DEVELOPMENT OF MOTION PLANER AND NAVIGATION SYSTEM FOR AN OFFICE ASSISTANT ROBOT

LOOI CHEN ZHENG

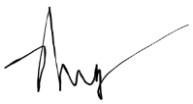
A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering

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

APRIL 2021

DECLARATION

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

Signature : 

Name : LOOI CHEN ZHENG

ID No. : 16UEB03663

Date : 17/4/2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DEVELOPMENT OF MOTION PLANER AND NAVIGATION SYSTEM FOR AN OFFICE ASSISTANT ROBOT"** was prepared by **LOOI CHEN ZHENG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ir. Danny Ng Wee Kiat

Date : 18/4/2021

ACKNOWLEDGEMENTS

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

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement.

ABSTRACT

In the past decades, service robot industry had rise rapidly. There are many types of service robot used in different field such as medical. Office assistant robot is one type of the service robot used to assist officers. The robot used in this research is a differential drive robot. In order for the robot to navigate autonomously in the office, navigation algorithm and motion planner were implemented on these robot. Robot Operating System (ROS) is one of the common platforms to developed these robots. Besides, map was generated by using SLAM algorithm, the costmap was setup and the parameters such as inflation_dist and cost_scaling_factor was studied and tuned based on this application. Next, the localization of robot was based on the LiDAR sensor and rotary encoder with AMCL algorithm. In this research the global planners, A* and Dijkstra algorithm and local planners, DWA and TEB algorithms were implemented and tested on the robot in in simulation and a real environment. Results from the experiments were used to evaluate and compare the performance of the robot with different planner and parameters. From the results obtained, the global planners, A* and Dijkstra algorithm both can achieve the required performance for this application whereas TEB outperforms DWA as the local planner due to its feasibility in avoiding dynamic obstacles in the experiments conducted.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

Q	sequence of pose
ΔT	Time interval
v	translational velocity
V_d	dynamic window
V_r	resultant search space
X	robot pose
α	weight of target heading
β	weight of clearance
γ	weight of velocity
θ	target heading angle, °
τ	sequence of time intervals
ω	rotational velocity
AGV	automated guided vehicles
AMCL	Abstract Adaptive Monte Carlo localization
DDQN	Double Q-network
DOL	Dynamic obstacle layer
DWA	Dynamic Window Approach
EBAND	Elastic Bands
IFR	International Federation of Robotics
LiDAR	light detection and ranging
ROS	Robot Operating System
SLAM	simultaneous localisation and mapping
TEB	Time Elastic Band

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Service robots are one of the types of robots that can move and perform various tasks based on the human order either fully autonomous or semi-autonomous. Based on The International Organization of Standardization, a service robot is defined as a robot that carry out useful work for humans. (ISO 8373::2012) Service robots are meant to be used in the task which is menial, repetitive, and time-consuming. This frees up more time for humans to perform more intellectual tasks. In the past decades, service robots had grown rapidly with technological advancement. In 2017, according to the International Federation of Robotics (IFR), the sales of service robots had increased 39% compared to 2016 which is around $6.6 billion. Besides, based on Len Calderon, it stated that service robots are predicted to grow rapidly and reach a sales value of around $37 billion. (Calderon,2019)

There are multiple types of service robot which include Pepper, a humanoid robot used to sell coffee maker in the store, Cobalt, a security robot that works in the office to detect unauthorized people or behaviours based on face recognition, employee scanner, and heat sensor and Medical robot which use for healthcare in the hospital. It can deliver medicine and detect patient messages based on built-in Artificial Intelligence. (Calderone,2019)

Office assistant robots are one type of service robot. The office assistant robot's main objective is to assist officers during working. For example, the robot is used for delivery application between the central cafe to offices done by Koubaa et al. (2016) and state prediction of the officers done by Kouno et al. (2012). The office assistant robot used in this research is a differential drive robot. This research is to develop a motion planner and autonomous navigation into the office assistant robot. This allows the robot to go somewhere based on the given goals without human intervention. The robot can localize itself and build a map based on the environment and sensor data. Besides, the robot can plan a path from the starting point and move to the goals with obstacles avoidance ability. The motion planner is separated into 2 parts which are global

planner and local planner. Global planner is used for optimal path planning with a known environment (global costmap) while local planner is based on the real-time sensor data such as LiDAR sensor to plan optimal trajectories which can modify the global path and also perform dynamic obstacles avoidance.

1.2 Importance of the Study

This study will mainly focus on developing the motion planner and navigation on the differential drive office assistant robot by using ROS. It allows the robot to navigate autonomously with given goals and perform obstacles avoidance. Besides, this study will discuss the results obtained of the motion planner in the office environment and give insight and compare different kind of path planner based on different journals, simulation and practical results.

1.3 Problem Statement

Currently, the office assistant robot used in this research is only able to perform manual control by a human. There are many limitations with manual control. Manual control robot is time-consuming. This is because the user required to focus on the robot to control the robot manually all the time. Besides, manual control is inaccurate and inconsistent. This is due to the controlled robot might be affected by the user's skill to control and physical condition. Moreover, although there are many studies regarding implementing different ROS planner to different kinds of wheel robot but only a few studies are comparing the different kind of planner based on its performance and parameters. Hence, this study is to implement the autonomous motion planner and navigation on the office assistant robot by using ROS and investigate how the parameters affecting the planner.

1.4 Aim and Objectives

This research aims to develop a motion planner and navigation system for an office assistant robot by using ROS. This allows the robot to autonomous navigate in the office environment based on the user requirement with the help of different sensor and algorithm. The detailed objectives are:

- Study and compare the feasibility of different type of global and local planner in ROS.

- Study and build a map with SLAM algorithm by using office assistant robot.

1.5 Scope and Limitation of the Study

The scope of this research is to develop and implement the motion planner and navigation system on an office assistant robot. The first part of the research is building a map by using SLAM algorithm and simulation in Gazebo. The second part will be comparing and discussing the result obtained to select the most suitable motion planner in this application.

There are other planners such as double Q-network (DDQN) deep reinforcement learning which are not included in the ROS packages. Due to the limitation of knowledge and time, this research will only discuss the global and local planner packages in ROS. Moreover, the office assistant robot used in this research is a differential drive robot. Therefore, due to this limitation, this study is only focusing on implementing motion planner on the differential drive robot instead of other types of mobile robot such as Omni and Ackerman drive robot.

1.6 Contribution of the Study

The contribution of the study is to have a details comparison between the local planner, implement motion planner on the office assistant robot and demonstration using ROS to autonomously navigate the robot in the office environment.

1.7 Outline of the report

Chapter 1 will have a brief introduction about service robot, the problem statement and the objectives needed to be achieved in this research.

Chapter 2 will undergo literature reviews about ROS, different global planner and local planer and its comparison. While Chapter 3 will explain the methodology about the step to implementing the motion planner and the experiment to compare different planner.

Chapter 4 will be tabulated and discuss the result obtained. Lastly, Chapter 5 will be concluding the results obtained, selecting the suitable planner and suggestion for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 ROS

ROS is an open-source meta-operating system. It was first developed by Willow Garage and Stanford University as a project named STAIR project. It is a useful tool for building a complex robotic system. This is because it provides many types of services such as hardware abstraction, message-passing between processes package management and low-level device control. (Dattalo,2018). Nowadays, it is widely used in the robotics community because it is easy to use, open-source and has many library/packages that are ready prepared. ROS can perform several types of communication including Services, Topics and Parameter Server.

Node is an individual module that performs computation. A robot control system usually consists of multiple nodes, each node is responsible for different tasks. For example, one node controlling the sensors, one node controlling the motor of the robot and one node perform path planning. The nodes can be communicated through Topic. Topic is a type of asynchronous communication which allow the nodes passing a message to each other with the help of publisher and subscriber. A node can publish the messages to a given topic while the other node is allowed to subscribe to the given topic to obtain the data. The topic is allowed to be subscribed/published by more than 1 of the publisher and subscriber and each node is allowed to be published and subscribed to many topics which are known as many-to-many or 1-way transport communication method. (Aitken, Veres and Judge,2014). Unlike Topics, Services is a synchronous communication method used for reply and request. For instances, nodes provide a service with a given name and a client send the request and wait for the reply by calling the services. Figure 2.1 shown the communication between nodes. Moreover, there is a Parameter Server which allows data to be stored in the server and can be obtained once needed. All this is controlled by the Master which offer a name registration. Figure 2.2 shows the basic structure of ROS.

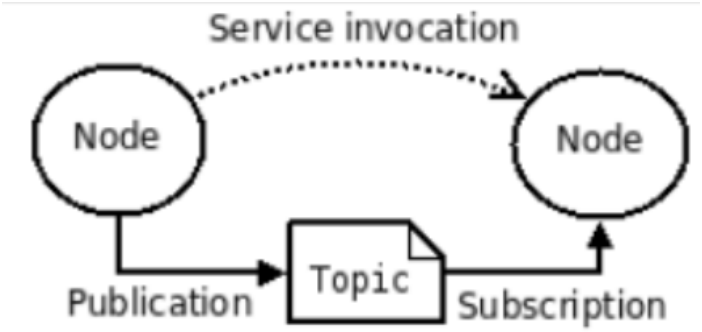


Figure 2.1: Communication between nodes (ROS Wiki, 2014).

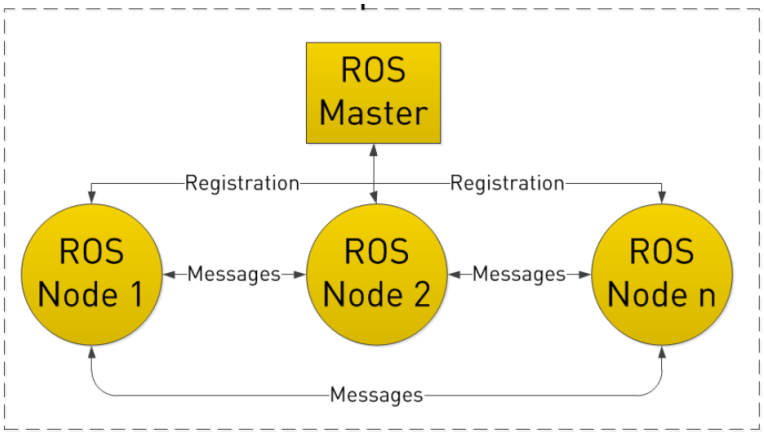


Figure 2.2: Basic structure of ROS (Clearpath Robotics,2014).

2.2 History of Motion Planning

In the year 1979, Path planning was first introduced in the autonomous mobile robot by Lozano-Pérez and Wesley with the concept of configuration space(C-space). (Lozano-Pérez and Wesley,1979).

In the past few decades, motion planning is playing a more and more important role in different field of robotics such as industry. This is because the robotics industry had growth and autonomous navigation had become one of the key elements in building a mobile robot especially service robot. Therefore, nowadays, there are many different kinds of researches on the path planning and motion control of the robots. Path planning is an algorithm used to generate the path for the robot to move to the given goal. There are many types of path planning used depends on the task needed to be performed by the robots. For instances, automated guided vehicles (AGV) robot used in the factory is usually

moving in a standard path while for more advance applications, more robust path planning algorithm is introduced which allow the robot to navigate in a dynamic environment. (Gasparetto, et al.,2015). While motion control is to smoothly control the robot by sending velocity commands to the robot to track the path planned.

The motion planning is the key for the mobile robot to perform navigation. There are many motion planning algorithms that had been developed.

2.3 Global Planner

The global planner is the algorithm used to generate an optimum path based on the map built. There are several methods on forming paths such as dividing the map into different cell/regions and compute called cell decomposition. The method review in this research is assigning value to each region of the map and compute to find a minimum cost path such as Dijkstra and A* algorithm.

2.3.1 Dijkstra Algorithm

The map is divided into cells and form a grid cell map as shown in Figure 2.3. The values are assigned to the cells or nodes that the robot can move freely based on the map. Start with the starting point, each node consists of a value based on the distance between the robot and the nodes. Each generation, it will compute the neighbour’s empty nodes and assign the values to the empty nodes until it reaches the goal as shown in Figure 2.3. The minimum values path is selected as the path generated. (Marin, et al.,2018).

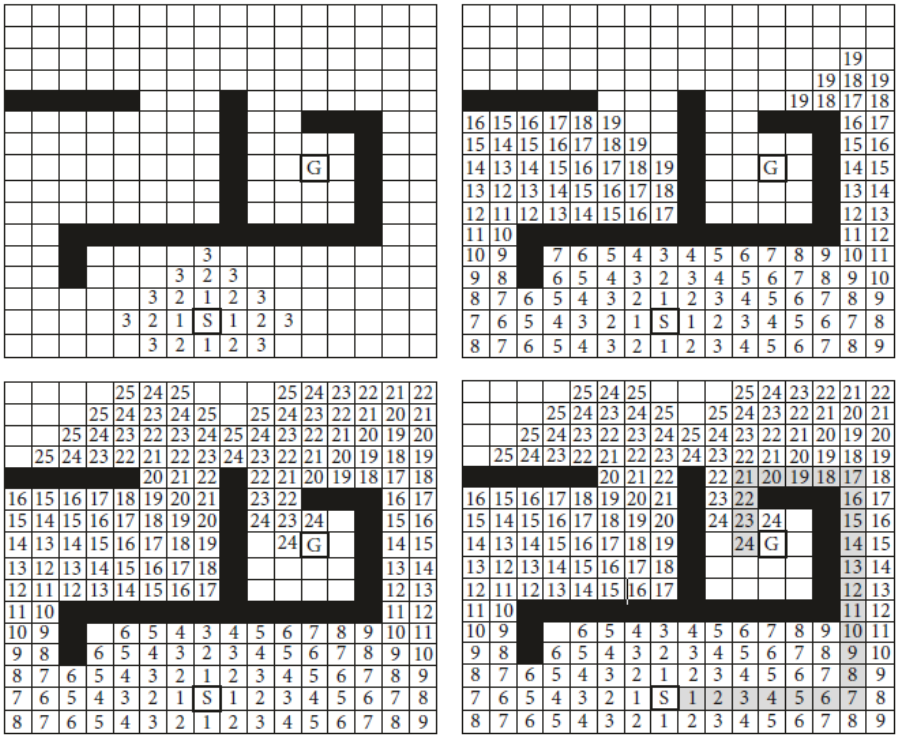


Figure 2.3: Dijkstra algorithm (Marin, et al.,2018)

2.3.2 A* Algorithm

A*algorithm is similar to Dijkstra while A* assigning one more value to the nodes with a heuristic function to estimate the distance between the goal point and the nodes. This heuristic function can be a Euclidian distance method or Manhattan distance method. With the combination of heuristic function and the original cost function, in the Dijkstra algorithm, the new cost function is generated. (Pittner, et al.,2018).

$$f(X) = g(X) + h(X) \qquad (2.1)$$

where

g(X) = original cost function of Dijkstra algorithm

h(X) = heuristic function

2.3.3 Comparison of the global planner

Dijkstra and A* algorithm are compared in term of their computation time and path length. In a simple environment, based on Table 2.1, with the same path length, A* algorithm has shorter computation time especially with Manhattan

distance equation as heuristic function. This is because Dijkstra algorithm explores unused cell/nodes in an undirected fashion. In a more complex environment, based on Table 2.2, A* algorithm is faster than Dijkstra algorithm. Based on Figure 2.4, One of the advantages of using Dijkstra is it can shorter the path length when using Gradient descent method to reduce the path length. A* algorithm is unable to reduce the path length effectively because Gradient descent method required more nodes explored to use for computation. However, with the combination of a local planner, A* algorithm can also shorter the path length with less computation power needed. (Pittner, et al.,2018).

Table 2.1: Comparison between Dijkstra and A* algorithm in a simple environment (Pittner, et al.,2018)

Algorithm	Path length	Computing time
Dijkstra	14, 35m	14, 45 μs
A*(Manhattan)	14, 35m	0, 15 μs
A*(Euclidean)	14, 35m	0, 29 μs

Table 2.2: Comparison between Dijkstra and A* algorithm in a complex environment (Pittner, et al.,2018)

Algorithm	Computing time
Dijkstra	70, 63 μs
A*(Manhattan)	4, 69 μs
A*(Euclidean)	37, 65 μs

Figure 2.4: Blue line indicates path planned without gradient descent method, red line indicate path with gradient descent method. (Pittner, et al.,2018)

2.4 Local Planner

Local planner is to generate a trajectory based on the path planned by global planner. It generates a suitable waypoint from the path planned which based on the dynamic constraints of the robots and real-time sensor data around the robot(local map). There are multiple local planners already built as packages in the ROS. (Marin, et al.,2018).

