HYBRID LIDAR AND VISION-BASED SLAM OF AUTONOMOUS FORKLIFT

by

Piyawat Apiwattanadej

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Mechatronics

Examination Committee: Prof. Manukid Parnichkun (Chairperson)
Prof. Matthew N. Dailey (Member)
Dr. Mongkol Ekpanyapong (Member)

Nationality: Thai
Previous Degree: Bachelor of Engineering in Production Engineering
King Mongkut's University of Technology
Thonburi, Thailand

Scholarship Donor: Siam Kubota Corporation Co., Ltd. Thailand

Asian Institute of Technology
School of Engineering and Technology
Thailand
July 2021

AUTHOR'S DECLARATION

I, Piyawat Apiwattanadej, declare that the research work carried out for this thesis was in accordance with the regulations of the Asian Institute of Technology. The work presented in it are my own and has been generated by me as the result of my own original research, and if external sources were used, such sources have been cited. It is original and has not been submitted to any other institution to obtain another degree or qualification. This is a true copy of the thesis, including final revisions.

Date: 13 July 2021

Name: Piyawat Apiwattanadej

Signature:

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor, Prof. Manukid Parnichkun for the valuable support in both study and thesis research, for treasure guidance, suggestion, motivation and immeasurable knowledge. His support and guidance helped me from struggling in many problems during my research. Without his support, this research is impossible to be done.

I wish to express my gratitude to Prof. Matthew N. Dailey and Dr. Mongkol Ekpanyapong, the examination committees, for their valuable comments and knowledge sharing.

Moreover, I would like to thank you to Mr. Prasitthichai Naronglerdrit, Miss Alisa Kunapinan and Mr. Dechatorn Subcharoen, Ph.D. students of Mechatronics Engineering field of study for their valuable support throughout my research.

I also wish to express my thanks to the Siam Kubota Corporation for the financial support which gives me this great opportunity to explore my master career.

Lastly, I wish to thank my parents and my friends for their support during my study.

ABSTRACT

In this research, the hybrid lidar and vision-based simultaneous localization and mapping (SLAM) system for autonomous forklift will be proposed. This SLAM system is capable of localize its position while building the map from surrounding environment. In case that system is kidnapped during the system is shutdown, the system also able to re-localization its position by given the map of the environment and compare with current scanned point cloud from lidar to recognize system's position and heading. The system is consisted of camera for Optical Flow based visual odometry in order to calculate relative transformation between two consecutive positions of system while lidar is used to detect environment around the system and using point cloud from lidar with relative position information from visual odometry for fine tuning the system's position using Iterative Closest Point (ICP) algorithm. This system will be installed on forklift to localize position of forklift during navigation task.

CONTENTS

	Page
ACKNOWLEDGMENTS	**iii**
ABSTRACT	**iv**
LIST OF TABLES	**vii**
LIST OF FIGURES	**ix**
CHAPTER 1 INTRODUCTION	**1**
1.1 Background	1
1.2 Statement of the Problem	1
1.3 Objective	2
1.4 Limitations and Scope	2
CHAPTER 2 LITERATURE REVIREW	**3**
2.1 Robot Position	3
2.2 Simultaneous Localization and Mapping (SLAM)	4
2.3 Lidar-Based SLAM	4
2.3.1 Gmapping	5
2.3.2 HectorsSLAM	6
2.3.3 Cartographer	7
2.4 Scan Matching	7
2.5 Visual-Based SLAM	8
2.5.1 Feature based SLAM	8
2.5.2 Direct SLAM	9
2.6 Visual-Lidar SLAM	10
CHAPTER 3 METHODOLOGY	**11**
3.1 Concept	11
3.2 Hardware Design	12
3.2.1 Overall Hardware Structure	12
3.2.2 Plant Characteristic	13

Page

3.2.3 Camera 13

3.2.4 Lidar 14

3.3 Software Design 14

3.4 Visual Odometry System 15

3.4.1 Camera Calibration 15

3.4.2 Visual Odometry 16

3.5 SLAM System 19

3.5.1 Interpretation of Row Data Measurement 19

3.5.2 Scan Matching Algorithm 22

3.5.3 Localization 33

3.5.4 Mapping 33

3.5.5 Re-Localization 34

3.6 Navigation System 34

3.7 Experiment Design 35

CHAPTER 4 RESULT AND DISCUSSION **40**

4.1 Overview 40

4.2 Mapping 40

4.3 Evaluation of Re-Localization System 40

4.4 Evaluation of Navigation System 53

4.5 Evaluation of Hybrid Lidar and Vision-Based SLAM System 57

CHAPTER 5 CONCLUSIONS AND RECOMMENDATION **61**

5.1 Conclusion 61

5.2 Recommendation 61

5.3 Future Improvement 62

REFERENCES **63**

LIST OF TABLES

Tables	Page
Table 2.1 Summary of Lidar-Based SLAM's Advantages and Drawbacks	5
Table 2.2 Summary of Visual-SLAM's Advantages and Drawbacks	10
Table 3.1 Camera Parameters	15
Table 3.2 Testing Conditions for Re-Localization System	38
Table 3.3 Testing Conditions for Navigation System	38
Table 4.1 Re-Localization Result of First Test Point	41
Table 4.2 RMSE Between Result of First Test Point and Actual Position	42
Table 4.3 Re-Localization Result of Second Test Point	42
Table 4.4 RMSE Between Result of Second Test Point and Actual Position	43
Table 4.5 Re-Localization Result of Third Test Poin	43
Table 4.6 RMSE Between Result of Third Test Point and Actual Position	44
Table 4.7 Re-Localization Result of Fourth Test Point	44
Table 4.8 RMSE Between Result of Fourth Test Point and Actual Position	45
Table 4.9 Re-Localization Result of Fifth Test Poin	45
Table 4.10 RMSE Between Result of Fifth Test Point and Actual Position	46
Table 4.11 Re-Localization Result of Sixth Test Poin	46
Table 4.12 RMSE Between Result of Sixth Test Point and Actual Position	47
Table 4.13 Re-Localization Result of Seventh Test Point	47
Table 4.14 RMSE Between Result of Seventh Test Point and Actual Position	48
Table 4.15 Re-Localization Result of Eighth Test Point	48
Table 4.16 RMSE Between Result of Eighth Test Point and Actual Position	49
Table 4.17 Re-Localization Result of Nineth Test Point	49
Table 4.18 RMSE Between Result of Nineth Test Point and Actual Position	50
Table 4.19 Re-Localization Result of Tenth Test Point	50
Table 4.20 RMSE Between Result of Tenth Test Point and Actual Position	51
Table 4.21 Re-Localization Result of Eleventh Test Point	51
Table 4.22 RMSE Between Result of Eleventh Test Point and Actual Position	52
Table 4.23 Re-Localization Result of Twelfth Test Point	52

Tables **Page**

Table 4.24 RMSE Between Result of Twelfth Test Point and Actual Position 53
Table 4.25 Evaluation of Re-Localization System 53
Table 4.26 RMSE Between Result of System's Initial Position and Actual Position 54
Table 4.27 RMSE Between Result of System's Position when Reach First Waypoint and Actual Position 55
Table 4.28 RMSE Between Result of System's Position when Reach Second Waypoint and Actual Position 56
Table 4.29 Evaluation of Navigation System 57
Table 4.30 RMSE Between Result of Lidar-Only Localization System's Position at Initial Position and Actual Position 58
Table 4.31 RMSE Between Result of Lidar-Only Localization System's Position when Reach First Waypoint and Actual Position 58
Table 4.32 RMSE Between Result of Lidar-Only Localization System's Position when Reach Second Waypoint and Actual Position 59
Table 4.33 Comparison Between RMSE of Distance from Hybrid Lidar and Vision-Based SLAM System and Lidar-Only Localization System 60

LIST OF FIGURES

Figures	Page
Figure 2.1 The Graph-Based Approach SLAM System	5
Figure 2.2 HectorSLAM and 3D State Estimation System	6
Figure 2.3 ORB SLAM System	9
Figure 2.4 LSD SLAM System	9
Figure 3.1 Overall System Structure	11
Figure 3.2 Overall Hardware Structure	12
Figure 3.3 Industrial Counter Balanced Electric Stacker Model: Xilin CQD15R	13
Figure 3.4 Picture of Monocular Camera	13
Figure 3.5 Picture of Lidar SICK s300 Expert	14
Figure 3.6 Dimension of Lidar SICK s300 Expert	14
Figure 3.7 Image Frame with Feature Points in Sky and Ground Region	16
Figure 3.8 Calculation of Feature Point's Angle Related to Center of Camera	17
Figure 3.9 Image Frame with Marked Coordinates at Each Tile Corner	17
Figure 3.10 Homography Transformation of Feature Points on the Ground Region	18
Figure 3.11 Structure of Raw Measurements from Lidar	20
Figure 3.12 Data Words from Sensor and Interpretation of the Words into Measurements	20
Figure 3.13 The First Sub Node and the Reference point	23
Figure 3.14 The Left Section of the Search Tree and the Reference Point in Section	24
Figure 3.15 The Completed K-D Tree	25
Figure 3.16 The Nearest Distance to the Search Point (r)	26
Figure 3.17 The Perpendicular Distance to the Adjacent Section	27
Figure 3.18 The Searching Process through the K-D Tree until the Closest Point is Found	28
Figure 3.19 The Matching of Current Point Cloud to the Previous Point Cloud	32
Figure 3.20 Map of the Testing Area	35
Figure 3.21 Reference Coordinates in the Map for Testing Re-Localization System	35
Figure 3.22 Real Environment of the Testing Area	36
Figure 3.23 Reference Coordinates in the Map for Testing Navigation System	39

Figures **Page**

Figure 4.1 Comparison of Graphical Map and the Scanned Map 40

Figure 4.2 Graphical and Actual Representation of Position of the System and the Scanned Area of the First Test Point 41

Figure 4.3 Graphical and Actual Representation of Position of the System and the Scanned Area of the Second Test Point 42

Figure 4.4 Graphical and Actual Representation of Position of the System and the Scanned Area of the Third Test Point 43

Figure 4.5 Graphical and Actual Representation of Position of the System and the Scanned Area of the Fourth Test Point 44

Figure 4.6 Graphical and Actual Representation of Position of the System and the Scanned Area of the Fifth Test Point 45

Figure 4.7 Graphical and Actual Representation of Position of the System and the Scanned Area of the Sixth Test Point 46

Figure 4.8 Graphical and Actual Representation of Position of the System and the Scanned Area of the Seventh Test Point 47

Figure 4.9 Graphical and Actual Representation of Position of the System and the Scanned Area of the Eighth Test Point 48

Figure 4.10 Graphical and Actual Representation of Position of the System and the Scanned Area of the Nineth Test Point 49

Figure 4.11 Graphical and Actual Representation of Position of the System and the Scanned Area of the Tenth Test Point 50

Figure 4.12 Graphical and Actual Representation of Position of the System and the Scanned Area of the Eleventh Test Point 51

Figure 4.13 Graphical and Actual Representation of Position of the System and the Scanned Area of the Twelfth Test Point 52

Figure 4.14 Initial Position of Forklift in Real Environment, System and Map 54

Figure 4.15 Position of Forklift in Real Environment, System and Map and Its Trajectory when Reach First Waypoint 55

Figure 4.16 Position of Forklift in Real Environment, System and Map and Its Trajectory when Reach Second Waypoint 56

Figure 4.17 Location of System in the Map and Current Scan of the System at Initial Position 57

Figures **Page**

Figure 4.18 Location of System in the Map and Current Scan of the System when Reach First Waypoint 58

Figure 4.19 Location of System in the Map and Current Scan of the System when Reach Second Waypoint 59

CHAPTER 1
INTRODUCTION

1.1 Background

An automated guided vehicle (AGV) has many impacts on the manufacturing industry. In the production processes, transportation and material handling process are needed to transport the materials from one process to another process. However, these processes are not the core process of the automobile industry, so called non-value-added activities.

In automobile industry, the demand of the product is very high. In order to handle the high production rate, increasing manpower is one of the solutions. This results in increasing the expenses on transportation and material handling process that using operator to transport the materials.

The development of AGVs have resulted in many kinds of autonomous transportation vehicle in the industrial floor, such as autonomous tow tractor and autonomous forklift. By implementation of these AGVs, manpower for operating the transportation process can be reduced. In case of autonomous forklift, unlike autonomous tow tractor that has the exact path for operating, forklift will be operated in the specific area that is assigned to work for minimizing safety hazard. Therefore, there is a flexibility for forklift when operating to choose the best path to reach the destination. To navigate the autonomous forklift, two systems are needed to be installed on the forklift which are localization system and navigation system. The most suitable localization system for the indoor forklift is Simultaneous Localization and Mapping (SLAM).

1.2 Statement of the Problem

SLAM can be used to solve the problem of robot operates in unknown environment. Common sensors that are usually implemented to SLAM are either lidar or camera with the help of wheel encoders or inertial measurement unit (IMU) for dead reckoning process to determine the relative position of the plant. However, both lidar and camera has their own weaknesses. Lidar can achieve high accuracy in sensing environment, but during moving through the environment, the scan point cloud that can be retrieved from lidar will change according to the environment without knowing the translation and

rotation of the plant. To localize the plant using lidar, wheel encoders and IMU will be used to compute for the relative position from previous pose to next pose. On the other hand, camera can also be used for features tracking and be able to localize the plant by itself without relying on other sensors. The downsides of camera are that, the map created from camera will be in 3D point cloud map and difficult to use for navigation process because of sparse map and, the accuracy of camera will be depended on resolution of camera and the lighting of the environment which is not as accurate as lidar.

1.3 Objective

This research is aiming to develop a hybrid lidar and vision-based SLAM for navigating the autonomous forklift. The following objectives must be achieved.

1. To develop a hybrid lidar and vision-based SLAM algorithm.
2. To navigate autonomous forklift using the map that is created from the proposed SLAM.

1.4 Limitations and Scope

1. The speed of the forklift will be limited to 30 cm/s.
2. The usage of the autonomous forklift will be limited only indoor application.
3. The size of the room for experiment will be roughly 1000 cm x 720 cm of the room in ISE building

CHAPTER 2
LITERATURE REVIREW

2.1 Robot Position

Dead reckoning is the concept of estimating the position of system relative to the sensor and acts as initial guess position of the robot. By measuring moving directions of the robot using encoders, gyros, and accelerometers, the relative position of robot is obtained and able localize the robot position by relate the measurements with the previous position. There are several ways to implement dead reckoning. One method to implement dead reckoning is to create system that consists of IMU, and encoders on two wheels of robot for heading angle reading, and ultrasonic measurements sub system that consists of ultrasonic to detect the environment with radio-frequency. Another method is to use encoders installed on wheels and inertial sensors only for localization for indoor environment and implementation of Kalman filter to optimize the estimations of orientation and velocity of mobile robot which gives a better result than standalone odometry. Another method is to track features of the scene of the system using camera with Optical Flow techniques. By tracking the features of the scene from the Optical Flow, the rotation and translation of the system can be determined. The features will be separated into two groups, sky region and ground region. Sky region features will be used to find rotation of system by calculating difference between mean angle of all features between consecutive frames. Ground region features will be used to find translation of system. The ground features will be perspective transformed into bird eye view. Using previous rotation that is obtained from sky features to transform the next frame into the previous frame coordinate, then subtract the mean coordinate of all features between consecutive frames to find the translation.

Even the robot position can be determined using dead reckoning, there are still some problems with using dead reckoning for localization of the robot. The error from the pose estimation using only dead reckoning will keep accumulate until the system fails. The best way to solve this accumulated error is to implement loop closure. Loop closure can solve the problem of re-observed places of the environment in mobile robot which robot is not able to recognize the place that it already visited. Still, the plant unable to locate itself in global position.

2.2 Simultaneous Localization and Mapping (SLAM)

SLAM is a algorithm for map creation of unknown environment using sensors while capable of localize the position of the robot. Most applications in factory are indoor applications which will encounter the problem of inaccurate of Global Positioning System (GPS) and SLAM can be used to solve this problem of indoor applications. The map that is created from SLAM represents landmarks that describe the environment that robot operates. The map can be used in path planning task as well as limit the error in estimation robot position.

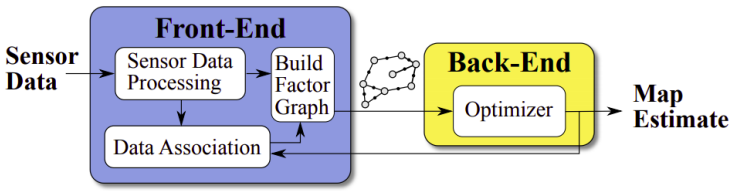
2.3 Lidar based SLAM

One of the main sensors of SLAM for indoor applications is lidar which is capable of low drift estimation. Lidar based SLAM can be separated into two main methods. Firstly, filtered-based approach which performs the estimation of the robot poses and map recursively. The robot poses and environment will be stored in the state vector. Error covariance matrix will be used to store uncertainties of the states along with correlation that is computed from the map and system location. The popular solution of SLAM that based on filtering approach and Extended Kalman Filter (EKF) and Particle Filter (PF). Secondly, graph-based method which use the pose graph to represent the robot trajectories. The node of the graph will represent the robot position in the environment while the edges represent the constraints between robot pose which is the relative transformation between two robot poses that can be computed from odometry between poses or, by aligning the observations that are collected from two poses. The graph-based method consists of two main tasks. First, graph construction which is the front-end of the system. Graph construction will take the raw measurement from the sensor and performs data association to build the graph. Then, graph optimization which is the back-end of the system will find the best estimation of the robot position using nonlinear least squares optimization technique. Scan-Matching and graph optimization is a technique to solve SLAM problem by using concept of Iterative Closest Point (ICP). ICP will extracts the closest point between each iteration of scanning process. After that, by matching between these closest points, transformation will be performed for the next scan. ICP will be repeated until current error is within the theshold. Solution for SLAM approach by optimizing pose graph that contains nodes of the robot and the world features. The target is to get the minimum error that

of the nodes given the constraints. The system of the graph-based approach is shown in the Figure 2.1.

