DEVELOPMENT OF A HYBRID MECANUM-OMNI-WHEEL MOBILE ROBOT WITH VISUAL SLAM

by

Ati Pongpachamnanwet

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 Technology in Electronics
Chitralada Technology Institute, Thailand

Scholarship Donor: Royal Thai Government Fellowship

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

AUTHOR'S DECLARATION

I, Ati Pongpachamnanwet, 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: 21 July 2021

Name: Ati Pongpachamnanwet

Signature:

ACKNOWLEDGMENTS

First of all, I would like to express my respect to my advisor, Prof. Manukid Parnichkun, for his valuable guidance, suggestion, and encouragement throughout the research. It is difficult to complete this thesis without his advice.

I want to extend my gratitude to Prof. Matthew M. Dailey and Dr. Mongkol Ekpanyapong, the examination committees, for their insightful remarks and information sharing.

I would like to extend my thanks to Mr. Hoang Hung Manh, the technical staff of the mechatronics department laboratory. He generously provides me with the resources I need to complete my research.

I'd also want to convey my gratitude to the Royal Thai Government for supporting me financially throughout my master's degree at AIT.

Finally, I want to express my gratitude to my family for their unwavering support and assistance during my education.

ABSTRACT

This master thesis focus on the develops a hybrid mecanum-Omni wheel configuration with visual SLAM for an autonomous mobile robot. This mobile robot is able to navigate autonomously in an indoor environment. The ORB SLAM 2, the open-source SLAM for monocular, stereo, and RGB-D camera is used for Mapping and localization. The RGB-D sensor is used to extract the feature for the SLAM and guild the robot inside the environment. The robot has the ability to make its own decision to avoid the obstacles with the Dynamic window approach (DWA). The robot is programmed based on the Robotic Operating System (ROS) which is a set of software and library for robotic applications.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

PID = Proportional Integral Derivative

ROS = Robotic Operating System

DWA = Dynamic Window Approach

SLAM = Simultaneous Localization and Mapping

CHAPTER 1
INTRODUCTION

1.1 Background of the Study

Automated guided vehicles (AGVs) become an important device in the industrial section. In the manufacturing process, it is used for transportation the material from the warehouse to the process or moving the product from the process to another process. The vehicle can navigate itself to the destination by following the guide which is prepared such as the mark on the floor or the magnet embedded in the ground.

In development of the navigation system, the mobile robot contains the ability to decide how to get to their destination. Simultaneous Localization and Mapping (SLAM) is used for creating a map from an unknown environment while localization the position of the robot. Additional path planning algorithm on created map allow the robot to move to the destination.

The transportation of the AGVs in the warehouse or the industrial environment needs suitable movement to avoid the collision with the product tray or the manufacturing machine. In this research, AGVs will be developed to have the ability to move in omnidirectional to expand the different move direction depend on the path and using the SLAM algorithm with a visual based for navigation system.

1.2 Statement of the Problem

The most mobile robot uses either omni-wheels or mecanum wheels to perform an omnidirectional motion. The difference of mecanum drive and omni-drive is that the mecanum drive provide more traction and friction than omni-wheel but the friction of mecanum wheel make it slower. The omni-wheel are light and fast. However, the friction of omni-wheel is low that from roller of wheel. The low friction led to low resistance to be pushed from a design direction.

Generally, for AGVs navigations system is mainly focus on the operation of following the path which navigate them. Therefore, the robot needs an ability to generate the optimal path in the workspace which is shortest and the collision free path.

1.3 Objective

The main objective of this research is to develop hybrid mecanum-omni-mobile robot with visual based SLAM. The list of objectives are as follows:

1. To design the Omni-direction mobile with hybrid mecanum and omni wheels.
2. To use SLAM algorithm to create a map for mobile robot.
3. To use camera for SLAM.

1.4 Limitations and Scopes

1. The mobile robot will be designed to operate only in an indoor environment.
2. The robot will operate only on the flat surface.

CHAPTER 2
LITERATURE REVIEW

2.1 Mobile Robot Wheel Designed

A robotic could be constructed as a holonomic and non-holonomic system. The holonomic system represented to the system that the number of the degree of freedom (DOF) of controlled is equal to a total degree of freedom (DOF). For the non-holonomic system, it means the total number of DOF of controlled less than the number of DOF. The characteristic of holonomic depends on the type of the robot wheel. In one hand, the robot with Ackerman wheeled system which cannot advance freely in any direction is called non-holonomic car. On the other hand, the robot that contain the ability of moving in the omni direction is a holonomic car (Shabalina, 2018).

The term of omnidirectional mobility describes the ability of a system to move instantaneously in any directions (X, Y, θ) at any time. (I. Doroftei et al., 2007) The wheel that can perform the omnidirectional movement is able to summaries as follow spherical, Swedish or mecanum wheel and universal omni-wheel.

The spherical wheel which is the ball shape wheel is able to perform an omnidirectional movement. The example of using spherical wheel is that the robot is based on the three-ball wheel with independent power by a motor. It can perform the excellent maneuverability. It can apply with other wheel to perform near-omnidirectional locomotion. However, the drawback of spherical wheel is that it is limited to flat surfaces and small load capacity. (Siegwart,Nourbakhsh, & Scaramuzza, 2011).

In 1973, A Mecanum wheel which is designed by Bengt Ilon has rollers at an angle of 45 degree to the plane of the wheel fastened on the periphery of the wheel. The angle of the rollers translates a portion of the force in the rotational direction of the wheel to a force normal to the wheel direction. The force vector from the translation depends on the direction and speed of each wheel. The sum of the force vector allows the platform to move freely in the direction of the result vector without changing the direction of the wheels. (F. Adascalitei and I. Doroftei, 2011) The configuration of the mecanum wheel need at least 4 mecanum wheels on square or rectangle shape platform, and angle of the

rollers of the opposite wheel must be in the opposite direction of each other to operate the omnidirectional motion.

The omni-wheel has a roller around the periphery of the wheel. The rollers are perpendicular to the plane of the wheel. The roller around periphery allows the wheel is able to roll with full force and slide laterally. The omni-wheels are employed as powered casters for differential drive robots to make turning faster. (S. Soni, 2014) The configuration of omni-wheel for omnidirectional movement required at least 3 wheel which the angle of each perform 120 degrees to each other like the triangular shape. For four-wheel configuration, the wheel locates at the side or corner of the square shape platform with the 90-degree angle between the wheel.

2.2 Kinematic Model of a Four Macanum Wheel

An omnidirectional motion can be achieved by using the kinematic relation of four mecanum wheel. In this paper, they explain about forward kinematic and inverse kinematic for four mecanum wheels mobile robot platform (Taheri et al.,2015).

Figure 2.1

The Configuration of a Robot with 4 Mecanum Wheel

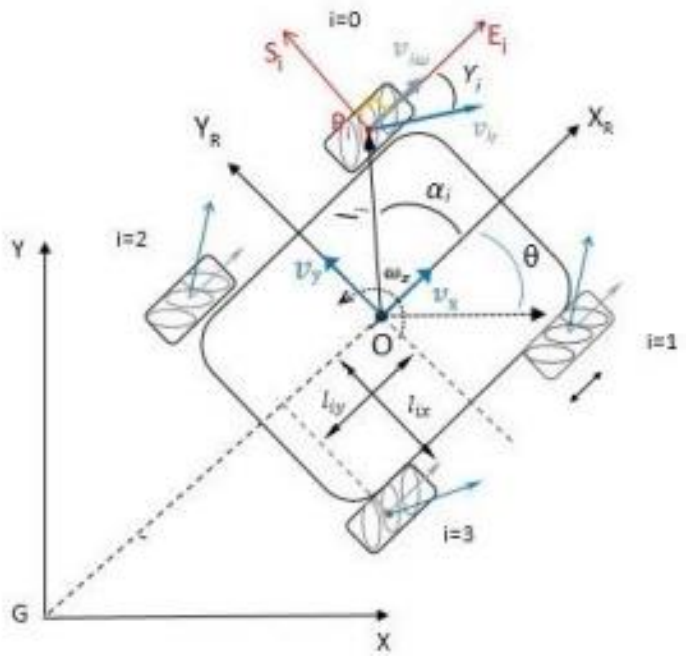
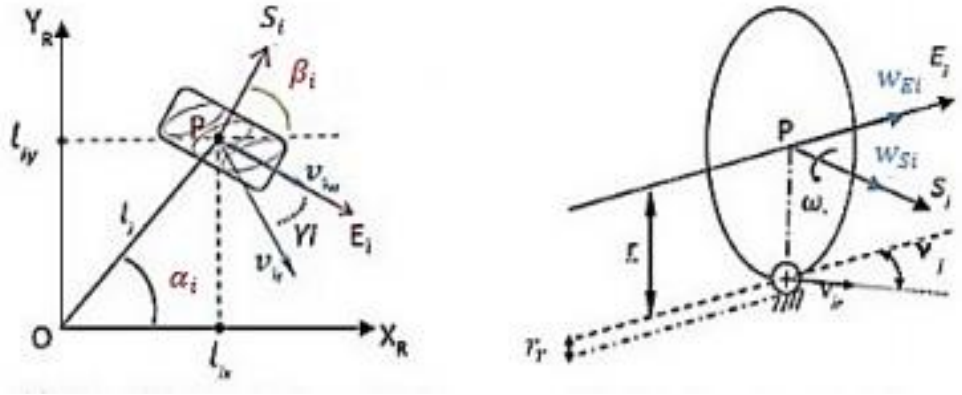


Figure 2.2

The Parameter of a Mecanum Wheel



As stated in **Figure 2.2**, assuming that a wheel is touching the ground. ω represent a wheel angular velocity and V_{ir} represent linear velocity of free roller wheel that touching to the floor (Taheri et al.,2015).

Figure 2.3

Inverse Kinematic of Mobile Robot with 4 Mecanum Wheel

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{-1}{r}\begin{bmatrix}\frac{\cos(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{\sin(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{l_1\sin(\beta_1-\gamma_1-\alpha_1)}{\sin\gamma_1}\\ \frac{\cos(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{\sin(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{l_2\sin(\beta_2-\gamma_2-\alpha_2)}{\sin\gamma_2}\\ \frac{\cos(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{\sin(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{l_3\sin(\beta_3-\gamma_3-\alpha_3)}{\sin\gamma_3}\\ \frac{\cos(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{\sin(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{l_4\sin(\beta_4-\gamma_4-\alpha_4)}{\sin\gamma_4}\end{bmatrix}\begin{bmatrix}v_X\\ v_Y\\ \omega_z\end{bmatrix}$$

Figure 2.4

Jacobian Matrix

$$T = \frac{-1}{r}\begin{bmatrix} \frac{\cos(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{\sin(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{l_1\sin(\beta_1-\gamma_1-\alpha_1)}{\sin\gamma_1} \\ \frac{\cos(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{\sin(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{l_2\sin(\beta_2-\gamma_2-\alpha_2)}{\sin\gamma_2} \\ \frac{\cos(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{\sin(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{l_3\sin(\beta_3-\gamma_3-\alpha_3)}{\sin\gamma_3} \\ \frac{\cos(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{\sin(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{l_4\sin(\beta_4-\gamma_4-\alpha_4)}{\sin\gamma_4} \end{bmatrix}$$

After calculation the value for the inverse kinematic, we can rewrite the matrix to a new equation. The following inverse and forward kinematic equation were used in velocity control program in order to send velocity commands for each wheel to the robot.

Figure 2.5

Inverse Kinematic Equations for 4-Wheel Mobile Robot Platform

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_x+l_y) \\ 1 & 1 & (l_x+l_y) \\ 1 & 1 & -(l_x+l_y) \\ 1 & -1 & (l_x+l_y) \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$$

$$\begin{cases} \omega_1 = \frac{1}{r}\left(v_x - v_y - (l_x+l_y)\omega\right), \\ \omega_2 = \frac{1}{r}\left(v_x + v_y + (l_x+l_y)\omega\right), \\ \omega_3 = \frac{1}{r}\left(v_x + v_y - (l_x+l_y)\omega\right), \\ \omega_4 = \frac{1}{r}\left(v_x - v_y + (l_x+l_y)\omega\right). \end{cases}$$

Figure 2.6

Forward Kinematic Equations for 4-Wheel Mobile Robot Platform

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} & -\frac{1}{(l_x+l_y)} & \frac{1}{(l_x+l_y)} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}$$

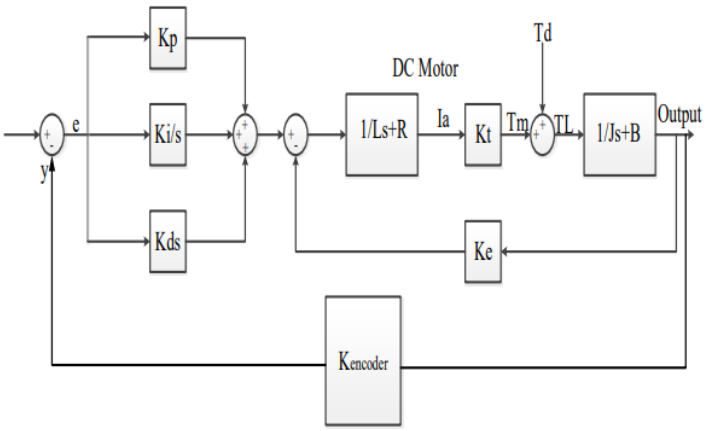
The designed structure of mecanum wheel for mobile robot from Salih et al. which used four mecanum wheel with mobile robot chassis. Each mecanum wheel are powered independently by using gear DC motor. **Figure 2.3** present the design structure of the omni-directional mobile robot.

2.3 Position and Velocity Control for Differential Drive Wheel Mobile Robot

The differential drive system is a simple system to create a movement for mobile robot. This system contains a robot chassis with two or more fixed wheels that are driven by an individual electric motor. This paper proposes a control system and navigation system of 2-wheel differential drive mobile robot for point-to-point motion without obstacle. Odometry is applied to estimate the current position of a navigation system. The PID Controller is used to control speed of each wheel for velocity control and position control. By using PID controller, robot can reach to desired goal (Cherry Myint et al.,2016).

