A Study on the Effect of Parameters for ROS Motion Planer and Navigation System for Indoor Robot

Chen Zheng Looi[1], Danny Wee Kiat Ng[1,*]
[1]Department of Mechatronics & Biomedical Engineering, Universiti Tunku Abdul Rahman, Malaysia,
Email: chenzheng98@hotmail.com, ngwk@utar.edu.my
[*] *Corresponding author*

Abstract: **In the past decades, the service robot industry had risen rapidly. The office assistant robot is one type of service robot used to assist officers in an office environment. For the robot to navigate autonomously in the office, navigation algorithms and motion planners were implemented on these robots. Robot Operating System (ROS) is one of the common platforms to develop these robots. The parameters applied to the motion planners will affect the performance of the Robot. In this study, the global planners, A* and Dijkstra algorithm and local planners, Dynamic Window Approach (DWA) and Time Elastic Band (TEB) algorithms were implemented and tested on a robot in simulation and a real environment. Results from the experiments were used to evaluate and compare the performance of the robot with different planners and parameters. Based on 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.**

Keywords: **DWA, Global Planner, Local Planner, ROS, TEB.**

I. INTRODUCTION

Service robots are a type 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 carries out useful work for humans. [1]. Service robots are meant to be used in the task which is menial, repetitive, and time-consuming. 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. [2] and state prediction of the officers. [3]. A typical autonomous robot will implement SLAM, global and local planner for autonomous navigation. The robot can localize itself and build a map based on the 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 the global and local planner. A global planner is used for optimal path planning with a known environment (global costmap) while a local planner is based on the real-time sensor data such as a LiDAR sensor to plan optimal trajectories which will track the global path while performing dynamic obstacles avoidance.

Global planning algorithms such as Dijkstra and A* algorithm; local planners such as Dynamic Window Approach (DWA) and Time Elastic Band (TEB) are available as pre-built packages in ROS. [4]. These packages used in multiple robot applications such as blind-guiding robot [5], intelligent transport robot [6] and driverless vehicle auto parking [7].

For global planners, based on the research done by [8], Dijkstra and A* algorithms are compared in term of their computation time and path length. In a simple environment, A* algorithm has a shorter computation time with the same path length especially with the Manhattan distance equation as a heuristic function. This is because the Dijkstra algorithm explores unused cell/nodes in an undirected fashion. Besides, one of the advantages of using Dijkstra is it can shorter the path length when using the Gradient descent method to reduce the path length.

As for local planners, DWA and TEB will be tested in this study. Local planners are responsible to perform obstacle avoidance while the robot is navigating towards the goal. Both DWA and TEB able to avoid obstacles while navigating. Research done by [9], concludes that each planner has its advantages, DWA has better consistency while TEB has the fastest computation time.

This study aims to determine the effects of parameters for motion planners and navigation algorithms on autonomous navigation in an office assistant robot. Studies are conducted to compare the feasibility of different type of global and local planner in ROS. DWA and TEB algorithm both has their advantages; it is worth to investigate more about its performance to determine which more suitable in this application. Therefore, A* and Dijkstra algorithm as a global planner, DWA and TEB algorithm as the local planner will be compared and selected to implement on the office assistant robot in this research.

II. ALGORITHMS

This section will cover the algorithms that were compared in this study. These include the Dijkstra, A*, DWA and TEB algorithms.

A. Dijkstra Algorithm

The map is divided into cells and form a grid cell map. 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 according to the distance between the robot and the nodes which represent the cost function, g(χ). Each generation will compute the neighbour's empty nodes and assign the values to the empty nodes until it reaches the goal as shown in Fig. 1. The minimum cost path is selected as the path generated. [4].

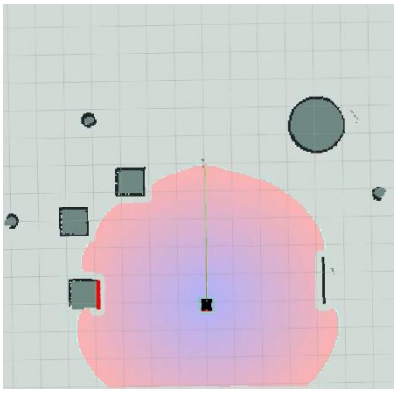

Fig. 1. The green line indicates the path and the circle search space generated by the Dijkstra algorithm.