2.4.1 Dynamic Window Approach

Dynamic Window Approach is a trajectory planning algorithm developed by the Dieter Fox, Wolfram Burgard and Sebastian Thrun in 1997. This algorithm is using the dynamics of the robot to do the planning. It will sample the velocities of the robot, and compute multiple approximations of trajectories in an interval of time. The approximation of trajectories will result in a 2-Dimensional search space (Ferrer Sánchez,2018).the trajectories in the search space are based on 3 criteria:-

i. Circular trajectories.
ii. Dynamic window.
iii. Admissible velocities.

For the robot to reach the goal with the trajectories planned, the velocity vector is computed at a certain time interval and input to the robot. This velocity vector which is determined by translational and rotational velocity of the robot is known as circular trajectories. Moreover, admissible velocities are the velocities that the robot able to move safely which mean is the velocity that the robot can stop before colliding the obstacles.

Besides, every robot had its acceleration limit. The trajectories in the search space will be reduced to the dynamic window based on the acceleration limit of the robot. The dynamic window,V_d only contain velocities that are reachable by the robot within the time interval. Hence the resultant search space, V_r is the combination of these three criteria shown in Figure 2.5.

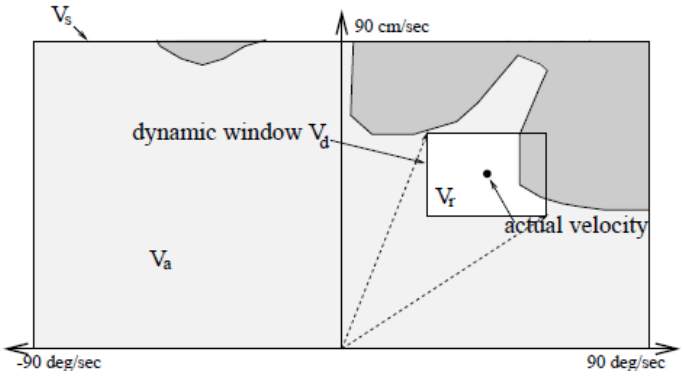


Figure 2.5: Search space (Ferrer Sánchez,2018).

The resultant search space will be optimized to obtain the maximum cost/ best trajectories. The remain trajectories in the search space will be computed with this cost function.

$$G(v, \omega) = \sigma(\alpha * \text{angle}(v, \omega) + \beta * \text{dist}(v, \omega) + \Upsilon * \text{vel}(v, \omega)) \qquad (2.2)$$

where

v = tranaslational velocity

ω = rotational velocity

α = weight of target heading

β = weight of clearance

Υ = weight of velocity

This cost function is respect to the current position and orientation of the robot. Trajectories with the highest cost will be selected. There are 3 parameters affecting the cost function.

i. Target heading, $\text{angle}(v, \omega)$
ii. Clearance, $\text{dist}(v, \omega)$
iii. Velocity, $\text{vel}(v, \omega)$

Each of the parameters is normalized to [1,0]. α, β, Υ is the weight of each parameter. Target heading is defined as the relative angle between the current orientation of the robot and the goal direction. Since the robot orientation is based on the circular trajectories, the target heading angle, θ will be calculated for the predicted position of the robot which is shown in Figure 2.6. Figure 2.7

shows the relationship between target heading with a set of rotational and translational velocity of the robot. It shows that when the rotational velocity increase, the function cost declined due to the robot is turning away from the target.

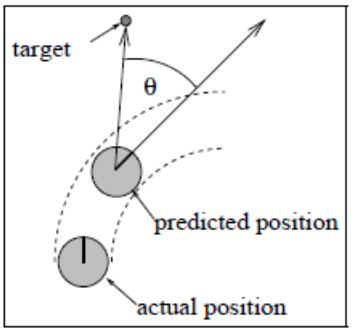


Figure 2.6: Target heading, θ and predicted position of the robot. (Fox, Burgard and Thrun,1997).

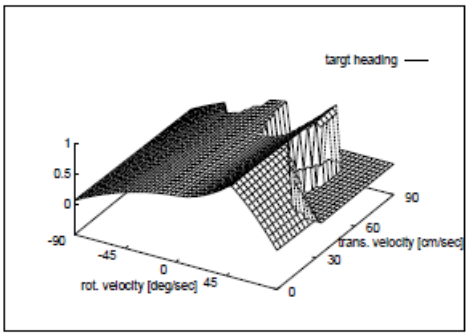


Figure 2.7: Target heading against rotational and translation velocity. (Fox, Burgard and Thrun, 1997).

Moreover, Clearance is based on the distance between the obstacles and robot position which intersect the trajectories in the resultant search space. While velocity is the robot velocity on the current trajectory. Hence, the cost is calculated according to these 3 criteria. The highest cost of trajectory will be the robot trajectory. (Fox, Burgard and Thrun, 1997).

2.4.2 EBAND local planner

Elastic bands (EBAND) local planner is one of the types of local planner which used to the autonomous navigation system. This algorithm is used to modify the path generated by the path planner. It uses two components to generate the free-

collision path which is ‘repulsive force’ and ‘contraction force’. Contraction force is used to form the tension of the band while the repulsion force is exerted on the elastic bands by the obstacles. Based on Figure 2.8-a, it is the path generated by the path planner, after applying the elastic bands with both forces the robot will move in the Figure 2.8-d path. (Quinlan and Khatib, 1993)

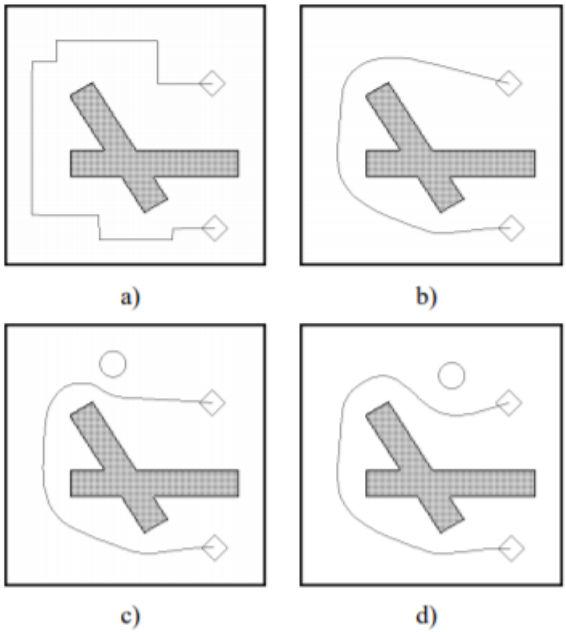


Figure 2.8: Elastic band planner (Quinlan and Khatib,1993)

2.4.3 TEB planner

Time Elastic Band (TEB) planner is an extension of the EBAND planner. It creates a local plan which consists of a sequence, n of robot poses, based on the path planned by the global planner. (Cybulski, Wegierska and Granosik, 2019). The robot poses, X consists of 3 elements which are position and orientation based on the related frame, {map} which shown in Figure 2.9. This sequence of the pose is defined as:

$$Q = \{X_i\}_{i=0\ldots n} \qquad (2.3)$$

where

X = robot pose

Q = sequence of robot pose

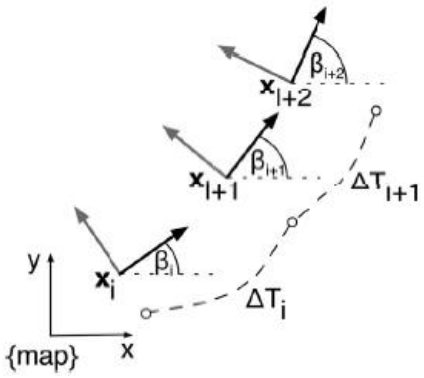


Figure 2.9: Sequence of the configuration of robot respect to map frame. (Rösmann, et al.,2012).

Unlike Elastic band planner, TEB is computed between 2 consecutive configurations between a time interval and forming a sequence of time difference which denoted as τ.

$$\tau = \{\Delta T_i\}_{i=0\ldots n-1} \qquad (2.4)$$

where

ΔT = Time interval

τ = Sequence of Time interval

The TEB is represented as:

$$B = (Q, \tau) \qquad (2.5)$$

Then the TEB, B is optimized by the objective function. The TEB, B with minimum cost will be selected. The objective function is based on several parameters such as the dynamic constraints of the robots and the distance between obstacles. Moreover, the translational and rotational velocity is computed based on the 2 consecutive configurations, and with the time intervals between these two consecutive poses. Then, the acceleration of the robot is just the velocity changes in the time intervals. (Rösmann, et al.,2012). Besides, TEB also can perform parallel trajectories planning. This means if one of the trajectories suddenly blocked such as closing the door, the robot can follow an

alternative trajectory. However, this will increase the computation power but result in robot able to move faster. (Rösmann, et al.,2012). Besides, based on (Marin, et al.,2018), one of the disadvantages of TEB is, when obstacles are moving across the robot as shown in Figure 2.10, the trajectory produced path length will increase.

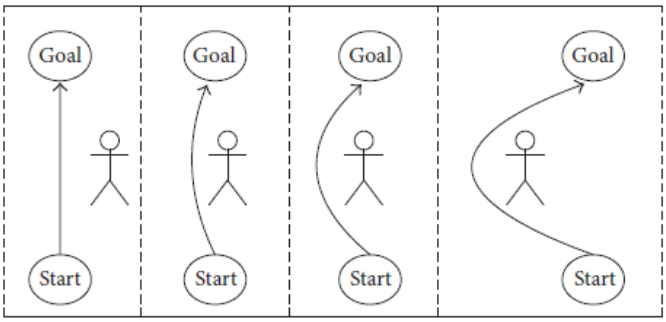


Figure 2.10: Dynamic obstacles moving from across the robot. (Marin, et al.,2018).

2.5 Comparison of local planner

When the start poses of the robot in the opposite direction to the goal. EBAND algorithm is generating the shortest path. This is due to there are no obstacles which exerted the repulsive force to the path planned. Therefore, the robot simply just rotates 180° and then drive straight to the goal. While the TEB algorithm computes the transition, the time required and the dynamic constraints of the robot between configurations. Hence, it generates a trajectory with a shorter transition time than EBAND. While DWA algorithm, has higher weighting on positive translational, hence the robot tends to move forward. The path generated by the different local planner is shown in Figure 2.11.

Moreover, in the case of static obstacles, EBAND algorithm failed to find an optimal path for the robot. While TEB and DWA successfully find an admissible path. This is because the obstacles exerted the ‘repulsive force’ on the EBAND trajectory which form a tangential rather than a circumnavigation. The result is shown in Figure 2.12. (Pittner, et al.,2018). Besides, in the case of dynamic obstacles, DWA algorithm collides with the obstacles which are shown in Figure 2.13. This is because, when moving forward DWA algorithm detect no obstacles in front. Hence it will move in high velocity. When the obstacles

suddenly moving toward the robot, DWA unable to stop immediately cause having a high cost of forward velocity.

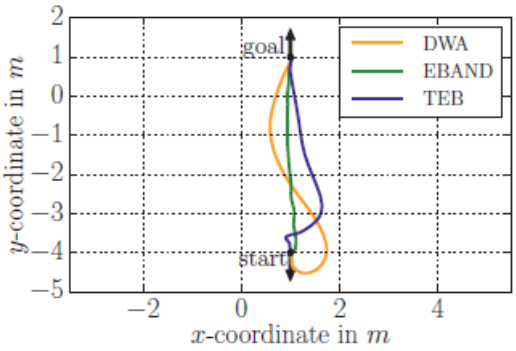


Figure 2.11: Path planned when the start position is the opposite direction to goal. (Pittner, et al.,2018).

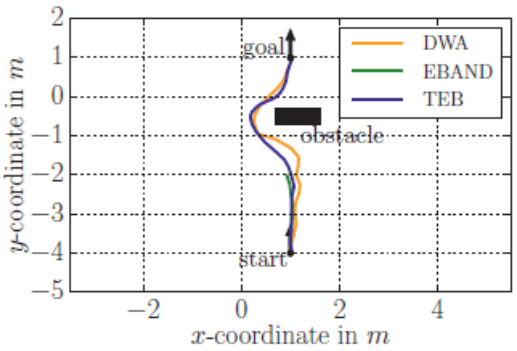


Figure 2.12: Path planned with static obstacles. (Pittner, et al.,2018).

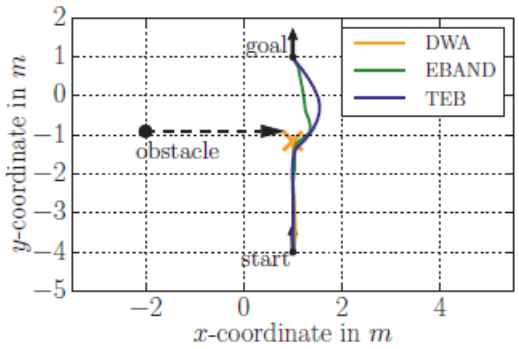


Figure 2.13: Path planned with dynamic obstacles. (Pittner, et al.,2018).

On top of another testing was undergo in different research which shown in Figure 2.14. EBAND algorithm had difficulties when going navigating. This may due to when turning to a corner, EBAND is sensitive to affect the

acceleration of the robot. Besides, Unlike DWA need extra time to replan the path when passing the obstacles, TEB able to move to pass the obstacles smoother and faster. Moreover, it conclude that each planner has its advantages. EBAND has higher accuracy, DWA has better consistency while TEB is the fastest time. (Cybulski, Wegierska and Granosiki, 2019)

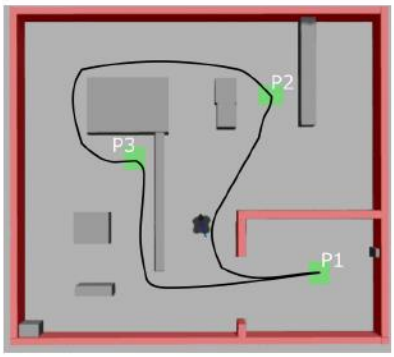


Figure 2.14: The path generated by the robot from P1 to P3. (Cybulski, Wegierska and Granosik,2019).

2.6 Summary

According to the literature review, A* algorithm is having the shorter path length and faster computation time than Dijkstra algorithm. Besides, Since DWA and TEB algorithm both has its advantages, it is worth to investigate more about its performance to determine which more suitable in this application. Therefore, A* algorithm as a global planner, DWA and TEB algorithm as the local planner will be compared and selected to implement in the office assistant robot in this research.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The motion planner of ROS on the office assistant robot was studied by simulated in Gazebo. Then, map was generated by using the SLAM algorithm with gmapping method. The office assistant robot is then implementing the navigation stack which included the map server and localization on it by using ROS. The global planners were compared by giving 2 goals for the robot to navigate. The computation time and search space created were recorded. Besides, the performance of the local planners was tested by moving around the simulation, moving toward and across the obstacles. The most suitable planner will be selected based on the results obtained. After that, the navigation stack was implemented on the office assistant robot in a real-world scenario. The parameters were studied and tuned based on the results obtained to ensure the robot having optimum results.

3.2 Simulation with gazebo

Gazebo is a free simulator used to simulate a population of robot efficiently in a complex indoor or outdoor environment. To simulate the office assistant robot in Gazebo, a UTAR KB 5th floor environment and an office assistant robot model was built which shown in Figure 3.1 and Figure 3.2 respectively.

The environment and model were built and inserted into the gazebo. The robot model is linked with different type of parts such as wheels and body. The main body link is called as base frame. Based on Figure 3.1, The model consists of two wheels, a body of the robot, LiDAR sensor and a depth camera. There is also robot_controller to control the robot in Gazebo such as differential_drive controller and join_state controller.

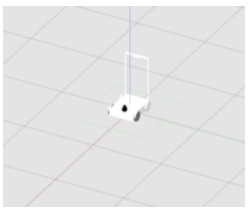


Figure 3.1: Robot model in Gazebo simulation

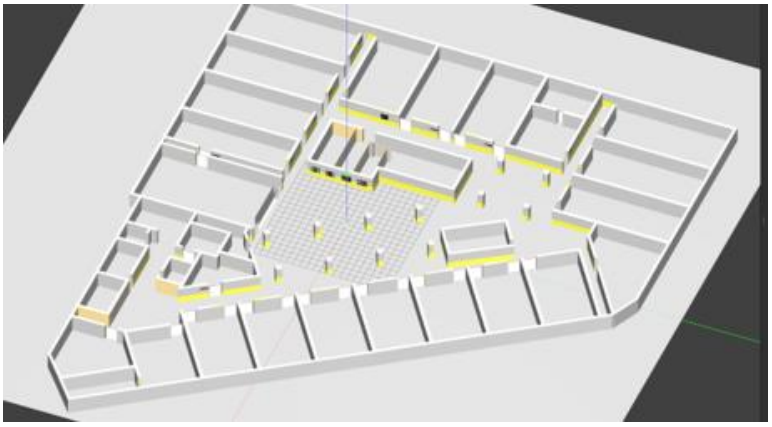


Figure 3.2: UTAR KB 5th floor

The robot can be controlled by using keyboard packages to publish the velocity topic to the robot model. The velocity topic named as diff_drive_controller/cmd_vel. The rqt graph was shown below in Figure 3.3.