Figure 2.7

Block Diagram of DC Motor with PID Controller



2.4 Mobile Robot Navigation

The problem of navigation of mobile robot can be summarized into two problems. The first problem is localization which provide robot an information of the current positions in that environment. The second problem is mapping which the ability to build the map of the surrounding area. The localization is a fundamental which is important for an autonomous mobile robot to navigate their own path. There are various techniques to achieve the localization.

The map-based method is that the method uses the map as a reference. The map-based technique can optimize the error by using loop-closure which know the area when it visits that area again. The error is reset when it returns to previously known map. (S. Campbell, 2020) However, the map of environment is not always realized, that is the drawback of the map-based method. The technique that can perform an accurate localization without the priori map is known as Simultaneous Localization and Mapping (SLAM).

2.5 Dynamic Window Approach

They describe the dynamic window approach that is They describe the dynamic window approach that is the obstacle avoidance approach for a mobile robot. The approach is obtained from the motion dynamics of the robot. It creates a dynamic window that consist of reachable linear velocity and angular velocity with in the short time limit. The dynamic window approach considers only admissible velocities that affect a robot trajectory which safe from crashing. From velocities in a dynamic window, the combination of angular velocity and linear velocity is selected from cost function. The cost function contains an estimation angle between the current robot angle and goal location, the current forward velocity of the robot and the distance to the obstacle (Dieter Fox et al.,1997).

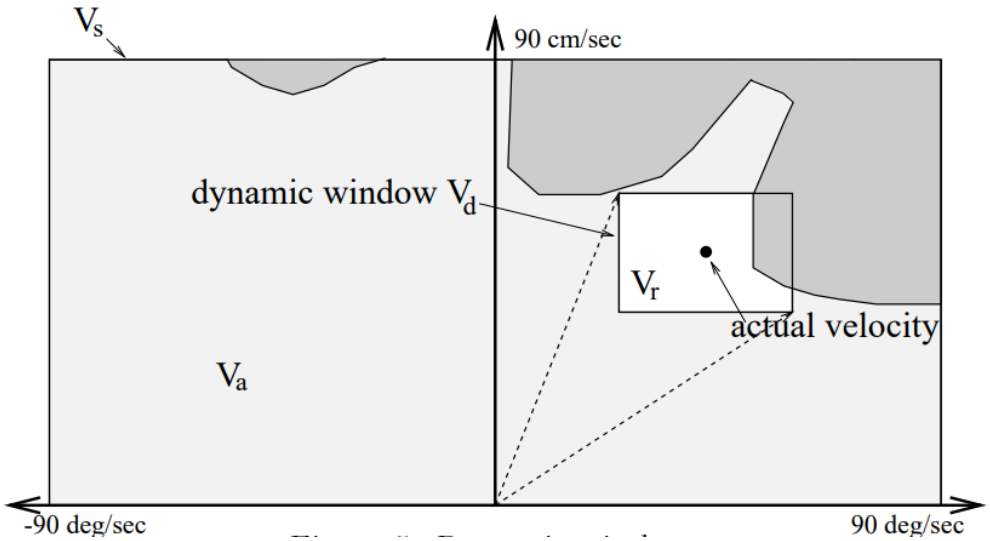
2.5.1 Search Space

Search space carried out the space of velocities that robot can reach. Thus, a dynamic window approach can select a proper velocity from reducing a velocity in search space. The reduction of search space work in 3 steps. In the first step, velocity in this approach is only in 2-dimensional space that is pair of angular velocity and linear velocity. In the

second step, the velocities that free from obstacle is considered admissible. In third step, the dynamic window is created from admissible velocity that robot can reach within a short time interval and limited acceleration.

Figure 2.8

Velocity Space and Dynamic Window



Following in **Figure 2.8**, the velocities range of the robot is 0 cm/sec to 90 cm/sec for linear velocity and -90 deg/sec to 90 deg/sec for angular velocity. It shows in big rectangle that is velocity space. On the other hand, the dynamic window is the small rectangle.

2.5.2 The Cost Functions

The cost function is the function for choosing the best velocity. The function considers 3 values that are Target heading, Clearance and Velocity. Target heading or function $angle$ is an angle between goal location and robot heading. If the robot moves directly to goal, the value is maximum. Clearance is the distance to the closest obstacle on the trajectory. The value is smallest when the trajectory desire to move around the obstacle. Velocity is a forward velocity of the robot. The function σ smooth is the sum of the three functions.

$$G(v,\omega) = \sigma(\alpha \cdot angle(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)) \quad (2.1)$$

Where $angle(v, \omega)$ is function of target heading, $dist(v, \omega)$ is a Clearance and $vel(v, \omega))$ is a velocity. α, β and γ are the weight.

2.5.3 The Result of Dynamic Window Approach

They demonstrate the robot with various experiment. The experiments show that the dynamic window approach is a robust obstacle avoidance technique. The robot can move safely without human supervision.

2.6 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is a method for estimating the sensor motion, create a map of an unfamiliar area and pinpoint the sensor's location. The major advantage of the SLAM is that it can accomplish localization without any prior knowledge about the surroundings. (Dissanayake et al., 2001) The several types of the sensor are able to perform a SLAM such as laser range, GPS, IMU and camera. From the different type of the sensor for SLAM, the price of the cameras is low and present the ton of information of the environment. The SLAM which the main sensor is cameras is known as Visual SLAM (VSLAM). The technique of visual SLAM is able to categorize into two main approaches: Feature-based approach which work on the key point from image and direct-based approach which work on the hold image. (T. Taketomi ei at. 2017)

2.6.1 Feature-Based Method

The feature-based method is based on keyframes and bundle adjustment optimization. These approaches extract the feature from the image and pick keypoints from the frame with diverse viewpoints through repetition and individuality. The map that is created by the feature-base technique is a very sparse map. However, the system can recieve the camera position from the map. (R. Mur-Artal, J. D. Tardos, 2015). The feature-base technique is developed in three techniques. MonoSLAM, The Parallel Tracking and Mapping (PTAM) an ORB-SLAM. (Taketomi T. ei at. 2017)

MonoSLAM which use Extended Kalmal filter (EKF) to estimate the unknown environments. The EKF's disadvantage is that the cost of computing rises in proportion to the size of an environment. In large surrounding area the size of the state vector

become large. Thus, the task of the real-time SLAM is difficult to achieve (Taketomi T. ei at., 2017). The issue of MonoSLAM is solved by using Parallel Tracking and Mapping (PTAM) technique which use two thread to perform the task in parallel. A foreground thread handles the feature tracking and camera registration, while a background thread operates the bundle adjustment (BA) to minimize the map which it can handle the more feature point in the map. However, in a large environment, it is difficult to get the global minimize. (G. Klein and D. Murray, 2009) In the case of Monocular VSLAM, the large problem is scale ambiguity if the global BA is not performed. In 2015, Ra´ul Mur-Arta et al. propose the ORB-SLAM which is able to perform the SLAM in the large environment with real-time performance. The ORB features are used to track the feature in changing of the viewpoint. The pose graph optimization is applied to global optimize the loop closing in real-time. (R. Mur-Artal et al., 2015)

The ORB-SLAM method is extended to stereo VSLAM and RGB-D VSLAM. From the implement of ORB-SLAM, Raul Mur-Artal and Juan D. Tardos developed the ORB-SLAM techniques into ORB-SLAM2 is an open-source SLAM system for monocular camera, stereo camera and RGB-D cameras. The advantage of using stereo camera or RGB-D camera is that each camera type contains the information of the depth from the first frame. Thus, it does not need the specific structure from the camera movement initialization same as the monocular camera. (Mur-Artal R. et al., 2017)

Form the analyzed of M. Filipenko and I. Afanasyev that compare various SLAM system which the method base on ROS-base SLAM for the mobile robot in an indoor environment. The experiment was conducted with a mobile robot operated on known perimeter of the square shape, and the surrounding area is typical office. The result of the experiment shows that the ORB-SLAM with a stereo camera perform a good result among the other is ORB-SLAM with RMSE ATE of 0.190 m. (M. Filipenko and I. Afanasyev., 2018). Moreover, the table and trajectory result of the demonstration of comparison show in table 2.1, **Figure 2.9**

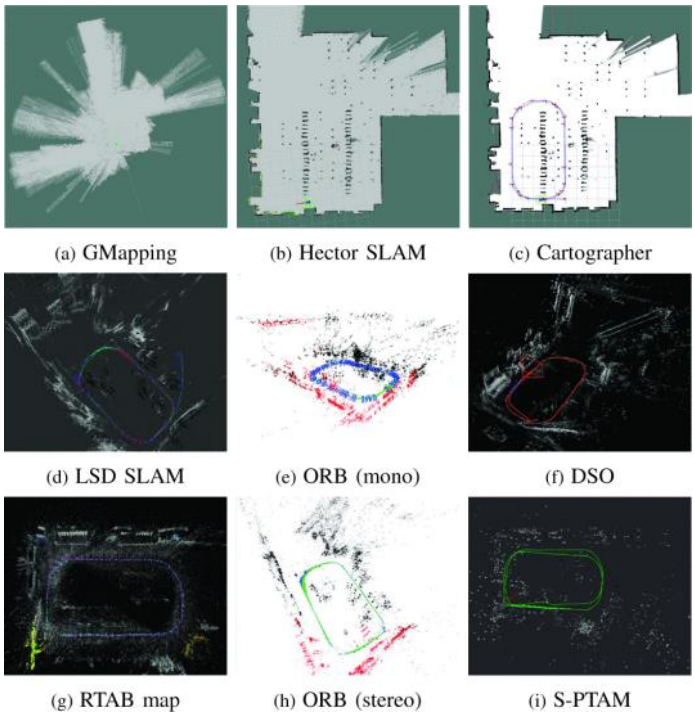
Table 2.1

Comparison Table of the Different SLAM Method

System	RMSE (m)	Mean (m)	Median (m)	Std. (m)	Min (m)	Max (m)
Cartographer	0.024	0.017	0.013	0.021	0.001	0.07
LSD SLAM	0.301	0.277	0.262	0.117	0.08	0.553
ORB SLAM (mono)	0.166	0.159	0.164	0.047	0.047	0.257
DSO	0.459	0.403	0.419	0.219	0.007	0.764
ZEDfu	0.726	0.631	0.692	0.358	0.002	1.323
RTAB map	0.163	0.138	0.110	0.085	0.004	0.349
ORB SLAM (stereo)	0.190	0.151	0.102	0.115	0.004	0.414
S-PTAM (no loop cl.)	0.338	0.268	0.244	0.206	0.001	0.768
S-PTAM (loop cl.)	0.295	0.257	0.242	0.145	0.006	1.119

Figure 2.9

Comparison Image of the Different SLAM Method



(a) GMapping (b) Hector SLAM (c) Cartographer

(d) LSD SLAM (e) ORB (mono) (f) DSO

(g) RTAB map (h) ORB (stereo) (i) S-PTAM

The result that the LIDAR-base SLAM with cartographer is the best result of the among the SLAM method. However, the cost of the LIDAR is very expensive compare with the cost of the camera. For the industrial applications, the visual SLAM is able to reduce the cost of developed the mobile robot which use for transportation the package with SLAM algorithm. Therefore, The ORB-SLAM is the techniques which match the task which use SLAM to localization. The drawback of the LIDAR is that the map from LIDAR is in 2D. Thus, some of the obstacle which lay on the different level of laser scanner will be invisible from LIDAR, and it led to accident. In case of camera-base SLAM, it provides the map in 3D which is more information than the LIDAR provided. The obstacles will be detected easily.

2.7 ORB-SLAM 2 Open-source SLAM System

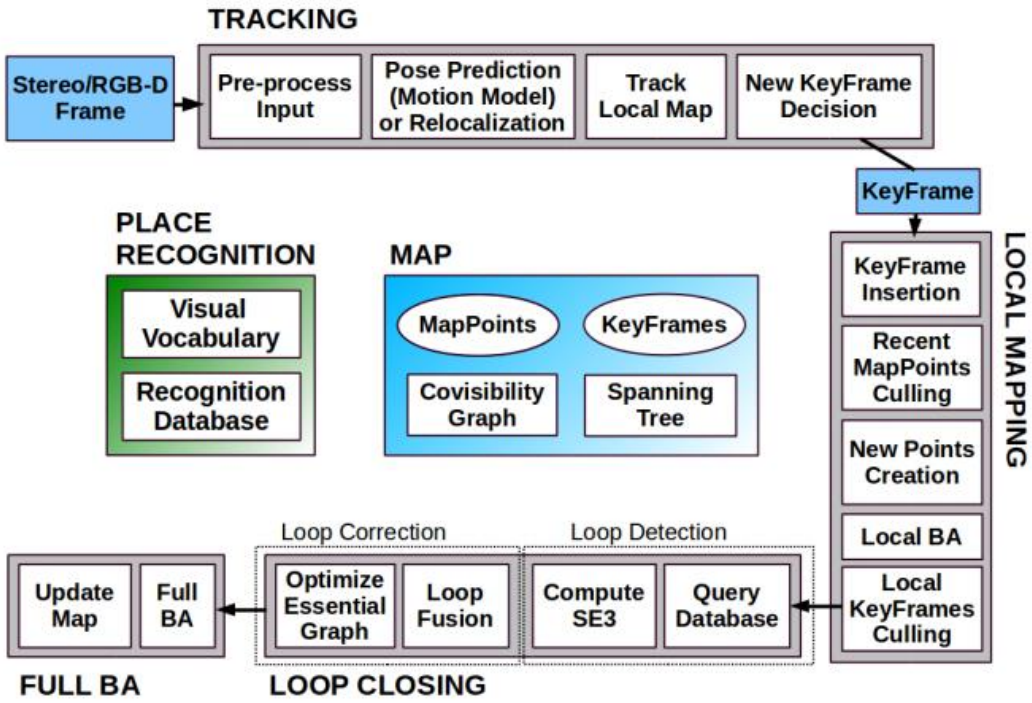
In 2017, Raul Mur-Artal et al., they present complete SLAM system for monocular, stereo and RGB-D camera. The ORB-SLAM 2 is an open-source SLAM system that can operate in real-time with ordinary CPUs in large environment. The drawbacks of monocular ORB-SLAM are scale drift and performing a pure rotation in the investigation. Those issues are solved by using a stereo or RGB-D camera.