B. A* Algorithm

A* assigned one more value to the nodes on top of those used in Dijkstra algorithm with a heuristic function, h(χ) 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 the heuristic function and the original cost function, g(χ) obtained from the Dijkstra algorithm, the new cost function, f(χ) is generated. [1]. The equation is shown in (1).

$$f(X) = g(X) + h(X) \tag{1}$$

C. DWA Algorithm

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 [10]. The trajectories in the search space are based on 3 criteria, circular trajectories, dynamic window and admissible velocities.

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

Next, the trajectories in the search space will be reduced to the dynamic window based on the acceleration limit of the robot. The dynamic window, Vd only contain velocities that are reachable by the robot within the time interval. Hence the resultant search space, Vr is the combination of these three criteria.

D. TEB Algorithm

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 [9]. The robot poses, X consists of 3 elements which are position and orientation based on the related frame, {map} which shown in Fig. 2. This sequence of the pose, Q is defined as:

$$Q = \{X_i\}_{i=0\ldots n} \tag{2}$$

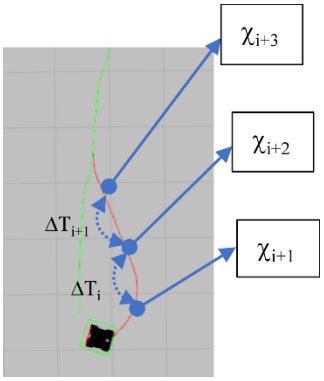

Fig. 2. The sequence of the configuration of a robot.

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

$$\tau = \{\Delta T_i\}_{i=0\ldots n-1} \tag{3}$$

The TEB is represented as:

$$B = (Q, \tau) \tag{4}$$

B is optimized by the objective function. The objective function is based on several parameters such as the dynamic constraints of the robots and the distance between obstacles. The path with the minimum cost will be selected as the path for the robot to travel on.

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. [11].

III. SYSTEM IMPLEMENTATION

In this section, the hardware specifications and the overall implementation of the system will be discussed.

A. Hardware Specification

The office assistant robot is a differential drive robot with four wheels. It has 2 omnidirectional free wheels mounted on the back of the robot and 2 front wheels independently powered by 2 DC brushed motors with built-in encoders. The built-in encoder is used to calculate the odometry data of the robot. A motor control board connected to a dual-channel motor driver powered by a 12V LiPo battery is used to drive the motor. A LiDAR sensor that provides 240-degree laser scanning data is mounted in the front of the robot. A laptop running Ubuntu 18.04 LTS with ROS Melodic processed the data obtained from the sensors and output the command velocity with Proportional-Integrative-Derivative (PID) controller to the control board for motor controls. The robot system is shown in Fig. 3 while the overview of the robot base is shown in Fig. 4.

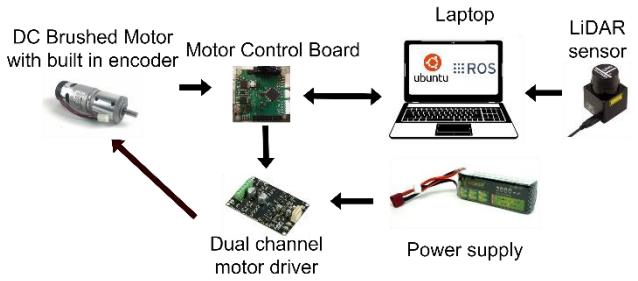

Fig. 3. System of the office assistant robot

Fig. 4. Office Assistant robot

B. *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. [12]. To implement the navigation stack, multiple configurations needed to be set up. The move_base package will take the sensor transform, odometry, map and sensors as inputs, and generate the path and trajectory and output the velocity commands to the base_controller to move the robot with the desired velocity.

