 4.21 below show the evaluated results for the mapping in Lab and Corridor respectively using LiDAR and D435 Camera.

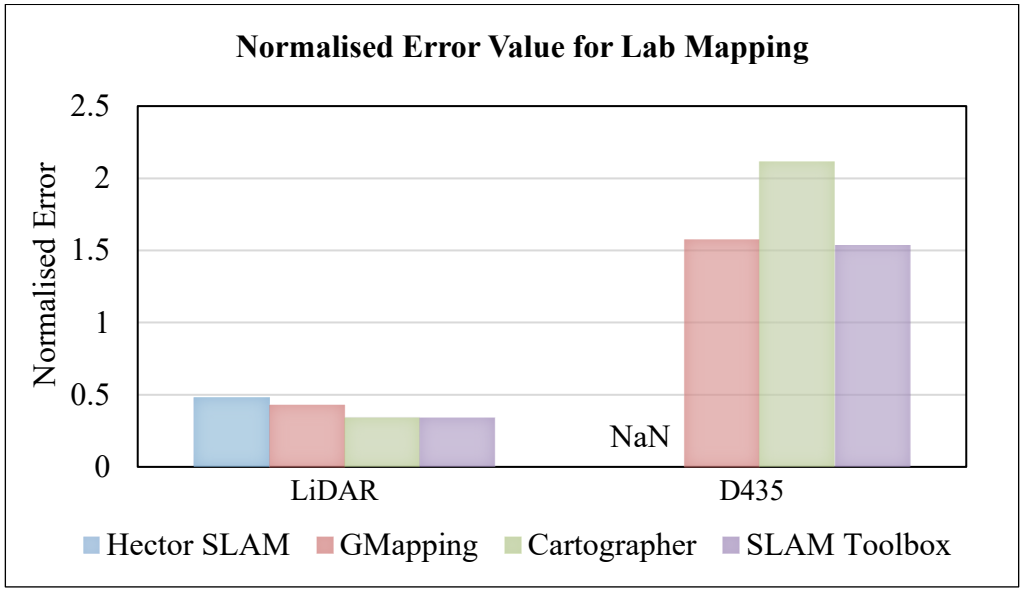


Figure 4.20: Normalised Error Value for Lab Mapping.

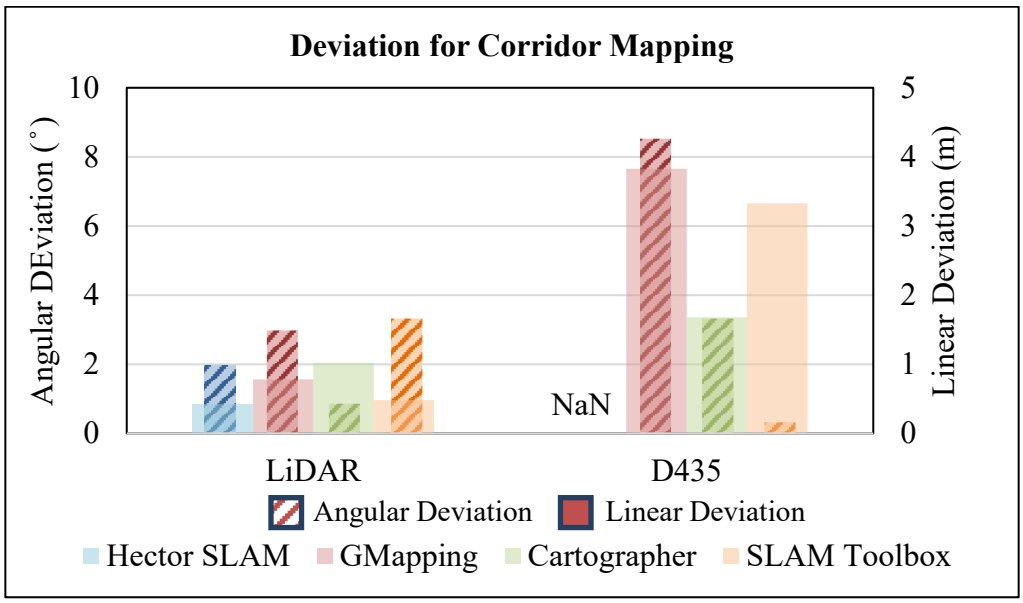


Figure 4.21: Deviation for Corridor Mapping.