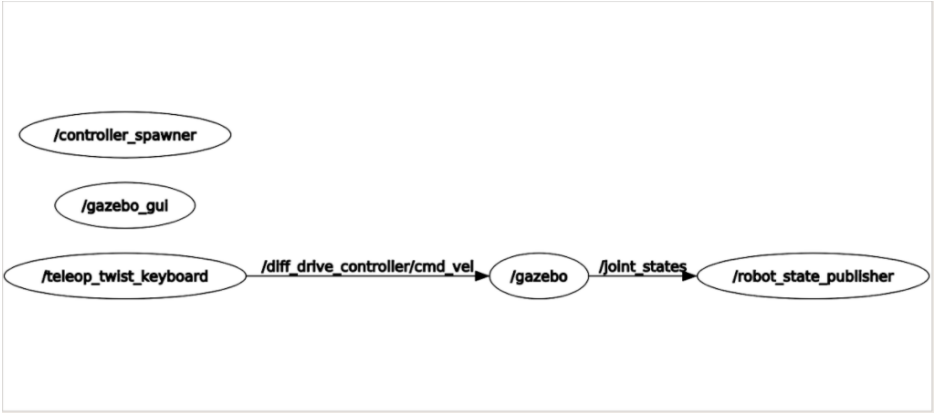


Figure 3.3: Rqt_graph of Gazebo simulation

3.3 Implement navigation stack

Navigation stack is used for the mobile robot to navigate autonomously by taking odometry and sensors information and output velocity commands to the robot. (Jasprit, 2018). To implement the navigation stack, multiple configurations needed to be setup. Based on Figure 3.4, the move_base package will take the sensor transform, odometry, map and sensors as the input, and generate the path and trajectory and output the velocity commands to the base_controller to move the robot with the desired velocity. To implement

navigation stack on the robot, a workspace was created. Then, a packages will be created to store all the configuration packages for the navigation stack.

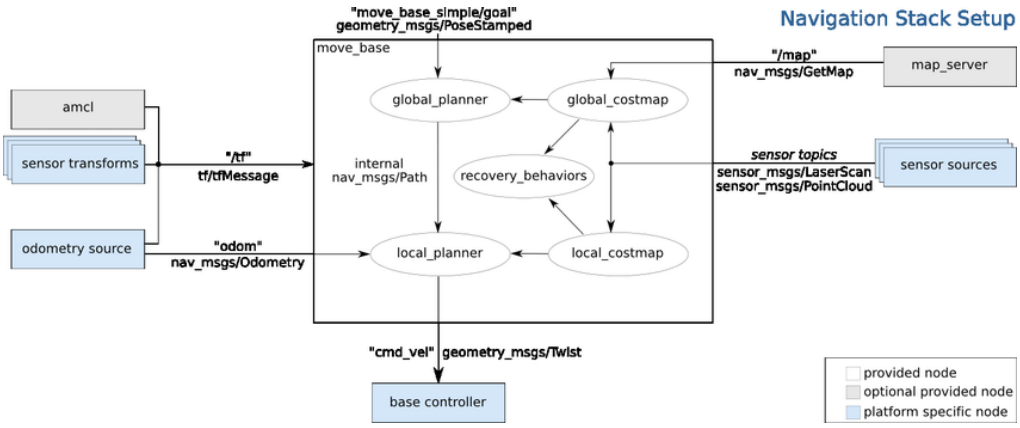


Figure 3.4: Navigation stack (Jasprit, 2018)

3.3.1 SLAM

Mapping is generated by using the sensor data and odometry of the robot. One of the method used for mapping is SLAM algorithm. It generates high quality and accurate map for the robot to navigate (Norzam, Hawari and Kamarudin, 2019). There are multiple ways of SLAM in ROS. The method used in this research is gmapping. The robot is then controlled manually by the user to generate a map for UTAR KB 5th floor. The map generated is known as global costmap while the local costmap will be generated by the real-time sensor data. The global costmap generated will be stored in the map server. The map generated can be tuned by manipulating the parameters in the parameter server. There is 3 configuration needs to be set by the costmap which are common configuration for global and local costmap, configuration for local costmap and global costmap. For example, the update and publish frequency, the resolution and the inflation radius of the map are the parameters that can be manipulated.

3.3.2 Localization

To localize the robot position in the map. Sensor data and map generated are used to localize the robot by using the Abstract Adaptive Monte Carlo localization(AMCL) algorithm. Then output the robot's estimate position as /tf topic. Figure 3.5 below shown the block diagram of the AMCL algorithm.

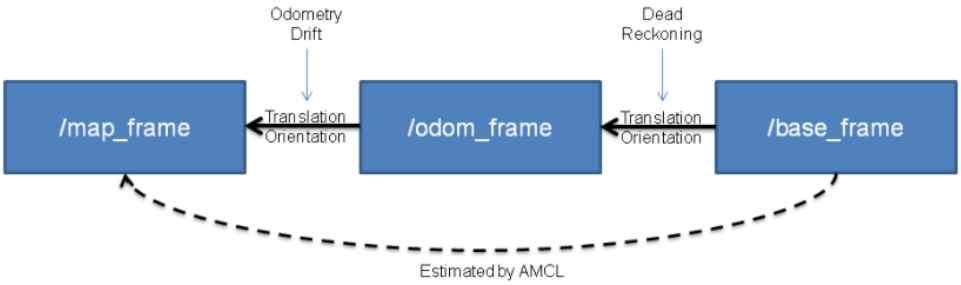


Figure 3.5: Block diagram of AMCL algorithm

3.3.3 Local planner configuration

A local planner parameter file was created. This file was used to store the parameters of a different type in the parameter servers. Then, the parameter will be taken by the move_base node and compute. The output will be a velocity commands to control the robot. After that, a launch file was created to launch multiples nodes such as configuration packages, map server, the AMCL packages and the move_base packages. Besides, the parameters for the costmap and the local planner was published to the parameter server through the launch file also.The overall rqt_graph is shown in Figure 3.6.

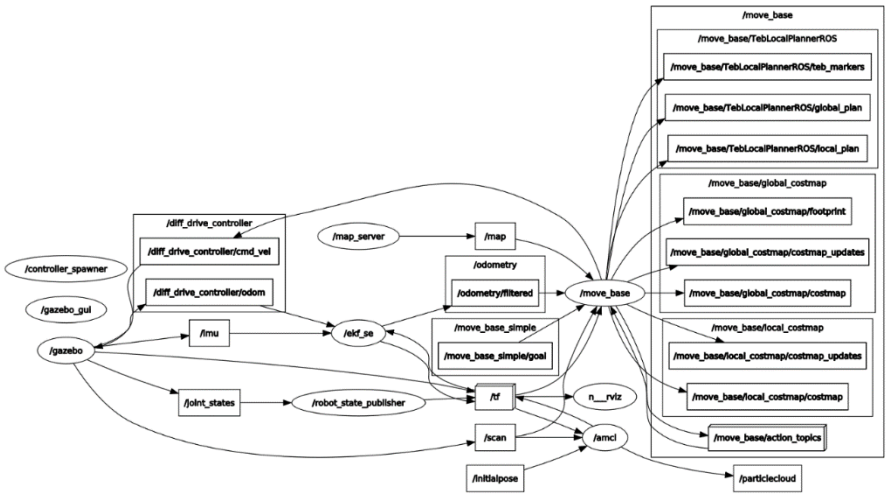


Figure 3.6: Overall rqt_graph of simulation with gazebo

3.4 Experiment in gazebo.

A node which called as simple_goal were created to publish different waypoint to the robot. The goal will specify the target position of x and y coordinate and also the targeted orientation of the robot. The computation time of the global planner will be recorded. Besides, the behaviour of the robot will also be recorded. Moreover, the local planners will be tested with multiple goals,

sudden obstacle in front 1m of the robot which shown in Figure 3.7 and dynamic obstacles which moving toward and moving across the robot to verify the feasibility of the local planner when facing dynamic obstacles which shown in Figure 3.8 and Figure 3.9.

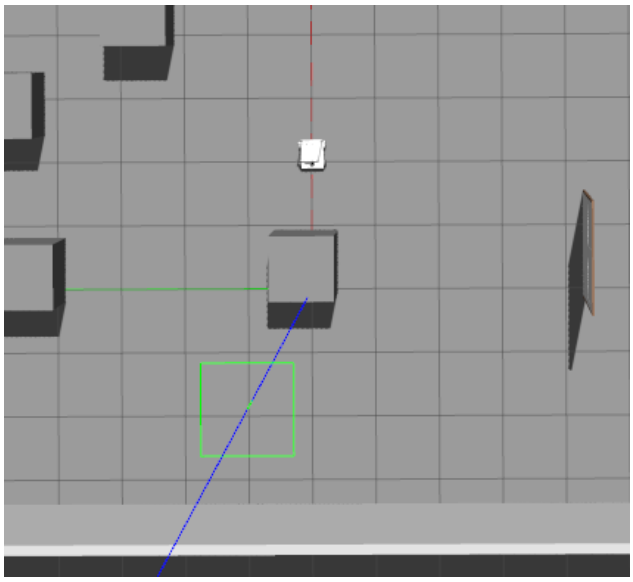


Figure 3.7: Robot facing sudden obstacle when moving to the goal.

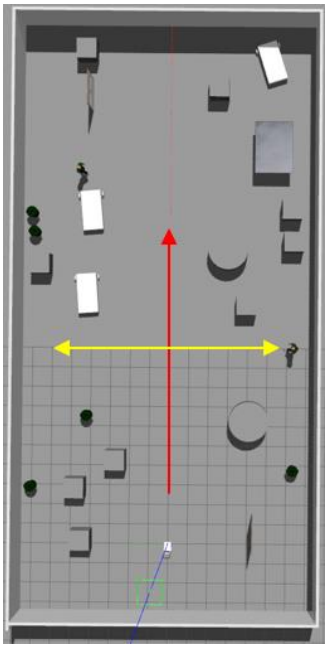


Figure 3.8: Simulation environment for dynamic obstacles moving across the robot. Red arrow indicate robot moving direction , yellow arrow indicate obstacles moving direction.

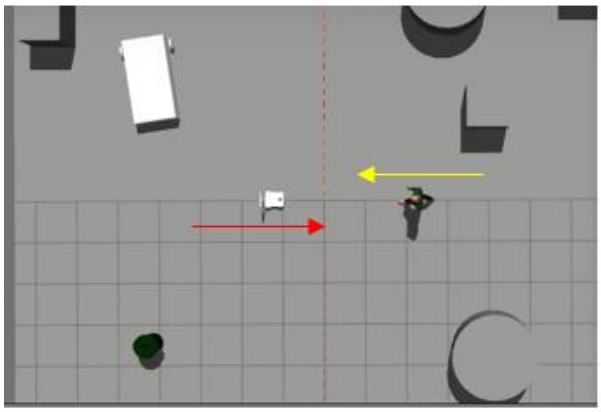


Figure 3.9: Simulation environment for dynamic obstacle moving toward the robot. Red arrow indicate robot moving direction , yellow arrow indicate obstacle moving direction.

3.5 Compare the results and Implement on office assistant robot

The results were compared based on their computation time and feasibility. The most suitable planner will be selected based on the simulation result and implemented on the real robot. There are some differences between the actual model of the robot and real robot such as the centre of gravity of the robot and the motor reaction time. These differences will affect the stability and inertia of the robot when navigating. Hence, the program needed to test with the real robot to do the final tuning.

3.6 Tuned the parameters

In a real-life scenario, there are many unpredictable issues and problem which are not able to be seen during simulation with Gazebo. Therefore, after implementing the program to the office assistant robot, the parameters of the planner need to be tuned to obtain the optimum results for the robot to smoothly navigate autonomously in the real environment. The office assistant robot will be given a sequence of a waypoint to determine its feasibility. Besides, it also will experiment with dynamic obstacles which moving across and toward the robot which shown in Figure 3.10, Figure 3.11 and Figure 3.12. The results will be recorded and discussed.

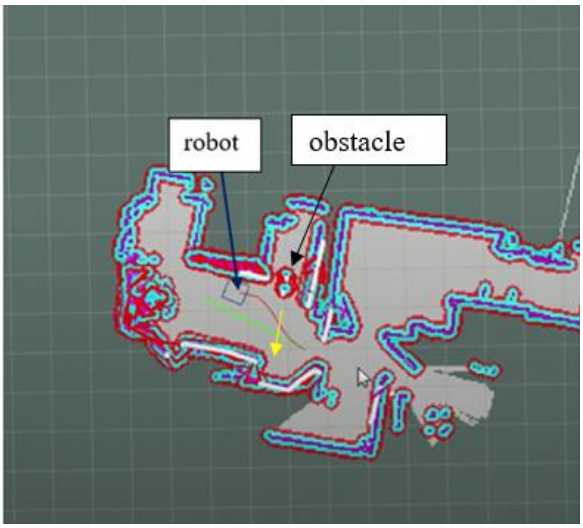


Figure 3.10: Dynamic obstacle moving across the robot in real environment. Yellow arrow indicate the movinf direction of obstacle.

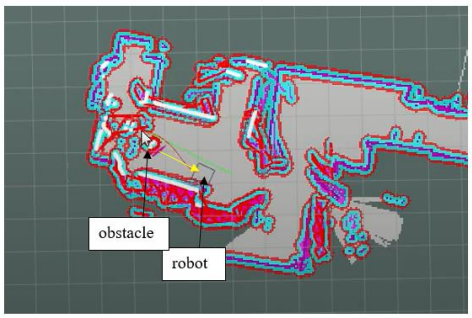


Figure 3.11: Dynamic obstacle moving toward the robot in real environment.Yellow arrow indicate the moving direction of obstacle.

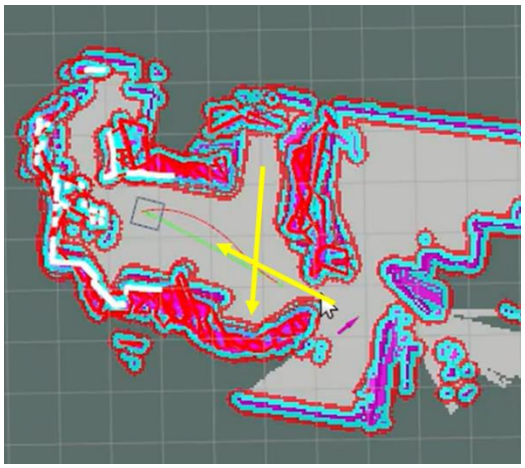


Figure 3.12: Dynamic obstacles moving toward and across the robot in real environment.Yellow arrow indicate the moving direction of obstacles.

3.7 Summary

The robot will first be tested with the simulation by using Gazebo. The advantages of using simulation is can reduce random error and time-saving. Then, the navigation stack will be implemented on the robot including the mapping and localization of the robot. Besides, there are multiple experiments to compare different local planner to select the most suitable planner used in this application. Finally, the planner will be implemented on the office assistant robot with fined tuned parameters.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this research, multiple simulation environments had setup to test the feasibility of different path planning algorithm which is shown in Figure 4.1, Figure 4.2 and Figure 4.3. Figure 4.2 had inserted multiple dynamic obstacles with specific velocity and goals. Besides, a simplified environment was generated to test the performance of the path planning algorithm with dynamic obstacles which is shown in Figure 4.3.

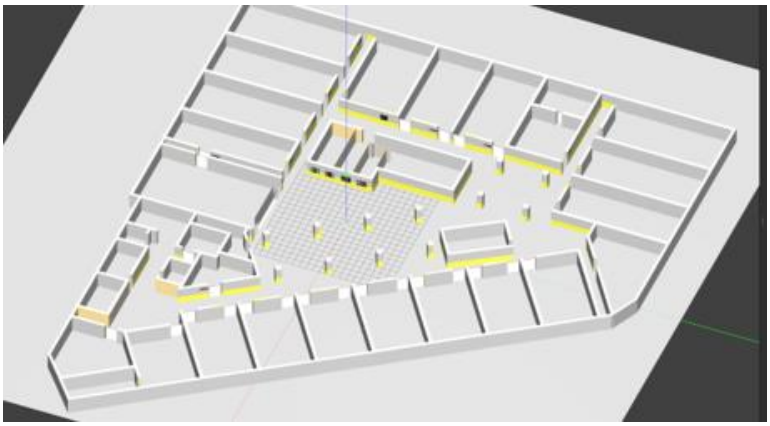


Figure 4.1: KB 5th floor simulation environment

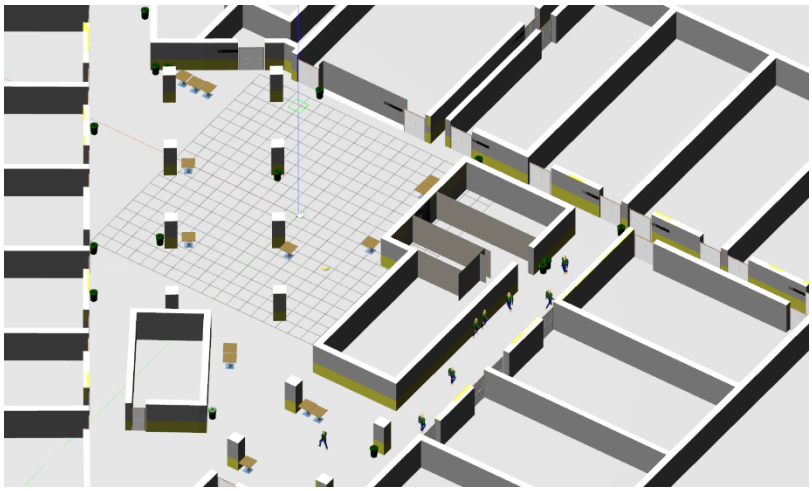


Figure 4.2: KB 5th floor simulation environment with multiple dynamic obstacles.