2.7.1 ORB-SLAM 2 System Overview

The ORB-SLAM 2 operates on three parallel threads. The first thread is the tracking threads. Every frame, the tracking thread detects a feature point and uses it to localize the camera. The local mapping thread is the second thread. It inserts a new keyframe to the local map, chooses the proper map point, and optimizes the local map. The final thread is the loop closing thread. The loop closing detects the loop and fix accumulated drift by performing a pose-graph optimization. After the loop closing, the fourth thread launch to perform full BA to optimize the map. The system overview of the ORB-SLAM 2 shown in **Figure 2.10**.

Figure 2.10

System Overview of ORB-SLAM 2



2.7.2 Monocular, Close Stereo and Far Stereo Keypoints

ORB-SLAM 2 receives an input image to extract feature then the input image is discarded. The hold system operates on the extracted feature. Monocular and stereo keypoints, categorized as close or far keypoints, are included in the system. Stereo keypoints contain three coordinates that are u_L, v_L *and* u_R. The coordinate u_L *and* v_L are the coordinates on the left and u_R which is the horizontal coordinate in the right image. For a stereo camera, the ORB features of both images are extracted and the feature from the left image has searched a match in the right image. Then, the stereo keypoints are generated by the coordinate of the left feature and the horizontal coordinate of the right match. For RGB-D cameras, the system extract ORB feature form RGB image. The depth image is transformed into a right coordinate by using equation (2.2).

$$u_R = u_L - \frac{f_x b}{d} \tag{2.2}$$

Where f_x is the horizontal focal length, b is the baseline and d are the depth value of RGB-D camera.

After getting stereo keypoint, the close and far keypoint are the two types of keypoints. Close keypoint is that its depth is less than 40 times the stereo or RGB-D baseline. Otherwise, it is defined as far keypoint. For close keypoints, the depth can be reliable. Thus, the triangulation provides accurate scale, translation and rotation information. For far keypoint, It gives precise rotation data but less accurate translation and scale data.

Monocular keypoints contain two coordinates $u_L \, and \, v_L$ on the left image. This keypoint is the point that a stereo match could not be found. It provides only rotation and translation information.

2.7.3 Bundle Adjustment in ORB-SLAM 2

The system performs BA to optimize the camera pose, local window of keyframe and all keyframe and points. To optimize the cost function of BA, they use the Levenberg–Marquardt method that implemented in g2o.

Motion-only BA optimize the camera pose in the tracking thread. The camera orientation $R \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$ is optimized to minimize the reprojection error between keypoint and matched 3D points in world coordinates.

$$\{R, t\} = argmin_{R,t} \sum_{i \in \mathrm{x}} \rho \left(\left\| x_{(\cdot)}^{i} - \pi_{(.)} (RX^i + t) \right\|_{\Sigma}^{2} \right) \tag{2.3}$$

Where, $x_{(.)}^{i}$ is keypoints, monocular $x_m^i \in \mathbb{R}^2$ and stereo $x_s^i \in \mathbb{R}^3$. ρ is the Huber cost function, Σ is the covariance matrix associated to the scale of the keypoint, $\pi_{(.)}$ is the projection function, $\pi_{(m)}$ is monocular and $\pi_{(s)}$ is rectified stereo.

$$\pi_m \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{X}{Z} + c_y \end{bmatrix} \tag{2.4}$$

$$\pi_s\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{X}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \quad (2.5)$$

The equation (2.4) and (2.5) is projection function of monocular and stereo camera. Where, f_x and f_y are the focal length, c_x and c_y are the principal point, b is a baseline. These parameters can obtain from camera calibration.

Local BA optimizes a set of covisible keyframes, the set of keyframes that share the observation of the same map point, and optimizes all points seen in those keyframes P_L.

$$\{X^i, R_l, t_l | i \in P_L, l \in K_L\} = argmin_{X^i, R_l, t_l} \sum_{k \in K_L \cup K_f} \sum_{j \in x_k} \rho\left(E_{kj}\right) \quad (2.6)$$

$$E_{kj} = \left\| x^i_{(\cdot)} - \pi_{(\cdot)}\left(R_l X^j + t_k\right) \right\|^2_{\Sigma} \quad (2.7)$$

Where, K_L is a set of covisible keyframe, K_f is other keyframe that not in K_L but observing points are in P_L, X_k is the set of match point between point in P_L and keypoints in a key frame k.

Full BA uses the same cost function as Local BA to optimize all keyframes and points in the map.

2.7.4 Localization Mode

In localization mode, the local mapping and loop closing thread are deactivated and use only tracking thread to localize in mapped area. The camera localizes by using visual odometry match and matches to the map point. Visual odometry matches are matches between 3D points created in the previous frame and ORB in current frame. For map point matches, the localization can perform without accumulation drift in mapped area. On the other hand, visual odometry can operate in unmapped area, but drift can be accumulated.

In conclusion, the ORB-SLAM 2 that is an open-source SLAM system operate in real-time on standard CPUs with large environment. The system achieves the highest accuracy compare with other SLAM systems and zero-drift localization in the mapped area. The comparison of translation RMSE (m.) is shown in **Table 2.2,** and **Figure 2.11** shows the point cloud reconstruction from estimation point cloud in TUM RGB-D dataset.

Figure 2.11

The Reconstruction from Estimation Point Cloud and Sensor Depth Maps in TUM RGB-D Dataset.

Table 2.2

Comparison of Translation RMSE (m) in TUM RGB-D Dataset

Sequence	ORB-SLAM2 (RGB-D)	Elastic-Fusion	Kintinuous	DVO SLAM	RGBD SLAM
fr1/desk	**0.016**	0.020	0.037	0.021	0.026
fr1/desk2	**0.022**	0.048	0.071	0.046	-
fr1/room	0.047	0.068	0.075	**0.043**	0.087
fr2/desk	**0.009**	0.071	0.034	0.017	0.057
fr2/xyz	**0.004**	0.011	0.029	0.018	-
fr3/office	**0.010**	0.017	0.030	0.035	-
fr3/nst	0.019	**0.016**	0.031	0.018	-

2.8 Autonomous Navigation of Mobile Robot Using Kinect Sensor

This paper presents the autonomous mobile robot with Kinect sensor. The robot used data from Kinect to detect surrounding obstacles. The obstacle detection receives data from Kinect that is a live video. Then, the data is converted into 3D point clouds. After getting the point clouds, those point cloud are flittered by voxel filtering to reduce the processing time. The clustering processes cluster the filtered point cloud by using Euclidean Cluster Extraction. This algorithm search through filtered point cloud to find the point neighbors of it in a sphere with a radius less than the distance threshold. In this state, the robot recognizes the wall, obstacle and the floor. When the robot detects the obstacle, the robot avoids obstacles and navigate to free path to goal (N. A. Zainuddin et al., 2014).

2.9 ROS-Based Autonomous Mobile Robot Positioning and Navigation System

In 2019, ZHU jian-jun et al., they design the autonomous mobile robot based on Robot Operating System (ROS). The sensor that uses for getting the environment is laser scanner. The industrial computer, which is a microcomputer with a dual-core CPU, controls the robot. It can run Linux operating system and use ROS. The hardware architecture of mobile robot shown in **Figure 2.9.** A cartographer SLAM that is SLAM algorithm for laser scanner is used for creating a map. They used Navigation stack that is a ROS stack for complete navigation system. The block diagram of the navigation stack shown in **Figure 2.10.** The Navigation stack contains 3 main packages that are Map server, AMCL and Move base. After getting a mapping process, it creates a 2D-cost map for the path planning. Global planner is A* search, while the local used dynamic window approach (DWA). The robot can reach the set point target and avoid obstacles.

Figure 2.12

Autonomous Mobile Robot Hardware Architecture

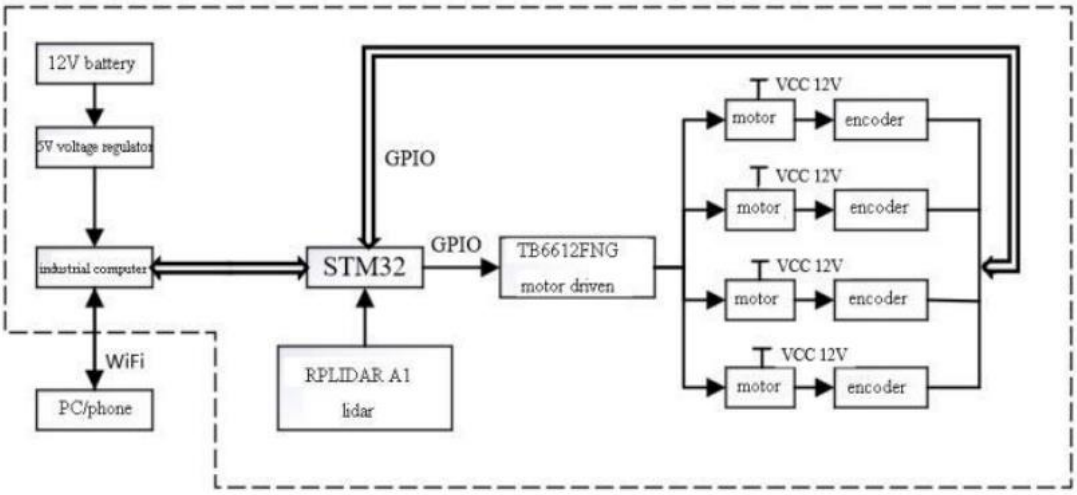
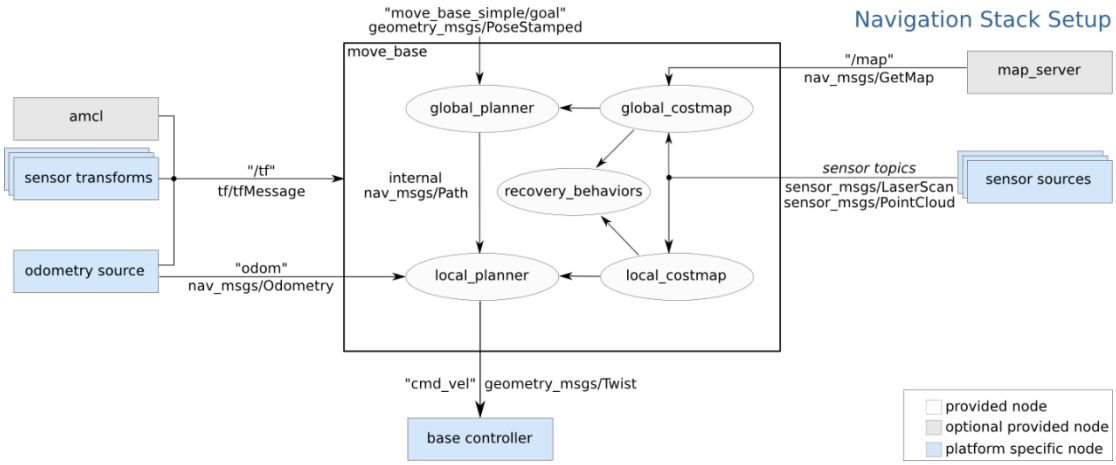


Figure 2.13

The Block Diagram of the Navigation Stack



CHAPTER 3
METHODOLOGY

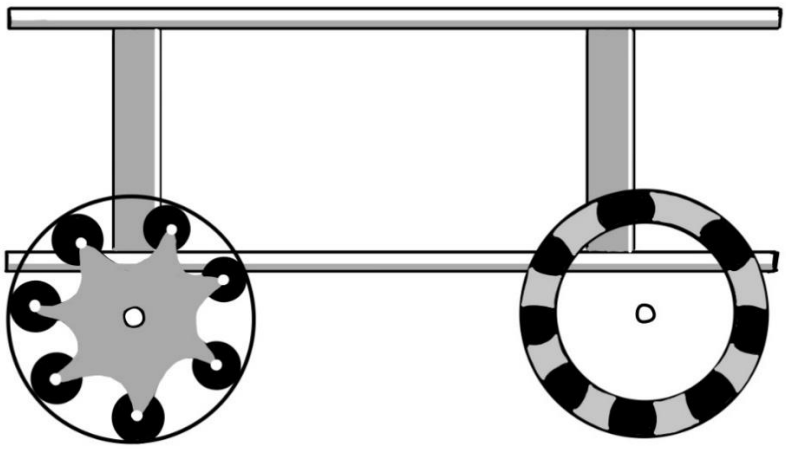
3.1 Mechanical Platform

The design of the mobile robot platform is defined base on the rectangle shape with four omnidirectional wheels. The robot platform is separated into two layers for mechanical system, sensor, and other necessary equipment. The wheel configuration is that the two mecanum wheels are in the front of the robot and two omni-wheels locate at the corner of rectangle with 90 degrees to the other omni-wheel which at the back of the platform. Each wheel is powered independently by the one actuator which is the differential drive. In additional, **Figure 3.1** shown the model of the proposed mobile robot.

In this study, the combination of the mecanum wheel and omni-wheel are used because of the advantage of the Omni-wheel perform a fast movement and good efficient on principle direction (diagonal) but the other direction (forward and sideway) will provide more error because of wheel slip. Mecanum wheel perform less slip than omni wheel but it provides a low speed because of friction on the wheels. Therefore, hybrid mecanum-omni wheel will reduce an error of omni wheel by using mecanum wheel to compensate error from omni wheel.