For the mapping of Lab using LiDAR, the best-performing methods were the Cartographer and SLAM Toolbox, both recording a low normalised error result of 0.35. Both packages contain a similar architecture, where they utilised the odometry data and loop closure to optimise their result. In contrast to GMapping which mapped both sides of the table as a single line, both Cartographer and SLAM Toolbox were able to represent the mapping of tables correctly. The worst performing method in this scenario was the Hector SLAM, as it did not utilise the odometry data and loop closure for optimisation, which caused the error as the robot rotated.

For the mapping of Lab using D435, the best performing method was SLAM Toolbox, which obtained the lowest normalised error of 1.54, which was just slightly better than GMapping at 1.58. Both SLAM Toolbox and GMapping differed in their approaches to performing loop closure optimization, whereby the prior optimizes the graph of submaps, and the latter updates its state estimation through resampling. Despite having a similar architecture as SLAM Toolbox, Cartographer had a higher normalised error value due to the distortion caused by the final loop closure optimisation. Hector SLAM performed the worst as it did not utilise the odometry data for optimization, which caused the error as the robot rotated.

For the mapping of Corridor using LiDAR, the best performing methods were the Hector SLAM and SLAM Toolbox, with a low linear deviation of -0.43 metres and -0.48 metres respectively, but the latter had the highest angular deviation. This shows that Hector SLAM had a robust scan matching algorithm, as compared to other methods which utilised the odometry data for optimization. Although Cartographer had a low angular deviation, it had the highest linear deviation, thus it was considered the worst-performing SLAM in this scenario.

For the mapping of Corridor using D435, all the methods performed badly, with Cartographer getting the best result at -3.32 metres of linear deviation. This shows that the lack of feature in both the D435 filtered scan data and the corridor had affected the performance of all state-of-the-art 2D SLAM methods. The Hector SLAM performed the worst in this scenario as it depended solely on scan matching for its operation.

In summary, the quality of the scan matching was the main factor that affects the performance of the state-of-the-art 2D SLAM. The scan matching process could be affected by the lack of features in the corridor, the lack of features in D435 filtered scan result, and the noisy scan data, as observed in this study. On other hand, the ability of 2D SLAM methods in performing loop closure detection and odometry data optimisation had greatly improved the performance of 2D SLAM, which could be observed through the comparison of the result of Hector SLAM with other methods.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The state-of-the-art 2D SLAM packages are based on the three main paradigms of SLAM, which are the EKF-based SLAM, RBPF-based SLAM, and graph-based SLAM. In this project, even though the SLAM packages are based on different architectures, they are capable of mapping KB613 Lab and 6th Floor Corridor, with a different performance and mapping quality for different scenarios. From the evaluated results, it is observed that the main feature of the state-of-the-art 2D SLAM is its scan matching capability, for instance, the external factors such as the noise in the sensors impacts and causes failure to the SLAM packages that mainly rely on the scan matching capability, which is the Hector SLAM. Other SLAM packages such as GMapping, Cartographer, and Hector SLAM are capable of using odometry data and loop closure detection to optimise their mapping result. On the other hand, the 2D SLAM packages have limitations in certain scenarios. For instance, when mapping a cluttered area such as the treadmills in the lab, the change of scan data from the Realsense Camera due to the change of viewpoints impacts the scan matching capability. When mapping the long and featureless corridor, together with the reduction of feature due to data captured using the Realsense Camera, the phenomenon of perceptual aliasing effect is much more obvious, therefore impacting the accuracy of mapping. In conclusion, the 2D SLAM packages could be implemented successfully with the use of accurate sensors and the correct tuning of SLAM parameters, based on a given scenario.