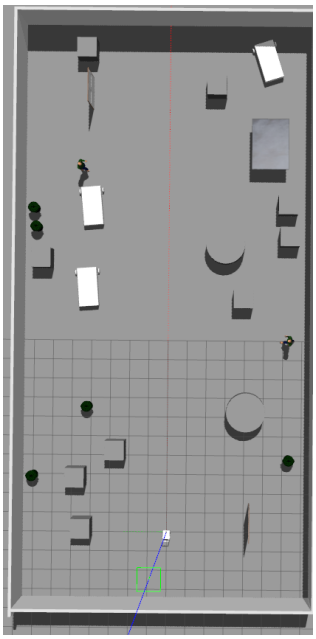


Figure 4.3: Simple simulation environment for experiment with dynamic obstacles.

Moreover, 2d map is generated by using the SLAM algorithm which is gmapping based on the data obtained from LiDAR sensors. On top of that, the path planning algorithms are compared based on the simulation results and the suitable local planner algorithm is selected and used in this office assistant robot. Next, the real robot is setup and the navigation pack is implement on the real robot and in a room environment to undergo further testing to tuned the parameters and verified the results obtained from the simulation.

4.2 Office assistant robot

The office assistant robot is a differential drive robot with four wheels. It has 2 omnidirectional freewheels mounted on the back of the robot and 2 front wheels independently powered by 2 DC brushed motor with built-in encoder. The built-in encoder provided the odometry data of the robot. Besides, it consists of a motor control board, dual channel motor driver to drive the motor, a LiDAR sensor which provided 240° laser scanning data mounted in the front of robot, a 12V Lipo Battery as power supply and a Laptop running Ubuntu 18.04 LTS with ROS Melodic to process the data obtained from the sensors and output the command velocity with PID controller to the control board to control the motors.

The robot system is shown in Figure 4.4 while the overview of the robot base is shown in Figure 4.5.

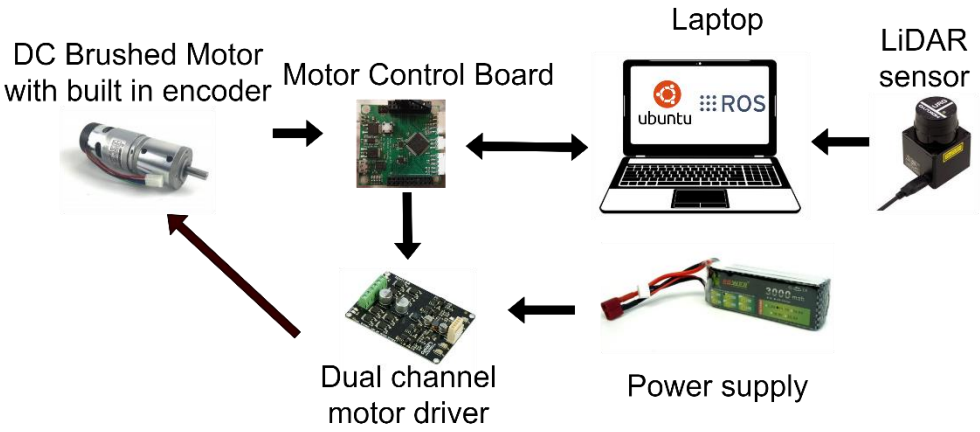


Figure 4.4: System of the office assistant robot

Figure 4.5: Office Assitant robot

4.3 SLAM

The map was generated by moving the robot around the environment to record the sensor data. Based on the LiDAR sensor data and the odometry data from the encoder feedback maps were generated by using the gmapping method which shown in Figure 4.6, Figure 4.7 and Figure 4.8. In simulation and real environment, the map generated will have some minor offset compare to the actual map. The generated map will input into the navigation pack for the robot used in the navigation and localization.

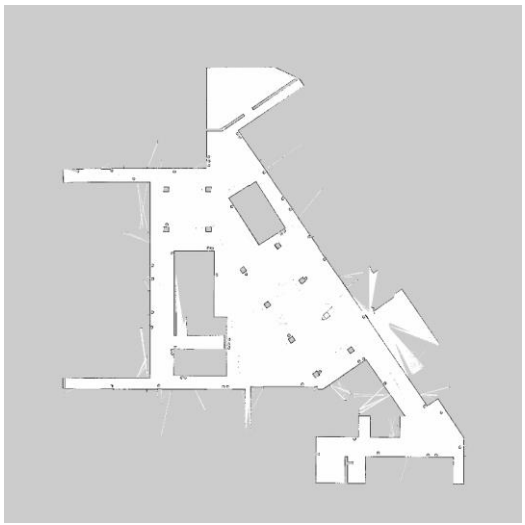


Figure 4.6: Map generated based on KB 5th floor simulation environment

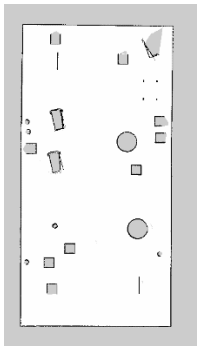


Figure 4.7: Map generated based on simple environment for dynamic obstacles.

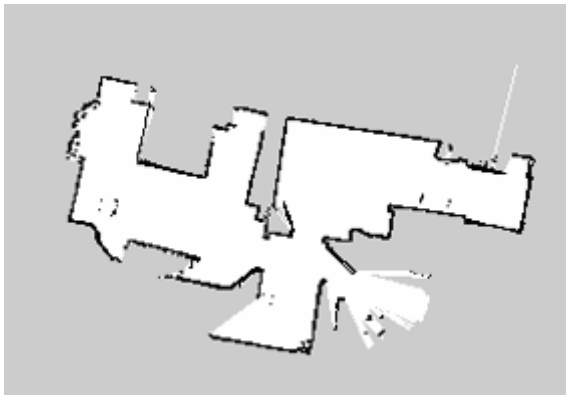


Figure 4.8: Map generated based on real environment(room)

4.4 Costmap setup

There are 2 costmap was implemented on the robot which is global and local costmap. Global costmap consists of 3 layers which are static map layer, inflation layer and obstacle map layer. Static map layer is used to define the map generated by SLAM which shown in Figure 4.9. While inflation layer was used to increase the cost around the static obstacles which is shown in Figure 4.10. There are two parameters to tune the inflation layer which are inflation_dist and cost_scaling_factor. Based on the result shown in Figure 4.10, the buffered zone around the obstacles increases when the inflation_dist increase. Besides, the cost_scaling_factor is used to adjust the inflation dist cost. The result was shown in Figure 4.11. Obstacle map layer used to track the dynamic obstacles or the obstacles not shown in the static map based on the sensor data. Unlike global costmap, local costmap only consists of inflation layer and obstacle map layer. The obstacle layer will be clear if the robot is running recovery behaviour due to oscillation, stuck in the position or unable to generate a valid path.

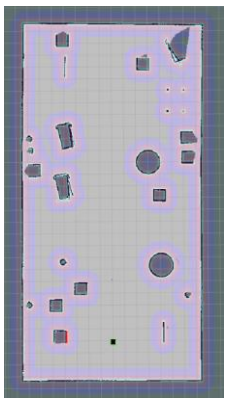
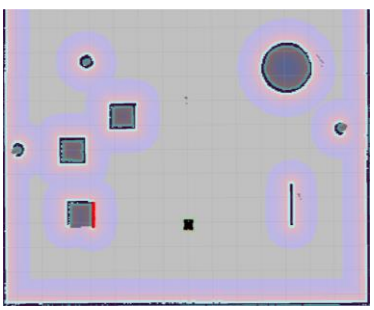


Figure 4.9: Map with Static map layer

(a)



(b)

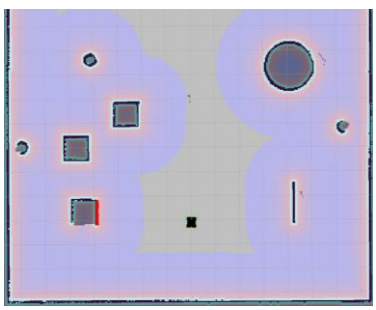


Figure 4.10: (a) indicate the inflation_dist = 1.0 and (b) indicate inflation_dist = 3.0 while both having cost_scaling_factor 3.0

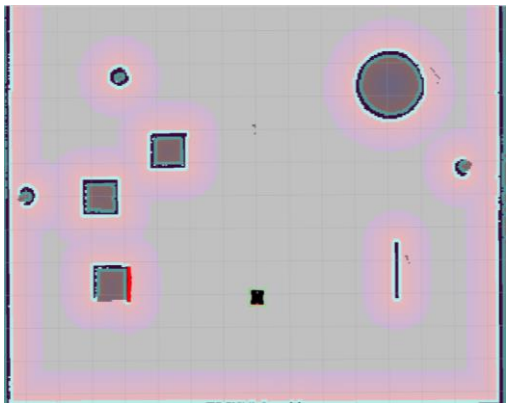


Figure 4.11: Cost_scaling_factor = 1.0

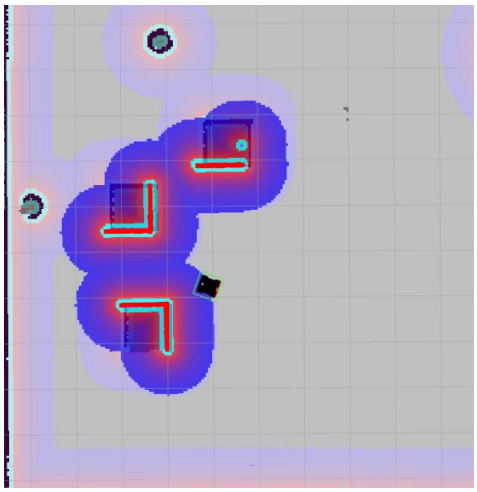


Figure 4.12: Obstacle layer

4.5 Localization of robot

The robot is localized based on the AMCL algorithm. The map generated, the odometry data and the real-time LiDAR sensor data is input to the algorithm to localize the position of the robot in the environment which shown in Figure 4.13. The red arrow around the robot indicates the predicted orientation and position of the robot in the map. Once the robot starts moving, it will auto-recalibrate the position of the robot based on the sensor data and the arrow will start to concentrate and point to the same direction which indicates the orientation of the robot which is shown in Figure 4.14.

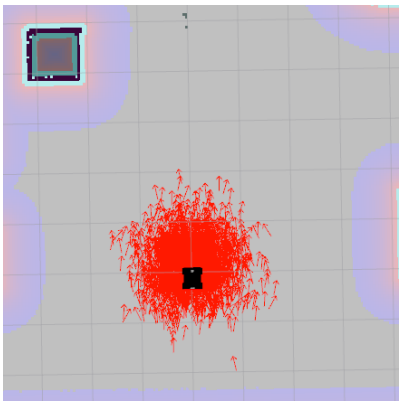


Figure 4.13: Robot implemented with AMCL algorithm in initial state.

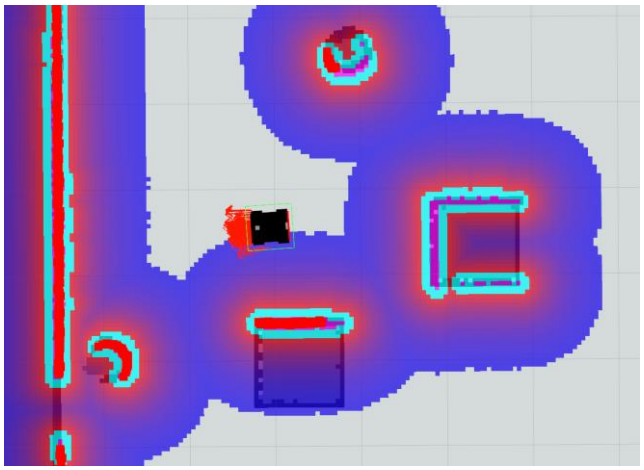


Figure 4.14: Robot implemented with AMCL algorithm after moving.

4.6 Comparison of global planner based on simulation

The package used in the robot is global_planner package. Figure 4.15 shown the path planned by the Dijkstra algorithm while Figure 4.16 shown the path planned by the A* algorithm. Based on the results obtained, Dijkstra will search the path and form a circle of search space which the robot act as a centre point until it reaches the goal, while A* will form a narrow search space toward the goals. When the robot moving toward the goals, the search space will be reduced based on the distance between the goal and the robot current position. Besides, the computation time of the global planner algorithms which shown in Table 4.1 and Table 4.2. Based on the results obtained, A* algorithm is 462.5% faster than Dijkstra algorithms when generating the path which 10m in front of the robot and 240.2% faster than Dijkstra algorithms when generating the path to the coordinate of (10,-25) of KB 5th floor which shown in Figure 4.18 and Figure

4.19. However, the computation time is not a main criteria in this application, because the global path planned is only in the initial state when the goal is set. Therefore, both global planners are able to achieve the required performance in this application.

Table 4.1: Computation time of global path planner algorithm with 10m goal.

No	Dijkstra, time (ms)	A*, time (ms)
1	7	1
2	7	1
3	9	1
4	6	3
5	7	2
6	8	1
7	7	2
8	6	1
9	6	1
10	11	3
Average	7.4	1.6

Table 4.2: Computation time of global path planner algorithm with coordinate(10,-25) goal.

No	Dijkstra, time (ms)	A*, time (ms)
1	41	23
2	41	22
3	44	16
4	40	19
5	38	15
6	46	17
7	50	19
8	49	16
9	50	17
10	43	20
Average	44.2	18.4

Moreover, the lethal cost is 253 in default parameters. In order for the robot to have a path planned further away from the obstacles and avoid stuck and unable to re-plan in a narrow corridor which shown in Figure 4.17. This is because, the local planner was avoiding travelling in the narrow corridor to maintain the minimum obstacle distance but the path generated by the global planner was conflicting. In order to solve this issue, the lethal cost is decreased

from 253 (default) every 10 steps until the robot was able to replan a new path which shown in Figure 4.17. The final lethal cost set is 150.

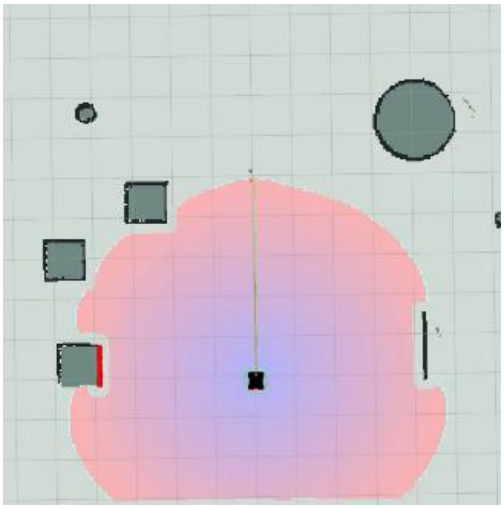


Figure 4.15: Path planned with Dijkstra algorithm

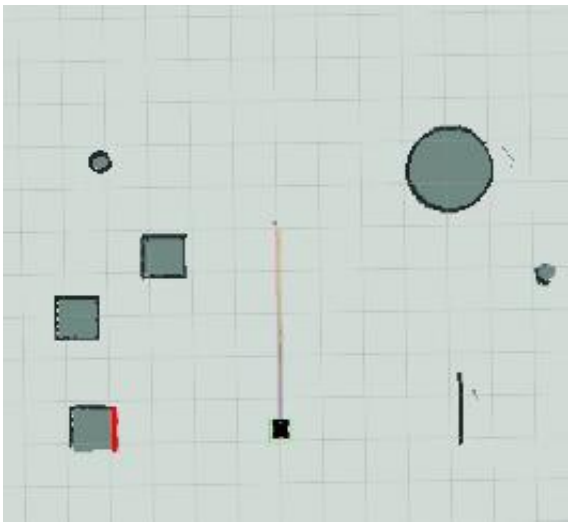


Figure 4.16: Path planned with A* algorithm

(a) (b)

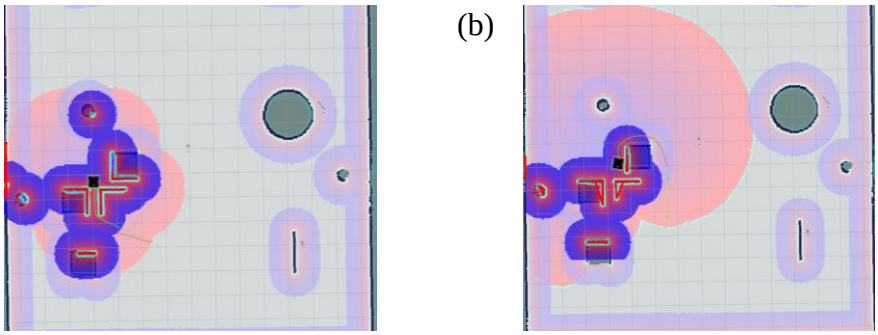


Figure 4.17: (a) Indicate the robot is stucked by the path planned with global planner with lethal cost = 253. (b) indicate the robot with lethal cost=150.