Figure 2.1

The Graph-Based Approach SLAM System



The lidar-based SLAM gives a satisfying result. However, there are still some drawbacks of each approach of the mentioned algorithms of lidar-based SLAM. The **Table 2.1** shows the advantages and disadvantages of each solution of lidar-based SLAM.

Table 2.1

Summary of Lidar-Based SLAM's Advantages and Disadvantages

	Lidar-based SLAM	
	Filtered-Based	**Graph-Based**
Advantages	2D lidar Standard technique. Easy to implement. Precise and Accurate result.	Easier loop closure than filtering technique. Capable of using in large environment because of removing process of raw data using optimization.
Disadvantages	Can be used for 2D technique only, Difficult to process for huge environment due to high memory required. Loop closure is difficult to implement.	The edges between position of system need to be accurately estimated.

2.3.1 Gmapping

Gmapping is Rao-Blackwellized particle filters (RBPF) based SLAM algorithm. The idea is to estimate the joint posterior between poses and the map.

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

Where, m is the map, $x_{1:t}$ is the trajectory of the robot from first position to the last position given the observations which is the measurement of the environment that is obtained from sensor $Z_{1:T}$ and odometry information which can be retrieved from

encoders and IMU $U_{1:T-1}$, Then RBPF for SLAM will factorizes the above equation into the following equation.

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \quad (2.2)$$

We can firstly estimate the path of the robot then create map of environment after we estimate robot path. Gmapping is an improved approach of the SLAM based on RBPF. Gmapping makes use of optimal proposal distribution.

$$p\left(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}\right) = \frac{p\left(z_t \mid m_{t-1}^{(i)}, x_t\right) \cdot p\left(x_t \mid x_{t-1}^{(i)}, u_{t-1}\right)}{p\left(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}\right)} \quad (2.3)$$

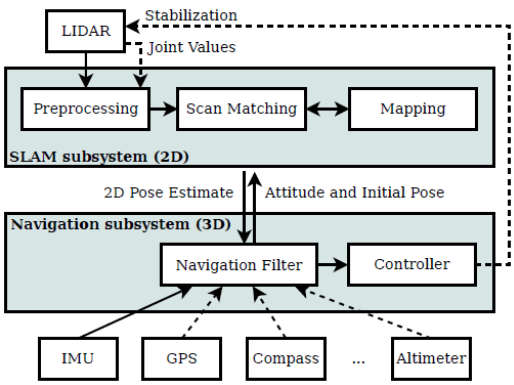
After that, the resampling process is improved using the effective number of particles to set the threshold for executing resampling process by comparing current set of particles with the target posterior. This can prevent particle depletion due to excessive resampling from the normal RBPF which also eliminate good particles in the process.

2.3.2 HectorSLAM

HectorSLAM was proposed for using in Urban Search and Rescue (USAR) application that is for indoor application but not only limited to only planar application but also 3D trajectory of the robot. The HectorSLAM is based on graph-based approach which only make use only the front-end of the system. Therefore, there is no optimization process for the HectorSLAM. The map creation is mainly depending on the scan-matching method without using the odometry data of the robot. The change of the robot pose will be recovered from the scan-matching method. Figure 2.2 shows the HectorSLAM system.

Figure 2.2

HectorSLAM and 3D State Estimation System



The scan matching method is used for pose estimation of HectorSLAM. In order to achieve the best alignment of the scans, the following cost function will be minimized.

$$\underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[1 - M\left(\mathbf{S}_i(\boldsymbol{\xi})\right)\right]^2 \tag{2.4}$$

Where ξ is the position of the system, $S_i(\xi)$ is the global coordinates of the scan endpoint and $M(S_i(\xi))$ is the map value. The change of the system position can be found by using Gauss-Newton method which gives the $\Delta\xi$ as the result.

$$\sum_{i=1}^{n} \left[1 - M\left(\mathbf{S}_i(\boldsymbol{\xi} + \Delta\boldsymbol{\xi})\right)\right]^2 \rightarrow 0 \tag{2.5}$$

2.3.3 Cartographer

This technique is one the most recent methods for lidar-based SLAM. This system was proposed in order to reduce computation requirement for loop closure and enable large map creation with real-time operating using Google's Cartographer. Cartographer is a graph-based SLAM. Solution of the SLAM is the same as HectorSLAM but with a different cost function as shown in the following equation.

$$\underset{\xi}{\operatorname{argmin}} \quad \sum_{k=1}^{K} \left(1 - M_{\text{smooth}}(T_\xi h_k)\right)^2 \tag{2.6}$$

The bicubic interpolation is used to find the map value from the pose of the robot to get a better minimization value while HectorSLAM use bilinear interpolation. For pose estimation after scan-matching on front-end, Branch and Bound will be used to find optimized pose in the search window.

2.4 Scan Matching

Scan matching is the algorithm which is widely used in lidar based SLAM. Scan matching aligns current point cloud with the previous point cloud and gives the affine transform between two point clouds. The standard scan matching is Iterative Closest Point (ICP) algorithm. ICP looks for the closest point pairs from two point clouds, current scan and reference scan. After that, the affine transformation that projects current point cloud to the previous point cloud is calculated based of following equation.

$$f\left(p_x, p_y, \omega\right) = \sum_{i=1}^{n} \left\| \mathbf{T}\left(a_i, \left(p_x, p_y, \omega\right)\right) - b_{\vartheta(i)} \right\|^2 \quad (2.7)$$

Where, T is the affine transformation, p_x and p_y are translations in x and y axis, ω is rotation that transform the current point cloud a_i to map with previous point cloud $b_{\vartheta(i)}$ when ϑ is a function that find the closest point in the previous point cloud with the current point cloud with index *i.* This error function will be minimized and gives the result of the best affine. ICP algorithm will be executed as following steps.

1. Find the correspondences points pairs between two point clouds.
2. Minimizes error function.
3. Repeat these steps until algorithm is converged when error is within the threshold.

2.5 Visual based SLAM

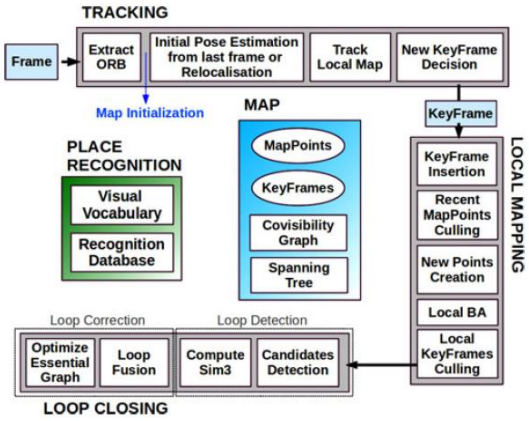
Another main sensor except lidar that commonly used is camera. Camera is capable of extracting information from the image. There are two main solutions for SLAM in case of using camera as sensor. First, feature-based method will detect prominent points in the image and track them. Another solution is direct SLAM which uses the whole image without using any features extraction.

2.5.1 Feature based SLAM

This method is depended on features detection and features extraction from the sequence of the images. The most used feature-based algorithm right now is ORB-SLAM which is able to use with all types of the camera. ORB-SLAM is based on pose graph optimization. The system of ORB-SLAM has three parallel for tracking the features in the scene, create local map and the last thread is loop closure. For pose graph optimization, g2o is used for optimization. Figure 2.3 shows the system of ORB-SLAM.

Figure 2.3

ORB SLAM syste

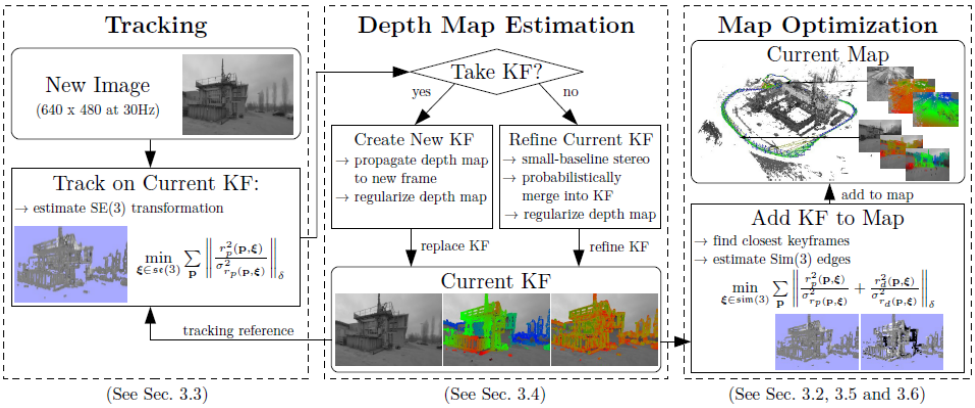


2.5.2 Direct SLAM

While feature-based SLAM only uses keypoints. This method uses the whole information in the image. Several popular techniques in direct SLAM are DTAM, LSD-SLAM, SVO and, DSO. Direct SLAM method can give a proper reconstruction of the scene unlike feature-based method that can only give sparse reconstruction. However, the downside is that in mostly cases, GPU is required and the accuracy of the localization is lower than feature-based method. The LSD SLAM is one of the best modern monocular SLAM direct method and also graph-based method as well. LSD SLAM has of three operations starting with tracking the features in the scene, estimation of depth map and loop closure. Figure 2.4 shows the LSD SLAM system.

Figure 2.4

LSD SLAM System



Even camera cost is much lower compare to LiDAR and Visual-SLAM can give a proper result, there are many drawbacks from using Visual-SLAM. The summary of advantages and disadvantages of Visual-SLAM is shown in Table 2.2.

Table 2.2
Summary of Visual- SLAM's Advantages and Disadvantages

	Visual-SLAM	
	Feature Based	**Direct**
Advantages	Light weight	Denser map than feature tracking techniques. No need for features detection
Disadvantages	Lighting of environment affects the system performance	Computation cost, GPU is needed

2.6 Visual-Lidar SLAM

The fusion between these two sensors was researched to get a better result from SLAM. There are several ways to combine lidar and camera for SLAM, such as using lidar as a depth sensor to improve visual based SLAM or, improve lidar based SLAM by using visual loop closing.

Another way is to combine result from lidar and visual SLAM. One research fused the visual and lidar measurements to improve odometry of the robot. Two maps are built separately using visual and lidar. By optimized the position estimated from two maps, the final pose estimation is improved.

Another work focuses on countering the downside of SLAM system which are features tracking lost and failure in localization. As the result, robust indoor SLAM on featureless environment is succeeded.

Another effective work performed graph optimization using specific cost function by consider both laser and features. After that, robot pose estimation can be obtained from laser and image.

CHAPTER 3
METHODOLOGY

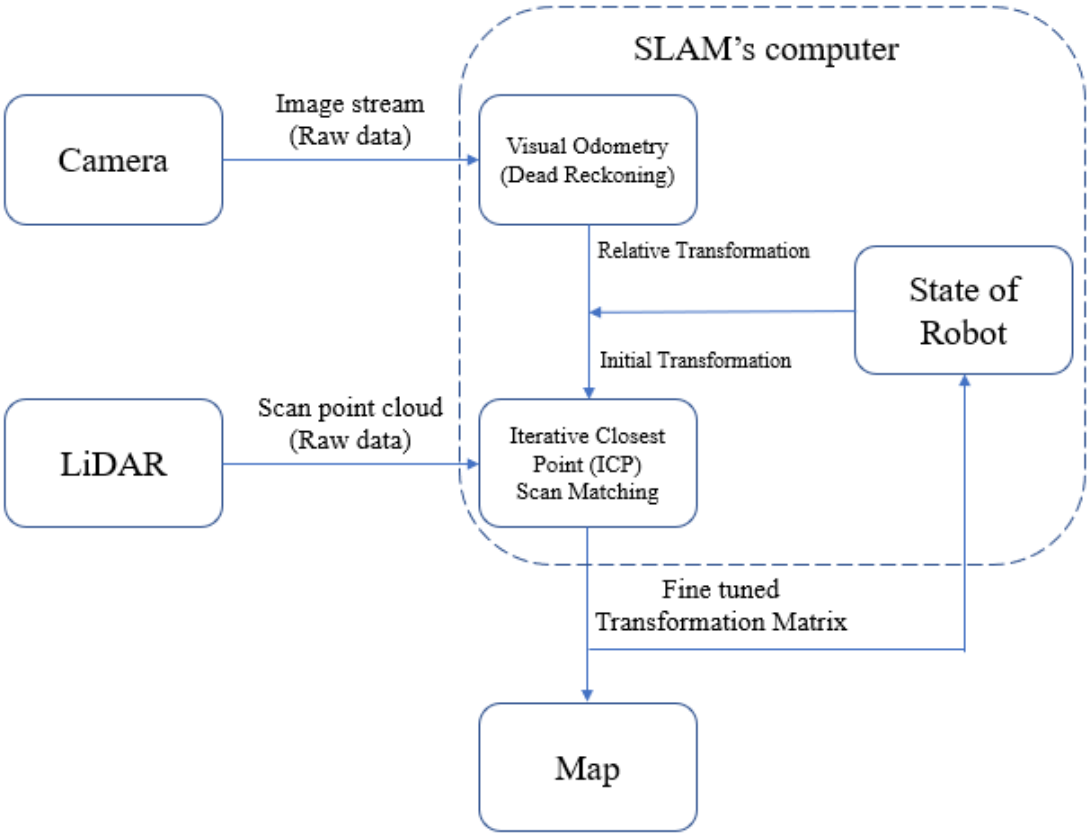
The details of methodology that is used in this thesis will be explained in this chapter. The overall structure of the system will be explained. Next, hardware design will be described. The software design which consists of dead reckoning using monocular camera, and SLAM system will be explained.

3.1 Concept

Figure 3.1 shows the overall structure of the system.

Figure 3.1

Overall Structure of System



Firstly, the camera will be used as substitution for wheel encoders for dead reckoning process. The camera will capture the scene and send to the visual odometry system in order to determine the position change (X, Y, Heading Change) between two consecutive frames. At the same time, lidar will scan the environment and input the

scans into scan matching system. Scan matching system takes in relative transformation from visual odometry as initial transformation to transform current scan to be as close as possible to previous scan then compute the scan matching algorithm which based on iterative closest point (ICP) algorithm which map current scan with the previous scan. After scan matching computation, the result from the system is the fine-tuned transformation matrix that will be used to align the scan to build the map as well as used for updating the position of the robot.

3.2 Hardware Design

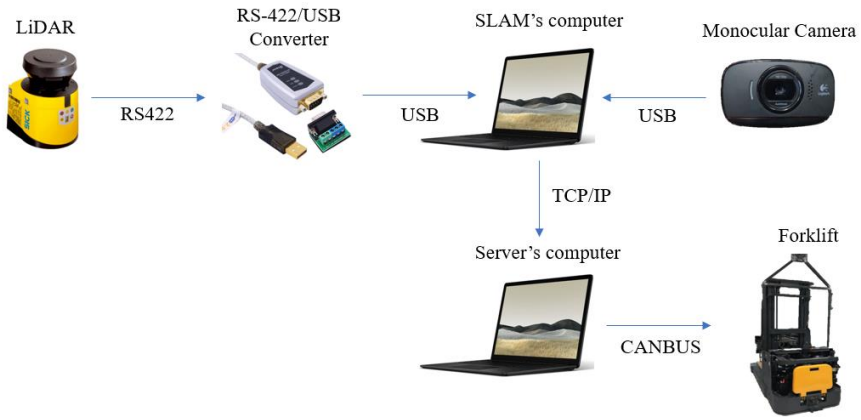
SLAM system will be consisted of two sensors which are lidar and monocular camera for map making and localization.

3.2.1 Overall Hardware Structure

In order to retrieve the scan data from lidar to the computer, the RS-422/USB converter is needed. The camera will be connected through USB connection to SLAM's computer. When the corrected position from SLAM system is out, the SLAM's computer will send the data to the server's computer embedded in the plant using TCP/IP communication. Client's computer will be used to send the desired position to the server's computer through TCP/IP communication as well. By comparing both positions from SLAM's computer and client's computer the server's computer will control and navigate the plant to the desired position through CANBUS protocol. The overall hardware structure is shown in Figure 3.2.

Figure 3.2

Overall Hardware Structure

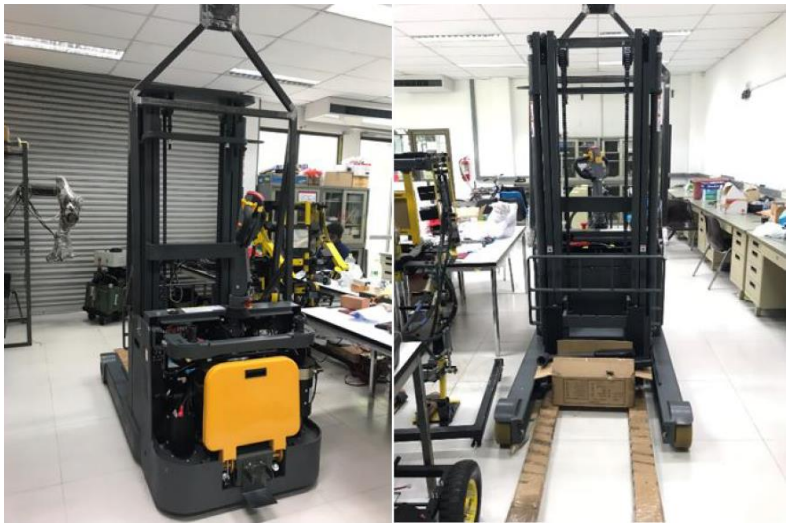


3.2.2 Plant Characteristic

The plant for this thesis will be forklift which will be used for navigation as shown in Figure 3.3.