Figure 3.1

The Model of the Proposed Mobile Robot



3.2 The Comparison of Regular Wheel and Hybrid Wheel

The new wheel configuration is designed based on the advantage of the mecanum wheel and the Omni wheel. The advantage of the mecanum wheel is that the friction of the wheel is higher than the Omni wheel with the same torque. Thus, it provides more traction more than the Omni wheel. However, the high friction of the mecanum wheel makes the robot move slower. For the Omni wheel, the Omni wheel can move faster than the mecanum wheel with the same torque. On the other hand, the omni wheel provide less friction than the omni wheel. The hybrid mecanum-omni wheel use the advantage of the mecanum wheel to compensate the drawback of the omni wheel. The comparison table show in Table 3.1. The velocity of the wheel was calculated based on the differential wheel. The velocity of the wheel shown in equation (3.1). The force of the differential wheel shown in equation (3.2).

$$\boldsymbol{V} = \boldsymbol{\omega} \times \boldsymbol{r} \tag{3.1}$$

Where, V is the linear velocity of the wheel, $\boldsymbol{\omega}$ is the angular velocity of the wheel and r is the wheel radian.

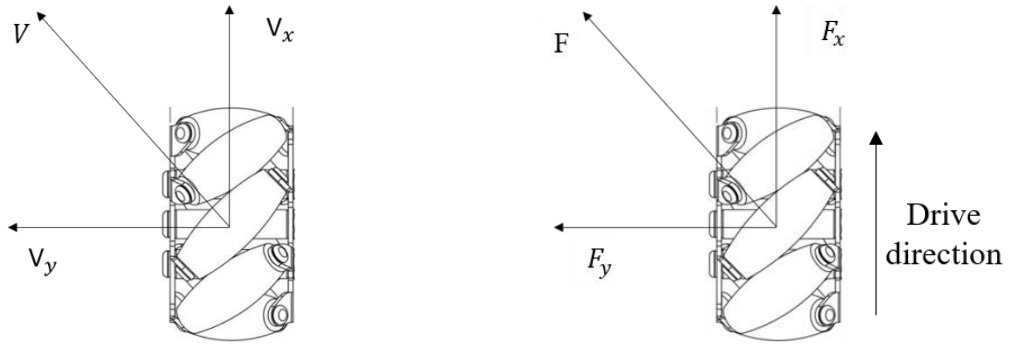
$$\mathrm{F} = \frac{\boldsymbol{\tau}}{\boldsymbol{r}} \tag{3.2}$$

Where, F is the force of the wheel, $\boldsymbol{\tau}$ is the angular velocity of the wheel and r is the wheel radian.

For the mecanum wheel, the friction and velocity are at 45 degrees of the drive direction. The omni directional the friction and velocity are the same as the differential wheel. The direction of force and velocity of mecanum wheel shown in **Figure 3.2.** and the direction of force and velocity of omni wheel shown in **Figure 3.2.** The direction of force and velocity base on the 4 mecanum wheel and 4 omni wheel configuration.e The force and velocity of each direction shown in Table 3.1.

Figure 3.2

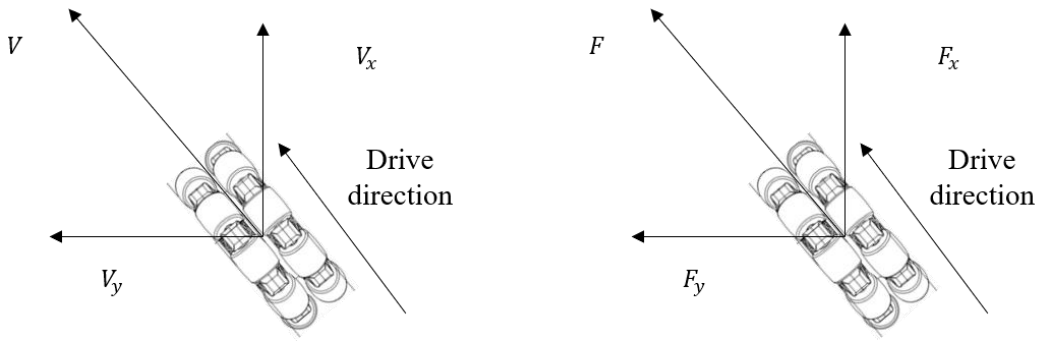
Velocity (a) and Force(b) Direction of Mecanum Wheel



(a) (b)

Figure 3.3

Velocity (a) and Force(b) Direction of Omni Wheel



(a) (b)

Table 3.1

The *Force and Velocity of Mecanum Wheel, Omni Wheel and Hybrid Wheel Configuration.*

	4 Mecanum wheel	4 Omni wheel	Hybrid wheel
Velocity(X)	$\boldsymbol{\omega \times r}$	$\boldsymbol{\sqrt{2}\omega \times r}$	$\boldsymbol{1.207\ \omega \times r}$
Friction(X)	$4\frac{\tau}{r}$	$4\frac{\tau}{\sqrt{2}\cdot r}$	$3.414\frac{\tau}{r}$
velocity(Y)	$\boldsymbol{\omega \times r}$	$\boldsymbol{\sqrt{2}\omega \times r}$	$\boldsymbol{1.207\ \omega \times r}$
Friction(Y)	$4\frac{\tau}{r}$	$4\frac{\tau}{\sqrt{2}\cdot r}$	$3.414\ \frac{\tau}{r}$
Diagonal velocity	$\boldsymbol{\frac{\omega \times r}{\sqrt{2}}}$	$\boldsymbol{\omega \times r}$	$\boldsymbol{0.853\ \omega \times r}$
Diagonal Friction	$\boldsymbol{2\sqrt{2} \times \frac{\tau}{r}}$	$\boldsymbol{2\frac{\tau}{r}}$	$2.414\frac{\tau}{r}$

According of the Table 3.2, the Hybrid wheel was calculated by combine the 2 mecanum wheel and 2 omni wheel.

Table 3.2

The comparison of mecanum wheel, omni wheel and hybrid wheel configuration.

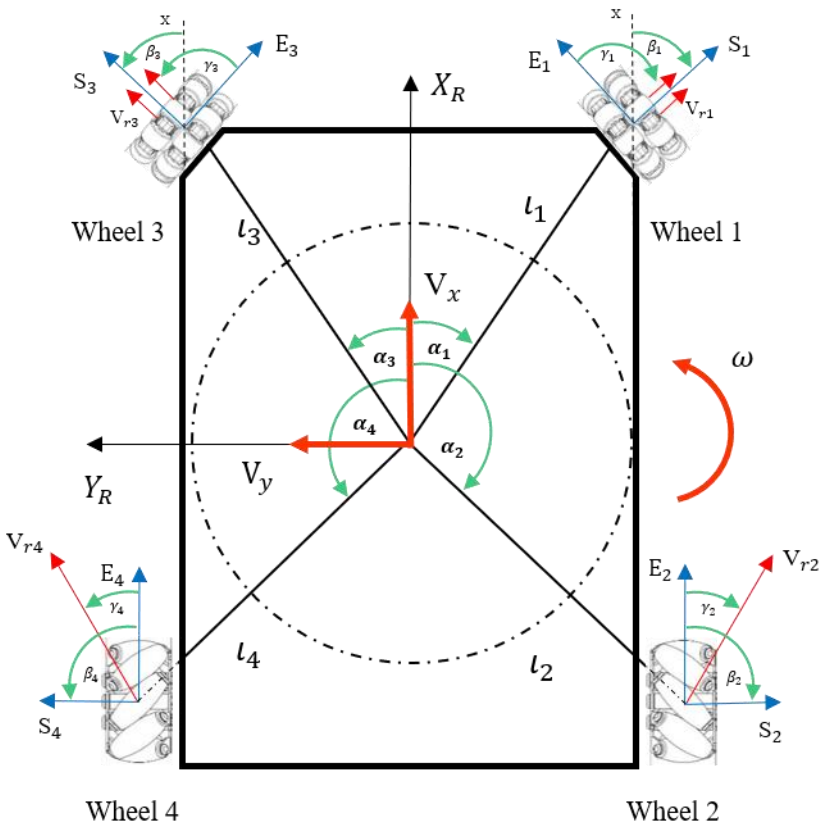
	Advantage	Disadvantage
4 Mecanum wheel	• The friction force of the 4 mecanum wheel is more than the omni wheel. Thus, it supports more weight than the 4 omni wheel.	• The wheel the velocity of the 4 mecanum wheel configuration less than the omni wheel. It makes the robot with 4 mecanum wheel move slower than the robot with 4 omni wheel.
4 Omni wheel	• The wheel performs the omnidirectional movement with more velocity than the mecanum wheel in any direction with the same torque and wheel radian.	• The force of the 4 Omni wheel is less than the mecanum wheel. It makes the robot slip easily and supports less weight than 4 mecanum wheels.
Hybrid wheel	• The performance of the hybrid wheel configuration is at the middle between 4 mecanum wheel and 4 omni wheel. The hybrid mecanum-omni perform with more velocity than mecanum wheel. It performs the omnidirectional movement with more force than the omni wheel. Thus, it will make the robot support more weight 4 omni wheel and move faster than the mecanum wheel.	• The kinematic model of the hybrid configuration wheel is more complicated the regular 4 mecanum wheel and 4 omni wheel.

3.3 Kinematics Analysis

The new kinematic model for 2 mecanum wheels and 2 Omni-wheels is designed following the model from the literature. The remaining model was used for 4 mecanum wheels mobile robot. The configuration of the new model and parameter follow **Figure 3.3**.

Figure 3.3

The Configuration of the New Model



The configuration parameter defines as follow:

X_R, Y_R: cartesian coordinate system of the robot base with the movement of the body center.

V_x, V_y, ω : linear velocity along x direction, linear velocity along y direction and angular velocity of the robot.

E_i, S_i : Coordinate system of the wheel i

V_{ri} : The velocity of the passive roller in wheel i

α_i : The angle between the center of wheel i and X_R

β_i : The angle between S_i and X_R

γ_i : The angle between E_i and V_{ir}

ι_i : The distances from the center of the robot to the center of the wheel i

The calculation for inverse kinematic model of mobile robot uses the Jacobian in **Figure 2.4** The Jacobian matrix for inverse kinematic of the model as follow:

$$T = \frac{-1}{r}\begin{bmatrix} \frac{\cos(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{\sin(\beta_1-\gamma_1)}{\sin\gamma_1} & \frac{l_1\sin(\beta_1-\gamma_1-\alpha_1)}{\sin\gamma_1} \\ \frac{\cos(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{\sin(\beta_2-\gamma_2)}{\sin\gamma_2} & \frac{l_2\sin(\beta_2-\gamma_2-\alpha_2)}{\sin\gamma_2} \\ \frac{\cos(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{\sin(\beta_3-\gamma_3)}{\sin\gamma_3} & \frac{l_3\sin(\beta_3-\gamma_3-\alpha_3)}{\sin\gamma_3} \\ \frac{\cos(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{\sin(\beta_4-\gamma_4)}{\sin\gamma_4} & \frac{l_4\sin(\beta_4-\gamma_4-\alpha_4)}{\sin\gamma_4} \end{bmatrix} \quad (3.3)$$

According from **Figure 3.3,** The parameter uses for calculating the mecanum wheel and Omni-wheel inverse kinematic model show in below **Table 3.1.**

Table 3.3

The Parameter of the 2 Mecanum Wheels and 2 Omni-Wheels.

Wheels	α_i	β_i	γ_i	l_i	r	type
1	$\frac{-\pi}{4}$	$\frac{-\pi}{4}$	$\frac{-\pi}{2}$	0.272	0.03	Omni
2	$\frac{-3\pi}{4}$	$\frac{-\pi}{2}$	$\frac{-\pi}{4}$	0.279	0.03	Mecanum
3	$\frac{\pi}{4}$	$\frac{\pi}{4}$	$\frac{\pi}{2}$	0.272	0.03	Omni
4	$\frac{3\pi}{4}$	$\frac{\pi}{2}$	$\frac{\pi}{4}$	0.279	0.03	Mecanum

The parameter in **Table 3.3** were substituted to the equation 3.1 and provide the inverse kinematic matrix in equation 3.4.