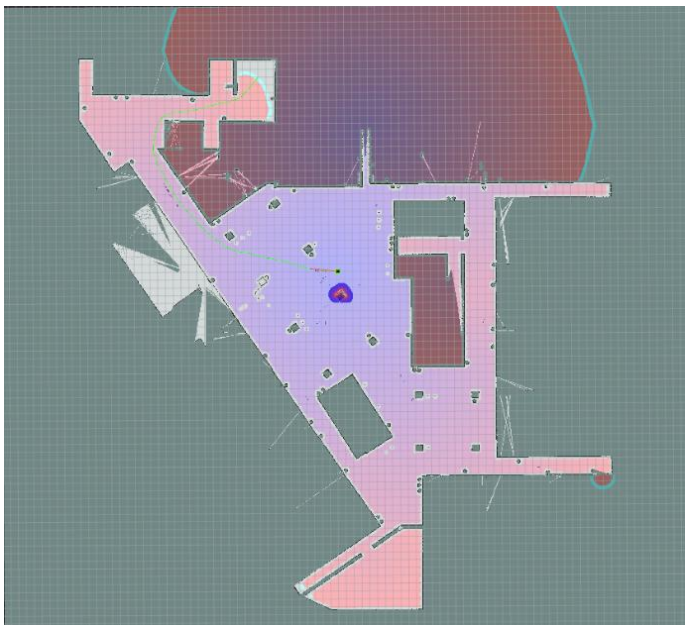


Figure 4.18: Search space of Dijkstra algorithm with target goal (10,-25)

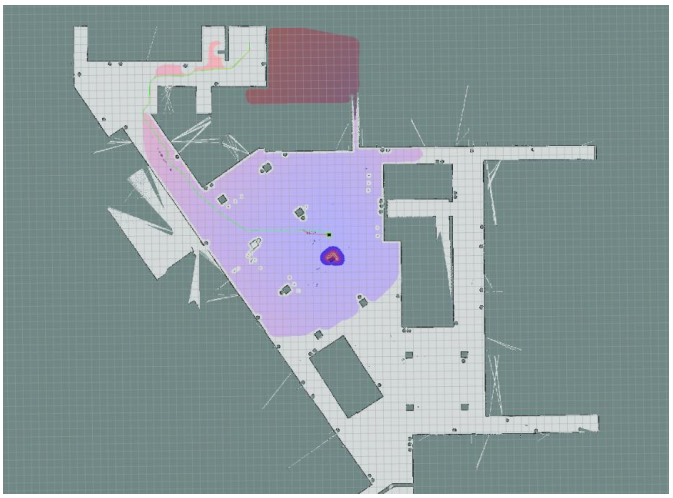


Figure 4.19: Search space of A* algorithm with target goal (10,-25)

4.7 Comparison of local planner

In this research, 2 local planner algorithm was implemented on the simulated robot which is DWA and TEB algorithm. In this research , the footprint model of the robot with TEB was changed to line instead of a point. This is because, the robot is a rectangle shape instead of a circular robot. Therefore, Line footprint model have better represantive than a point foot print model. Besides, due to the limitation of computation power, the enable_homotopy_class_planning was changed from true to false to reduce the computation power. The first experiment was tested by sending a specific goal to the robot with different local planner algorithms. The results were recorded and shown in Figure 4.20 and Figure 4.21. Based on the results obtained, in default parameters of DWA, the robot is unable to move in backward motion and stuck in the position. However, by tuning the min_vel_x parameter to negative value , we can change the maximum velocity for the robot to move backwards. But, this cause the robot will tend to move backwards than forward motion if the goals is behind the robots which shown in Figure 4.22. This is not suitable in this application. This is because, the LiDAR sensor is mounted in front of the robot, hence it unable to detect the dynamic obstacles that are moving behind the robot. Based on Figure 4.21, the robot is able to travel in backward motion unlike DWA. However, to reduce the robot to travel in backward motion, the algorithm can tune the parameter which are weight_kinematics_forward_drive from 1.0 to 800, so the robot will tend to move forward but it also able to move backwards when it is needed. The weight_kinematics_forward_drive indicate the cost of robot to moving in forward direction. The higher the value, the higher the tendency of the robot to moving in forward direction.

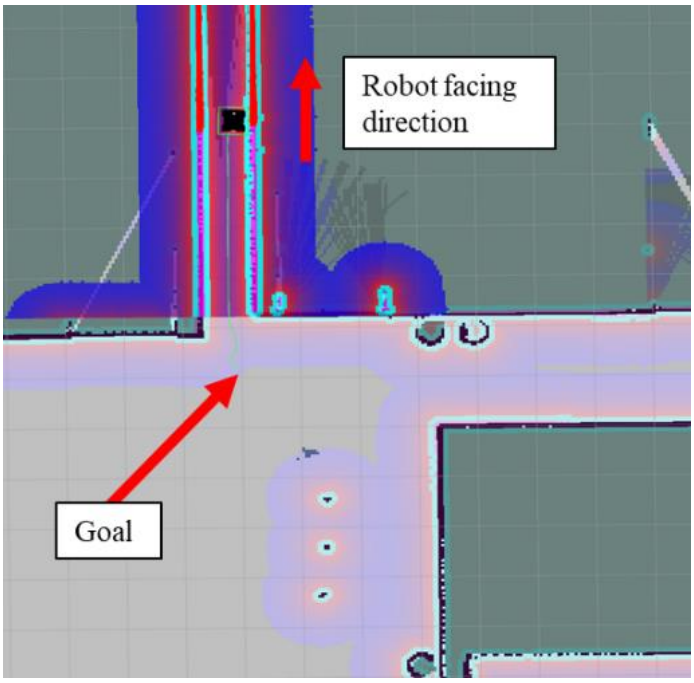


Figure 4.20: Robot stuck due to unable to move in backward motion in DWA.

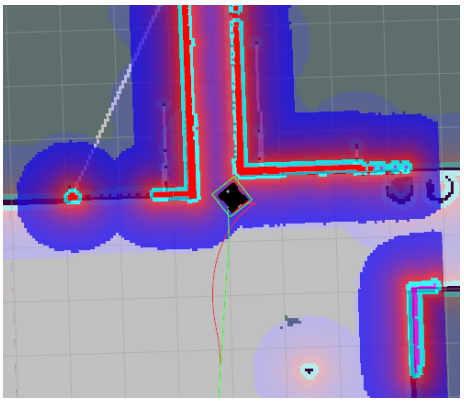


Figure 4.21: Robot able to travel in backward motion with TEB.

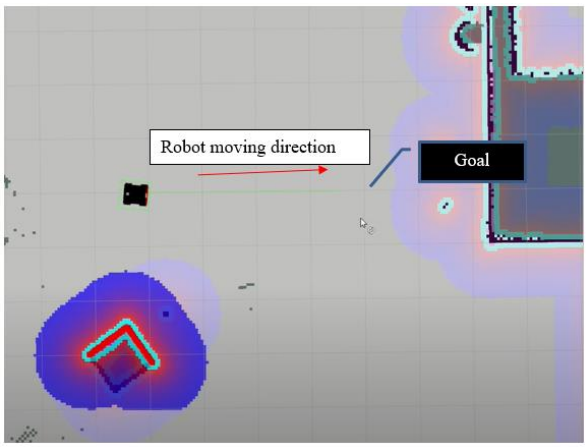


Figure 4.22: Robot tend to move in backward motion with DWA.

On top of that, to verify the performance of the algorithm in avoiding dynamic obstacles a new simulation environment is built-in Gazebo which shown in Figure 4.3. According to the results shown in Figure 4.23 and Figure 4.24, both algorithm able to stop without colliding with the obstacle when it suddenly appears 1m in front of the robot when the robot moving. Besides, based on the results obtained when the robot moving across dynamic obstacle, with default parameters, both algorithms have bad performance when facing dynamic obstacle especially DWA algorithm. The path generated by DWA when moving across the dynamic obstacle was shown in Figure 4.25. Based on Figure 4.26, the robot will only reduce its velocity when the obstacle are right in front of the robot and the angular velocity remain approximately constant. Hence, the robot is unable to stop before hitting the dynamic obstacle. To improve the performance of avoiding obstacle, a few parameters of DWA algorithms was increased which are:-

i. Sim_time –The amount of time to simulate forward trajectories.
ii. Vx and vth_samples – Number of forward and angular sampling velocity.
iii. Occdist_scale – Weight of the cost to avoid obstacle.

However, the performance of the DWA algorithm is not improved which shown in Table 4.3. With TEB algorithm, the robot has slightly better performance than DWA algorithm when moving across the dynamic obstacle. Based on the results shown in Figure 4.27, the robot will increase its angular velocity to approximate 0.75 rad/s when the obstacle moving across the robot unlike robot with DWA which remain approximately zero. To improve the results of TEB algorithm, there are also few parameters was tuned, which included the following:-

i. include_dynamic_obstacles.
ii. costmap_converter_plugin.

Costmap_converter_plugin is used to convert the costmap cells geometric primitives such as line, point and polygons as obstacles. By applying the costmap_converter_plugin, the obstacles are converted into polygon and a dynamic obstacle layer was applied. It able to track the dynamic obstacles according to the update of costmap. According to Albers et al. (2019), the

working of algorithms starts with a background subtractor and blob detector to track and locate the obstacles. New track will be generated if there are obstacles are not tracking and some track will be removed due to inactivate a period time. The tracks are generated based on the current and past position of the obstacles. In order to solve the data acquisition problem, Hungarian algorithm was implemented. Lastly, Kalman filter with first constant velocity model was applied to estimate the velocity of the dynamic obstacles. With these two parameters modified, the robot with TEB algorithm is able to avoid the dynamic obstacles most of the time which shown in Figure 4.28. Based on the results shown in Figure 4.28, the angular velocity increase significantly from 0.25 rad/s to maximum, 1.0 rad/s and at the same time the forward velocity dropped gradually from 1.0 m/s to around 0 m/s. This is because, with the converter plugin, the robot is able to predict the moving obstacle motion and able to stop and turn before hitting the dynamic obstacle, which shown in Figure 4.29.

Besides, based on the results shown in Figure 4.30, the robot with TEB+DOL will increase its angular velocity to rotate and avoid the obstacle moving toward the robot which shown in Figure 4.31.

In conclusion, based on the simulation results, TEB algorithm was chosen to apply on the office assistant robot due to its feasibility in avoiding dynamic obstacles and able to tune the weight of moving backward motion which is more suitable in this application.

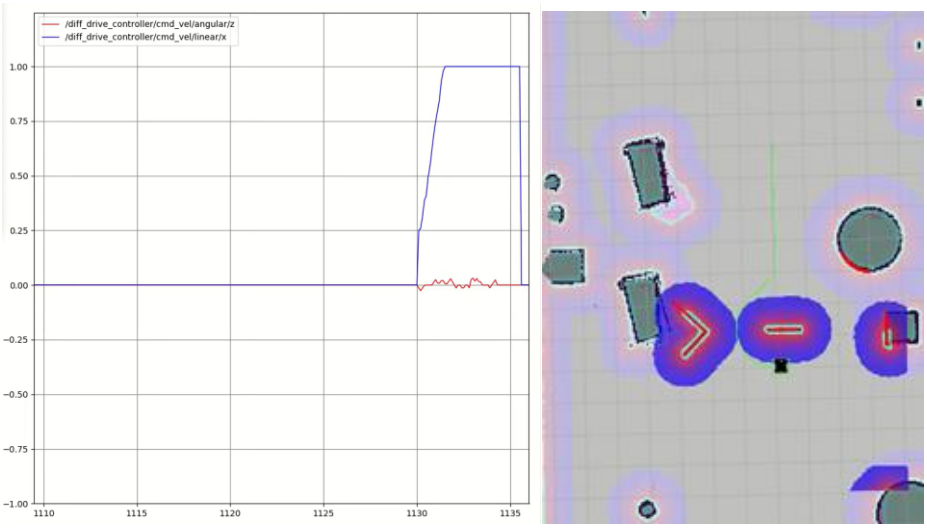


Figure 4.23: Robot having sudden obstacle infront the robot when moving forward (DWA)

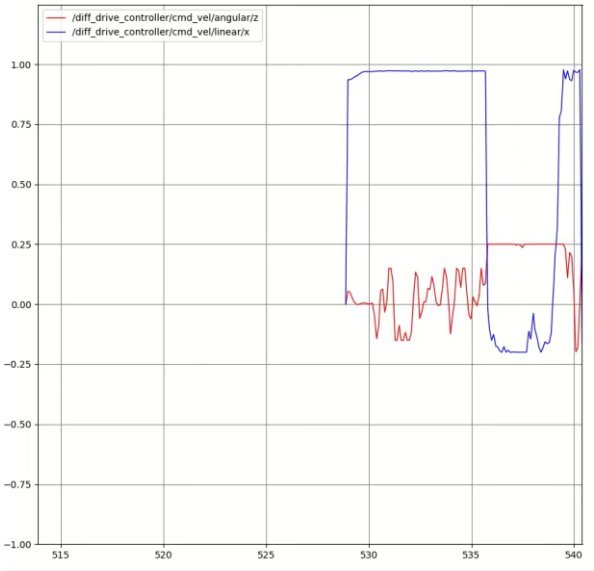


Figure 4.24: Command velocity graph with default parameters sudden obstacle in front of the robot (TEB)

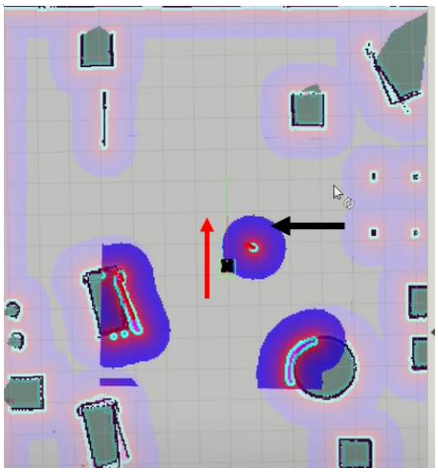


Figure 4.25: Robot moving across dynamic obstacle (black arrow indicate direction of obsctales while red arrow indicate robot moving direction) (DWA)