5.2 Recommendations for future work

Mobile robots, as they are getting commercialised nowadays for home service robot applications, require a much more stable development for their use in different kinds of scenarios. While the state-of-the-art 2D SLAM packages available in ROS are capable of mapping, they are not stable enough for the

applications in different kind of indoor scenarios, not to mention the presence of unstructured conditions and dynamic objects in the scene. While this project offers an insight into the implementations of 2D SLAM, it does not cover the implementations of 3D SLAM, computer vision, and AI.

To extend the scope of this project for future development, further analysis towards the architecture of customised SLAM frameworks such as ORB-SLAM and LOAM, as well as frameworks which have integrated the use of machine learning, as mentioned earlier in Section 2.4.1, could be carried out to provide more insights into their usage. In terms of the scalability and the robustness of the SLAM frameworks, the project could be carried out in a much more challenging environments, such as the presence of dynamically moving objects, change of objects in the scene, and the change of environmental lightings at the different time of day. The result of mapping could then be evaluated by analysing their capabilities and accuracy in representing the scenes, such as the alignment of the surfaces of objects, and the presence of noise in the point cloud data.

REFERENCES

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J.J., 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, *32*(6), pp.1309-1332.

Carlevaris-Bianco, N.D., 2015. *Long-term simultaneous localization and mapping in dynamic environments*. PhD. University of Michigan.

Cartographer, 2019. *Running Cartographer ROS on a demo bag*. [Online] Available at: <https://google-cartographer.readthedocs.io/en/latest/> [Accessed 14 August 2021].

Durrant-Whyte, H. and Bailey, T., 2006. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine*, 13(2), pp.99-110.

Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W., 2010. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), pp.31-43.

Grisetti, G., Stachniss, C. and Burgard, W., 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, *23*(1), pp.34-46.

Hess, W., Kohler, D., Rapp, H. and Andor, D., 2016, May. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1271-1278). IEEE.

Kohlbrecher, S., Von Stryk, O., Meyer, J. and Klingauf, U., 2011, November. A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics* (pp. 155-160). IEEE.

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B. and Vincent, R., 2010, October. Efficient sparse pose adjustment for 2D mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 22-29). IEEE.

Liu, Y. and Miura, J., 2021. RDS-SLAM: real-time dynamic SLAM using semantic segmentation methods. *IEEE Access*, *9*, pp.23772-23785.

Luknanto, B.K., 2020. *A review of 2D SLAM algorithms on ROS*. Master. Politecnico di Milano. Available at: <http://hdl.handle.net/10589/164687> [Accessed 14 August 2021].

Marks, T.K., Howard, A., Bajracharya, M., Cottrell, G.W. and Matthies, L.H., 2009. Gamma-SLAM: Visual SLAM in unstructured environments using variance grid maps. *Journal of Field Robotics*, *26*(1), pp.26-51.

Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B., 2002. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, *593598*.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y., 2009, May. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

ROBOTIS, 2021. *ROBOTIS e-Manual.* [Online] Available at: <https://emanual.robotis.com/docs/en/platform/turtlebot3/features/> [Accessed 2 September 2021].

Santos, J.M., Portugal, D. and Rocha, R.P., 2013, October. An evaluation of 2D SLAM techniques available in robot operating system. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (pp. 1-6). IEEE.

Shi, X., Li, D., Zhao, P., Tian, Q., Tian, Y., Long, Q., Zhu, C., Song, J., Qiao, F., Song, L. and Guo, Y., 2020, May. Are we ready for service robots? The OpenLORIS-scene datasets for lifelong SLAM. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3139-3145). IEEE.

Takleh, T.T.O., Bakar, N.A., Rahman, S.A., Hamzah, R. and Aziz, Z.A., 2018. A brief survey on SLAM methods in autonomous vehicle. *International Journal of Engineering & Technology*, *7*(4), pp.38-43.