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \begin{bmatrix}23.5702 & 23.5702 & 9.0667\\ 33.3333 & -33.3333 & 13.152\\ 23.5702 & -23.5702 & -9.0667\\ 33.3333 & 33.3333 & -13.152\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ w_z\end{bmatrix} \tag{3.4}$$

Where, ω_i is angular velocity of wheel I, V_x and V_y are linear velocity of mobile robot and w_z is angular velocity the robot. The inverse kinematic matrix shows the velocity of each wheel for sending the velocity command to move the mobile robot.

$$\omega_1 = 23.5702 \times (V_x) + 23.5702 \times (V_y) + 9.0667 \times (w_z) \tag{3.5}$$

$$\omega_2 = 33.3333 \times (V_x) + (-33.3333) \times (V_y) + 13.152 \times (w_z) \tag{3.6}$$

$$\omega_3 = 23.5702 \times (V_x) + (-23.5702) \times (V_y) + (-9.0667) \times (w_z) \tag{3.7}$$

$$\omega_4 = 33.3333 \times (V_x) + 33.3333 \times (V_y) + (-13.152) \times (w_z) \tag{3.8}$$

Equations 3.5 to 3.8 are the angular velocity of each wheel. They were used to command the velocities of the mobile robot. After getting inverse kinematic for velocity command,

the matrix was calculated to find the forward kinematic model by taking the inverse of the matrix in equation 3.3 to get the forward kinematic model.

$$\begin{bmatrix} V_x \\ V_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0.0071 & 0.01 & 0.0071 & 0.01 \\ 0.0107 & -0.0071 & -0.0107 & 0.0071 \\ 0.027 & 0.019 & -0.027 & -0.019 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (3.9)$$

The forward kinematic use for getting feedback velocities of the mobile robot. The equation of linear velocity and angular velocity of mobile robot show in equation 3.12 to 3.14

$$V_x = 0.0071 \times (\omega_1) + 0.01 \times (\omega_2) + 0.0071 \times (\omega_3) + 0.01 \times (\omega_4) \quad (3.12)$$

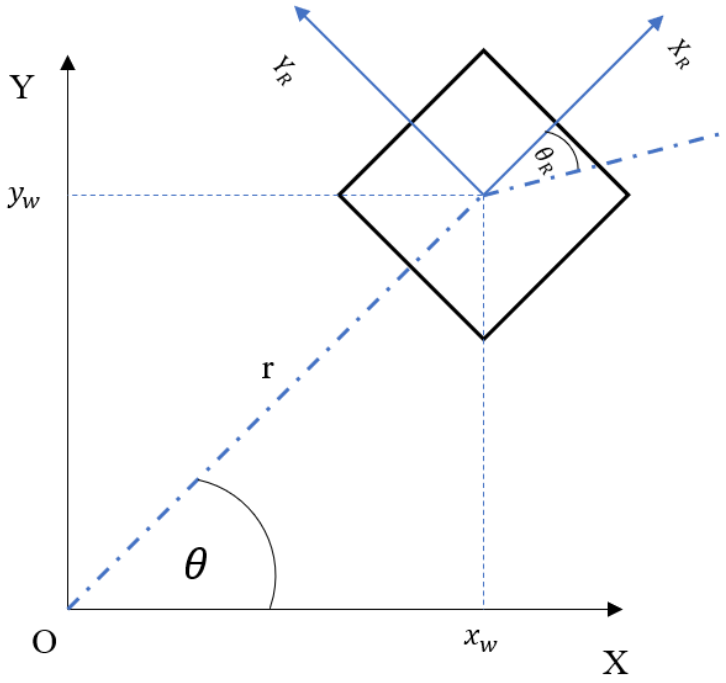
$$V_y = 0.0107 \times (\omega_1) - 0.0071 \times (\omega_2) - 0.0107 \times (\omega_3) + 0.0071 \times (\omega_4) \quad (3.13)$$

$$w_z = 0.027 \times (\omega_1) + 0.019 \times (\omega_2) - 0.027 \times (\omega_3) - 0.019 \times (\omega_4) \quad (3.14)$$

3.4 The Robot Coordinate System

Figure 3.4

The Coordinate System of Mobile Robot in Global Frame



According to Figure 3.4, the global coordinate system contains position and orientation of the mobile robot. The calculation of coordinate system follows in below equation.

$$X_w = r \times \cos\theta \quad (3.15)$$

$$Y_w = r \times \sin\theta \quad (3.16)$$

Where, X_w, Y_w *and* θ is position of the robot in global coordinate, r travel distance of the mobile robot w.r.t global coordinate.

3.5 Mechanical Design and Equipment Selection

This thesis focus on improves the hybrid mecanum-omni wheel configuration. The movement of this configuration is the combination of force from the wheel's rotation direction. The length and width of the robot chassis is 420 × 300 mm which was used to connect all actuator and microcontroller. The angle between each wheel is 90 degrees. The designed robot platform shown in **Figure 3.5.** The mecanum wheel that was used

for the robot shown in **Figure 3.6** and the specification of mecanum wheel shown in table 3.2 The both type of mecanum wheel that are left type and right were used. The omni wheel at the front of the robot and wheel specification presented in **Figure 3.7** and table 3.3.

Figure 3.5

Robot's Chassis

Figure 3.6

Mecanum Wheel

Figure 3.7

Omni Wheel

Table 3.4

Mecanum Wheel's Specification

Parameter	Description
Diameter	60 mm.
Body material	Aluminum Alloy
Weight	86 g
Load capacity	10 kg.
Number of rollers	8

Table 3.5

Omni Wheel's Specification

Parameter	Description
Diameter	60 mm.
Body material	Aluminum Alloy
Weight	73 g
Load capacity	3 kg.
Number of rollers	10

The model of mobile robot is represented in following figure.

Figure 3.8

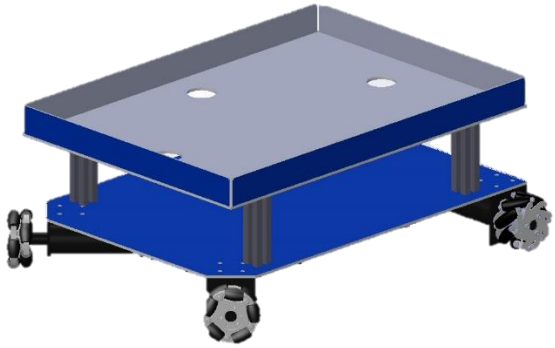
Front View

Figure 3.9

Rear View

Figure 3.10

Side View

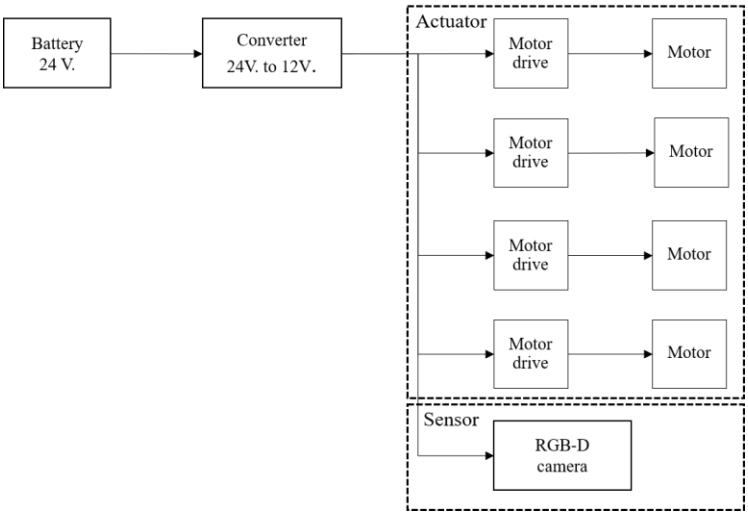


3.6 Electrical System Design and Equipment Selection

The one of the important parts of the mobile robot is electrical system. The electrical design of the robot separates into two parts the power system and the communication system. The power system of the robot shown in **Figure 3.11**.

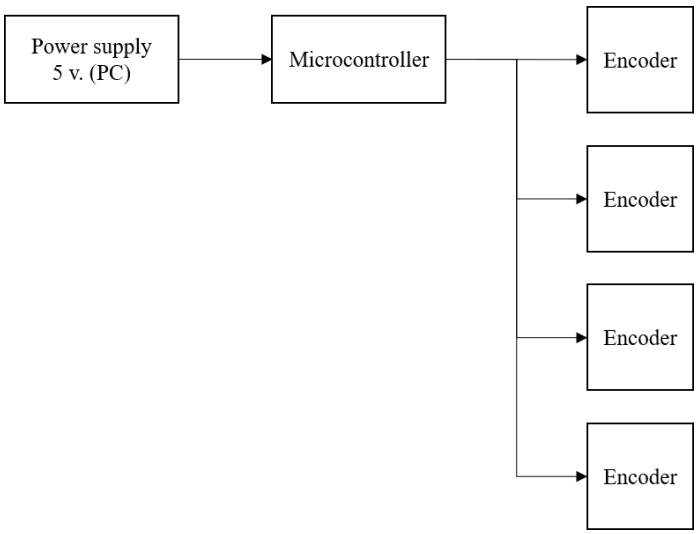
Figure 3.11

The Block Diagram for Electrical System (a) the Electrical System for 12 V. Power Supply (b) the Electrical System for 5 V. Power Supply



(a)

According to Figure 3.10 (a), the block diagram presents the power system from 24 V. battery. The battery connected with the converter to convert 24 VDC to 12 VDC. The voltage from the converter was stable.

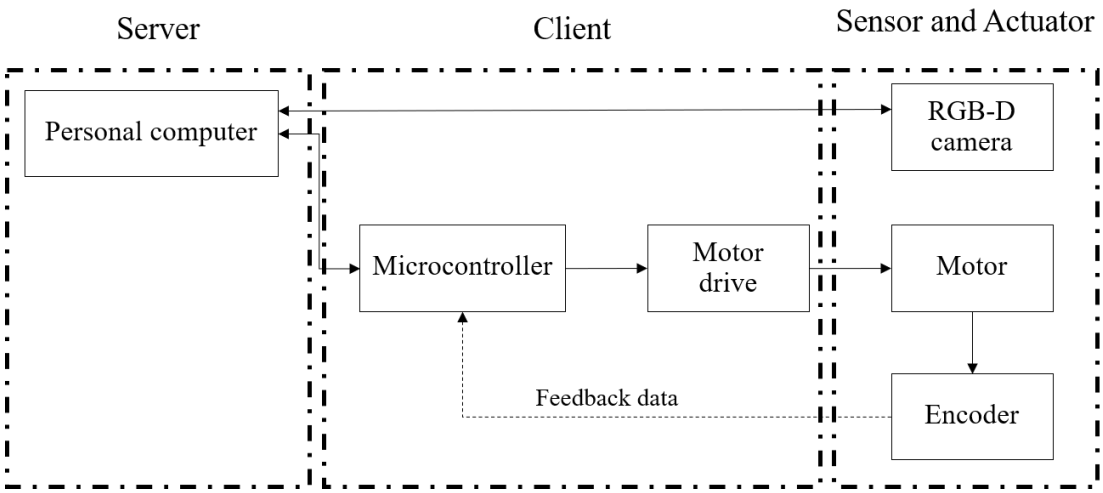


(b)

The communication system contains the server which is a personal computer and the client that is a microcontroller. The microcontroller was used for low-level communication which is sent Pulse Width Modulation (PWM) via motor driver to control motor and receive feedback data from encoder. The microcontroller communicates with PC through ROS topic which is ros_seriel. RGB-D was connected with a PC by using a driver from ROS. The block of the communication represents in **Figure 3.12**.

Figure 3.12

The Block Diagram for Electrical System for Communication Between Master and Slave Device



3.7 Microcontroller, Sensor and Actuator

The Kinect XBOX 360 is the main sensor for the mobile robot. The sensor was used for creating a map with a SLAM algorithm and navigating the robot via obstacles. This RGB-D camera is a low-cost RGB-D camera and widely used in computer vision or SLAM. Moreover, ROS develops a driver to communicate with the Kinect 360. The camera was calibrated by ROS camera calibration package RGB-D camera shown in **Figure 3.13** The specification of camera is represented in table 3.6. the encoder was connected to the motor to get feedback from motor for speed control. The encoder and its specification shown in **Figure 3.14** and table 3.7.

Figure 3.13

RGB-D Camera

Table 3.6

RGB-D's Specification

Parameter	Description
Name	Kinect XBOX360
Power supply	12 V.
Camera resolution	640 × 480 at 30 fps
IR camera resolution	320 × 240 at 30 fps
Field of view of RGB camera	62° × 48.6°
Field of view of Depth camera	57° × 43°
Operative measuring range	0.8 m – 4 m

Figure 3.14

Encoder

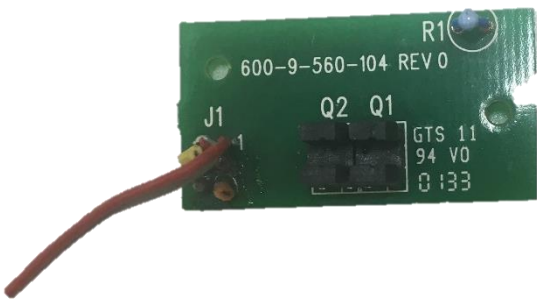


Table 3.7

Encoder's Specification

Parameter	Description
Power supply	5 V.
Count per revolute	12 CPR
Channal	2

The microcontroller that used to control motor speed and send a feedback data to PC via ROS is STM32 Nucleo-64 F401-RE. The stm32 has an encoder mode to connect the encoder without complex coding.

Figure 3.15

Microcontroller

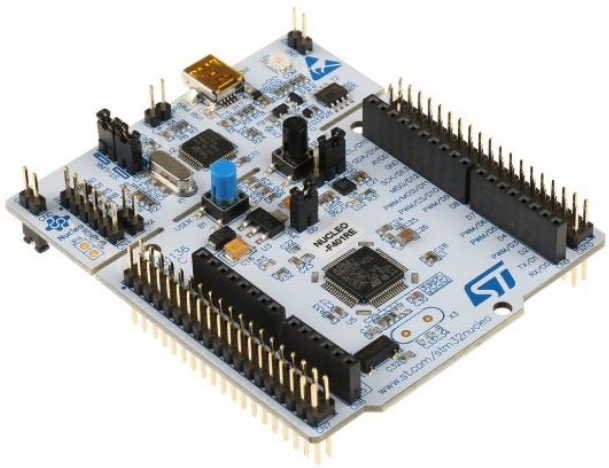


Table 3.8

Microcontroller's specification

Parameter	Description
Name	STM32 Nucleo-64 F401-RE
MCU	Arm Cortex-M4F
Operating voltage	3.6 v.
Clock speed	85 MHz
SRAM	96 KB
Flash memory	512 KB
Power supply	5 – 12 V.