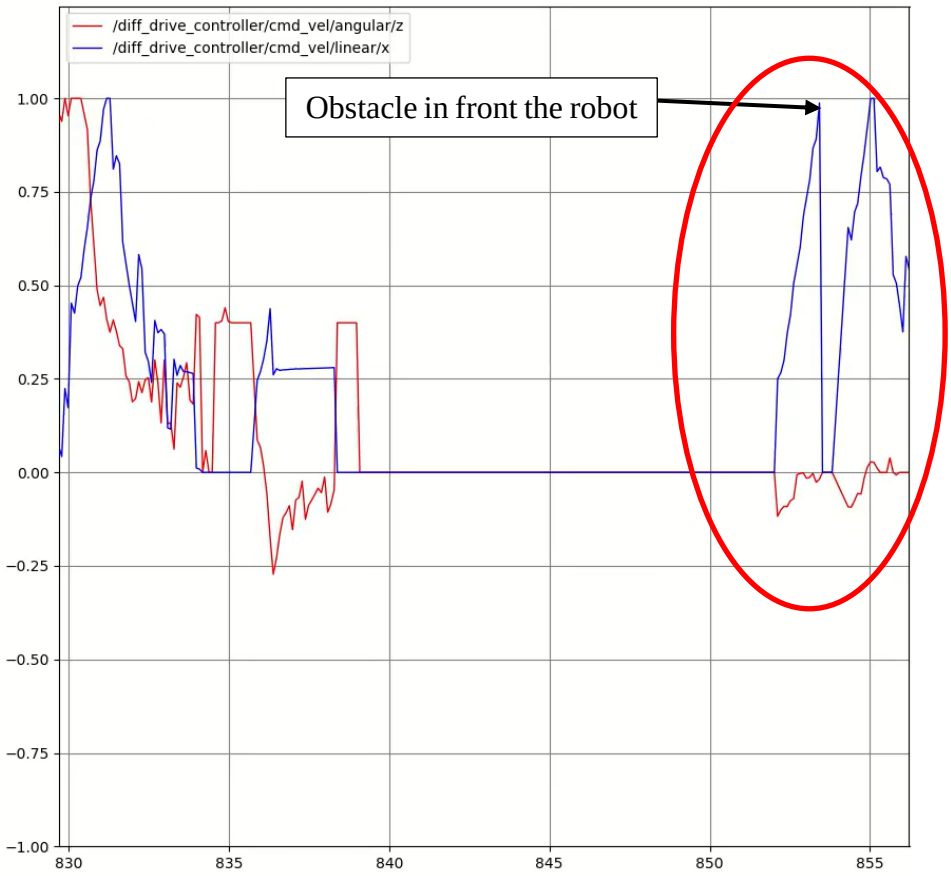
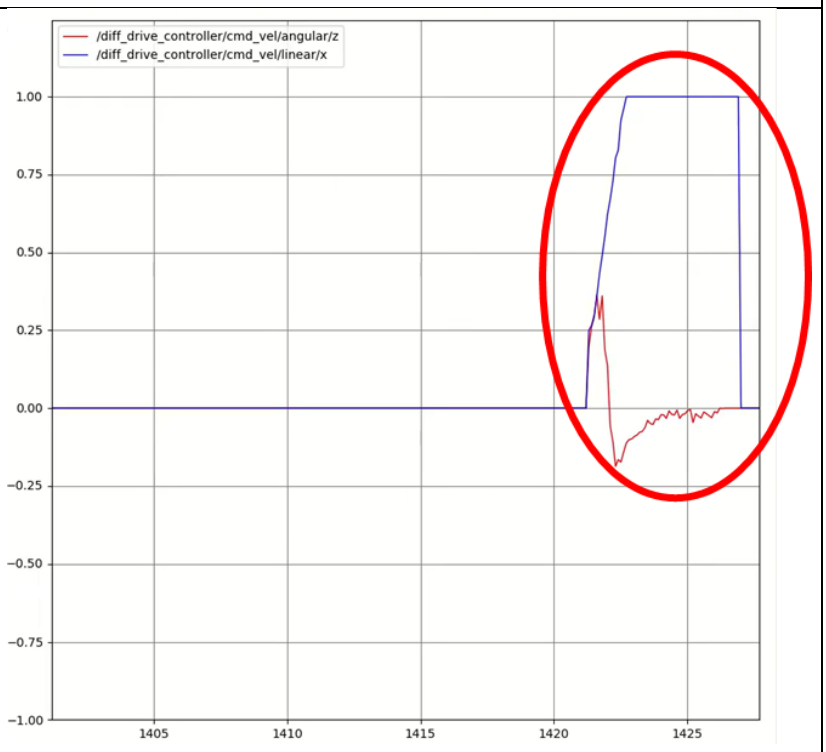
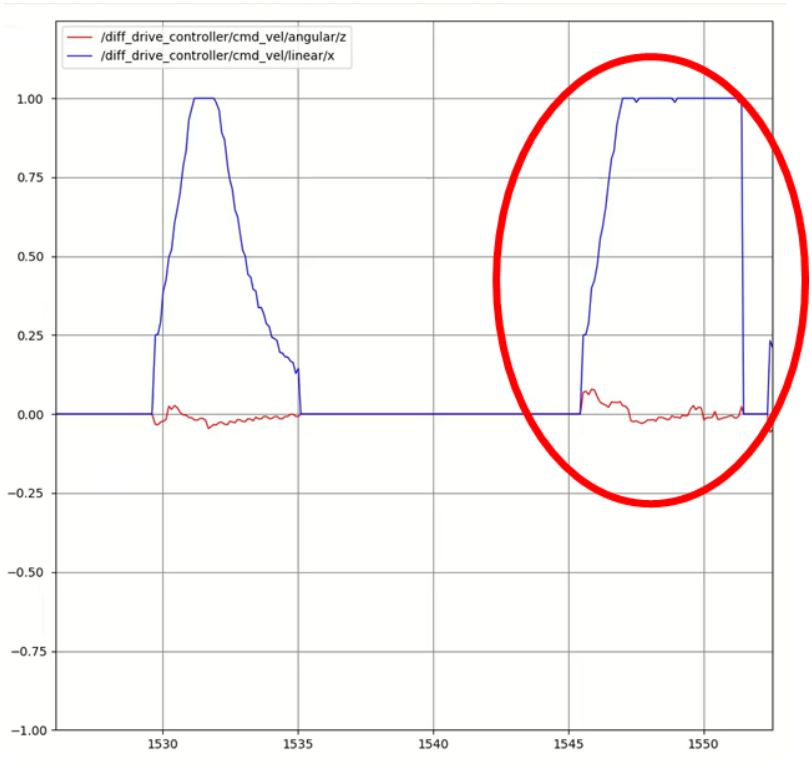
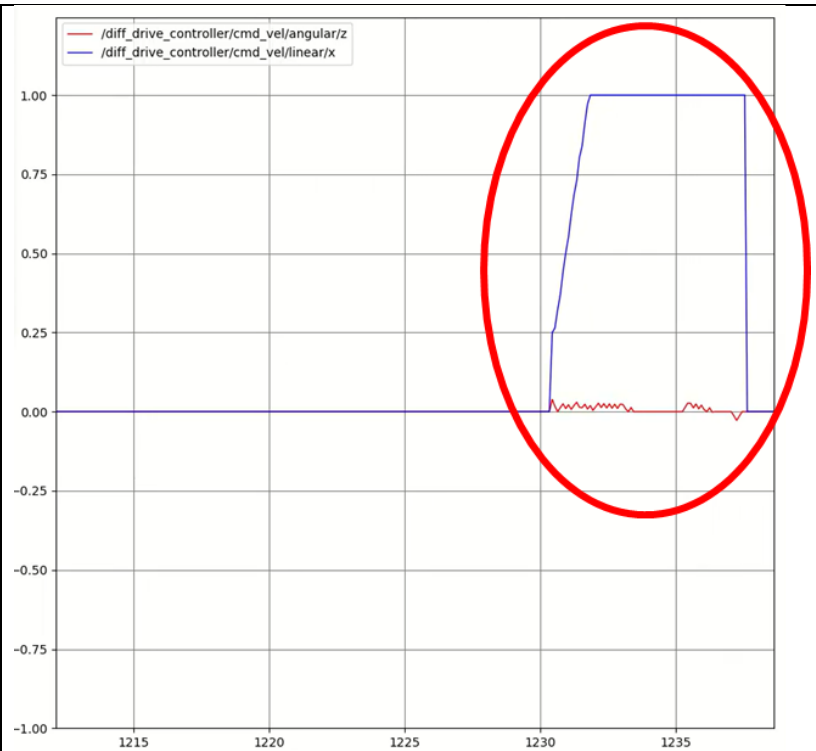


Figure 4.26: Command velocity graph of robot moving across obstacle (DWA)

Table 4.3: Results of changing 3 different parameters of DWA

Parameters	Results
Sim_time (1.7 to 3.5)	
Occ_dist_scale (0.01 to 1.0)	

Vx (3 to 20) Vth (20 to 40)	

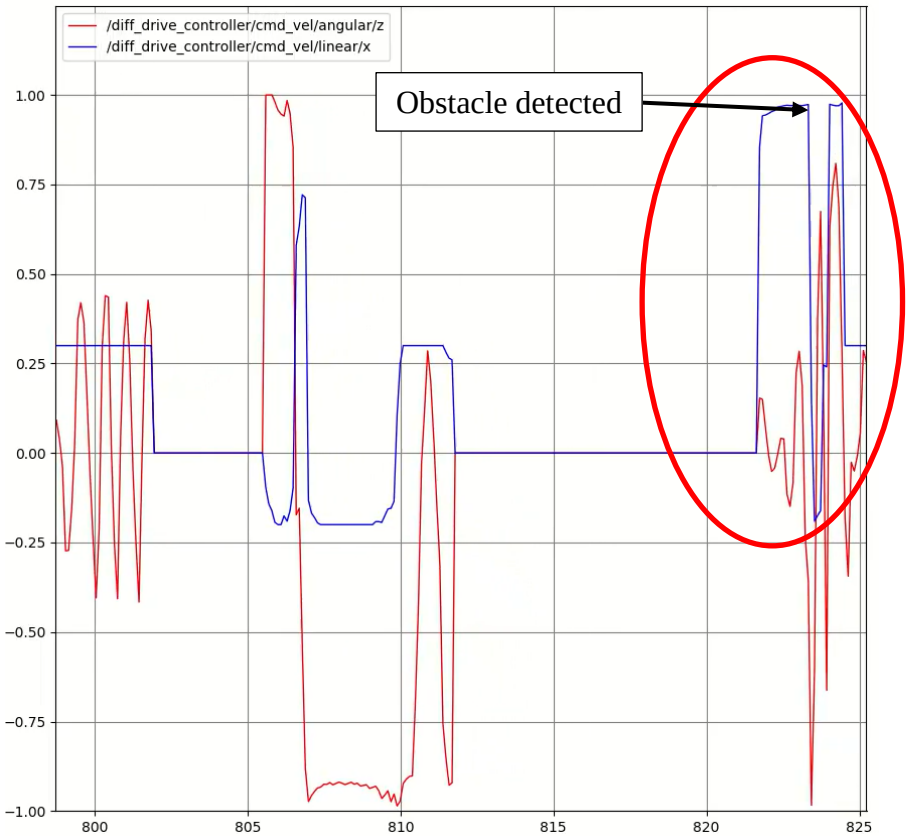


Figure 4.27: Command velocity when obstacle moving across (TEB)

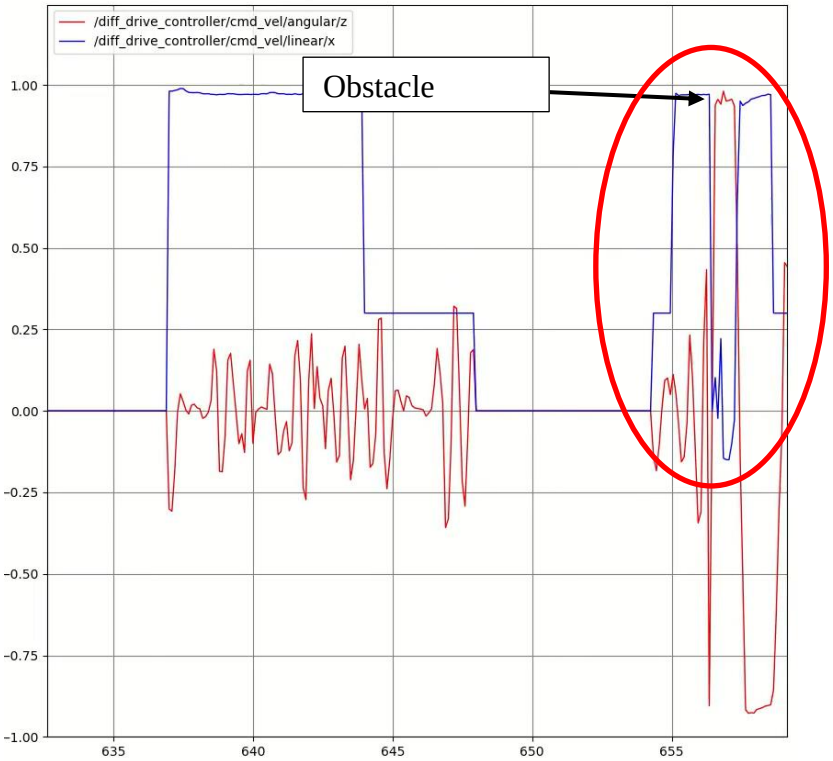


Figure 4.28: Command velocity when obstacle moving across (angular velocity increase) in simulation (TEB+DOL).

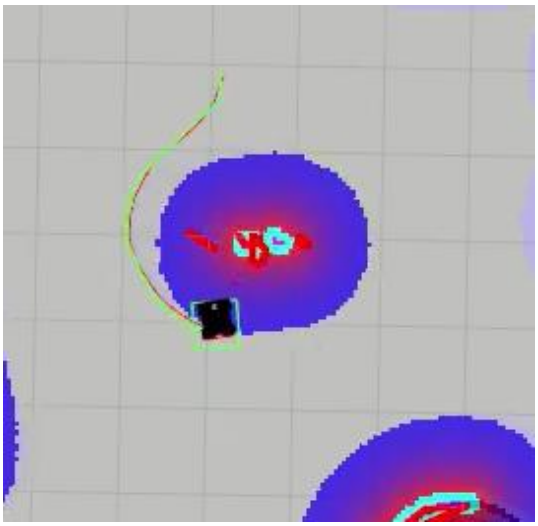


Figure 4.29: Robot slow down and path re-planned to avoid the dynamic obstacle which moving across in simulation.

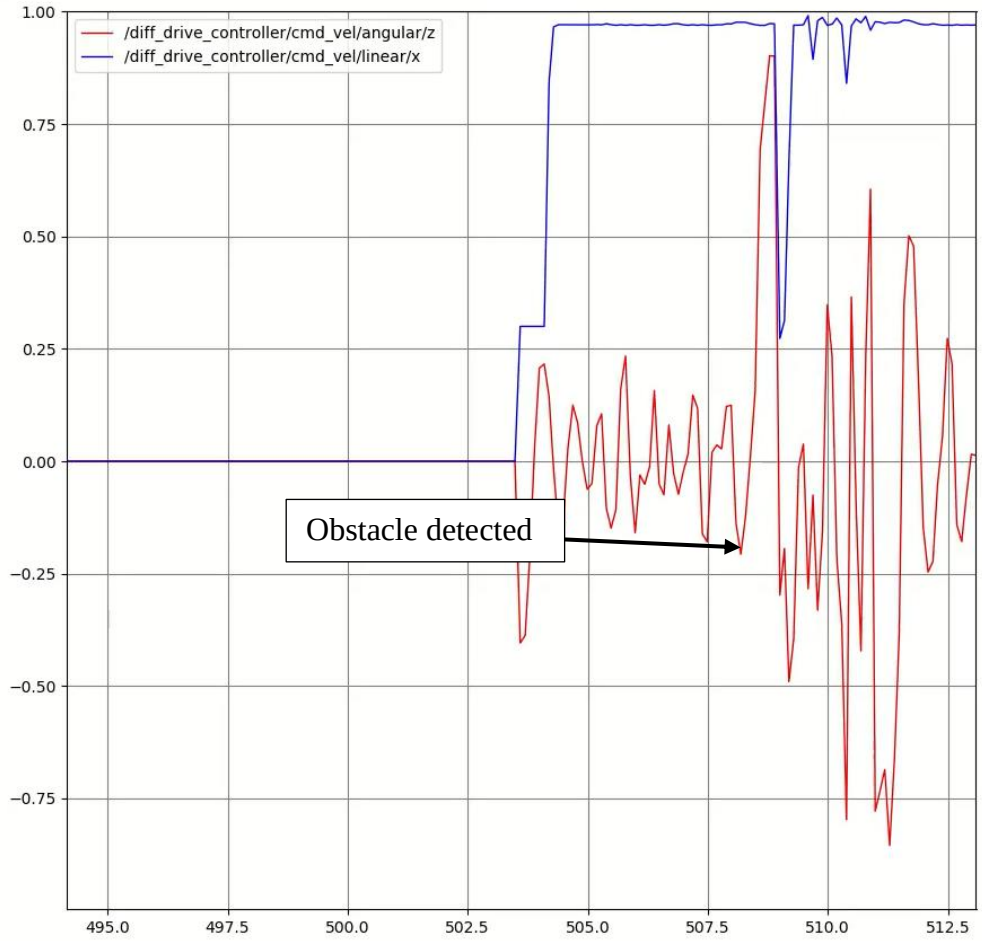


Figure 4.30: Obstacle moving toward (angular velocity increase) in simulation (TEB+DOL)

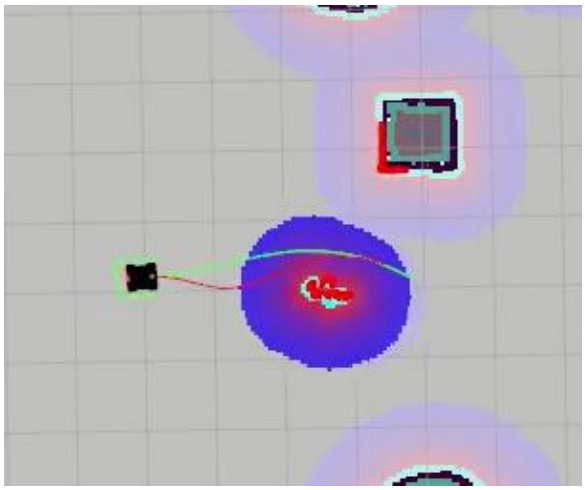


Figure 4.31: Robot turn to avoid the dynamic obstacle which moving toward in simulation (TEB+DOL).