Figure 3.3

Forklift for Navigation Task



The forklift must be able to switch between manual drive for map creation and autonomous drive when navigation and control.

3.2.3 Camera

The monocular camera will be used for determining the relative translation and rotation between two consecutive poses of the plant. The camera is shown in the Figure 3.4.

Figure 3.4

Picture of Monocular Camera

3.2.4 Lidar

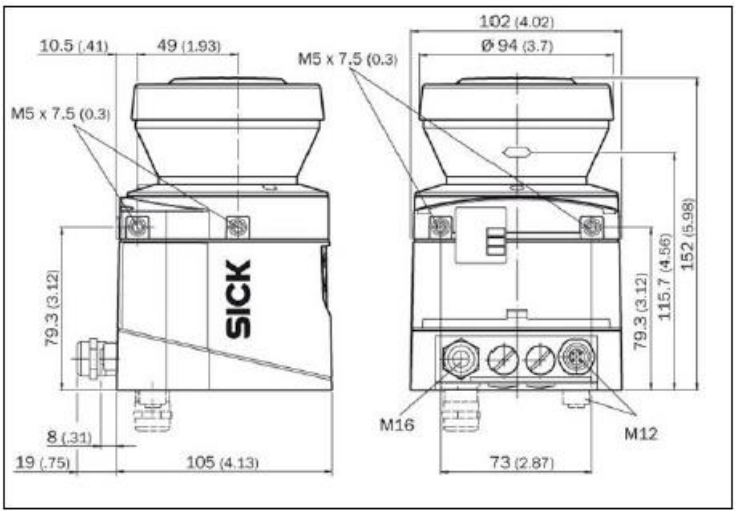
The model of lidar that is used in this thesis is SICK s300 Expert. This laser has 270-degree scanning area. Maximum range of sensor is 30 meters. Scan time of each scan revolution is 80 ms. The lidar is shown in Figure 3.5 and, the dimension of sensor is shown in Figure 3.6.

Figure 3.5

Picture of Lidar SICK s300 Expert

Figure 3.6

Dimension of Lidar SICK s300 Expert



3.3 Software Design

This SLAM system is implemented on Visual C#. The system is consisted of two main parts, Visual Odometry and SLAM. Firstly, the SLAM system is developed in SLAM's computer which lidar and camera are connected to. The camera will be used for visual odometry to find the relative transformation between two consecutive poses of the forklift. Lidar will be used for sensing the unknown environment and gives the environment in form of point cloud at each position of forklift. Then, by sending the

change of position and orientation from camera with the point cloud from lidar at these positions into the SLAM system, the system will compute for the localization of forklift and mapping. After receives location of the robot, this information will be sent to the server's computer that is used for controlling and navigating the forklift to compare with the desired position that is sent from the client's computer.

3.4 Visual Odometry System

The plant must be able to know how far it was travelled which consists of translation change and heading change. Most of the case, the wheel encoders are selected as sensor for tracking the translation and heading change. However, by implementing wheel encoders, a modification of the plant is needed. In contrast, camera can be used to track the translation and heading change as well without modification of the plant. This system is developed using .NET OpenCV which is called EMGU CV.

3.4.1 Camera Calibration

The camera calibration will be needed to get the properties of the camera. Intrinsic parameters that consist of focal length and center of the camera will be used to calculate the feature points distance and angle related to the center of the camera. Distortion parameters will be used to undistort input image. Table 3.1 shows camera parameters from calibration process.

Table 3.1
Camera Parameters

Intrinsic Parameters	
fx	0.731318846e+003
fy	0.708229696e+003
cx	3.54491786e+002
cy	2.35166125e+002

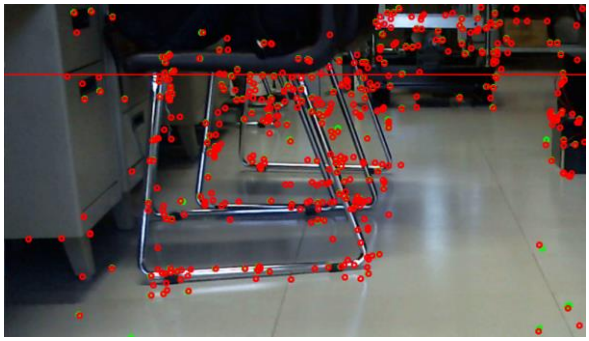
Distortion Parameters	
k1	-5.833e-003
k2	-8.3797e-002
k3	0
p1	-3.210e-003
p2	1.060e-003

3.4.2 Visual Odometry

Firstly, the input image will be undistorted by using the distortion parameter from camera calibration process to compensate lens distortion. After that, the algorithm will generate the first feature points group. The good features of the original image will be detected using function goodFeaturesToTrack() in OpenCV that used Shi-Tomasi corner detection to find the prominent corner points in the image. Then each feature points will also be undistorted and stored as history feature points. After retrieved the first set of feature points, the algorithm will take previous frame, current frame and history feature points as inputs for Optical Flow calculation to get current set of feature points. Then, current feature points will be stored as history feature points for next frame. The algorithm continues as new frame comes in. When two consecutive feature points are retrieved, the calculation of heading change and translation change can be done based on location of these feature points. Image frames will be separated into two sections, sky region and ground region. Feature points in sky region will be used to find heading change. While, feature points on ground region will be used to find translation change. Figure 3.7 shows image frame that consists of feature points in both sky region and ground region.

Figure 3.7

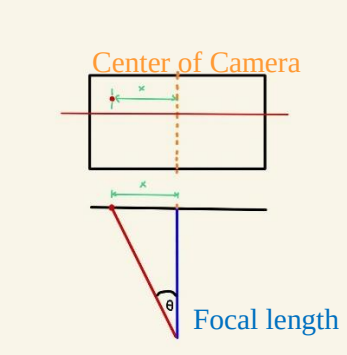
Image Frame with Feature Points in Sky (Above the Red Line) and Ground Region (Below the Red Line)



In order to find heading change of the camera, the feature points in sky region will be considered. As the depth distance of feature points in the sky region of two consecutive frames is not different that much, the different in depth will be neglected. By finding the distance in x-axis of each feature points of both frames related to the center of camera with focal length, the angle (θ) of each feature points related to center of camera can be obtained as shown in Figure 3.8.

Figure 3.8

Calculation of Feature Point's Angle related to Center of Camera



The heading can be calculated as the following equation:

$$\theta_{feature} = arctan \frac{(X_{feature} - X_{center\ of\ camera})}{Focal\ Length} \quad (3.1)$$

After each individual θ are retrieved from calculation, the angle change of each feature point can be found. The mean of all angle change will be selected as the candidate of heading change between two frames as shown in the following equation:

$$\theta_{change} = \frac{\sum_{i=0}^{n}(\theta_{feature,current} - \theta_{feature,previous})i}{n} \quad (3.2)$$

To find translation change, feature points in ground region will be considered. However, the features on the ground must be perspective transformed into bird eye view in order to retrieve real coordinate of features. The camera will be set at the specific height and angle then capture the scene. Four corners of the tile of size 600 mm x 600 mm in the image frame will be manually marked to get their coordinates. Figure 3.9 shows the image frame with marked coordinates at each tile corner.

Figure 3.9

Image Frame with Marked Coordinates at Each Tile Corner

Homography matrix can be calculated as following equation:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\text{8x9 } A \text{ matrix}} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\text{9x1 } h \text{ matrix}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad (3.3)$$

Where,

x_n is x-coordinate of n^{th}corner of the tile in the image frame.

y_n is y-coordinate of n^{th}corner of the tile in the image frame.

x'_n is actual x-coordinate of the tile.

y'_n is actual y-coordinate of the tile.

h is homography matrix.

Which can be re-written as:

$$Ah = 0 \quad (3.4)$$

To find homography matrix, Singular Value Decomposition (SVD) will be used to factorize matrix A which gives the following result:

$$A = UDV^T \quad (3.5)$$

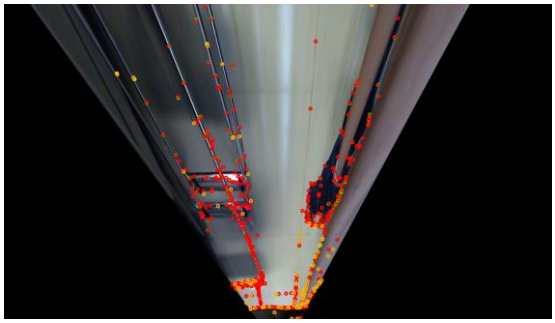
homography matrix will be equal to the last column of matrix V.

$$h = \begin{bmatrix} -1.8998 & -0.0022 & 1338.7137 \\ -0.1311 & 0.5700 & -1918.0493 \\ -3.5679 - 05 & -0.0095 & 1 \end{bmatrix}$$

The image and features after perspective transformed is shown in Figure 3.10.

Figure 3.10

Homography Transformation of Feature Points on the Ground Region



Feature points in ground region are affected by the heading change. To find pure translation change, the heading change effect will need to be eliminated. By transforming the current frame coordinate system back to previous frame coordinate system using heading change that was retrieved previously, rotation effect can be eliminated and the pure translation change of each feature points can be obtained as the following equation:

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{change}) & -\sin(\theta_{change}) \\ \sin(\theta_{change}) & \cos(\theta_{change}) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.3)$$

Mean value of translation change in both x direction and y direction will be selected as the candidate of translation change as shown in the following equation:

$$x_{change} = \frac{\sum_{i=0}^{n} (x_{corrected,current} - x_{corrected,previous})_i}{n} \quad (3.4)$$

$$y_{change} = \frac{\sum_{i=0}^{n} (y_{corrected,current} - y_{corrected,previous})_i}{n} \quad (3.5)$$

3.5 SLAM System

After received relative position of the camera, this information will be input for the SLAM system. SLAM system will be consisted of three parts. Firstly, the interprets of the raw measurement from lidar. Secondly, using information from camera as initial transformation between two poses to compute scan-matching between two consecutive point cloud in order to obtain better transformation between two poses. Thirdly, creating the map as well as localize the pose of the robot in global coordinate.

3.5.1 Interpretation of Raw Data Measurements from Lidar

In order to sense the unknown environment, SICK s300 lidar is used to scan the environment. However, this lidar has its unique structure for output raw data. The sensor will continuously send the data over to the laptop. Figure 3.11 shows the structure of measurement.

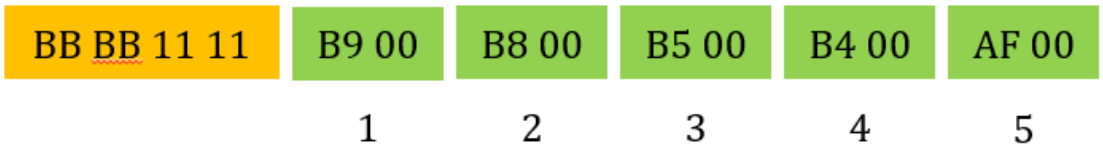
Figure 3.11

Structure of Raw Measurements from Lidar

The information starts with a block number 1 which contains of all zeros. The second block is used for defining the current configuration of the sensor. There are three modes of sensing in this sensor, I/O data, distance data, reflector data. For I/O data, the third block will be started with "AA AA 11 11". Reflector data will be started with "CC CC 11 11". While in this case, only the distance of the environment will be considered. Therefore, second block will be started with "BB BB 11 11". The third block is the most important block which contains the distance of each scan varies from -45 degree to 225 degree. While connecting sensor to laptop, sensor will send the data continuously to the laptop. The data can be acquired by following the data structure mentioned previously. Firstly, looks up for the first block to detect the starting of the scan round. Then using BB BB 11 11 as the starting index for reading scan data. This lidar has 0.5 degree of scan resolution and 270 degrees scan range which means, the scan end points will be 541 points and equal to 2164 data words. In order to retrieve the measurement data from data words, converting from hexadecimal to decimal will be processed. The example below shows the converting from data words to measurement data. Figure 3.12 shows the example of data words from the sensor.

Figure 3.12

Data Words from Sensor and Interpretation of the Words into Measurement



(Measured data no.1) B9 00 measurements at -45 degree is 0x00B9
Convert into bit: 0000 0000 1011 1001
Measurements are varying from bit 0 to bit 12
(128 cm + 32 cm + 16 cm + 8 cm + 1 cm = 185 cm)

(Measured data no.2) B8 00 measurements at -44.5 degree is 0x00B8
Convert into bit: 0000 0000 1011 1000
Measurements are varying from bit 0 to bit 12

(128 cm + 32 cm + 16 cm + 8 cm = 184 cm)

(Measured data no.3) B5 00 measurements at -44 degree is 0x00B5
Convert into bit: 0000 0000 1011 0101
Measurements are varying from bit 0 to bit 12
(128 cm + 32 cm + 16 cm + 4 cm + 1 cm = 181 cm)

(Measured data no.4) B4 00 measurements at -43.5 degree is 0x00B4
Convert into bit: 0000 0000 1011 0100
Measurements are varying from bit 0 to bit 12
(128 cm + 32 cm + 16 cm + 4 cm = 180 cm)

(Measured data no.5) AF 00 measurements at -43 degree is 0x00AF
Convert into bit: 0000 0000 1010 1111
Measurements are varying from bit 0 to bit 12
(128 cm + 32 cm + 8 cm + 4 cm + 2 cm + 1 cm = 175 cm)

As the data is the distance from sensor to the obstacle at that specific scan angle. Therefore, converting of the distance with scan angle to coordinate x and y must be processed for mapping. The example of converting will be shown below.

(Data block no.1) Measured distance: 185 cm at -45.0
x : 185 * cos (-45.0) = 130.81 cm
y : 185 * sin (-45.0) = -130.81 cm

(Data block no.2) Measured distance: 184 cm at -44.5 degree
x : 184 * cos (-44.5) = 131.24 cm
y : 184 * sin (-44.5) = -128.97 cm

(Data block no.3) Measured distance: 181 cm at -44.0 degree
x : 181 * cos (-44.0) = 130.20 cm
y : 181 * sin (-44.0) = -125.73 cm

(Data block no.4) Measured distance: 180 cm at -43.5 degree
x : 180 * cos (-43.5) = 130.57 cm
y : 180 * sin (-43.5) = -123.90 cm

(Data block no.5) Measured distance: 175 cm at -43.0 degree
x : 175 * cos (-43.0) = 127.99 cm
y : 175 * sin (-43.0) = -119.35 cm

3.5.2 Scan Matching Algorithm

While visual odometry is capable of finding relative transformation between two consecutive poses of the robot, it contains a lot of error in the process. If localization process only depends on visual odometry alone, the error will keeps accumulating until diverge from the true path of the robot. Scan matching algorithm will be used to corrected the error from visual odometry by aligning two consecutive point cloud that are retrieved from lidar with initial transformation between two consecutive poses from visual odometry. After scan matching is done, corrected transformation both rotation and translation can be obtained. This transformation will be used to transform next point cloud to align with previous point cloud and also used updating the pose of the robot.

Iterative closest point scan matching algorithm consists of three processes. Firstly, apply transformation from Visual Odometry to current point cloud which will map the current point cloud to the previous point cloud. Secondly, the algorithm will find the pair of each point in current point cloud with previous point cloud using Euclidean distance. This method can be easily done through brute force, by finding the distance of one point of next point cloud with all of points in previous point cloud, the nearest point will be selected as the pair. Thirdly, these point pairs will be aligned together each iteration until overall error between two point clouds is within the threshold. The transformation between these two point clouds will be the corrected version of transformation between two poses as well. However, using brute force method requires huge computation resource. The binary search K-D tree will be applied to deal with the computation process in order to reduce Euclidean distance computation of unlikely points pair.

3.5.2.1 **Correspondences Points Finding using K-Dimensional Tree.**

K-Dimensional Tree (K-D tree) is a binary search tree that will divide the previous point cloud into section and build the search tree. When the current point cloud is inputted into ICP algorithm, each point will be searched through the tree of reference point cloud. The building of the search tree follows these steps in the following example.

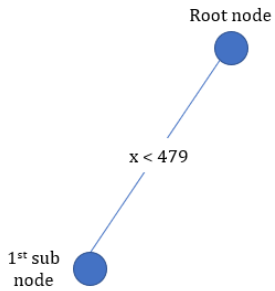
Given the point cloud of the following points.

(70,721), (207,313), (343,858), (479,449), (615,40), (751,177), (888,585)

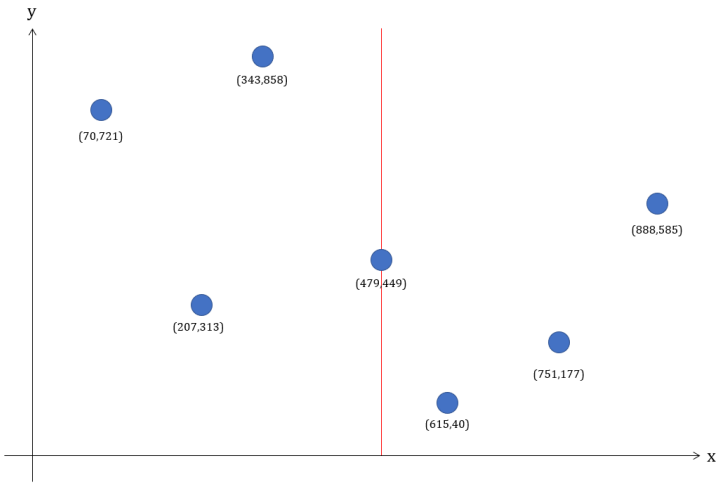
1. Arrange all reference points in x coordinate.
2. Find median point in x coordinate, which is (479,449).
3. Split reference point cloud into left and right using median point as center. Left side contains points that have x value less than median point. Right side contains points that have x value greater or equal to median point.
4. Consider points that have x value less than 479 which are (70,721), (207,313), And (343,858). The first sub node and the reference point are shown in Figure 3.13.