One of the most important parts of the mobile robot is the motor. The 17 watts 12 volts motors were selected to drive the wheels. The motor connects with gearbox ration 1:64 to increase the rated torque for carrying a high payload. The motor for the mobile robot represents in **Figure 3.16**. The motor specification is shown in table 3.9

Figure 3.16

17 Watts 12-volt Motor

Table 3.9

Motor's Specification

Parameter	Description
Name	Fualhaber 2342 CR012
Operating voltage	12 v.
Rate current	1.5 A.
Output power	17 W
No load speed	8100 rpm.
Rate torque	17 mNm.
Gear box ratio	1:64

Finally, the L298M motor driver was selected to drive the motor. This motor driver can input with 2 channel PWM to control speed of the motors and the price is low. The motor driver depicted in **Figure 3.17.** Moreover, the specification of L298M represented in table 3.10

Figure 3.17

Motor Driver

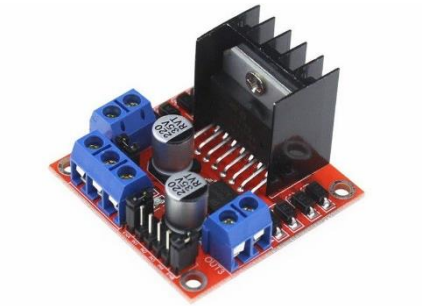


Table 3.10

Motor Drive's Specification

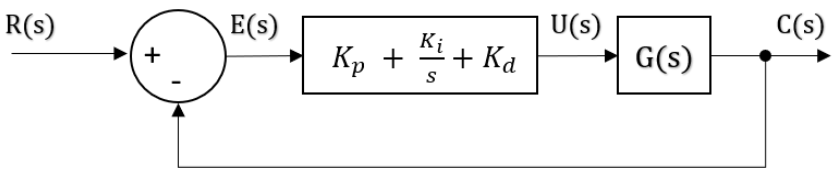
Parameter	Description
Chip	L298N
Operating voltage	7 – 30 V.
Maximum rate current	2 A.
Commutation frequency	25 – 40 KHz
PWM channel	2

3.8 PID Velocity and Position Control

STM32 microcontroller is the device for low level control. For microcontroller part, a PID controller was implemented in order to control velocity of the mobile robot. A PID controller receives the error between output and a setpoint as an input. It adjusts an output relate to the error to control the system. In this case, the system is a motor. Thus, the output is velocity. The controller receives velocity feedback from an encoder. The calculation of wheel speed in RPM from encoder shown in equation 3.13. ROS receive velocity in rad/sec. Thus, the velocity was converted from RPM to rad/sec. in equation 3.18

Figure 3.18

PID Controller



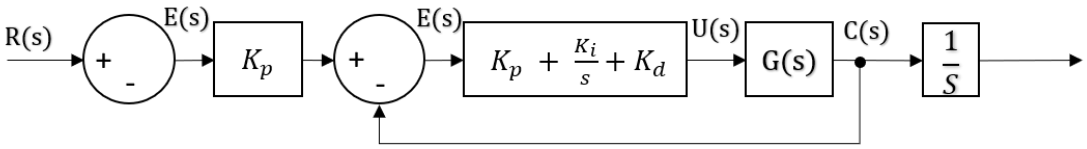
$$V(RPM) = \frac{Pulse\ different \times 60}{CPR \times time\ different} \quad (3.17)$$

$$\omega\ (red/sec.) = V(RPM) \times 0.1047197 \quad (3.18)$$

After getting velocity, the robot needs to move within designed distance such as “moving forward for 1 kg” or “turning heading to 90 degrees”. It can be achieved by using position control with PID. The figure below shows the block diagram of position control with PID.

Figure 3.19

PID Position Control



3.9 ROS

The Robot Operating System (ROS) is a collection of software libraries and tools designed to develop robot applications. ROS defines a standard for implement a code in the same way. Thus, the software that develops with ROS standards can be connected easily with the other. This mobile robot was programmed by using ROS melodic version with Linux Ubuntu 18.04. This version of ROS was developed in 2018. Hench, it supports a variety of package in ROS including “ros_orb_slam”.

3.9.1 Control Implement

The STM32 send data to PC with “serial_node_stm32” node. The topics that contain the wheels velocity from STM32 are “vel_n” for wheel n. In the same time, the velocity command for the robot was sent by the topic “cmd_vel”.

Figure 3.20

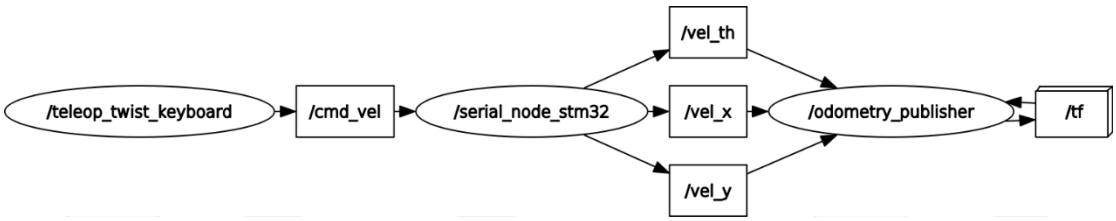
Serial_node Architecture

The nodes that publish the topic “cmd_vel” are “DWA” which is the node of Dynamic window approach and the node “teleop_twist_keyboard”. Basically, the node

"teleop_twist_keyboard" send velocity setpoint via the topic "cmd_vel". The application of this node is that the program starts with the default velocity, and it can increase or decrease the velocity level in the step of 10 % with the input keyboard. However, the node cannot input velocity level with the number. In this thesis, this node was applied to be able to input with the number. The modified node is used to publish position setpoint for position control of the robot via "cmd_vel". Thus, the application of this node is applied to use in the manual mode of the robot. In this mode, the robot receives setpoint which include the direction in x, y or heading from the modified node. Then, the robot moves to the setpoint and collect the data with node "odometry_publisher" which is the node for calculating odometry.

Figure 3.21

The Architecture of Node in Manual Mode



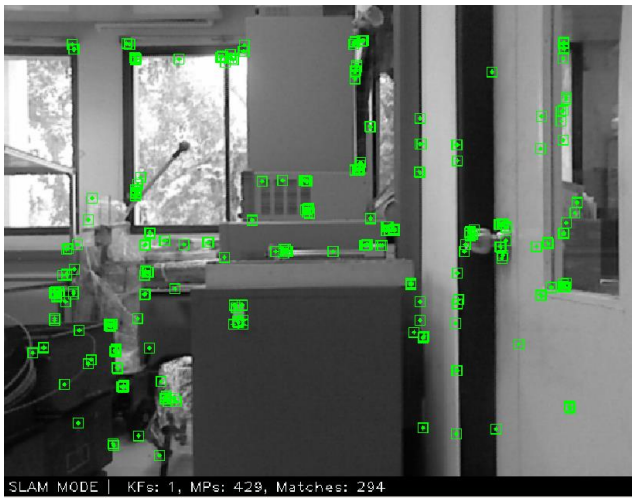
3.9.2 Ros ORB SLAM 2

This thesis use ORB SLAM2 with RGB-D to creating a map of the surround environment and localize the location of the robot. The map which is created with the ORB SLAM2 is sparse 3D point clouds. The system was descripted in the **chapter 2, 2.6**. The ORB SLAM2 the real-time SLAM for monocular, stereo camera and RGB-D camera was implemented by using ROS. This implementation of the ORB SLAM2 removes some dependency of the original version that is pangolin which is the software for monitoring the map. RVIZs which is the ROS visualizer is used to monitor the map instead of pangolin. All input and output are handled via ROS topic.

The mapping process of ORB SLAM 2 start with tracking thread. RGB-D camera extract ORB feature on the RGB image. For each feature, the depth was transformed into the right coordinate that describe in equation 2.2.

Figure 3.22

The ORB Feature in RGB Image



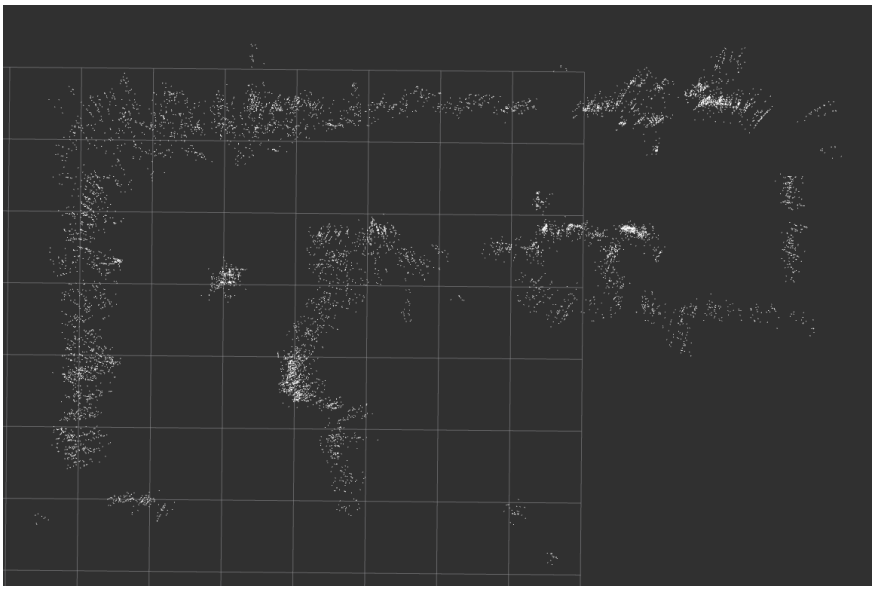
The keyframe is created with the first frame that the feature was extracted, and the pose of the first frame is set as an origin of the map. Then, the system creates an initial map from all stereo keypoints within the first keyframe. The system project the local map to the current frame and search map point correspondences. For the local map, it contains the set of the keyframe that share map points with the current frame and a set of keyframes which is a neighbor of K_1. The map points seen in are searched and optimized. Finally, the tracking thread insert the new keyframe if it meets the condition.

The tracking thread send the new keyframe to the local mapping thread. The new keyframe is added into the map and perform the Local BA to get the optimal reconstruction of the environment around the camera location.

Finally, the loops are searched with every keyframe by the loop closing thread. Once the loop is detected, the system calculates a similarity transform which contains the drift in the loop. Both sides of the loop are closed up and the duplicate points are combined. The pose graph optimization is performed to achieve an effectivity close the loop and corrected the scale drift.

Figure 3.23

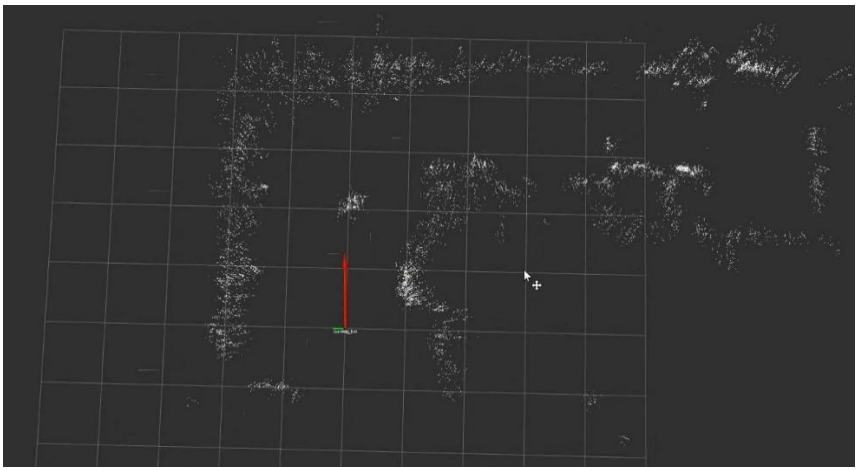
The Complete Sparse 3D Maps in RVIZs



For the localization without mapping, the system turns off the local mapping thread and the loop closing thread. The system localizes with visual odometry match that the match between ORB in current frame and the 3D points in the previous frame from the depth information. The other method for the localization is that the system matches the feature point in the current frame and the Mappoint in the local map to localize the camera location. The localization is represented in **Figure 3.24** The red arrow in the map is the location of the camera.

Figure 3.24

The Localization in Mapped



3.9.3 Dynamic window approach, path planning and obstacle avoidance

Dynamic window approach (DWA) is the one of the path planning algorithms. The approach calculates the proper velocity that contain linear velocity and angular velocity for the robot. The system creates the area of reachable velocity within the limit of time. This area is called “dynamic window”. This window relates with max/min liner velocity, max/min angular velocity, linear acceleration, angular acceleration, limit time constant and the current speed. The velocity in the dynamic window is chosen by the cost function. The cost function contains the heading of the robot to a goal location, the forward velocity of the robot and the distance to the obstacles on the current location. Basically, DWA is used for differential wheel. In this thesis, the mechanical of the robot is omni-directional wheel. Thus, the cost function of DWA was adjusted.

The velocity cost function:

$$vel = V_{max} - V_{current} \tag{3.19}$$

Where, vel is velocity cost function, V_{max} is maximum linear velocity in X direction in the dynamic window and $V_{current}$ is the current linear velocity in X direction of the robot. If the speed is near the max speed, the velocity cost function is low.

The heading cost function:

$$heading = \|\theta robot - \theta goal\| \quad (3.20)$$

Where, *heading* is heading cost function, $\theta robot$ is the heading angle of the robot that can be obtained with localization process and $\theta goal$ is the goal angle relate with the robot position and goal position.

The equation for $\theta goal$:

$$\theta goal = atan(\frac{dy}{dx}) \quad (3.21)$$

$$gx = x_{robot} - x_{goal} \quad (3.22)$$

$$gy = y_{robot} - y_{goal} \quad (3.23)$$

Where, gx is the distance of robot position in x coordinate and goal position in x coordinate, gy is the distance of robot position in y coordinate and goal position in y coordinate.

The heading cost function is the calculation of the heading relate with goal. The heading cost function is low if the heading of the robot point to the goal or rotate to the goal.

In this thesis, the heading cost function was adjusted for the calculation of the omni direction movement. This cost function removes the heading calculation. The new cost function was called goal cost function. The goal cost function was represented in equation 3.24.

The goal cost function:

$$goal = \sqrt{gx^2 + gy^2} \quad (3.24)$$

The goal cost function calculates the Euclidean distance between the current robot position and the goal position. The robot moves to the goal in x, y, or diagonal directions. It makes the cost function decrease.

For obstacles cost function, the robot calculates the trajectory that relates to all velocities in the "dynamic window". For all trajectory, the distance between the trajectory and the obstacles were calculated. The trajectory that hits an obstacle is maximized in order to remove from admissible velocity.