4.8 TEB algorithm on real office assistant robot

The real environment had changed to a room instead of UTAR KB 5th floor. The TEB algorithm were implemented on the office assistant robot. The costmap parameters was tuned based on the environment. Moreover, the maximum forward velocity was reduced to 0.5 m/s because the room is not wide enough for the robot to travel with 1.0 m/s. Besides, the maximum angular velocity was reduced to 0.5 rad/s to reduce the odometry error. The first experiment was tested by sending a specific goal for the robot to travel in the room. According to the results shown in Figure 4.32, the robot are stucked in front of the door. This problem can be solved by tuning the min_obstacles_dist parameters. Min_obstalces_dist indicate the minimum distance between the robot and the obstacles. In this research, it was reduced to 0.3 m to allow the robot pass through narrow environment.

Moreover, the robot was tested with the dynamic obstacle move across itself which shown in Figure 3.10 and the results were shown in Figure 4.33 and Figure 4.34. Based on the results obtained, the robot is approximately same as the simulation results. The robot is able to plan a path in order to avoid the dynamic obstacle which shown in Figure 4.35. When there is no converter plugin, the robot is unable to predict the obstacle motion and it did not increase the angular velocity to avoid hitting the obstacle only until the obstacles is right in infront the robot which shown in Figure 4.36. However, with the converter plugin, the robot are able to predict the dynamic obstacles and it increased the angular velocity to avoid the obstacle more efficient which shown in Figure 4.35. Hence, the robot with TEB + DOL has better performance. Besides , It is observed that the robot forward velocity will dropped to zero and remain constant until the obstacle moved away which shown in Figure 4.37 when the obstacle is moving across the path planned and is near to the robot which shown in Figure 4.38.

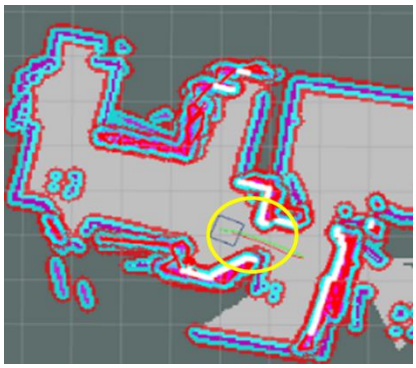


Figure 4.32: Robot was stucked in narrow environment

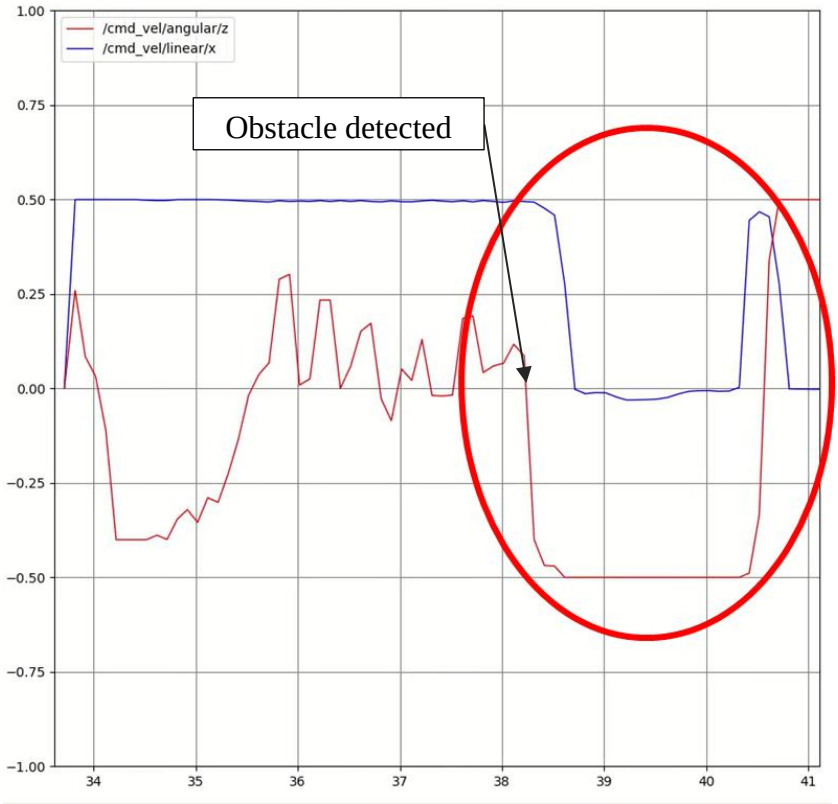


Figure 4.33: Velocity graph when obstacle moving across robot with TEB.

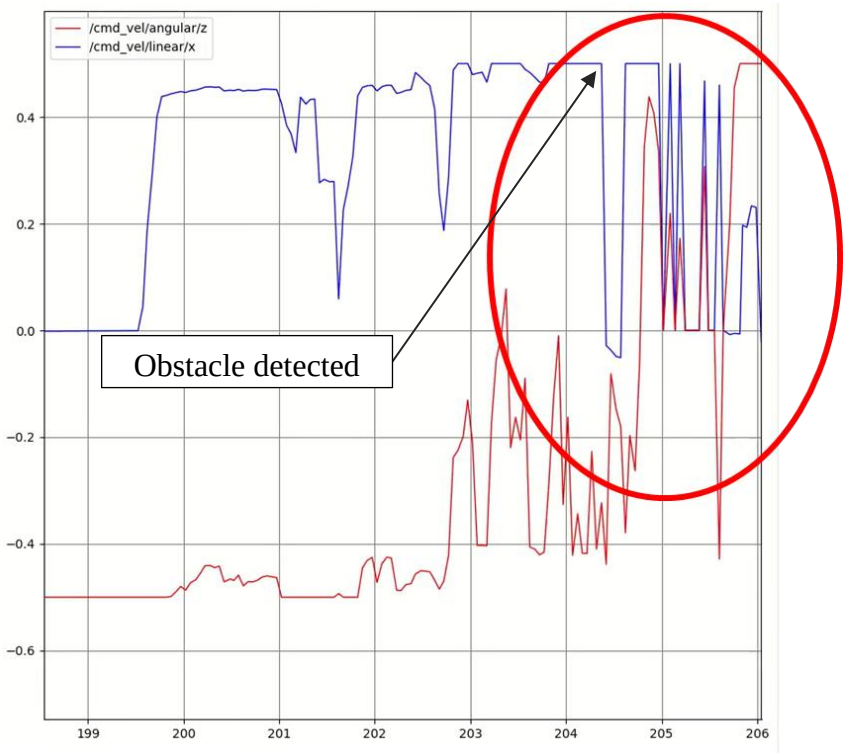


Figure 4.34: Velocity graph when obstacle moving across robot with (TEB+DOL)

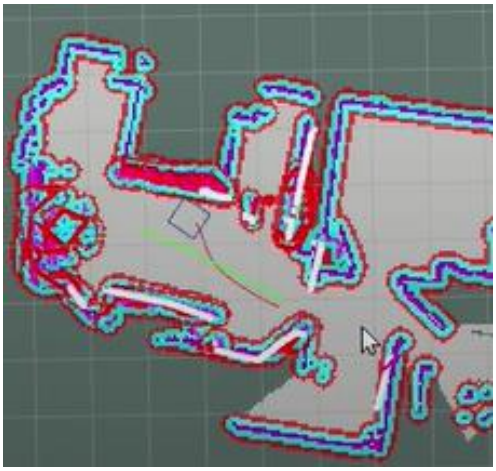


Figure 4.35: Obstacle moving across the robot path planned (TEB+DOL)

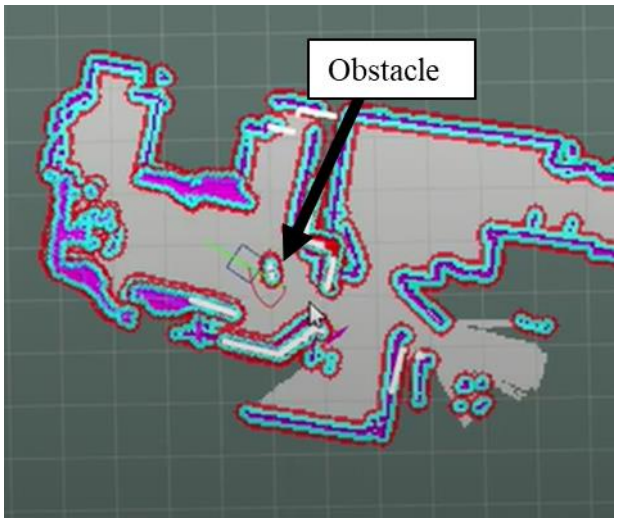


Figure 4.36: Obstacle moving across the robot path planned (TEB)

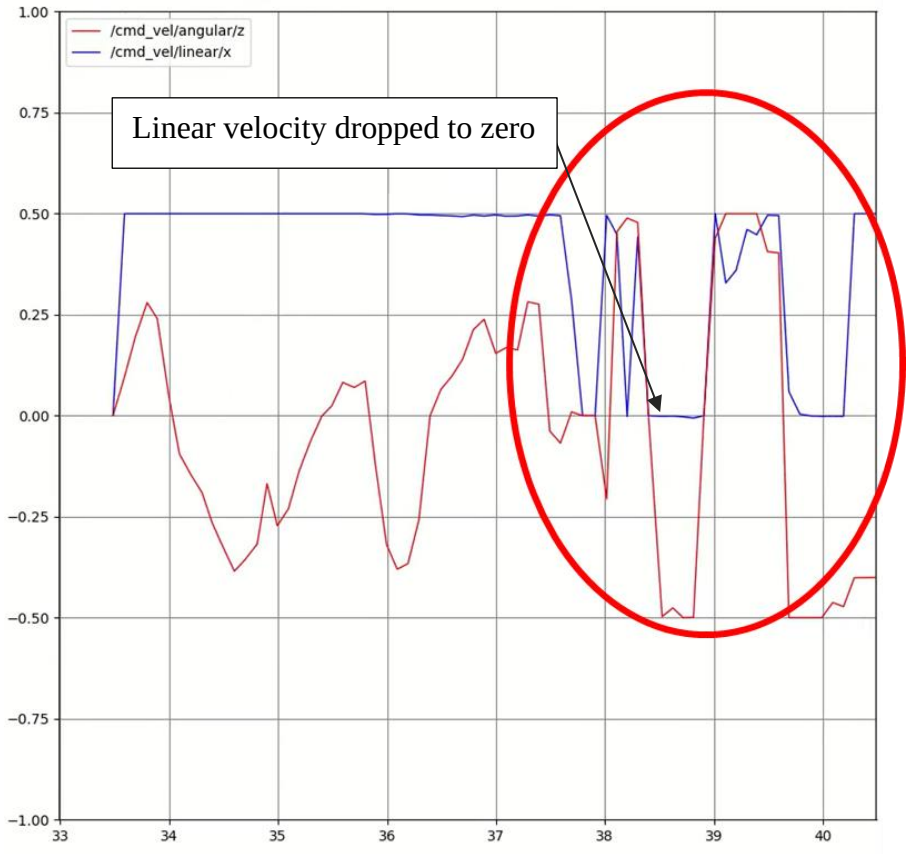


Figure 4.37: Command velocity graph of real robot when obstacle moving across path planned (TEB+DOL)

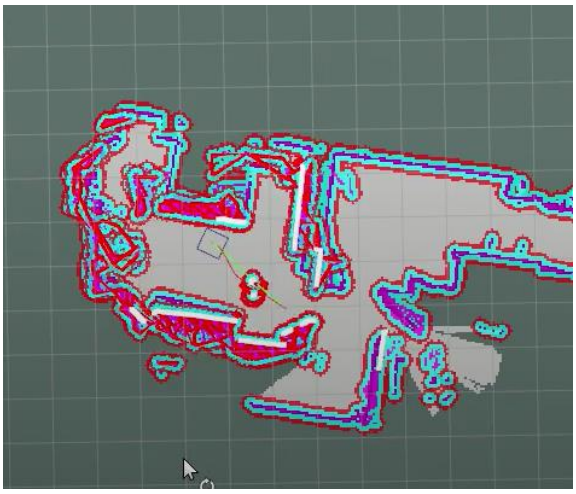


Figure 4.38: Obstacle moving across the path planned (TEB+DOL)

On top of that, another experiment was undergone which are the obstacle mowing toward the robot which is shown in Figure 3.11. Based on the result shown in Figure 4.39, the robot will slow down or stop and increase its angular velocity to avoid the obstacle moving toward and generate a path which shown in Figure 4.40. Besides, if the obstacle continue moving toward the robot, the robot will collide with the obstacles while the robot will tend to move backwards afterwards. If the obstacle stop right before hitting the robot, the robot will rotate and follow the path planned in Figure 4.40.

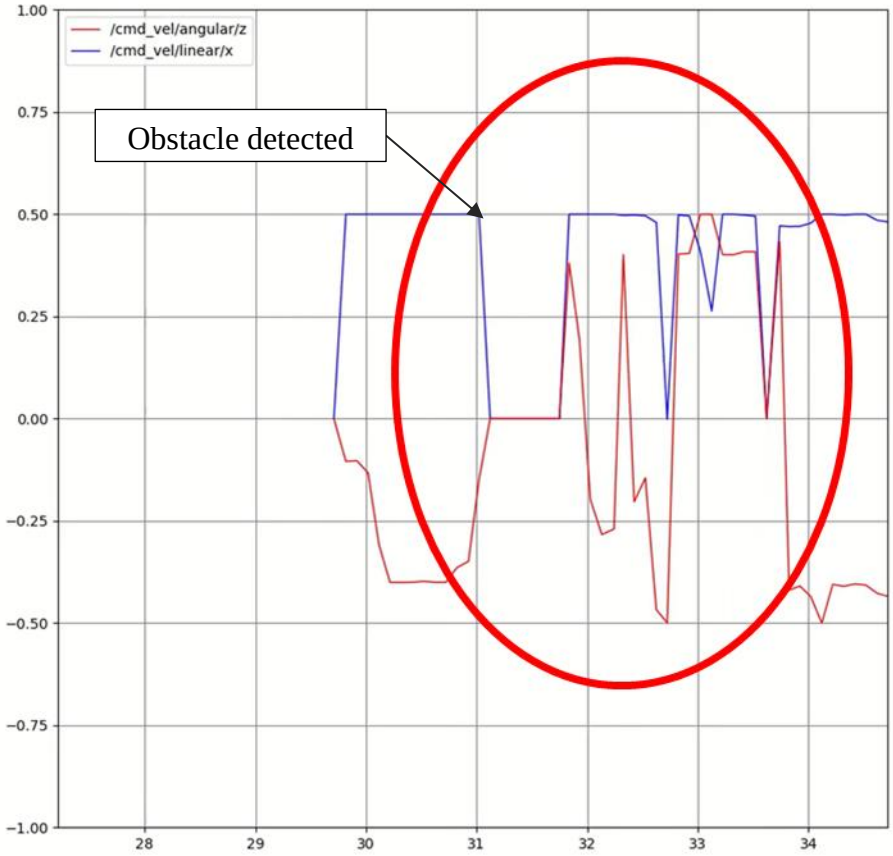


Figure 4.39: Velocity graph when obstacle moving toward the robot. (TEB+DOL)

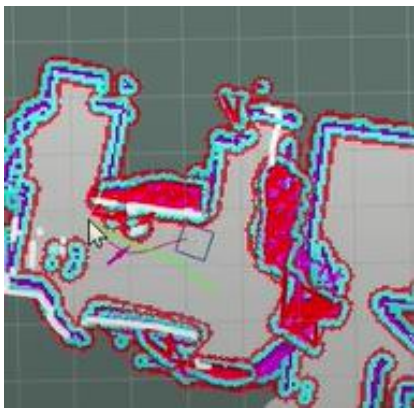


Figure 4.40: Obstacle moving toward robot

Moreover, the robot can avoid the dynamic obstacles when there is dynamic obstacles moving toward and across the robot which is shown in Figure 3.12. It will slow down its linear velocity and generate a path to avoid the obstacle which shown in Figure 4.41 and after the dynamic obstacles move away from the robot, the path will back to normal which shown in Figure 4.42.