Figure 3.13

First Sub Node of the Search Tree



The Reference Point Divided into Section

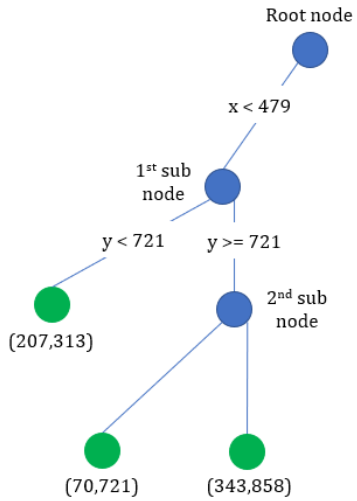


5. Consider left hand section, arrange these three points again in y direction. The median point this time is (70,721).
6. Split into upper and lower section using median point as center. The lower section contains (207,313) while the upper section contains (70,721) and (343,858).

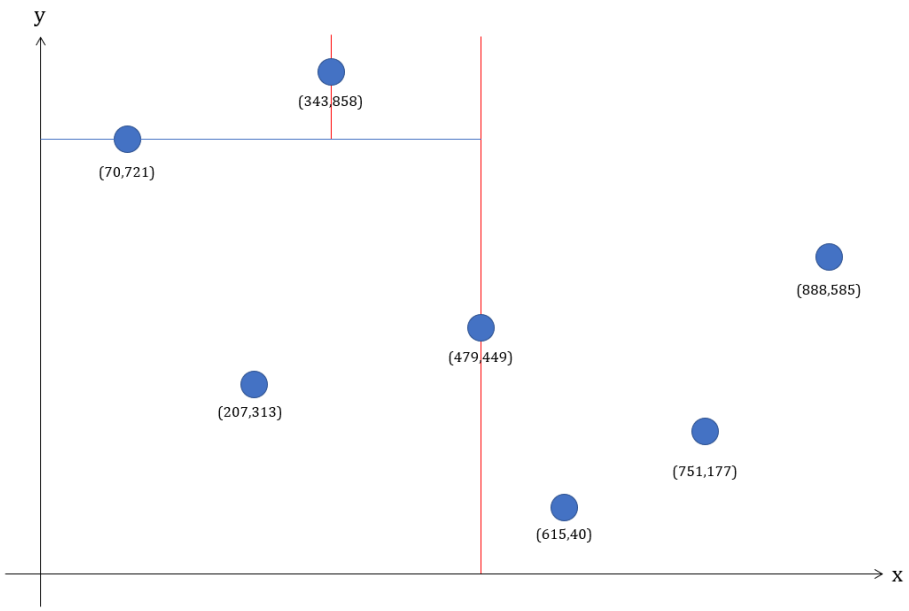
7. The lower section only contains one point. Consider upper section and split section in x direction around median which is (343,858). The left part of the search tree is now completed. Figure 3.14 shows the left section of the search tree and the reference point in section.

Figure 3.14

Left Section of the Search Tree



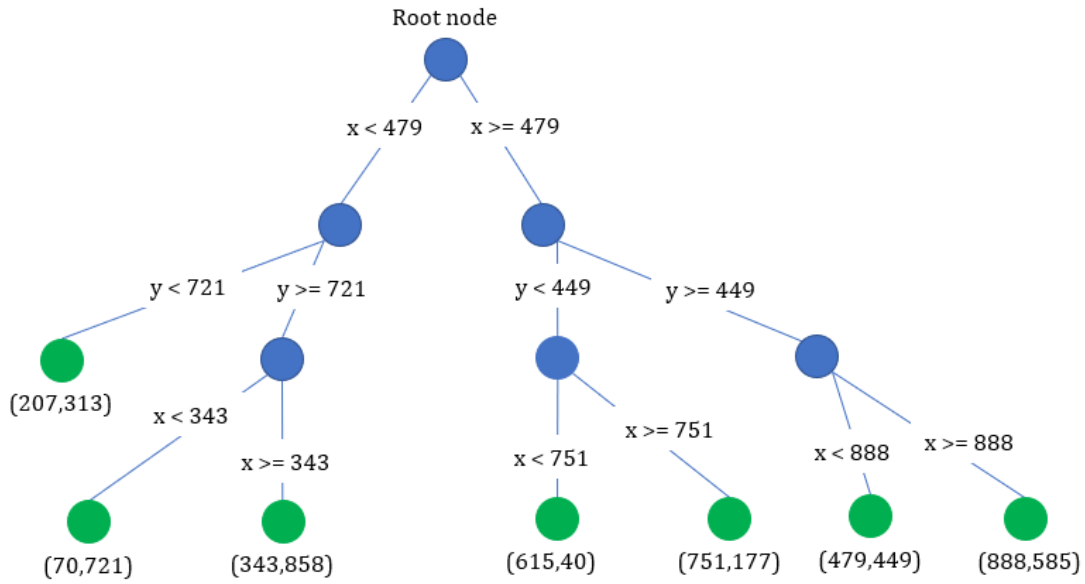
The Reference Point Divided into Section



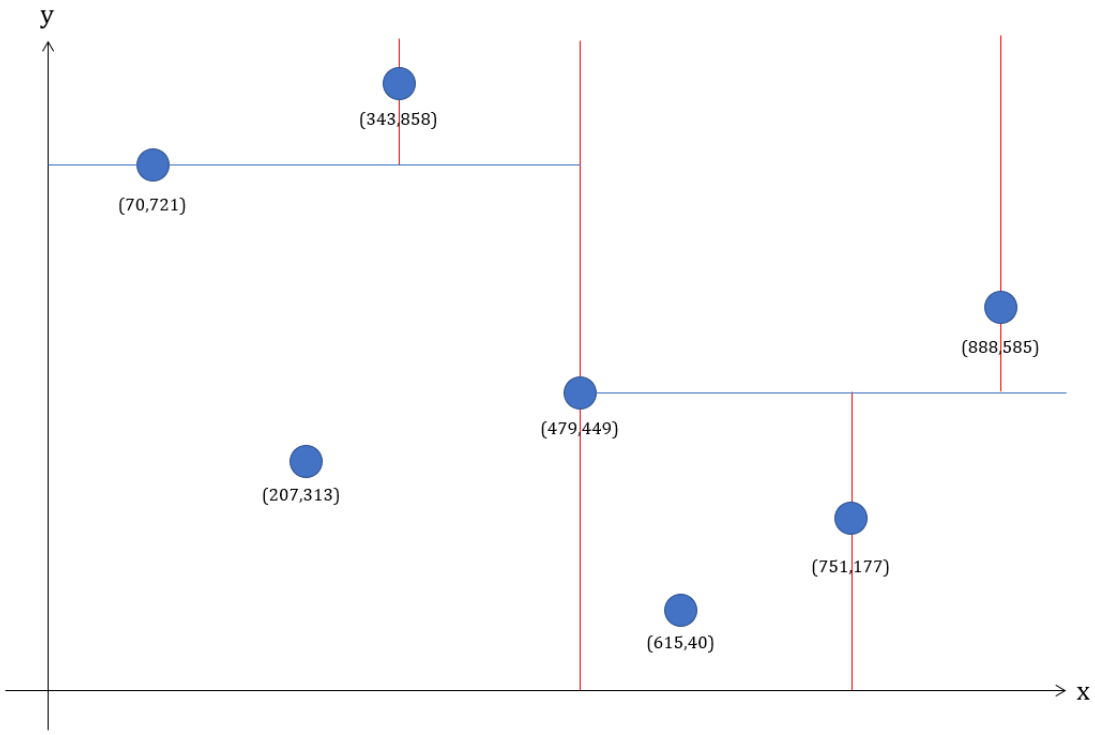
8. Repeat all the previous step recursively to all the points switching between x and y directions until each section contains only one point (exclude median point). The completed search tree is shown in Figure 3.15.

Figure 3.15

Completed K-D Tree



The Reference Point Divided into Section

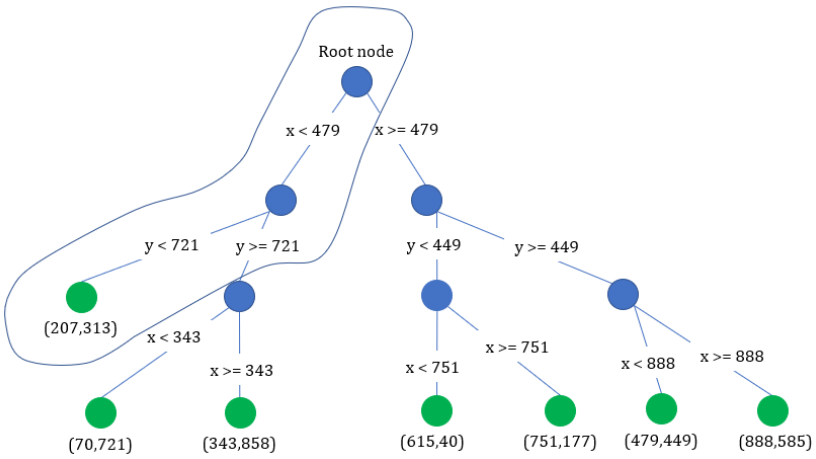


After building K-D tree, the search point will be inputted to find the nearest neighbor in the search tree. For example, to find the nearest point of (438,681) from the reference point cloud. The steps of finding the closest point are as follow.

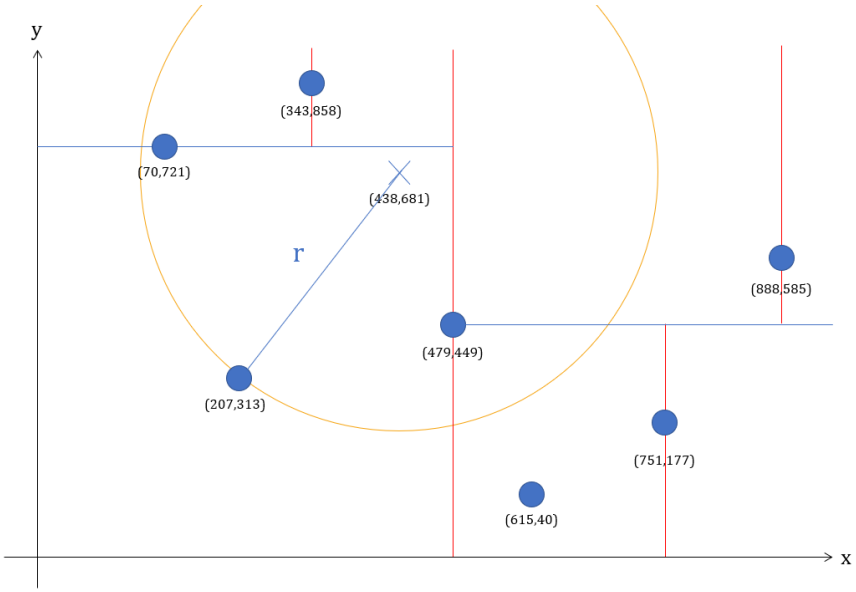
1. Starting with the root node, x value of search point is less than 479. Therefore, this takes the left section. Next, y value is less than 721. After the leaf cell is reached, compute distance to every point in that leaf and find the closest point. In this case, (207,313) gives the nearest distance to the search point (r). As shown in Figure 3.16.

Figure 3.16

Searching for Nearest Neighbor



Search Point with Nearest Neighbor with Radius (Distance from Search Point to Nearest Point)



2. Euclidean distance will be computed to find the distance between two points as shown in the following equation:

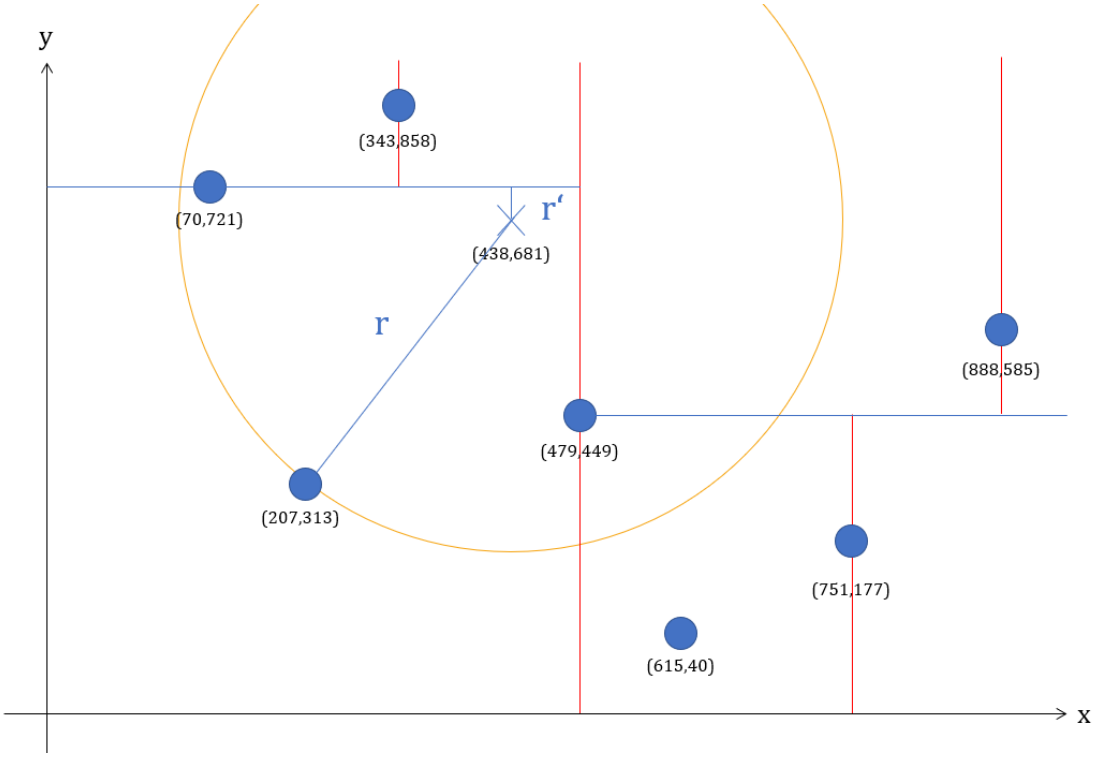
$$d(x,y)^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 \qquad (3.6)$$

When (x_1, y_1) is the coordinate of the first point and (x_2, y_2) is the coordinate of the second point.

3. After we obtained the distance to the nearest neighbor (r), we need to check the distance perpendicular from search point to the other adjacent section that is within the r value that we did not visit and call this distance as r' value. If this perpendicular distance is shorter than the nearest distance that we obtained. There is a chance that that section may contain closer point. The perpendicular distance to the adjacent section is shown in Figure 3.17.