For trajectory i and obstacle j:

$$dx = traj_{i,x} - O_{j,x} \tag{3.25}$$

$$dy = traj_{i,y} - O_{j,y} \tag{3.26}$$

$$\text{r} = \sqrt{dx^2 + dy^2} \tag{3.27}$$

$$clear = \frac{1}{r} \tag{3.28}$$

Where, clear is obstacle cost function, r is the Euclidean distance between trajectory i and obstacle j, $traj_{i,x}$ is trajectory i of the robot in x direction and $O_{j,x}$ is obstacle j position in x coordinate. The obstacle cost function is low if the trajectory of the robot is far from the obstacle.

For all set of velocity in Dynamic window, the system calculates the sum of the cost functions which are vel cost function, *heading* cost function and *clear* cost function. The sum of the cost functions that is minimum is the best velocity.

$$V_{(x,\omega)} = min((\alpha \times vel(x_i, \omega_i)) + (\beta \times heading(x_i, \omega_i)) + (\gamma \times clear(x_i, \omega_i))) \tag{3.29}$$

$$V_{(x,y)} = min((\alpha \times vel(x_i, y_i)) + (\beta \times goal(x_i, y_i)) + (\gamma \times clear(x_i, y_i))) \tag{3.30}$$

Where, $V_{(x,\omega)}$ is the best set of curvature velocity, $V_{(x,y)}$ is the best set of omni directional velocity, α is a weight factor for *vel* cost function, β is a weight factor for *heading* cost function, min is the minimizing function and γ is a weight factor for $clear$ cost function.

the robot was designed to move with both curvature movement and omnidirectional movement. Equation 3.29 was used for curvature movement. When the angle between the goal and heading of the robot is more than 40 degrees, The curvature movement is active. It changes the heading of the robot to the goal location. Equation 3.30 was used for omnidirectional movement.

CHAPTER 4
RESULT AND DISCUSSION

4.1 Overview

In this thesis, ROS was implemented in the robot to use the DWA with ORB SLAM2. "DWA" node receives the data from the "orb_slam2_rgbd" node and navigates the robot to the goal. The data contain the location of the robot in the world coordinate and the position of the obstacle in the 3D point cloud. The point cloud data was filtered to remove the point at the ground and the 3D point that is higher than the robot heigh. The path planner receives only the x and y positions of the point cloud to navigate the robot. The map of the ORB SLAM 2 is a 3D sparse map. Moreover, the map and position of the robot are visualized in RVIZ.

The robot has two operation modes which are the manual mode and navigation and mapping mode. In the manual mode, the robot receives the position setpoint from the user via the "teleop_twist_keyboard" node in ROS. The RGB-D camera are deactivated. The robot localizes with wheel odometry, although the current position did not visualize on the map. The PID and the robot's kinematic model was implemented to control the position and movement of the robot. In the mapping and navigation mode, DWA node receives the goal from the user and navigates the robot to the goal with a safe path. This mode is separated into two modes. First, the mapping mode, the main task of this mode is creating the map of the environment. Thus, the robot will check the unknown space and rotate the camera to the unknown to explore the new area before heading to the goal location. The flow chart for mapping mode is showed in **Figure 4.1.** Second, the navigation mode, the main task of this mode is traveling to goal with mapped area. Moreover, this mode was operated with the localization mode in the ORB SLAM node. Thus, the robot not collect the new map point. The flow chart for navigation mode is showed in **Figure 4.2.**

Figure 4.1

The Flow Chart for Mapping Process

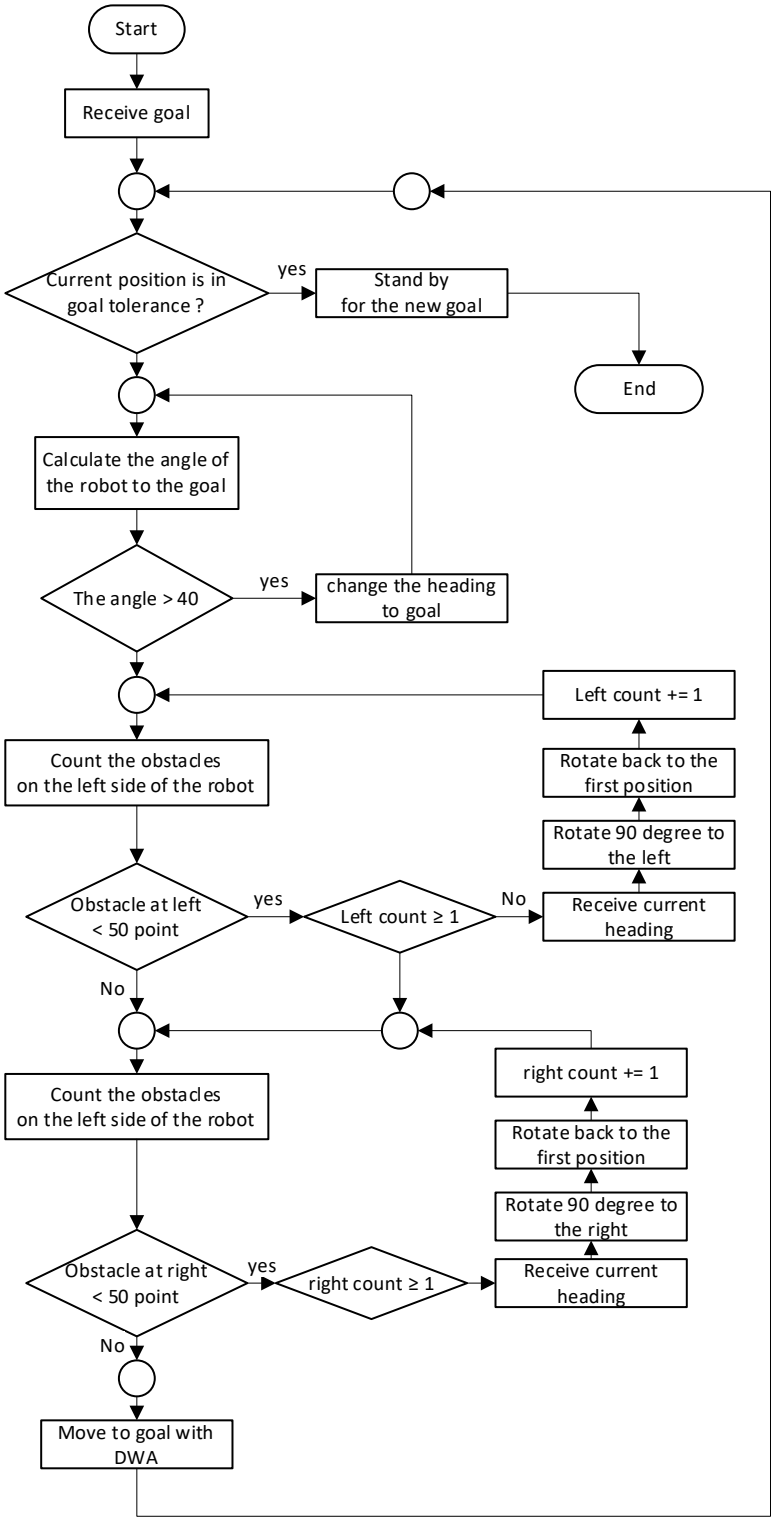
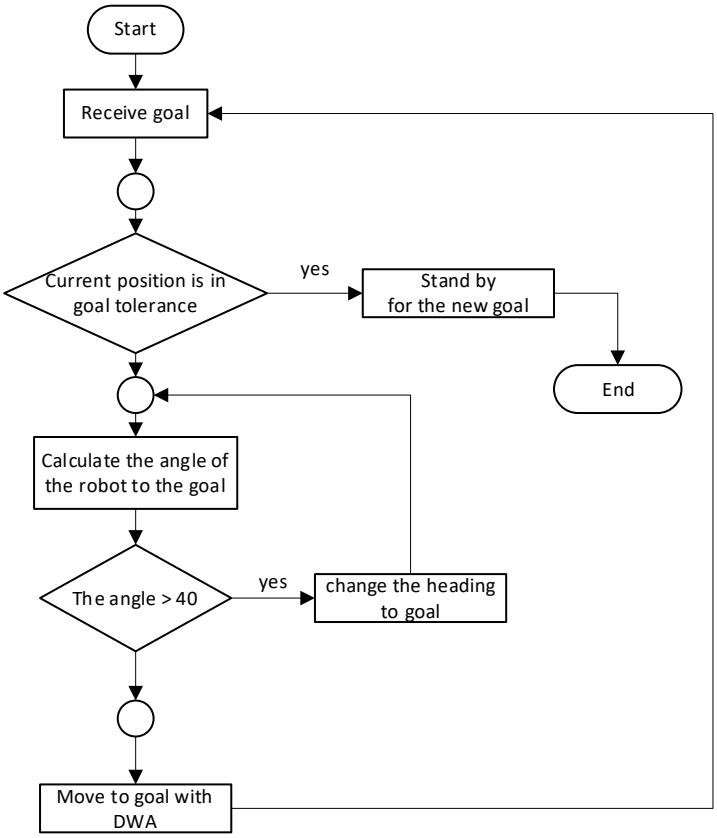


Figure 4.2

The Flow Chart for Navigation With Mapped Area



4.2 Robot Performance

This section presents the robot's performance in manual mode. The PID controller was implemented to control the position of the mobile robot. For the experiment, the robot will receive the command to move in 1 meter 5 direction and change the heading direction. These directions are forward, left, right, left diagonal and right diagonal. The result of the moving experiment shown in table 4.1 and 4.2 and the result of the changing heading direction shown in table 4.3. The result compares the setpoint with the feedback data from the robot and the data from the measuring tool. For each setpoint, the robot moves with the same setpoint and measure the error 3 times. In additional, the set point that sent to the robot is the pair of the distance (x, y). For example, the command is moving forward in 1 meter. The setpoint is (1,0). If the command is moving left in 1 meter. The setpoint is (0,1). The direction of each setpoint represent in **Figure 4.3.**

Figure 4.3

The Setpoint Direction

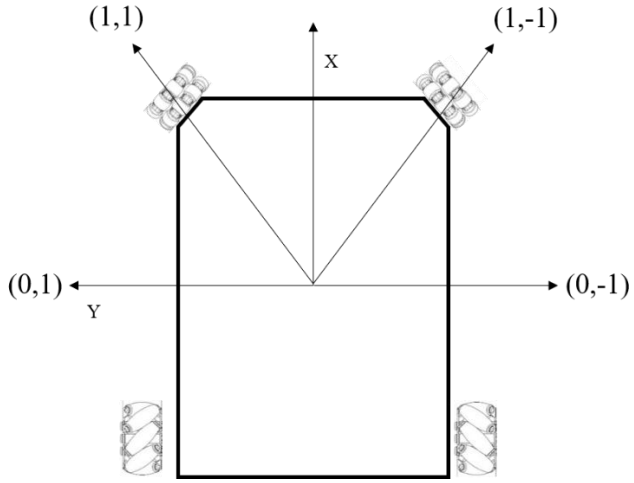


Table 4.1

The Position Control Analysis (Translation)

	Coordinate (X, Y)	measuring(error)	Encoder (X, Y)	r (measuring)
1	(0,0)	(0,0)	(0,0)	0.000
2	(1,0)	(0.95,0.03)	(0.97,0.005)	0.058
3	(1,0)	(0.97,0.02)	(0.99,-0.001)	0.036
4	(1,0)	(0.97,0.02)	(0.99,-0.007)	0.036
5	(0,1)	(-0.06,0.91)	(-0.006,0.97)	0.100
6	(0,1)	(-0.04,0.91)	(-0.01,0.97)	0.089
7	(0,1)	(-0.05,0.9)	(-0.003,0.96)	0.051
8	(0,-1)	(-0.003,-0.9)	(-0.01,-0.97)	0.100
9	(0,-1)	(-0.01,-0.89)	(0.01,-1.08)	0.110
10	(0,-1)	(0.0,-0.89)	(0.01,-0.96)	0.110
11	(1,1)	(0.95,0.91)	(0.98,0.97)	0.094
12	(1,1)	(0.93,0.92)	(0.98,0.96)	0.106
13	(1,1)	(0.89,0.93)	(0.98,0.96)	0.130
14	(1,-1)	(0.99,-0.9)	(0.98,-0.97)	0.010
15	(1,-1)	(0.95,-0.94)	(0.98,-0.97)	0.050
16	(1,-1)	(0.98,-0.92)	(0.98,-0.97)	0.020
	Average error			0.07

SAccording to the data in table 4.1, All data is in meter unit. The r (measuring) is the error from measurement with the measuring tool. The error calculates by using Euclidean distance between the final position of the robot and goal location. The result uses the measuring tool as a reference. The result shows that the maximum error is 0.13 m. from setpoint (1,1). The minimum error is 0.01m. from setpoint (1, -1). Moreover, the measuring from feedback of the robot and the error from measuring tool are not equal. The error of the measuring shown in Table 4.2.

Table 4.2

The Error of the Feedback Data

Coordinate (X, Y)	r (measuring tool)	r (encoder)	r (Encoder to measuring tool)
(0,0)	0.000	0.000	0.000
(1,0)	0.058	0.030	0.028
(1,0)	0.036	0.010	0.026
(1,0)	0.036	0.012	0.024
(0,1)	0.100	0.031	0.069
(0,1)	0.089	0.032	0.058
(0,1)	0.051	0.040	0.011
(0,-1)	0.100	0.032	0.068
(0,-1)	0.110	0.081	0.030
(0,-1)	0.110	0.041	0.069
(1,1)	0.094	0.036	0.058
(1,1)	0.106	0.045	0.062
(1,1)	0.130	0.045	0.086
(1,-1)	0.010	0.045	0.035
(1,-1)	0.050	0.045	0.005
(1,-1)	0.020	0.045	0.025
Average error	0.07	0.036	0.041

According to table 4.2, the data show the error of feedback data from the robot compare with data from measuring tool. The average error from encoder is 0.036 and the average error from the comparing is 0.041. On the other hand, the result from the robot is from the wheel encoder that counts only the wheel rotation. Thus, the wheel slip can cause an error for the sensor.