Thrun, S., Burgard, W. and Fox, D., 2005. *Probabilistic Robotics*. Cambridge: The MIT Press.

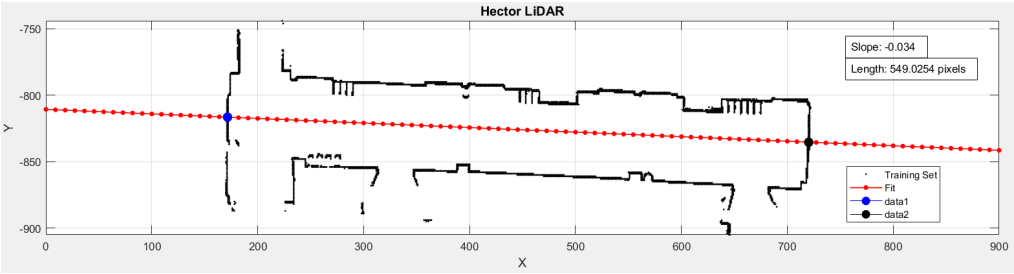
Tipaldi, G.D. and Arras, K.O., 2010, May. Flirt-interest regions for 2d range data. In *2010 IEEE International Conference on Robotics and Automation* (pp. 3616-3622). IEEE.

Wang, Y., Zhang, W. and An, P., 2017, October. A survey of simultaneous localization and mapping on unstructured lunar complex environment. In *AIP Conference Proceedings* (Vol. 1890, No. 1, p. 030010). AIP Publishing LLC.

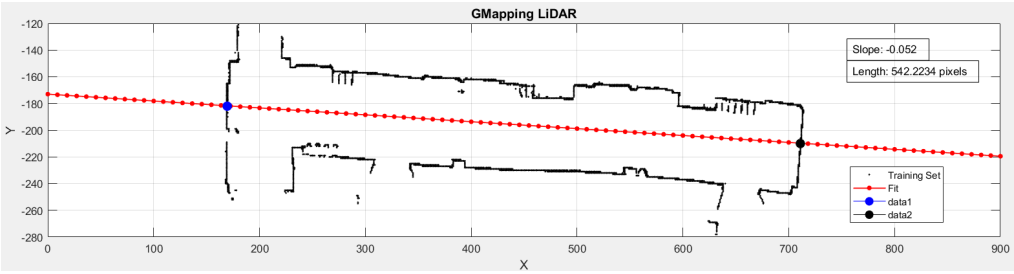
Yuan, X. and Chen, S., 2020. SaD-SLAM: A Visual SLAM Based on Semantic and Depth Information. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4930-4935). IEEE.