Figure 3.17

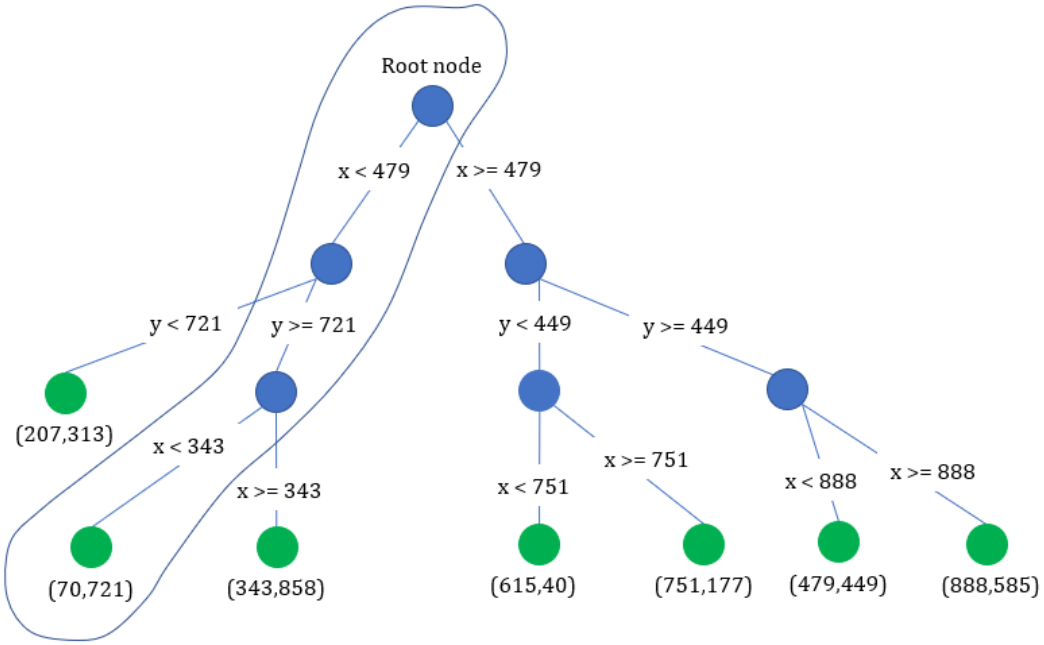
Perpendicular Distance (r') from Search Point to Adjacent Section



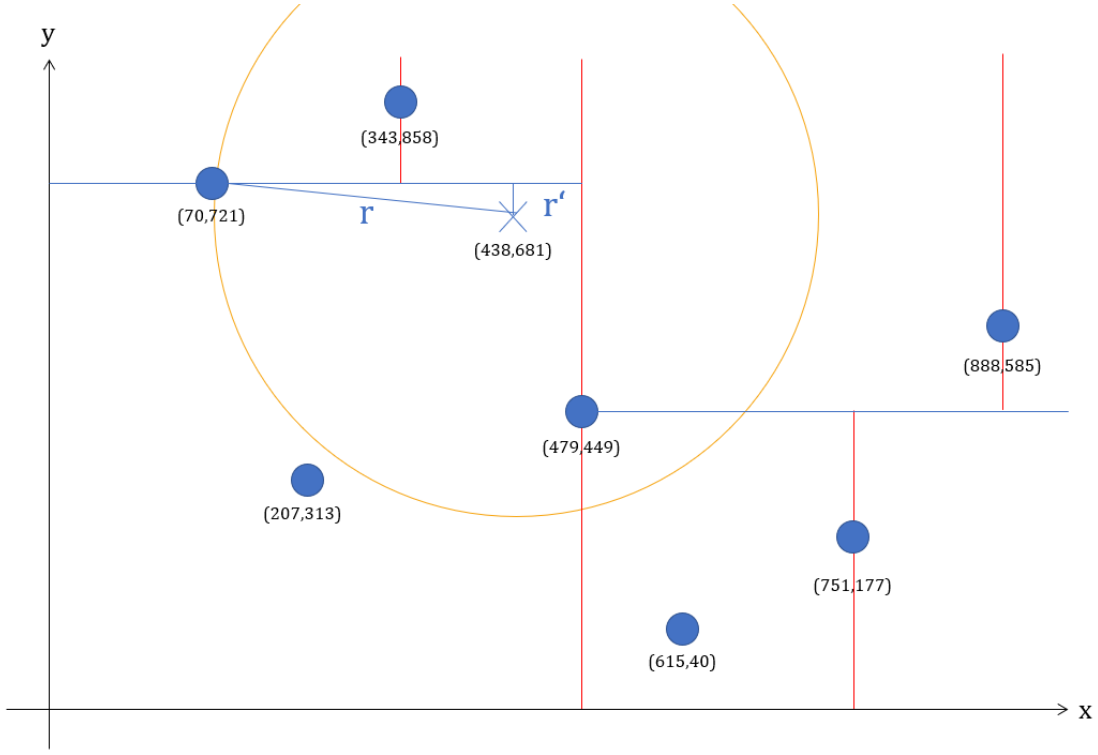
4. Check the distance of each point in the adjacent section with the search point as previous step. If the new point is closer than the previous point, replace the old point with the new one. Figure 3.18 shows the searching process through the K-D tree until the closest point is found.

Figure 3.18

The Searching Process through the K-D Tree until the Closest Point is found. The Search Tree if r' is less than best r. (Step 1)

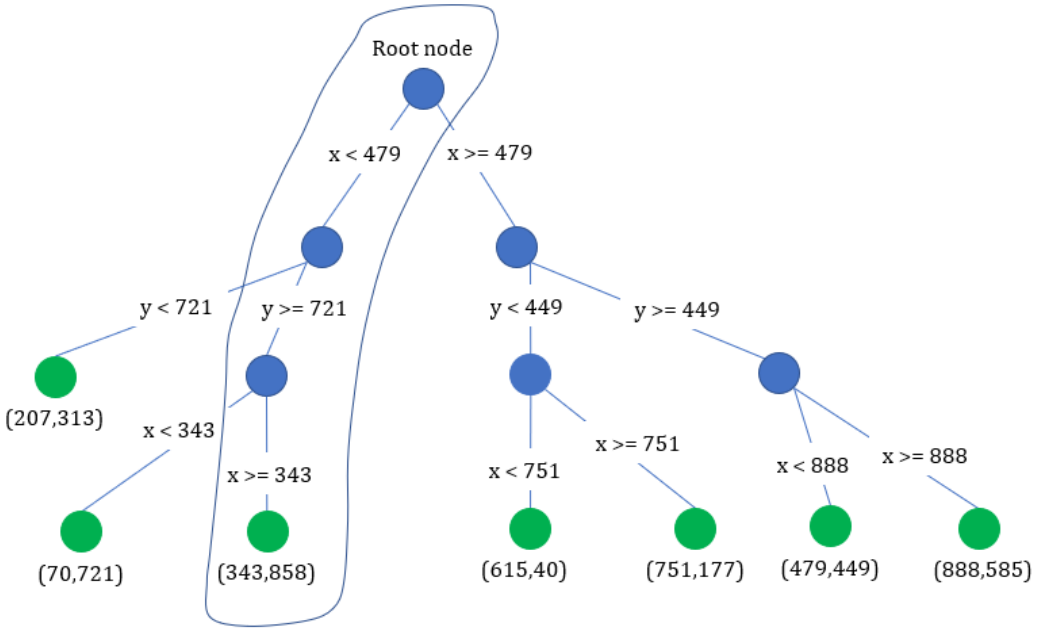


Search Point with New Nearest Neighbor. (Step 1)

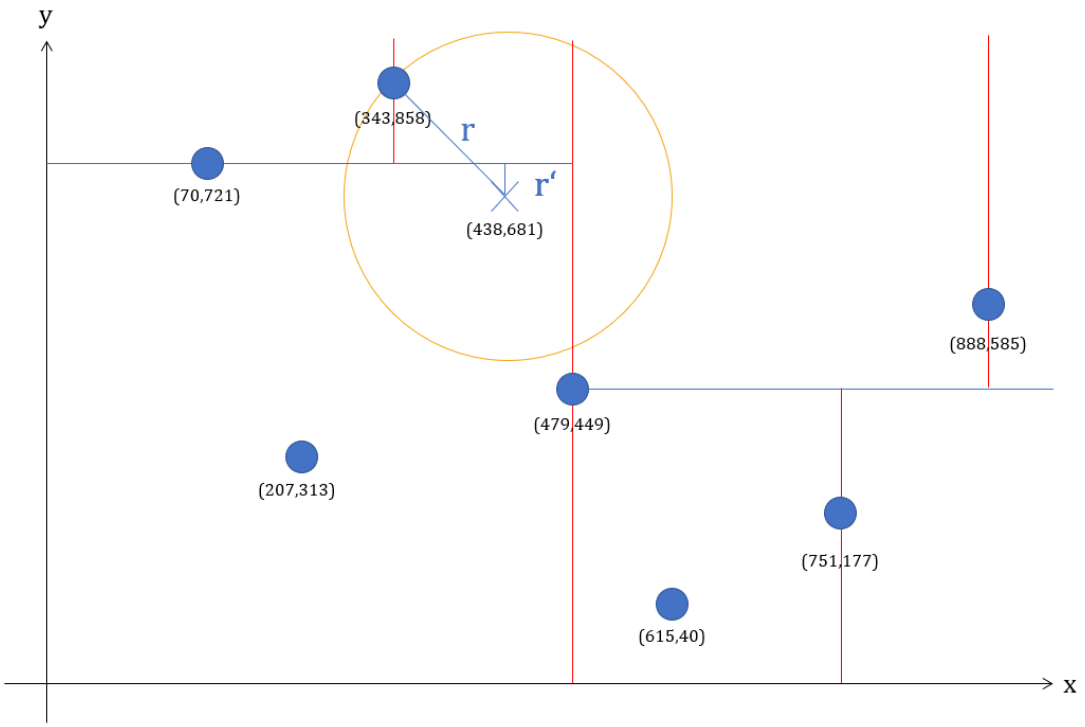


The r' value is still less than the new r value. Therefore, recurse back again to another section.

The Search Tree after r' is less than best r. (Step 2)

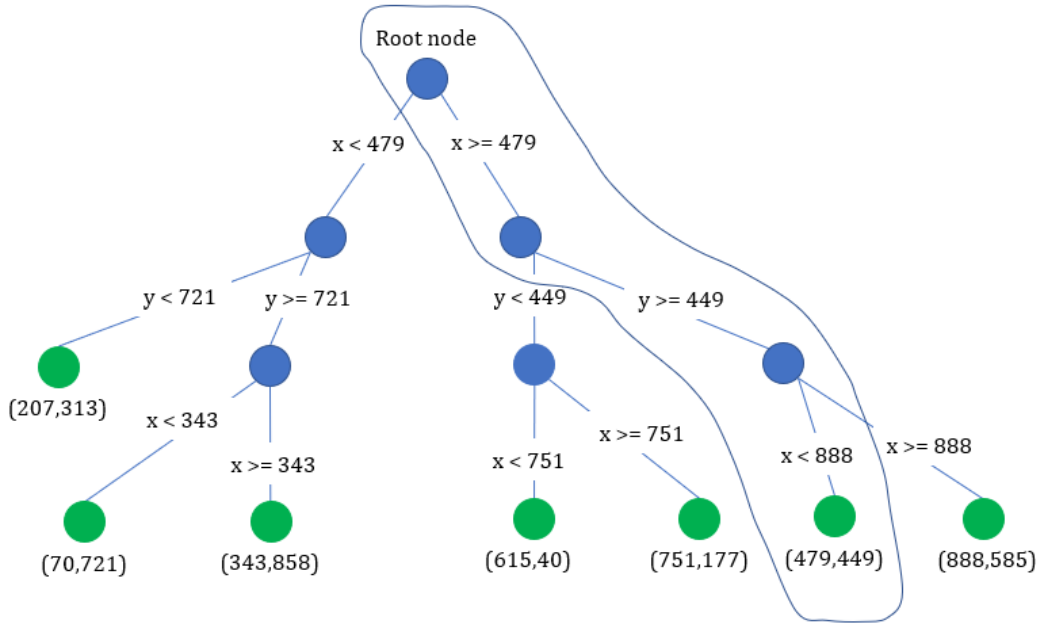


Search Point with New Nearest Neighbor. (Step 2)

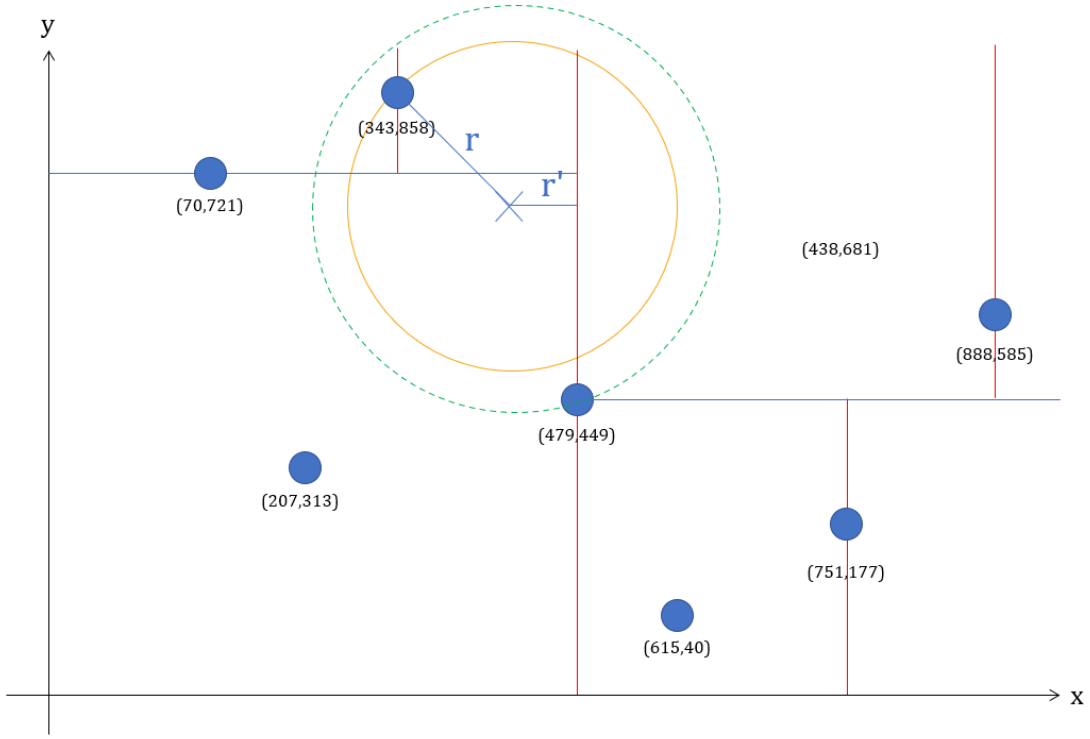


Now, all the points in the upper section are searched. Therefore, the new r' is calculated based on right-handed section that still overlap with the latest radius.

After Reaching the Best Possible Solution in the Left-Handed Side of the Tree. The Right Section is Searched (Step 4).



Search Point with New Nearest Neighbor. (Step 4)



However, the new r distance is still not the best. Therefore, the closest point is (343,858). And the searching in now completed.

All these steps will be repeated until all nearest neighbors of the current search point cloud are found. The result nearest neighbor points will be stored and used in Iterative Closest Point algorithm.

3.5.2.2 Determining the Best Alignment Between Two Point Clouds.

After finding the correspondences points, the transformation between these two point clouds will be calculated using the following error function.

$$E(R,t) = \sum_{(i,j)\in N} \| \ q_i - Rp_j - t \ \|^2 \tag{3.7}$$

Where,

$E(R,t)$ is the error function of rotation and translation matrix to be minimized.

N is the number of points in point cloud.

p is the current point cloud.

q is the point cloud of correspondences points that is retrieved from the previous step and will be considered as reference point cloud in ICP algorithm.

R is the rotation matrix that transform current point cloud to map with correspondences point cloud.

t is the translation matrix that transform current point cloud to align with correspondences point cloud.

3.5.2.2.1 Finding Rotation. By taking in two point clouds as inputs, the current point cloud is affected by both translation and rotation. To find the rotation, the translation can be neglected by subtracting with its own centroid as shown in the following equations. New coordinate of two point clouds will be related to the same reference which is origin point (0,0).

$$p' = p - Centroid(p) \tag{3.7}$$

$$q' = q - Centroid(q) \tag{3.8}$$

Cross-covariance matrix (W) between these point clouds will be computed.

$$W = \sum_{(i,j)\in N} q'_i p'^{\mathsf{T}}_j \tag{3.7}$$

Singular Value Decomposition (SVD) will be used to decompose this cross-covariance matrix.

$$W = UDV^{\mathsf{T}} \tag{3.8}$$

Rotation matrix can be calculated as follows.

$$R = VU^{\mathsf{T}} \tag{3.9}$$

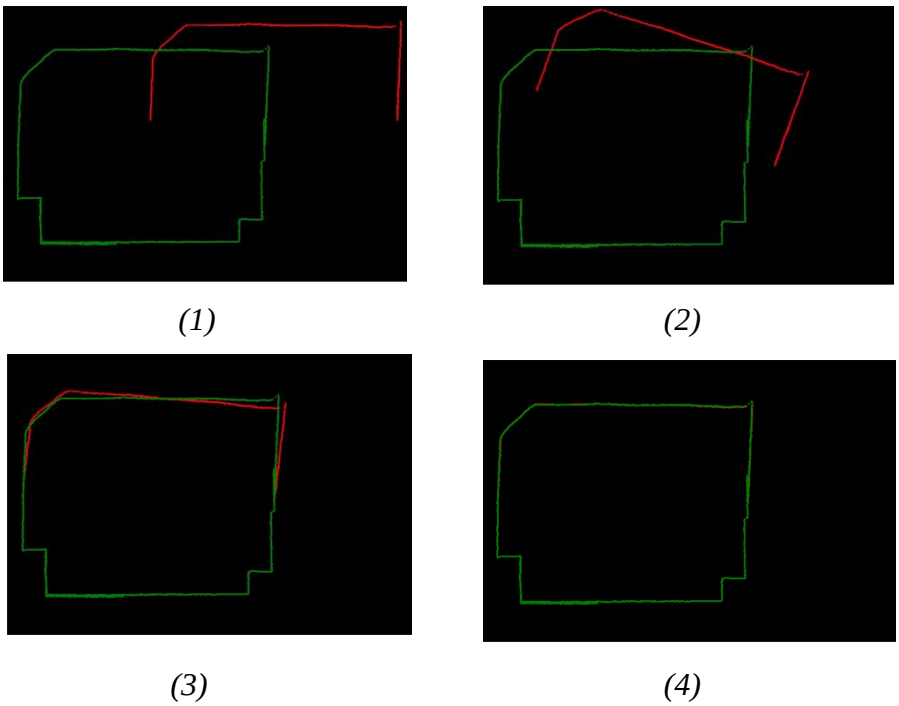
3.5.2.2.2 Finding Translation. The current point cloud will be transformed using rotation matrix. New centroid of current point cloud will be calculated again. By subtract new centroid of current point cloud from centroid of previous point cloud. The translation can be obtained.

$$t = Centroid(q) - R * Centroid(p) \quad (3.10)$$

3.5.2.2.3 Error Computation. After obtaining of the transformation both rotation and translation, the error function as mentioned in equation (3.7) will be computed. At first, the error will be set to be maximize. When ICP is computed each time, the current error will be compared with the previous minimum error. If the current error is less than previous best but still more than threshold, these processes from determining the correspondences points will be repeated. The final transformation matrix will be used to aligned the current point cloud to the reference point cloud. The Figure 3.19 shows the alignment of current point cloud to the reference point cloud.