Table 4.3

The Position Control Analysis (Rotation)

	Theta(rad)	measuring tool(rad)	encoder(rad)	Dth(measuring)
1	1.571	1.64	1.55	0.069
2	1.571	1.62	1.56	0.049
3	1.571	1.61	1.54	0.039
4	-1.571	-1.589	-1.543	0.018
5	-1.571	-1.589	-1.542	0.018
6	-1.571	-1.571	-1.52	0
7	0.786	0.768	0.774	0.018
8	0.786	0.803	0.787	0.017
9	0.786	0.803	0.779	0.017
10	-0.786	-0.821	-0.772	0.035
11	-0.786	-0.803	-0.769	0.017
12	-0.786	-0.803	-0.772	0.017
	Average error			0.026

According to Table 4.3, the data show that the green box means the highest error in that setpoint. The error from measuring tool and the error from the encoder show in table 4.4

Table 4.4

The Error of the Feedback Data

	Setpoint(rad)	Dth(measuring)	Dth(encoder)	Dth (Encoder to measuing)
1	1.571	0.069	0.021	0.09
2	1.571	0.049	0.011	0.06
3	1.571	0.039	0.031	0.07
4	-1.571	0.018	0.028	0.046
5	-1.571	0.018	0.029	0.047
6	-1.571	0	0.051	0.051
7	0.786	0.018	0.012	0.006
8	0.786	0.017	0.001	0.016
9	0.786	0.017	0.007	0.024
10	-0.786	0.035	0.014	0.049
11	-0.786	0.017	0.017	0.034
12	-0.786	0.017	0.014	0.031
	Average error	0.026	0.020	0.044

4.3 Localization

This part shows the localization of the mobile robot in mapped area with ORB SLAM. The ORB SLAM node is activated, and the localization mode, the mode in ORB SLAM is turned on. The robot was placed in the different position in the map. The system is restarted when the robot is placed on the new location in the map. The ground truth is measured by using measuring tool. The error of the localization is represented in Table 4.3. The error is the Euclidian distance of the robot to the goal and the distance along the x, y direction in meter.

Table 4.5

The Localization Analysis

	Coordinate(X,Y)	Robot(X,Y)	Dx	Dy	r
1	(0,0)	(0,0)	0	0	0.00
2	(0,1)	(0.17,0.95)	-0.17	0.05	0.18
3	(-0.5,-1.0)	(-0.45,-1.03)	0.05	0.03	0.06
4	(1.0,0.0)	(1.07,-0.006)	0.07	-0.006	0.07
5	(1.5,0.8)	(1.59,0.78)	-0.09	0.02	0.09
6	(3.0,0.6)	(3.14,0.59)	0.14	0.01	0.14
7	(3.5,-3.1)	(3.68,-3.13)	-0.18	0.03	0.18
8	(3.5,-5.1)	(3.73,-5.12)	-0.23	0.02	0.23
9	(3.5,-7.1)	(3.78,-7.11)	-0.28	0.01	0.28
	Average error				0.14

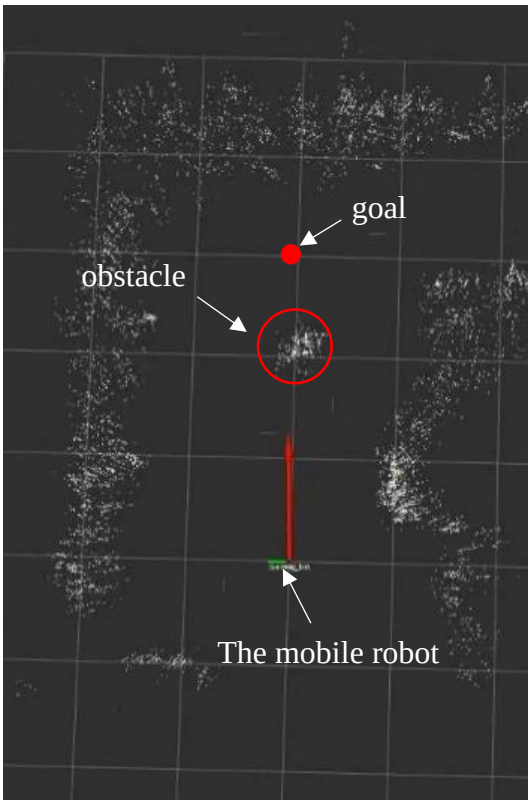
From the Table 4.3, It can be seen that the error increase when the location is far from the origin point (0,0). According to Dx and Dy it shows the accumulated error from Dx is more than Dy, especially the point that far from the starter point (0,0).

4.4 Navigation

This section presents the navigation system of the robot. The sets of parameters in the dynamic window were adjusted. The set of parameters contain the weight factor of the goal cost function and the weight factor of the obstacle cost function to find the cost function that can navigates the robot to the goal and avoid the obstacle. The goal was expanded from point to square shape. The size of the square shape is that 10 cm. from the center point. For the experiment, the navigation mode of the robot is activated. Thus, it knows only the obstacles in the maps. Then, the robot was placed on the origin point of the mapped area. The 1 obstacle was installed in 2 meters at the front of the robot. The goal of the robot is behind of the obstacle for 1 meter. The map for the experiment shown in **Figure 4.4.** The result of the experiment shown in Table 4.4

Figure 4.4

The Map for Navigation Experiment



According to the **Figure 4.4,** the red point is the location of the goal, The beginning of the arrow is the robot and the group of point cloud in the red circle is obstacle.

Table 4.6

The Navigation Analysis

	goal gain	ob gain	Goal tolerance	Avoidance	Goal(X,Y)
1	0.2	0.2	0.1 m.	False	False
2	0.2	0.4	0.1 m.	False	False
3	0.2	0.6	0.1 m.	False	False
4	0.2	1	0.1 m.	False	False
5	0.4	0.2	0.1 m.	True	True
6	0.4	0.4	0.1 m.	False	False
7	0.4	0.6	0.1 m.	False	False
8	0.4	1	0.1 m.	False	False
9	0.6	0.2	0.1 m.	True	True
10	0.6	0.4	0.1 m.	False	False
11	0.6	0.6	0.1 m.	False	False
12	0.6	1	0.1 m.	False	False
13	1	0.2	0.1 m.	True	True
14	1	0.4	0.1 m.	True	True
15	1	0.6	0.1 m.	True	True
16	1	1	0.1 m.	False	False

According to avoidance and goal in table 4.4, the avoidance in the table refer to the ability to avoid the obstacle of the parameter. If it is True, the set of parameters can avoid the collisions. The column of Goal represents the condition of the robot that can travel to the goal location. If it is True, the set of parameters can navigate the robot to the goal location. The data show that the parameter that the goal gain is more than ob gain can navigate the robot to goal and avoid the obstacle except for the set of the goal gain of 0.6 and the ob gain of 0.4. However, for each parameter that cannot navigate the robot to the goal, the robot stops in front of the obstacle and stop in the gap between the wall and the obstacle. Each parameter that can navigate the robot to the goal, the errors from goal was calculated with the data from the robot. The result show in Table 4.5

Table 4.7

The Error of the Navigation

	measuring(error)	Camera(X,Y)	r (measuring)	r (camera)
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	-	-
5	(2.75 , 0.01)	(2.91 ,- 0.21)	0.25	0.23
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	(2.71,0.03)	(2.95 , 0.06)	0.29	0.08
10	-	-	-	-
11	-	-	-	-
12	-	-	-	-
13	(2.78,0.04)	(2.97, - 0.01)	0.22	0.03
14	(2.82,0.04)	(3.01,- 0.14)	0.18	0.14
15	(2.74,0.06)	(2.94,- 0.09)	0.27	0.11
	Average error		0.24	0.12

According to Table 4.5, the data show that the final position of the robot stops in the goal tolerance.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATION

5.1 Conclusion

The mobile robot is designed based on the new configuration wheels, and the robot was able to perform SLAM by using a camera. The main idea of the new design is the combination of the benefit of the mecanum wheel and the Omni wheel. I was able to build the new mathematic model by using the regular mecanum wheel mathematic model as a reference. Finally, the new kinematic model was derived to the new wheel configuration and test the performance. The robot can perform the omnidirectional movement. After confirming the new concept, the PID controller was applied to control the movement of the robot. Next part was the SLAM system, The Kinect, the low-cost RGB-D camera was chosen as the main sensor. ROS was used to develop the robot SLAM and navigation system, including the control system. ROS has supported ORB SLAM2, the open-source SLAM for monocular, stereo and RGB-D camera. Moreover, ROS has supported the "openni_camera" package, the driver for the RGB-D camera in ubuntu, and the camera calibration package. For the microcontroller, the "serial_node_stm32" was used to connect the stm32 board with the other nodes. After implement ROS system and ROS package, the navigation system was implemented based on the idea of an autonomous mapping process. The dynamic window approach was used to navigate the robot to the goal, avoid the obstacle and create the map. The cost function in the approach was adjusted in order to support the omnidirectional movement. Finally, the robot has 3 operation mode. The first mode is that the manual mode used position control to control the robot. The robot will receive a direction and a distance to travel. The second mode is mapping mode. The dynamic window approach and ORB SLAM was activated to navigate the robot and create the map. The robot will receive the goal in a 2D coordinate. Then, the robot travels to the target and rotates to the unknown space of the map around it. In this mode, the robot can avoid the obstacle along with creating the map. mode is similar to the mapping mode, but the navigation mode operates in the mapped area. Thus, the robot did not need to find the unknown space to complete the map. It only focuses on travel to the goal and avoids collision. The only navigation is the last mode of the robot. The operation of this mode is similar to the

mapping mode, but the navigation mode operates in the mapped area. Thus, the robot did not need to find the unknown space to complete the map. It only focuses on travel to the goal and avoids collision. The mobile robot was experimented to find the robot performance. The result show that the robot can travel to goal and autonomously create the map while avoid the obstacles. Moreover, it was able to navigate itself to the goal in the mapped area.

To summarize, this thesis designs an autonomous mobile robot with a hybrid mecanum-Omni wheel configuration. The kinematic model can be used with the hybrid wheel configuration. The ORB SLAM2, with a dynamic window approach, be able to create the map and navigate the robot to the goal while avoiding the obstacles.

5.2 Recommendation

1. Design the global path planning. The DWA is the local path planning. In some situations, it needs the people to lead it to the goal. Global planning might solve this issue, and the DWA uses to navigate the robot to the global path instead.

2. Create the local map for the robot. The navigation of the robot uses a static map to navigate the robot. If the robot has the local map, the environment around it will be updated. Thus, the robot will detect the dynamic obstacle and avoid it. Moreover, the obstacles that did not stay in the real world will be removed from the local map to reduce the operation time.

3. Apply the kinematic model of the mobile robot to the ORB SLAM. The tracking process of the ORB SLAM localizes the position of the camera by using the velocity motion model. If the kinematic model of the robot is applied to the motion model, the SLAM system will estimate the camera pose with more accuracy.

REFERENCES

Adăscăliţei, Florentina, and Ioan Doroftei (2011). Practical applications for mobile robots based on mecanum wheels-a systematic survey. *The Romanian Review Precision Mechanics, Optics and Mechatronics* 40 21-29.

Taketomi, T., Uchiyama, H. & Ikeda, S (2017). Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ T Comput Vis Appl* 9, 16

Mur-Artal, Raul & Tardos, Juan. (2015). Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM.

S. Campbell, N. O'Mahony, A. Carvalho, L. Krpalkova, D. Riordan and J. Walsh (2020). "Where am I? Localization techniques for Mobile Robots A Review," 6th International Conference on Mechatronics and Robotics Engineering (ICMRE), Barcelona, Spain, 2020, pp. 43-47.

Fox, Dieter & Burgard, Wolfram. (1997). The Dynamic Window Approach to Collision Avoidance. Robotics & Automation Magazine, IEEE. 4. 23 - 33. 10.1109/100.580977.

R. Mur-Artal, J. M. M. Montiel and J. D. Tardós (2015). "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct.

N. A. Zainuddin, Y. M. Mustafah, Y. A. M. Shawgi and N. K. A. M. Rashid (2014)."Autonomous Navigation of Mobile Robot Using Kinect Sensor," International Conference on Computer and Communication Engineering pp. 28-31.

Younes, Georges, et al. (2017). "Keyframe-based monocular SLAM: design, survey, and future directions." *Robotics and Autonomous Systems* 98: 67-88.

Dissanayake, M.W.M. & Newman, P. & Clark, Steven & Durrant-Whyte, Hugh & Csorba, M.A.. (2001). A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. Robotics and Automation, IEEE Transactions on. 17. 229 - 241.

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

G. Klein and D. Murray (*2009).* "Parallel Tracking and Mapping on a camera phone," *8th IEEE International Symposium on Mixed and Augmented Reality*, Orlando, FL, 2009, pp. 83-86.

R. Mur-Artal and J. D. Tardós (2017). "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct.

Myint, Robot Cherry and Nu Nu Win (2016). “Position and Velocity control for Two-Wheel Differential Drive Mobile.”

Soni, Sanket, Trilok Mistry, and Jayesh Hanath (2014). "Experimental Analysis of Mecanum wheel and Omni-wheel." *International Journal of Innovative Science, Engineering & Technology* 1.3 292-295.

Shabalina, Ksenia & Sagitov, Artur & Magid, Evgeni. (2018). Comparative Analysis of Mobile Robot Wheels Design. 175-179.

K. Krinkin, E. Stotskaya and Y. Stotskiy (2015). "Design and implementation Raspberry Pi-based omni-wheel mobile robot," Artificial Intelligence and Natural Language and Information Extraction, social media and Web Search FRUCT Conference (AINL-ISMW FRUCT), St. Petersburg, 2015, pp. 39-45.

Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza (2011). *Introduction to autonomous mobile robots*. MIT press.