APPENDICES

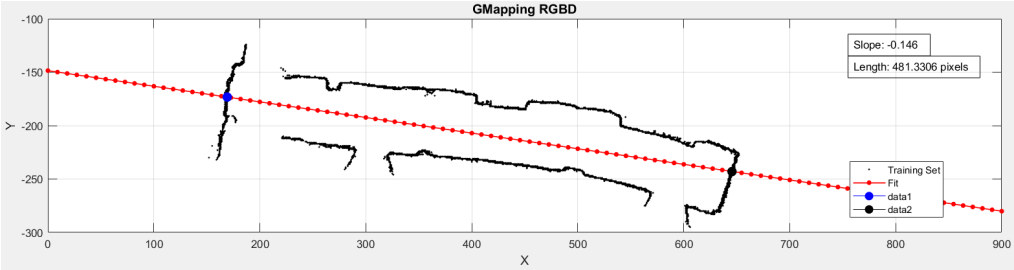
Appendix A: Graphs



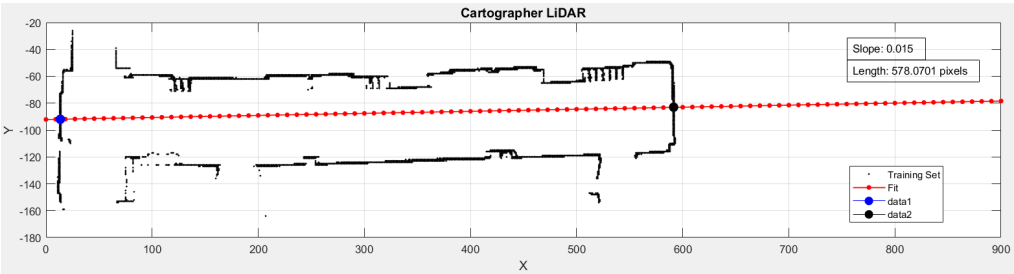
GraphA-1: Deviation Calculation Result for Hector SLAM using LiDAR.



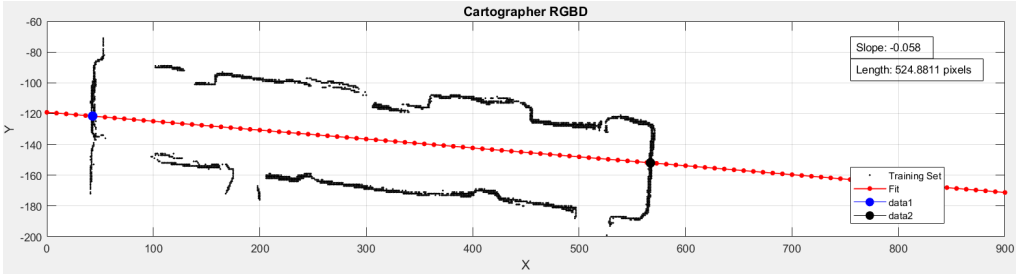
GraphA-2: Deviation Calculation Result for GMapping using LiDAR.



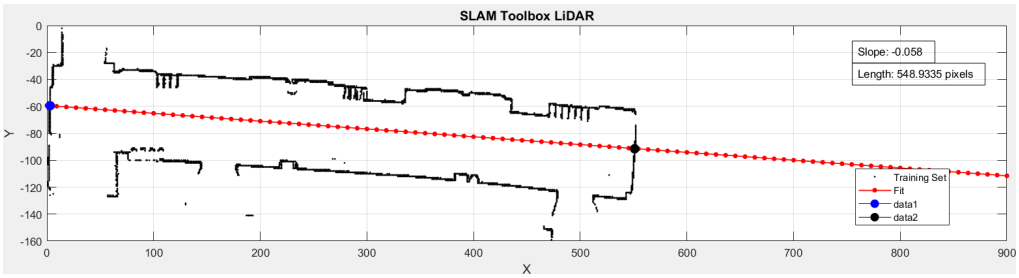
GraphA-3: Deviation Calculation Result for GMapping using D435.



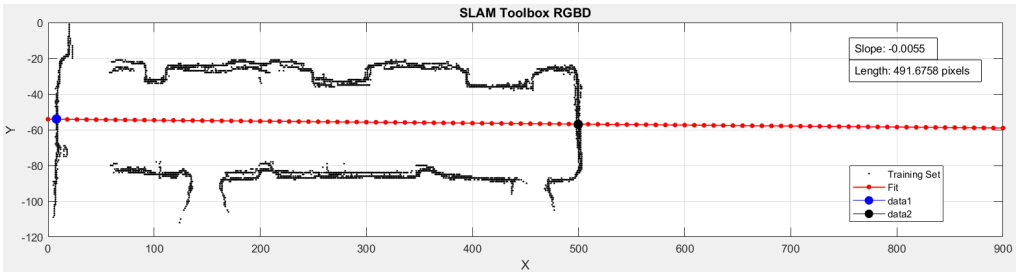
GraphA-4: Deviation Calculation Result for Cartographer using LiDAR.



GraphA-5: Deviation Calculation Result for Cartographer using D435.



GraphA-6: Deviation Calculation Result for SLAM Toolbox using LiDAR.



GraphA-7: Deviation Calculation Result for SLAM Toolbox using D435.