Figure 3.19

Alignment of Current Point Cloud (Red) to Reference Point Cloud (Green)



(1) *(2)*

(3) *(4)*

3.5.3 Localization

In order to update the system position, the transformation matrix from ICP algorithm will also be used to update the system position each time ICP is completed. The robot state updating is as following:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Where,

x is the initial x coordinate of system.

y is the initial y coordinate of system.

x' is the x coordinate of robot after transformed by transformation matrix from ICP.

y' is the y coordinate of robot after transformed by transformation matrix from ICP.

r is the rotation component of the rotation matrix from ICP.

t is the translation component of the translation matrix from ICP.

3.5.4 Mapping

The point cloud that is scanned by lidar will be transformed by transformation matrix from ICP algorithm by aligning current scanned point cloud with reference point cloud as shown in the following equation.

$$\begin{bmatrix} p'_{1x} & \cdots & p'_{nx} \\ p'_{1y} & \cdots & p'_{ny} \\ 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{1x} & \cdots & p_{nx} \\ p_{1y} & \cdots & p_{ny} \\ 1 & \cdots & 1 \end{bmatrix}$$

Where,

p_{nx} is the x coordinate of the n^{th} point of the current point cloud.

p_{ny} is the y coordinate of the n^{th} point of the current point cloud.

p'_{nx} is the x coordinate of the n^{th} point of the transformed current point cloud.

p'_{ny} is the y coordinate of the n^{th} point of the transformed current point cloud.

The point cloud that is already transformed by ICP algorithm will be stored. These set of point clouds will be used to build the map of the environment which will be used as the global map by assigning the first scanning position as zero position ($x = 0\,, y = 0\,, \theta = 0$).

3.5.5 Re-Localization

When starting the system, the map from the previous process will be used as the input for re-localization. The system will compare the current scan from lidar with reference point cloud which is the map to determine where the current position is. However, the ICP algorithm may result in local minimum due to no prior knowledge of where it is. The pose of the robot with different heading angle will be generated. In this case, the 72 poses with the heading varies from 0 degree to 355 degree (5 degrees difference from each pose) will be generate to give the initial transformation of current point cloud. The ICP algorithm will be computed using the current point cloud in each heading with the map until the minimum error from ICP can be achieved. The result transformation matrix then will be used to update the position of the system and transform the current point cloud into corrected position. To summarize, the re-localization can be described as follows:

1. Read the current scanned point cloud.
2. Rotate the current point cloud 5 degrees starting from 0 degree until 355 degrees.
3. Each time the point cloud is rotate by 5 degrees, execute the ICP algorithm.
4. The minimum error that can be obtained from all ICP executions is the best possible location of the system.

3.6 Navigation

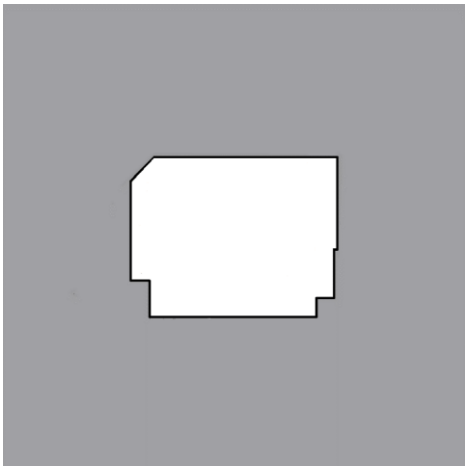
After the system is capable of localize itself, the next step is to implement the system to navigate the plant. The SLAM system will be connected to the plant which in this case is forklift. The forklift is modified and the program to receive location of the system and to drive the forklift is installed in the embedded PC that is installed in the forklift. The command to drive the forklift will be sent through CAN BUS which consists of direction of steering and speed of forklift. To receive location from SLAM system, the TCP/IP communication is used to send location by connecting embedded system to the SLAM system laptop using LAN. The TCP listener is set in the embedded PC as the server. The SLAM system laptop will act as TCP client and will continuously send the location of the system which consists of x, y and heading to the server. The forklift will be navigated until reaching the final waypoint.

3.7 Experiment Design

The room for experiment has the size of 1000 cm x 720 cm in the rectangle shape. Each corner of the room will be modified to be unique from each corner in order to help re-localization system able to recognize the direction of the system, or the system will be unable to recognize the position due to similar feature on the map. Figure 3.20 shows graphical map of the environment.

Figure 3.20

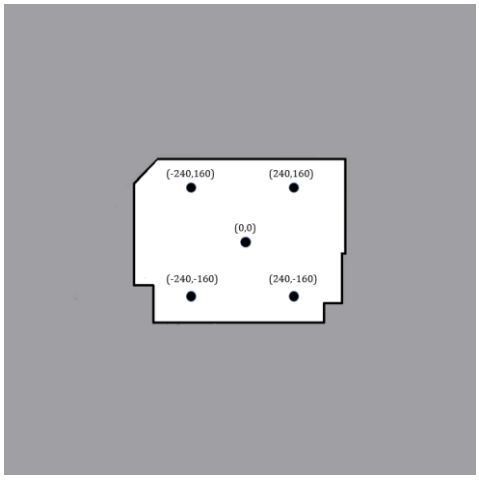
Graphical Map of the Testing Area



The center of the room is set as origin when facing upright in the map ($x = 0\,, y = 0\,, \theta = 0$). Figure 3.21 shows reference coordinates that are set for testing the re-localization system.

Figure 3.21

Reference Coordinates in the Map for Testing Re-Localization System



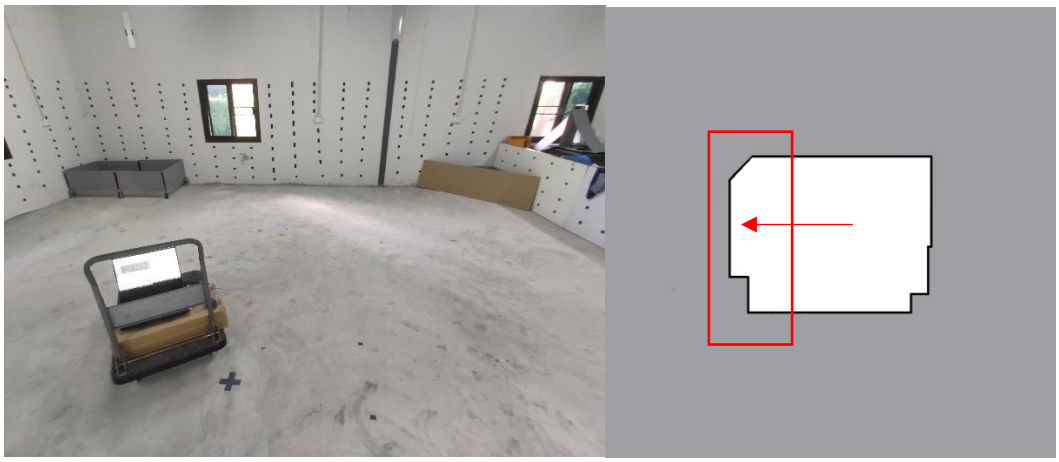
The real environment is shown in Figure 3.22.

Figure 3.22

Real Environment of the Testing Area

Real Environment of the Testing Area (Heading: 0 Degree)

Real Environment of the Testing Area (Heading: 90 Degree)



Real Environment of the Testing Area (Heading: 180 Degree)

Real Environment of the Testing Area (Heading: 270 Degree)

The testing conditions of re-localization system are shown in the Table 3.2.

Table 3.2

Testing Conditions for Re-Localization System

Test Point	Ground Truth Position			Distance to wall
	x	y	heading	
1	-240	160	45	Near
2	-240	-160	135	
3	240	-160	225	
4	240	160	315	
5	-240	160	225	Far
6	-240	-160	315	
7	240	-160	45	
8	240	160	135	
9	0	0	0	Medium
10	0	0	90	
11	0	0	180	
12	0	0	270	

There are total of 12 conditions for testing which are divided mainly into 3 groups based on distance of sensor facing to the wall, near, far and medium. Each group will be tested to localize the system in 4 different corners of the room.

To test the navigation system, the forklift will be kidnapped to any place in the map and will be navigated to two waypoints. The forklift must past the first waypoint and then navigate itself to reach the second waypoint.

The testing conditions of navigation are shown in the Table 3.3. and Figure 3.23.

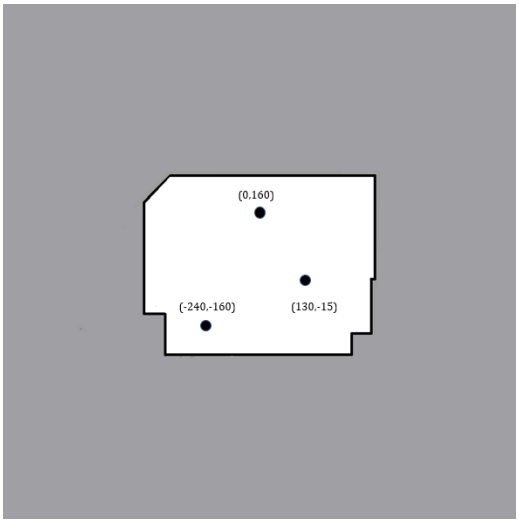
Table 3.3

Testing Conditions for Navigation

Waypoint	Ground Truth Position	
	x	y
1	130	-15
2	-240	-160

Figure 3.23

Reference Coordinates in the Map for Testing Navigation System.



The evaluation of SLAM system will be based on comparison between RMSE of the hybrid lidar and vision-based SLAM system and RMSE of the lidar-only localization system using ICP algorithm without the aid from visual odometry which, the scans between poses of the robot will be used to find the transformation between two poses without knowing the relative transformation from dead reckoning system.

CHAPTER 4
RESULT AND DISCUSSION

4.1 Overview

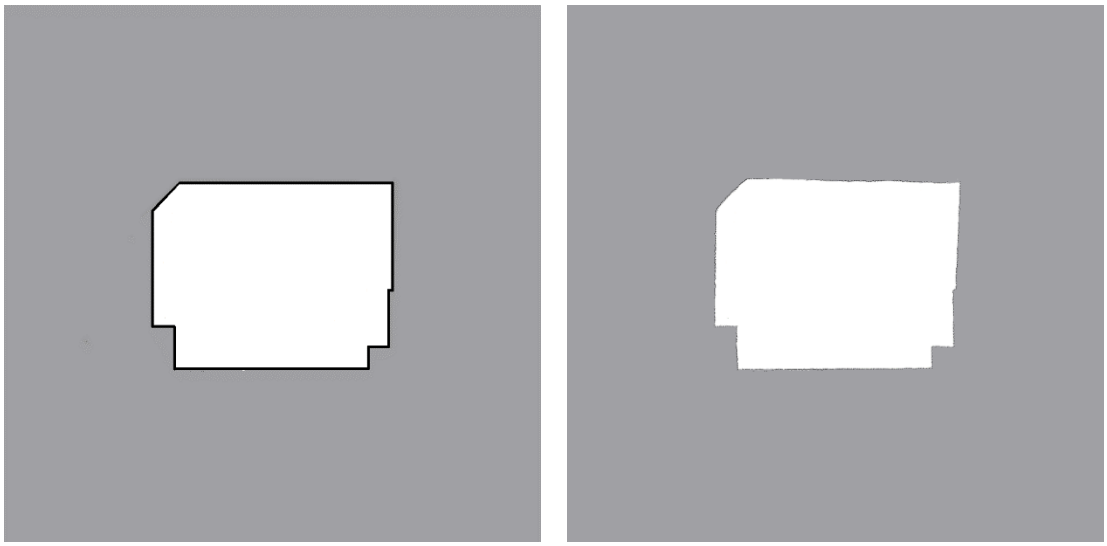
The map result of hybrid lidar and vision-based SLAM and the re-localization result will be shown in this section including evaluation of the system.

4.2 Mapping

The result for hybrid lidar and vision-based SLAM is the map of the environment that was scanned from the system by placing system at center of the map in order to set this location as position of $(x = 0\,, y = 0\,, \theta = 0)$. Figure 4.1 shows the comparison of graphical map and scanned map.

Figure 4.1

Comparison of Graphical Map (Left) and the Scanned Map (Right)



4.3 Evaluation of Re-Localization System

To evaluate the re-localization system, the map retrieved from SLAM system will be given to the system. By comparing the current scan point cloud with the stored map point cloud, the system will be able to localize itself in the given map. Root Mean Square Error (RMSE) will be used to evaluate the error of the localization of the system. The calculation of RMSE is expressed in the following equations:

$$RMSE, X = \sqrt{\frac{1}{n}\sum_{i-1}^{n}\left(X_{c,i} - X_{e,i}\right)^2} \quad (4.1)$$

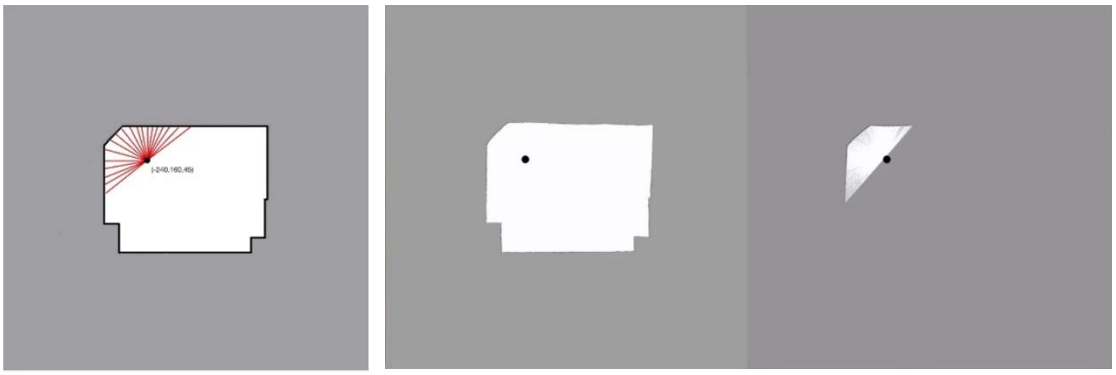
$$RMSE, Y = \sqrt{\frac{1}{n}\sum_{i-1}^{n}\left(Y_{c,i} - Y_{e,i}\right)^2} \quad (4.2)$$

$$RMSE, \theta = \sqrt{\frac{1}{n}\sum_{i-1}^{n}\left(\theta_{c,i} - \theta_{e,i}\right)^2} \quad (4.3)$$

Firstly, the system will be placed at position of ($x = -240$, $y = 160$, $\theta = 45$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.2 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.2

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the First Test Point ($\boldsymbol{x = -240}$, $\boldsymbol{y = 160}$, $\boldsymbol{\theta = 45}$)



(a) *(b)*

The Table 4.1 shows the re-localization result of first test point.

Table 4.1

Re-Localization Result of First Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
1	-240	160	45	-246.8	162.6	48.7

The Table 4.2 shows the RMSE between result of first test point and actual position.

Table 4.2

RMSE Between Result of First Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
1	6.8	2.6	3.7	7.28011

Second test point, the system will be placed at position of ($x = -240$, $y = -160$, $\theta = 135$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.3 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.3

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Second Test Point ($\boldsymbol{x = -240}$, $\boldsymbol{y = -160}$, $\boldsymbol{\theta = 135}$)

(a) *(b)*

The Table 4.3 shows the re-localization result of second test point.

Table 4.3

Re-Localization Result of Second Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
2	-240	-160	135	-230.2	-161.4	134.5

The Table 4.4 shows the RMSE between result of second test point and actual position.

Table 4.4

RMSE Between Result of Second Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
2	9.8	1.4	0.5	9.89949

Third test point, the system will be placed at position of ($x = 240$, $y = -160$, $\theta = 225$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.4 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.4

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Third Test Point ($\boldsymbol{x = 240}$, $\boldsymbol{y = -160}$, $\boldsymbol{\theta = 225}$)

(a) *(b)*

The Table 4.5 shows the re-localization result of third test point.

Table 4.5

Re-Localization Result of Third Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
3	240	-160	225	248.9	-164.1	230.7

The Table 4.6 shows the RMSE between result of third test point and actual position.

Table 4.6

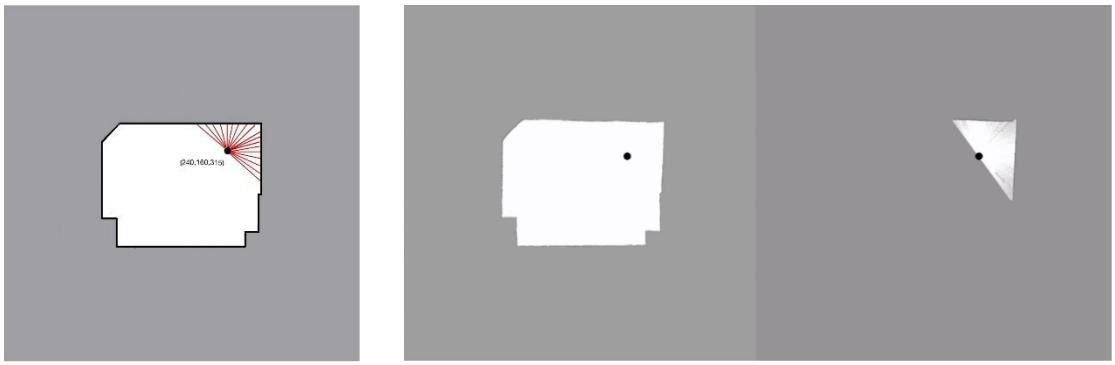
RMSE Between Result of Third Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
3	8.9	4.1	5.7	9.79898

Fourth test point, the system will be placed at position of ($x = 240$, $y = 160$, $\theta = 315$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.5 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.5

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Fourth Test Point ($\boldsymbol{x = 240}$, $\boldsymbol{y = 160}$, $\boldsymbol{\theta = 315}$)



(a) *(b)*

The Table 4.7 shows the re-localization result of fourth test point.