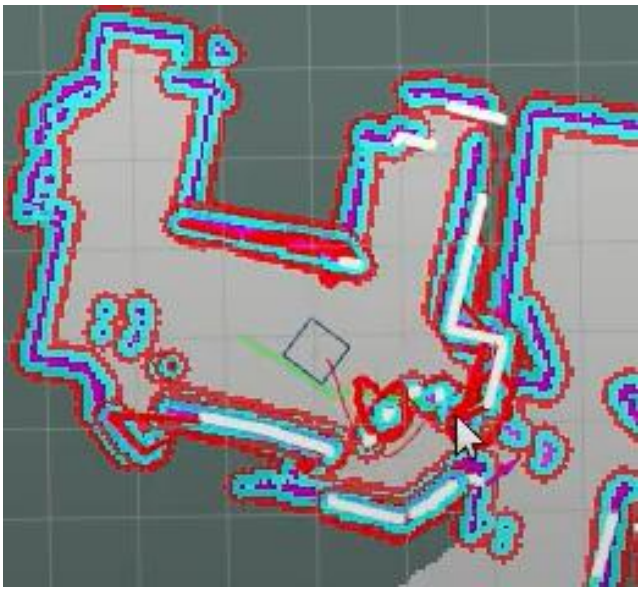


Figure 4.41: Path generated when obstacles move toward and across the robot.

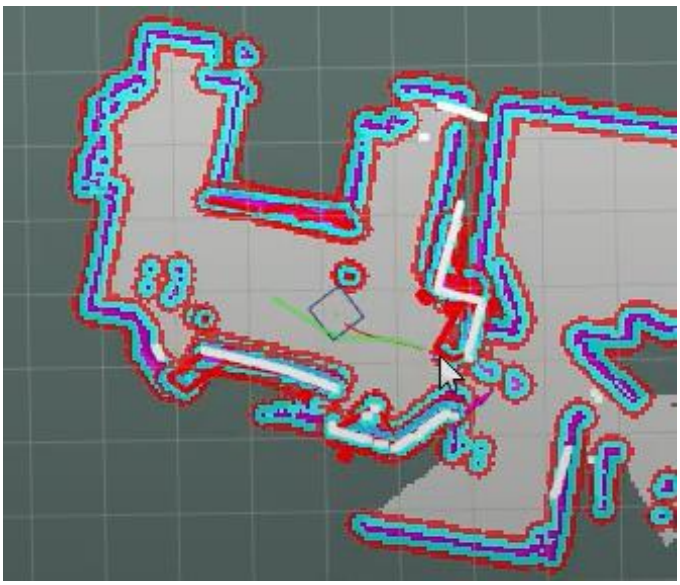


Figure 4.42: Path planned after the obstacles move away.

According to the simulation and real life results, it can conlude that office assistant robot with TEB and DOL are able to avoid obstacles more efficient than TEB only. Therefore, TEB with DOL are applied to the office assistant robot.

4.9 Parameters affect dynamic obstacles avoidance.

Besides, some of the parameters were tuned to investigate the performance of the robot when facing dynamics obstacles which are:-

i. weight_dynamic_obstacle – the weight of distance between dynamic obstacles and robot.
ii. weight_dynamic_obstacle_inflation – Optimization cost of inflation penalty of dynamic obstacles.
iii. dynamic_obstacle_inflation_dist – Buffered zone around the predicted position of dynamic obstacles.

The weight_dynamic_obstacle, weight_dynamic_obstacle_inflation and dynamic_obstacle_inflation_dist was increased to study the effect of modifying the parameters. The weight_dynamic_obstacle was increased from the default value, 50 to 500. Based on the simulation results shown in Figure 4.43, the path planned is further away from the moving obstacle when it moving toward the robot which same as the results obtained from the room which is shown in Figure 4.45.

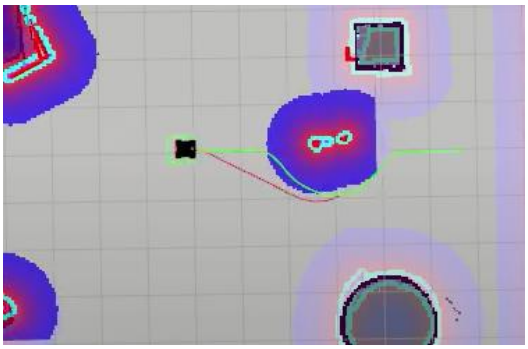


Figure 4.43: Simulation results when weight_dynamic_obstacle = 500

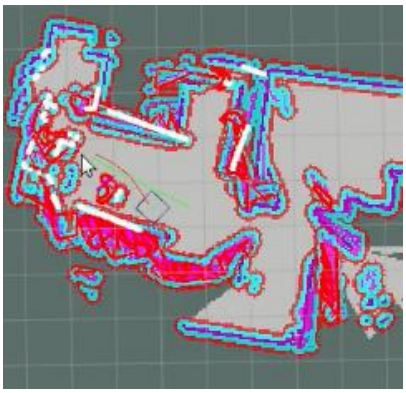


Figure 4.44: Weight_dynamic_obstacle = 50

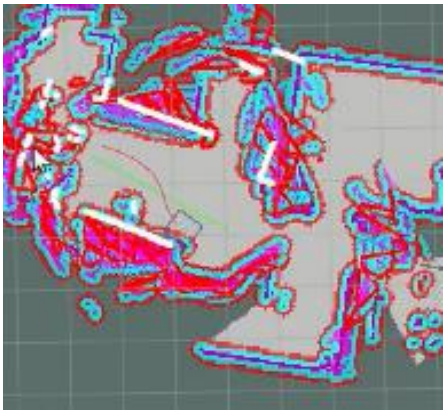


Figure 4.45: Weight_dynamic_obstacle =500

Besides, based on the results shown in Figure 4.46, when the obstacle moving across the dynamic obstacle, the local path planned will be distorted and these results obtained same as the simulation results shown in Figure 4.47. This is because, the higher the dynamic_obstacle_inflation_dist, the greater the local path planned will be distorted further. Moreover, based on the results obtained, the weight_dynamic_obstacle_inflation is not having any significant effect to the path planned by the TEB algorithms in simulation and room environment.

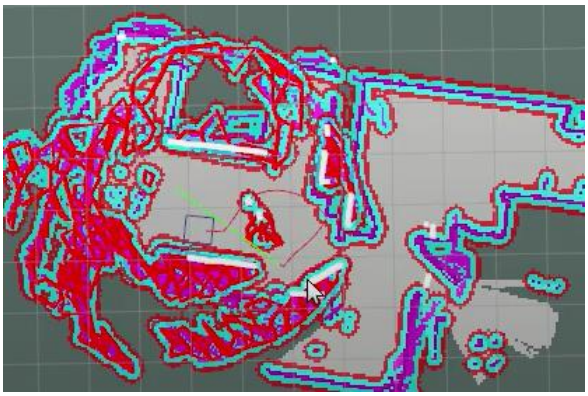


Figure 4.46: Dynamic_obstacle_inflation_dist = 4.0

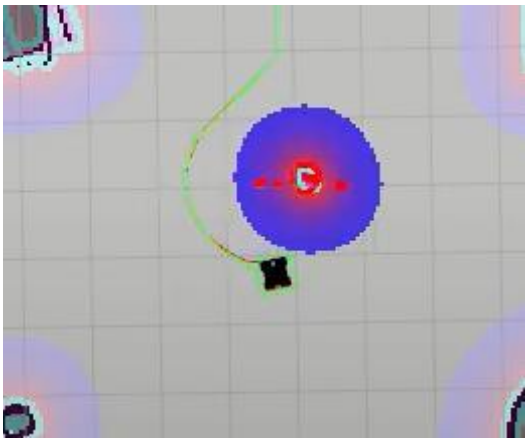


Figure 4.47: Simulation result when dynamic_obstacle_inflation_dist = 4.0

4.10 Summary

The navigation pack had successfully implemented on the office assistant robot. Besides, the map had generated by using the SLAM algorithm. Moreover, the costmap parameters are modified based on this application. On top of that, the 2 global planners which are A* and Dijkstra had compared. Other than the global planner, the 2 local planners which are DWA and TEB had compared with simulation and real environment. The most suitable local planner was selected which are TEB due to its navigation behaviour and the feasibility on avoiding dynamic obstacles. Besides, the parameters of TEB had studied and implemented on the office assistant robot which are shown in Table 4.6, Table 4.5 and Table 4.4. The overall rqt_graph was shown in Figure 4.48.

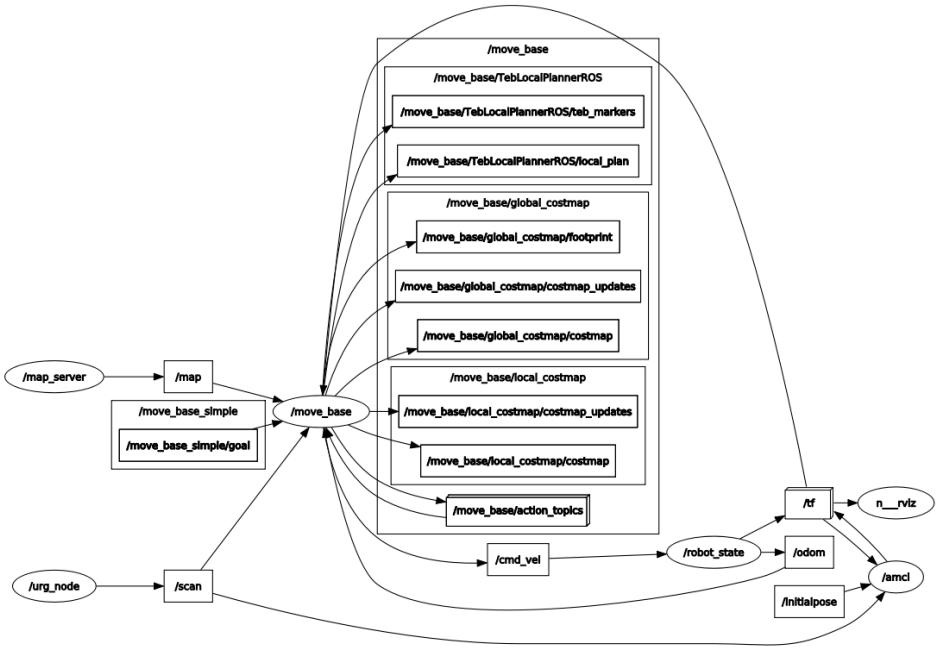


Figure 4.48: Overall rqt_graph

Table 4.4: Costmap Setup

Inflation_dist	1.0
Cost_scaling_factor	3.0

Table 4.5: Parameters modified for Global planner

Parameters	Values
Lethal_cost	150

Table 4.6: Parameters modified for TEB local planner

Parameters	Values
max_vel_x	0.5 m/s
max_vel_tetha	0.5 rad/s
footprint_model/type	“line”
min_obstacle_dist	0.3 m
include_dynamic_obstacle	True
costmap_converter_plugin	costmap_converter::CostmapToDynamicObstacles
static_converter_plugin	costmap_converter::CostmapToPolygonsDBSMCCH
weight_kinematics_forward_drive	800
enable_homotopy_class_planning	False

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The simulation environment and the differential drive robot model was built with gazebo and several experiments with the robot was tested. Moreover, the navigation pack had successfully implemented on the office assistant robot. The behaviour and the feasibility of the robot on navigation were studied.

In conclusion, for the differential drive robot to navigate, map are needed to generate with gmapping method of SLAM algorithm. Besides, LiDAR sensor data are used to detect obstacles and also used with odometry data for localization with AMCL algorithm. Moreover, costmap are needed to implement on the robot. The parameters inflation_dist and cost_scaling_factor are able to adjust the cost of the buffer zone of the obstacles. The higher the inflation_dist, the bigger the cost of the buffer zone around the obstacle. On top of that, A* has faster computation time than Dijkstra algorithm but both global planners are able to achieve the requirements in this application. Besides, the parameter, lethal_cost is affecting the path planned by the global planner. The higher the lethal_cost, the path planned will tend to travel in a narrow corridor and conflicting the local planner. Besides, the TEB local planner are selected for this application. This is because, it can modify the parameter of weight_forward_drive to adjust the robot weight in travel in forward motion and reduce the weight in backwards motion.

Besides, TEB can avoid dynamic obstacles by using the dynamic obstacle layer. It can track and predict dynamic obstacles and avoid it. Besides ,the parameters min_obstacle_dist and footprint model are affecting the distance between the obstacle and the robot. The higher the min_obstacle_dist, the higher the distance between the robot and obstacles. dynamic_obstacles_inflation_dist and weight_dynamic_obstacle also affect the behaviour of robot when facing dynamic obstacles. The higher the dynamic_obstacles_inflation_dist and the weight_dynamic_obstacle, the robot will tend to avoid the obstacle. Lastly, The optimum value of parameters which

are shown in Table 4.4, Table 4.5 and Table 4.6 were implemented on the differential drive office assistant robot to navigate in the room and simulation environment with obstacle avoidance.

5.2 Recommendation for future direction

In this research, only 2 or less dynamic obstacles was tested in the real environment. One of the future recommendations is to increase the number of dynamic obstacles and evaluate the performance of the local planner, TEB and the parameters affected in the crowded environment. Besides, in this research, the robot is unable to avoid the dynamic obstacles when it moving toward the robot non-stop. In future study, the TEB algorithm needed to be improved in order to avoid the obstacle.

REFERENCES

Lozano-Pérez, T. and Wesley, M.A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10), pp.560-570.

Gasparetto, A., Boscariol, P., Lanzutti, A. and Vidoni, R., 2015. Path planning and trajectory planning algorithms: A general overview. In *Motion and operation planning of robotic systems* (pp. 3-27). Springer, Cham.

Ferrer Sánchez, J., 2018. Implementation and comparison in local planners for Ackermann vehicles.

Fox, D., Burgard, W. and Thrun, S., 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*(1), pp.23-33.

Quinlan, S. and Khatib, O., 1993, May. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation* (pp. 802-807). IEEE.

Cybulski, B., Wegierska, A. and Granosik, G., 2019, July. Accuracy comparison of navigation local planners on ROS-based mobile robot. In *2019 12th International Workshop on Robot Motion and Control (RoMoCo)* (pp. 104-111). IEEE.

Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F. and Bertram, T., 2012, May. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics* (pp. 1-6). VDE.

Marin-Plaza, P., Hussein, A., Martin, D. and Escalera, A.D.L., 2018. Global and local path planning study in a ROS-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, *2018*.

Pittner, M., Hiller, M., Particke, F., Patino-Studencki, L. and Thielecke, J., 2018, June. Systematic analysis of global and local planners for optimal trajectory planning. In *ISR 2018; 50th International Symposium on Robotics* (pp. 1-4). VDE.

Norzam, W.A.S., Hawari, H.F. and Kamarudin, K., 2019, November. Analysis of Mobile Robot Indoor Mapping using GMapping Based SLAM with Different Parameter. In *IOP Conference Series: Materials Science and Engineering* (Vol. 705, No. 1, p. 012037). IOP Publishing.

Jasprit, S., 2018. *Navigation/Tutorials/Robotsetup - ROS Wiki*. [online] Wiki.ros.org. Available at: <http://wiki.ros.org/navigation/Tutorials/RobotSetup> [Accessed 2 September 2020].

Clearpath Robotics, 2020. *ROS 101: Intro To The Robot Operating System | Robohub*. [online] Robohub.org. Available at: <https://robohub.org/ros-101-intro-to-the-robot-operating-system/> [Accessed 10 September 2020].

Dattalo, A., 2018. *ROS/Introduction - ROS Wiki*. [online] Wiki.ros.org. Available at: <http://wiki.ros.org/ROS/Introduction> [Accessed 10 September 2020].

Calderone, L., 2019. *What Are Service Robots? | Roboticstomorrow*. [online] Roboticstomorrow.com. Available at: <https://www.roboticstomorrow.com/article/2019/02/what-are-service-robots/13161> [Accessed 10 September 2020].

Aitken, J.M., Veres, S.M. and Judge, M., 2014. Adaptation of system configuration under the robot operating system. *IFAC Proceedings Volumes*, *47*(3), pp.4484-4492.

Albers, F., Rösmann, C., Hoffmann, F. & Bertram, T. (2019) Online Trajectory Optimization and Navigation in Dynamic Environments in ROS.

In: Koubaa, A. (Ed.) Robot Operating System (ROS). Springer International Publishing: Cham, pp. 241–274.

ISO 8373::2012 Robots and robotic devices — Vocabulary.

Koubaa, A., Sriti, M.-F., Javed, Y., Alajlan, M., Qureshi, B. & Ellouze, F. et al. (2016 - 2016) Turtlebot at Office: A Service-Oriented Software Architecture for Personal Assistant Robots Using ROS. In: 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), 5/4/2016 - 5/6/2016, Bragança, Portugal. IEEE, pp. 270–276.