Table 4.7

Re-Localization Result of Fourth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
4	240	160	315	243.9	158.8	306.8

The Table 4.8 shows the RMSE between result of fourth test point and actual position.

Table 4.8

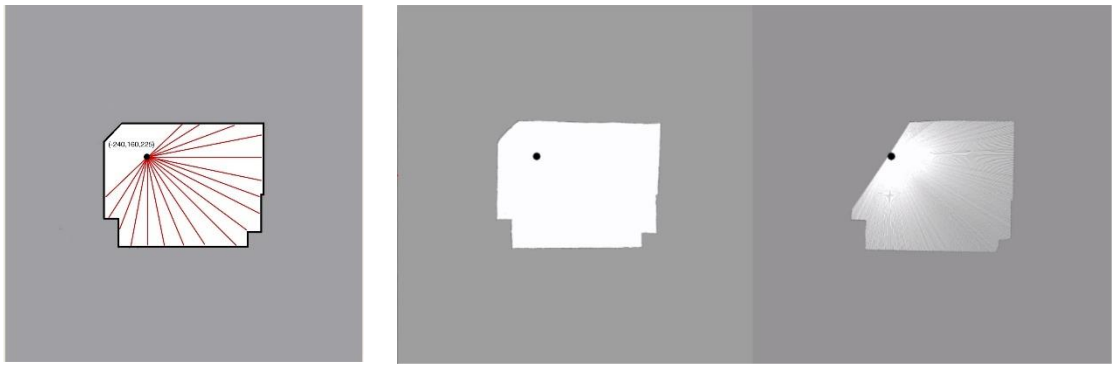
RMSE Between Result of Fourth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
4	3.9	1.2	8.2	4.08044

Fifth test point, the system will be placed at position of ($x = -240$, $y = 160$, $\theta = 225$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.6 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.6

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Fifth Test Point ($\boldsymbol{x = -240}$, $\boldsymbol{y = 160}$, $\boldsymbol{\theta = 225}$)



(a) *(b)*

The Table 4.9 shows the re-localization result of fifth test point.

Table 4.9

Re-Localization Result of Fifth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
5	-240	160	225	-243.2	163.8	235.2

The Table 4.10 shows the RMSE between result of fifth test point and actual position.

Table 4.10

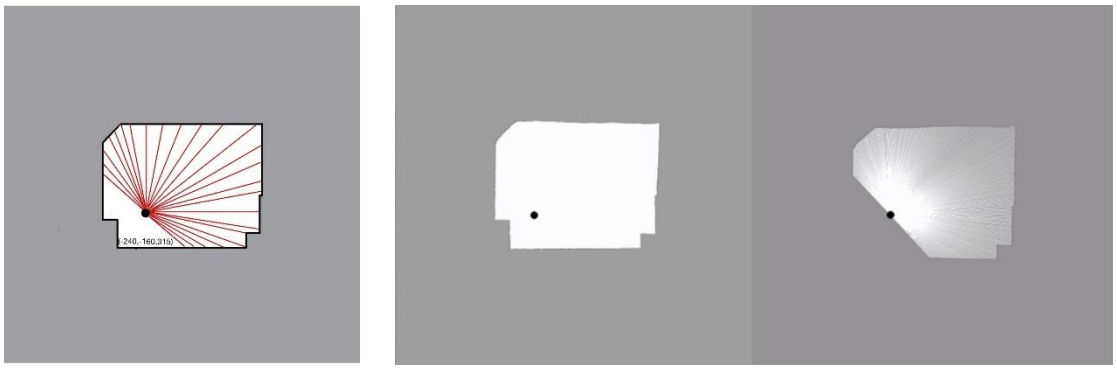
RMSE Between Result of Fifth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
5	3.2	3.8	10.2	4.9679

Sixth test point, the system will be placed at position of ($x = -240$, $y = -160$, $\theta = 315$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.6 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.7

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Sixth Test Point ($\boldsymbol{x = -240}$, $\boldsymbol{y = -160}$, $\boldsymbol{\theta = 315}$).



(a) *(b)*

The Table 4.11 shows the re-localization result of sixth test point.

Table 4.11

Re-Localization Result of Sixth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
6	-240	-160	315	-239.7	-161.8	314.1

The Table 4.12 shows the RMSE between result of sixth test point and actual position.

Table 4.12

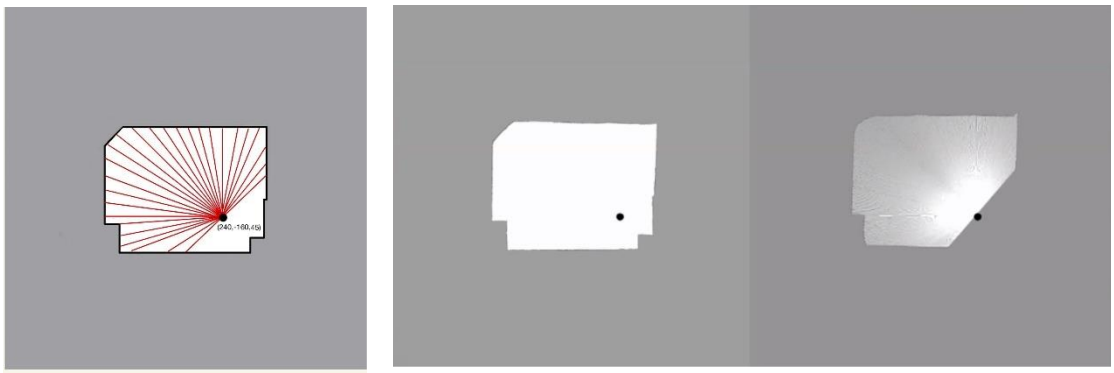
RMSE Between Result of Sixth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
6	0.3	1.8	0.9	1.82483

Seventh test point, the system will be placed at position of ($x = 240$, $y = -160$, $\theta = 45$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.8 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.8

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Seventh Test Point ($\boldsymbol{x = 240}$, $\boldsymbol{y = -160}$, $\boldsymbol{\theta = 45}$)



(a) *(b)*

The Table 4.13 shows the re-localization result of seventh test point.

Table 4.13

Re-Localization Result of Seventh Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
7	240	-160	45	239.15	-169.5	47.4

The Table 4.14 shows the RMSE between result of seventh test point and actual position.

Table 4.14

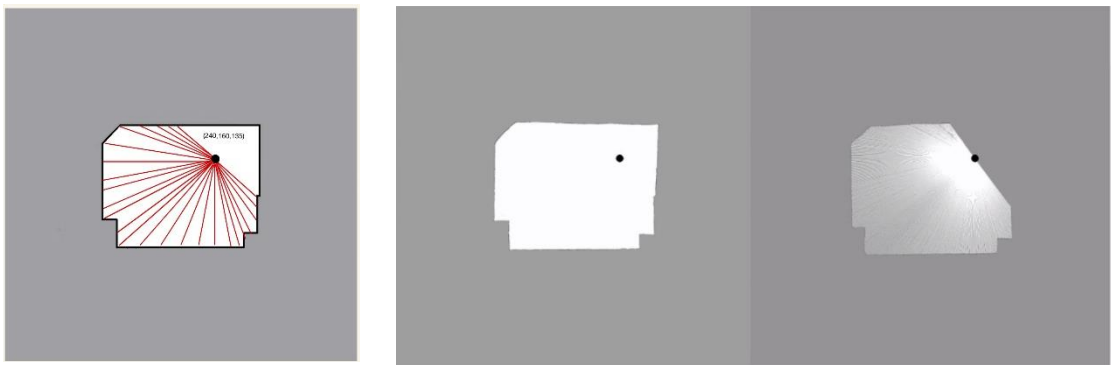
RMSE Between Result of Seventh Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
7	0.85	9.5	2.4	9.53795

Eighth test point, the system will be placed at position of ($x = 240$, $y = 160$, $\theta = 135$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.9 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.9

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Eighth Test Point ($\boldsymbol{x = 240}$, $\boldsymbol{y = 160}$, $\boldsymbol{\theta = 135}$)



(a) *(b)*

The Table 4.15 shows the re-localization result of eighth test point.

Table 4.15

Re-Localization Result of Eighth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
8	240	160	135	238	165.7	128.9

The Table 4.16 shows the RMSE between result of eighth test point and actual position.

Table 4.16

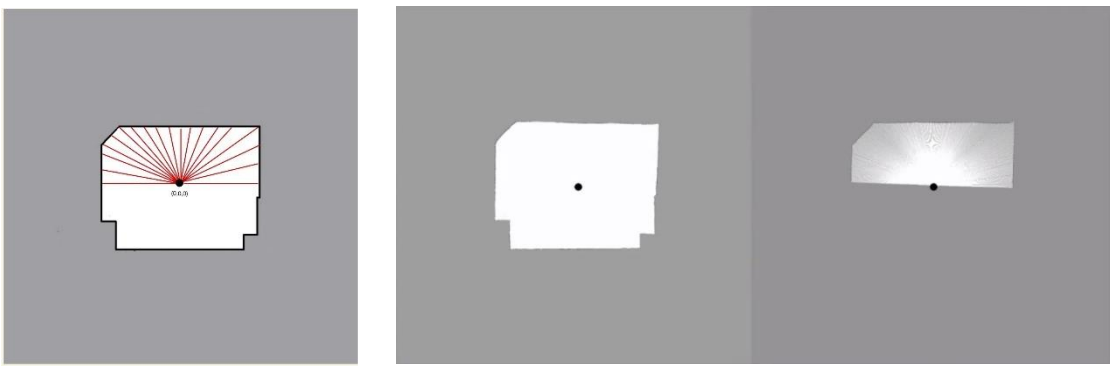
RMSE Between Result of Eighth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
8	2	5.7	6.1	6.0407

Nineth test point, the system will be placed at position of ($x = 0$, $y = 0$, $\theta = 0$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.10 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.10

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Nineth Test Point ($\boldsymbol{x = 0}$, $\boldsymbol{y = 0}$, $\boldsymbol{\theta = 0}$)



(a) *(b)*

The Table 4.17 shows the re-localization result of nineth test point.

Table 4.17

Re-Localization Result of Nineth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
9	0	0	0	-0.9	0.5	-2.1

The Table 4.18 shows the RMSE between result of nineth test point and actual position.

Table 4.18

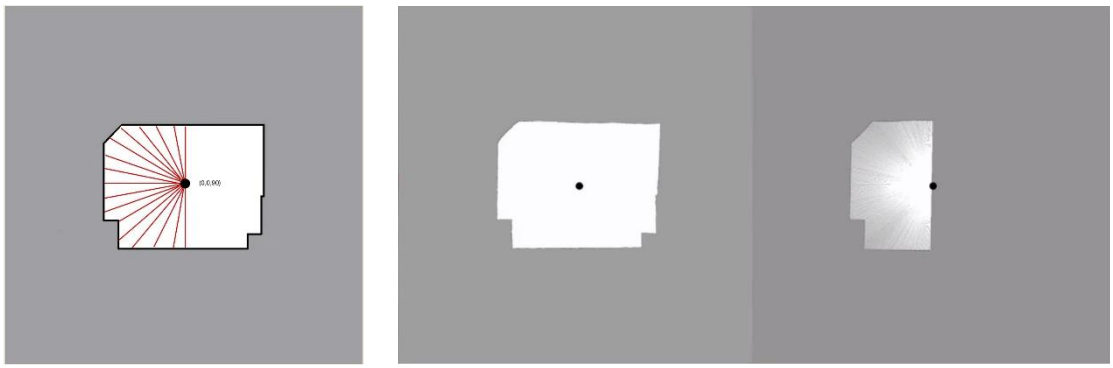
RMSE Between Result of Nineth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
9	0.9	0.5	2.1	1.02956

Tenth test point, the system will be placed at position of ($x = 0$, $y = 0$, $\theta = 90$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.11 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.11

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Tenth Test Point ($\boldsymbol{x = 0}$, $\boldsymbol{y = 0}$, $\boldsymbol{\theta = 90}$)



(a) *(b)*

The Table 4.19 shows the re-localization result of tenth test point.

Table 4.19

Re-Localization Result of Tenth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
10	0	0	90	-1	1.6	89.6

The Table 4.20 shows the RMSE between result of tenth test point and actual position.

Table 4.20

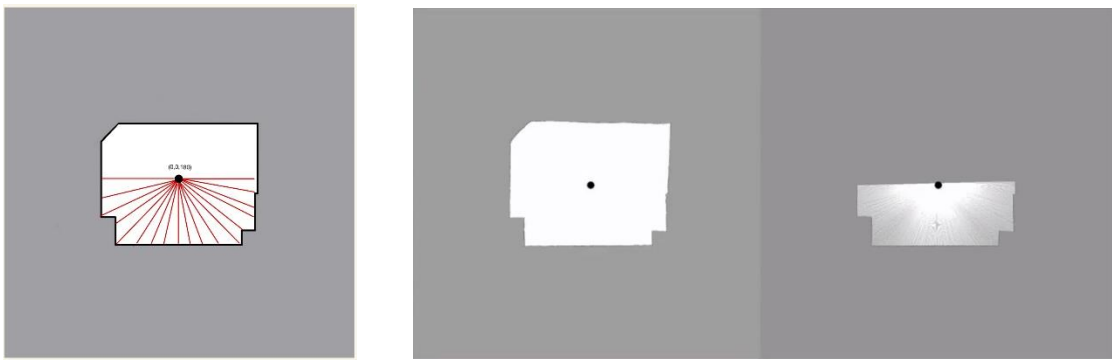
RMSE Between Result of Tenth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
10	1	1.6	0.4	1.8868

Eleventh test point, the system will be placed at position of ($x = 0$, $y = 0$, $\theta = 180$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.12 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.12

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Eleventh Test Point ($\boldsymbol{x = 0}$, $\boldsymbol{y = 0}$, $\boldsymbol{\theta = 180}$)



(a) *(b)*

The Table 4.21 shows the re-localization result of eleventh test point.

Table 4.21

Re-Localization Result of Eleventh Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
11	0	0	180	-0.5	-1.8	181.2

The Table 4.22 shows the RMSE between result of eleventh test point and actual position.

Table 4.22

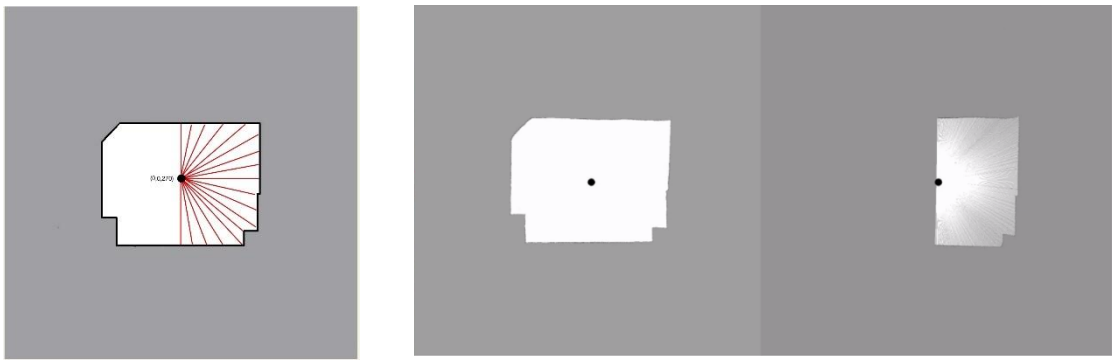
RMSE Between Result of Eleventh Test Point and Actual Position.

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
11	0.5	1.8	1.2	1.86815

Twelfth test point, the system will be placed at position of ($x = 0$, $y = 0$, $\theta = 180$). The current scanned point cloud will be compared with the point cloud of the given map. Figure 4.13 shows the graphical representation of the scanned area and position of the system and actual scanned area and position of the system.

Figure 4.13

Graphical (a) and Actual (b) Representation of Position of the System and the Scanned Area of the Twelfth Test Point ($\boldsymbol{x = 0}$, $\boldsymbol{y = 0}$, $\boldsymbol{\theta = 270}$)



(a) *(b)*

The Table 4.23 shows the re-localization result of twelfth test point.

Table 4.23

Re-Localization Result of Twelfth Test Point

Test Point	Ground Truth Position			Experiment Position		
	X (cm)	Y (cm)	θ (degree)	X (cm)	Y (cm)	θ (degree)
12	0	0	270	-1.5	-1.3	269

The Table 4.24 shows the RMSE between result of twelfth test point and actual position.

Table 4.24

RMSE Between Result of Twelfth Test Point and Actual Position

Test Point	RMSE			
	X (cm)	Y (cm)	θ (degree)	Distance (cm)
12	1.5	1.3	1	1.98494

The summarize of re-localization system evaluation is shown in Table 4.25.

Table 4.25

Evaluation of Re-Localization System

Test Point	Ground Truth Position			Experiment Position			RMSE				Condition
	x	y	heading	x	y	heading	x	y	heading	distance	
1	-240	160	45	-246.8	162.6	48.7	6.8	2.6	3.7	7.28011	Near
2	-240	-160	135	-230.2	-161.4	134.5	9.8	1.4	0.5	9.89949	
3	240	-160	225	248.9	-164.1	230.7	8.9	4.1	5.7	9.79898	
4	240	160	315	243.9	158.8	306.8	3.9	1.2	8.2	4.08044	
						Avg.Close	7.35	2.325	4.525	7.76476	
5	-240	160	225	-243.2	163.8	235.2	3.2	3.8	10.2	4.9679	Far
6	-240	-160	315	-239.7	-161.8	314.1	0.3	1.8	0.9	1.82483	
7	240	-160	45	239.15	-169.5	47.4	0.85	9.5	2.4	9.53795	
8	240	160	135	238	165.7	128.9	2	5.7	6.1	6.0407	
						Avg.Far	1.5875	5.2	4.9	5.59284	
9	0	0	0	-0.9	0.5	-2.1	0.9	0.5	2.1	1.02956	Medium
10	0	0	90	-1	1.6	89.6	1	1.6	0.4	1.8868	
11	0	0	180	-0.5	-1.8	181.2	0.5	1.8	1.2	1.86815	
12	0	0	270	-1.5	-1.3	269	1.5	1.3	1	1.98494	
						Avg.Mid	0.975	1.3	1.175	1.69236	
						Avg.Total	**3.3042**	**2.9417**	**3.5333**	**5.0167**	

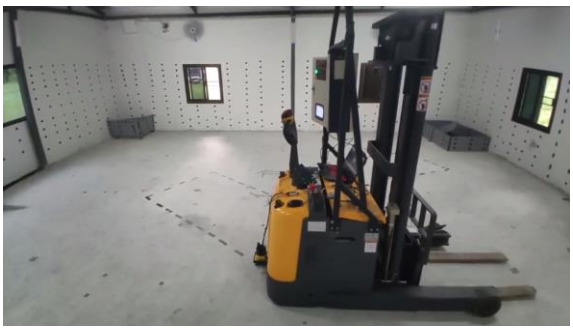
4.4 Evaluation of Navigation System

In navigation task, the forklift will be kidnapped when system is shutdown. When the system starts, it must re-localization itself in the map to get the current position, then navigate itself through the assigned waypoints.

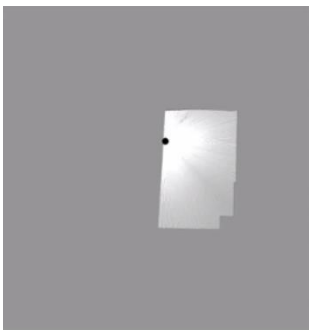
In the test, forklift is kidnapped to the $(x = 0, y = 160, \theta = 270)$ as shown in the Figure 4.14.

Figure 4.14

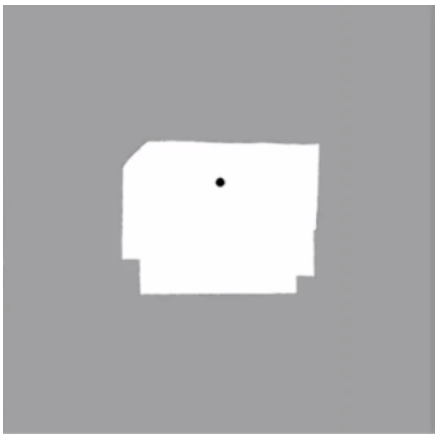
(a) Initial Position of Forklift in Real Environment (The scene was captured facing the 180 degree of the map) , (b) System and (c) Map



(a)



(b)



(c)

The Table 4.26 shows the RMSE between result of system's initial position and ground truth.

Table 4.26

RMSE Between Result of System's Initial Position and Ground Truth

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
0	0	160	-6.26	169.8	6.26	9.8	11.6287

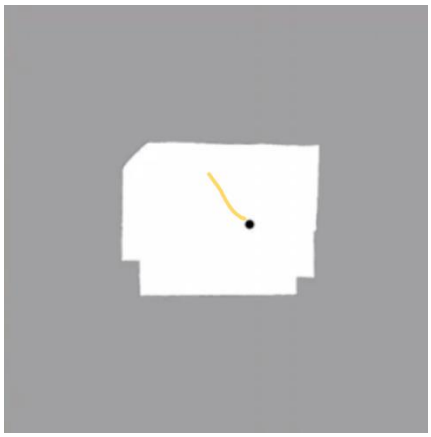
Forklift will be navigated to the first waypoint which is at $(x = 130, y = -15)$ regardless the heading. The result is shown in Figure 4.15.

Figure 4.15

(a) Position of Forklift in Real Environment (The scene was captured facing the 180 degree of the map), (b) System and (c) Map and Its Trajectory (orange line) when Reach First Waypoint

(a)

(b)



(c)

The Table 4.27 shows the RMSE between result of system's position when reach first waypoint and ground truth.

Table 4.27

RMSE Between Result of System's Position when Reach First Waypoint and Ground Truth

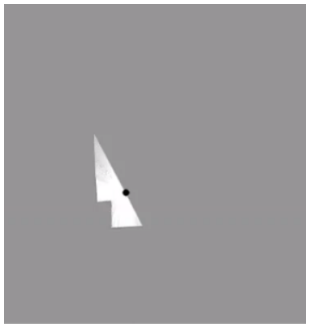
Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
1	130	-15	131.18	-15.36	1.18	0.36	1.23369

Forklift will be navigated to the second waypoint which is at $(x = -240\,, y = -160)$ regardless the heading. The result is shown in Figure 4.16.

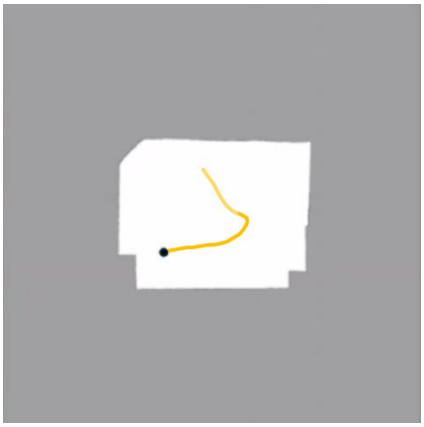
Figure 4.16

(a) Position of Forklift in Real Environment (The scene was captured facing the 180 degree of the map), (b) System and (c) Map and Its Trajectory (orange line) when Reach Second Waypoint

(a)



(b)



(c)

The Table 4.28 shows the RMSE between result of system's position when reach second waypoint and ground truth.

Table 4.28

RMSE Between Result of System's Position when Reach Second Waypoint and Ground Truth

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
2	-240	-160	-245.98	-170.83	5.98	10.83	12.3713

The summarize of navigation system evaluation is shown in Table 4.29.

Table 4.29

Evaluation of Navigation System

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	x	y	x	y	x	y	distance
0	0	160	-6.26	164.8	6.26	4.8	7.88845
1	130	-15	131.18	-15.36	1.18	0.36	1.23369
2	-240	-160	-245.98	-170.83	5.98	10.83	12.3713

4.5 Evaluation of Hybrid Lidar and Vision-Based SLAM System

To evaluate the performance of hybrid lidar and vison-based SLAM system, the RMSE of the system during navigation task will be used to compare with the RMSE of the lidar-only localization system using ICP algorithm. The result from lidar-only localization system at initial point $(x = 0\,, y = 160)$ is shown in Figure 4.17.

Figure 4.17

Location of System in the Map (left), Current Scan of the System (right) at Initial Position

The Table 4.30 shows the RMSE between result of lidar-only localization system's position at initial position and ground truth.

Table 4.30

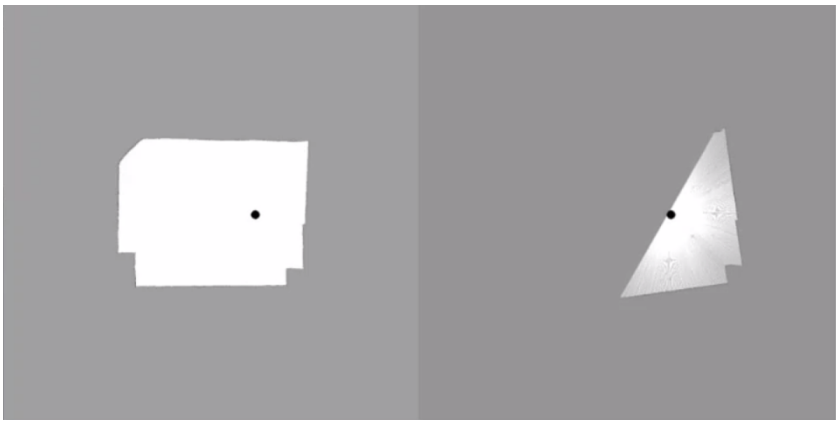
RMSE Between Result of Lidar-Only Localization System's Position at Initial Position and Ground Truth

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
0	0	160	-5.8	161.73	5.8	1.73	6.0525

The result from lidar-only localization system when reach first waypoint ($x = 130$, $y = -15$) is shown in Figure 4.18.

Figure 4.18

Location of System in the Map (left), Current Scan of the System (right) when Reach First Waypoint



The Table 4.31 shows the RMSE between result of lidar-only localization system's position when reach first waypoint and ground truth.

Table 4.31

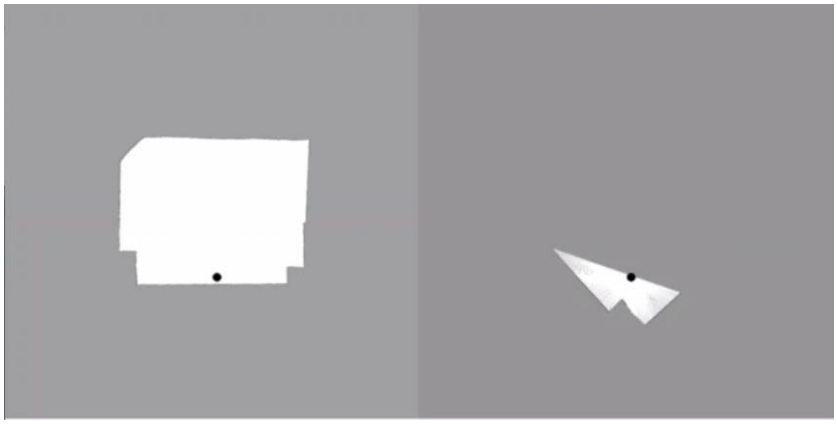
RMSE Between Result of Lidar-Only Localization System's Position when Reach First Waypoint and Ground Truth

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
1	130	-15	144.48	0.197	14.48	15.197	20.9909

The result from lidar-only localization system when reach second waypoint ($x = -240\,, y = -160$) is shown in Figure 4.19.

Figure 4.19

Location of System in the Map (left), Current Scan of the System (right) when Reach Second Waypoint.



The Table 4.32 shows the RMSE between result of lidar-only localization system's position when reach first waypoint and ground truth.

Table 4.32

RMSE Between Result of Lidar-Only Localization System's Position when Reach Second Waypoint and Ground Truth

Waypoint	Ground Truth Position		Experiment Position		RMSE		
	X (cm)	Y (cm)	X (cm)	Y (cm)	X(cm)	Y (cm)	Distance (cm)
2	-240	-160	9.323	-317.66	249.3	157.66	294.989

The Table 4.33 shows comparison between RMSE of Distance from hybrid lidar and vision-based SLAM system and lidar-only localization system.

Table 4.33

Comparison Between RMSE of Distance from Hybrid Lidar and Vision-Based SLAM System and Lidar-Only Localization System

Waypoint	Hybrid Lidar and Vision (cm)	Lidar-Only (cm)
0	7.888	6.053
1	1.234	20.991
2	12.371	294.989

As the result of comparison, at initial position, the re-localization will be executed which depends on lidar only by comparing current scan with given map. Therefore, results from both systems are similar to each other. After the robot moves to the first waypoint, the localization of lidar-only system starts to drift due to lack of initial guess of the relative transformation. When reaching the second waypoint, the RMSE of distance between lidar-only system and the ground truth is huge as 3 meters error. On the other hand, the hybrid lidar and vision-based SLAM system can localize the system with RMSE distance around 10 cm compare with ground truth when reaching the second waypoint.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION

5.1 Conclusion

The hybrid lidar and vision-based SLAM system has been successfully developed and were evaluated. There are two main parts of the system, visual odometry which use the monocular web camera and SLAM system which use lidar. Visual odometry uses the concept of detecting the movement of feature of the image frame. The feature is detected using concept of Shi-Tomasi corner detection to find the prominent corner points in the image. Features are separated into two sections, sky region and ground region. Features in sky region are used to find the heading change between two consecutive frames. Features on the ground region are used to find the change of translation components between two consecutive frames. The relative transformation between two consecutive frames in this process is unusable for localization due to accumulation of error. Therefore, SLAM system will use this information as the initial transformation for iterative closest points algorithm which aligns the scene of two consecutive frames together and gives the result of the better transformation matrix. The point cloud that is align with the reference will be stored to build the map of the environment. The transformation matrix from iterative closest point will also be used to update the position of the system. The system is capable of re-localization itself given the map. By comparing the current point cloud with the map, the system is able to localize itself in the map without any information related to where it is. The system is implemented to the forklift and is used for localization forklift during navigation task. The forklift is successfully navigated through two assigned waypoints.

5.2 Recommendation

Even visual odometry is better than wheel odometry in terms of installation and does not have a problem of wheel slip. However, the downsides of visual odometry are that the scene must heavily contains the features or else, the odometry system will not be able to recognize the change of the scene. Also, the environment must have enough lighting to detect the features of the scene.

5.3 Future Improvement

The features in the image frame are manually separated into sky region and ground region. This problem can be improved by implementation of deep learning techniques using semantic segmentation in order to determine the region of the floor and the wall. Then, the features can be automatically separated into the corrected region. Another improvement is related to the re-localization system. As of now, the re-localization system is capable of matching the first scan with the map in order to determine the location of the robot when the system is started. However, if the actual robot's position is too close to the wall, which the scan cannot capture the feature of the map, the system might unable to correctly map the current scan with the map which results in failure in localization. To solve this problem, particle filter can be used to corrected the position of the robot in the map each time the robot moves and the new scan is input into the system.

REFERENCES

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., ... Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics, 32(6), 1309–1332. https://doi.org/10.1109/TRO.2016.2624754

M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," 2018 International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 400-407, doi: 10.1109/IS.2018.8710464.

Zhang, Ji, and Sanjiv Singh. "Low-drift and real-time lidar odometry and mapping." Autonomous Robots 41.2 (2017): 401-416.

Debeunne, César, and Damien Vivet. "A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping." Sensors 20.7 (2020): 2068.

K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai and R. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, 2010, pp. 22-29, doi: 10.1109/IROS.2010.5649043.

G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in IEEE Transactions on Robotics, vol. 23, no. 1, pp. 34-46, Feb. 2007, doi: 10.1109/TRO.2006.889486.

Doucet, A., Freitas, N. De, Russell, S., & Murphy, K. P. (2000). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In Uncertainty in Artificial Intelligence Proceedings 2000 (pp. 176–183).

Kohlbrecher, Stefan, et al. "Hector open source modules for autonomous mapping and navigation with rescue robots." Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2013.

W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 1271-1278, doi: 10.1109/ICRA.2016.7487258.

R. Yagfarov, M. Ivanou and I. Afanasyev, "Map Comparison of Lidar-based 2D SLAM Algorithms Using Precise Ground Truth," 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2018, pp. 1979-1983, doi: 10.1109/ICARCV.2018.8581131.

Strasdat, Hauke, José MM Montiel, and Andrew J. Davison. "Visual SLAM: why filter?." Image and Vision Computing 30.2 (2012): 65-77.

Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Robot. 2015, 31, 1147–1163.

Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.

Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 834–849.

Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 15–22.

Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 40, 611–625.

Graeter, J.; Wilczynski, A.; Lauer, M. Limo: Lidar-monocular visual odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7872–7879.

Liang, X.; Chen, H.; Li, Y.; Liu, Y. Visual laser-SLAM in large-scale indoor environments. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; pp. 19–24.

Jiang, Guolai, et al. "A simultaneous localization and mapping (SLAM) framework for 2.5 D map building based on low-cost LiDAR and vision fusion." Applied Sciences 9.10 (2019): 2105.

Kurth, Derek. "Range-only robot localization and slam with radio." (2004): 04-29. 23, no. 1, pp. 34-46, Feb. 2007, doi: 10.1109/TRO.2006.889486.

X. Ji, L. Zuo, C. Zhang and Y. Liu, "LLOAM: LiDAR Odometry and Mapping with

Loop-closure Detection Based Correction," 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 2019, pp. 2475-2480, doi: 10.1109/ICMA.2019.8816388.

Ching-Chih Tsai, "A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements," in IEEE Transactions on Instrumentation and Measurement, vol. 47, no. 5, pp. 1399-1404, Oct. 1998, doi: 10.1109/19.746618.

Cho, Bong-Su, et al. "A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding." Journal of mechanical science and technology 25.11 (2011): 2907-2917.

A. Hemami, M. G. Mehrabi and R. M. H. Cheng, "A new control strategy for tracking in mobile robots and AGVs," Proceedings., IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 1990, pp. 1122-1127 vol.2, doi: 10.1109/ROBOT.1990.126146.

Wahid, Nurbaiti & Hassan, Nurhaffizah. (2012). Self-Tuning Fuzzy PID Controller Design for Aircraft Pitch Control. Proceedings - 3rd International Conference on Intelligent Systems Modelling and Simulation, ISMS 2012. 10.1109/ISMS.2012.27.

Y. Seo and C. Chou, "A Tight Coupling of Vision-Lidar Measurements for an Effective Odometry," 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 1118-1123, doi: 10.1109/IVS.2019.8814164.

Y. Xu, Y. Ou and T. Xu, "SLAM of Robot based on the Fusion of Vision and LIDAR," 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), Shenzhen, 2018, pp. 121-126, doi: 10.1109/CBS.2018.8612212.

Sünderhauf, Niko. Robust optimization for simultaneous localization and mapping. Diss. Technischen Universitat Chemnitz, 2012.

Campbell, Jason, et al. "A robust visual odometry and precipice detection system using consumer-grade monocular vision." Proceedings of the 2005 IEEE International Conference on robotics and automation. IEEE, 2005.