- SLAM: Mapping is generated by using the sensor data and odometry of the robot. It generates high quality and accurate map for the robot to navigate. [13]. The map was generated by moving the robot around the environment. Maps were generated by using the Gmapping method [14].
- Localization of robot: The robot is localized based on the Adaptive Monte Carlo localization (AMCL) algorithm [15]. The map generated, the odometry data and the real-time LiDAR sensor data is fed to the algorithm to localize the position of the robot in the environment. Then output the robot's estimate position as /tf topic.
- Costmap setup: There were 2 costmap implemented on the robot. The global and local costmap. Global costmap consists of 3 layers which are the static map layer, inflation layer and obstacle map layer. There are two parameters to tune the inflation layer which are inflation_dist and cost_scaling_factor. The higher the inflation_dist, the higher the buffered zone around the obstacles. Obstacle map layer used to track the dynamic obstacles or the obstacles not shown in the static map. Unlike global costmap, local costmap only consists of inflation layer and obstacle map layer.
- 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 velocity commands to control the robot. After that, a launch file was created to launch multiples nodes such as the map server, the AMCL packages and the move_base packages. Besides, the parameters for the costmap and the local planner were published to the parameter server through the launch file also.

IV. EXPERIMENT SETUP

To analyse the performance of the different planners, multiple simulation environment which shown in Fig. 5 and Fig. 6, a differential drive robot model was created which shown in Fig. 7 and the correspond map was generated which shown in Fig. 8.

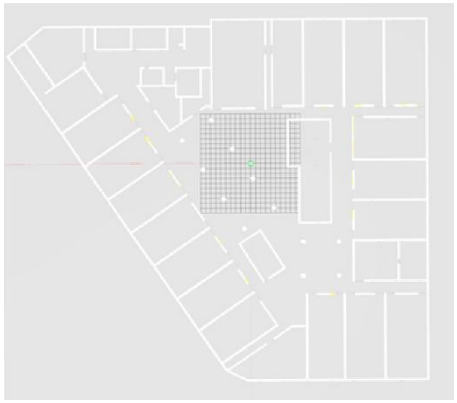

Fig. 5. Simulation environment A

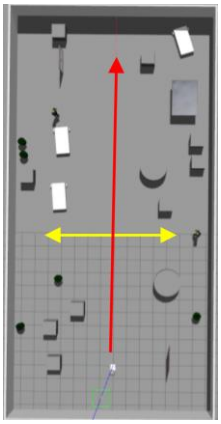

Fig. 6. Simulation environment B with dynamic obstacles

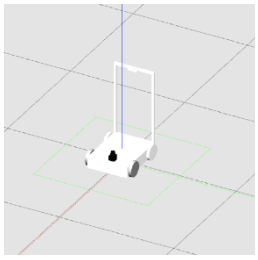

Fig. 7. Robot model in simulation

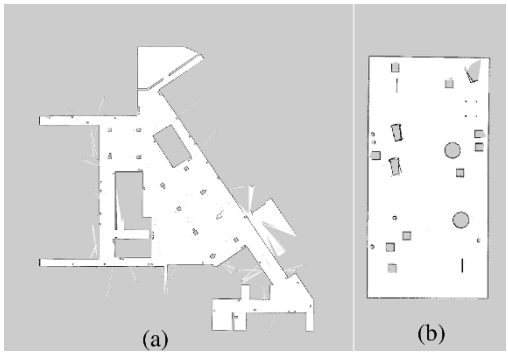

Fig. 8. Floor maps generated by SLAM. (a) Simulation Environment A (b) Simulation environment B.

A. Simulation on Simulation Environment A

To compare the global planner, the robot was given 2 specific goals in the simulation environment. One of the goals is 10 meter in front of the robot, while another goal is the coordinate of (10, -25) of Fig. 5. The computation time and the path planned will be recorded.

Besides, to compare the path generated by the different local planner, 2 local planner algorithms were implemented on the simulated robot which are DWA and TEB algorithm. In this research, the footprint model of the robot with TEB was changed to a line instead of a point. This is because the robot is a rectangle shape instead of a circular robot. Therefore, the line footprint model has better represented than a point footprint model. Besides, due to the limitation of computation power, the enable_homotopy_class_planning was changed from true to false to reduce the computation requirement. The first experiment was tested by sending a specific goal to the robot with different local planner algorithms the robot will be given different goals. The behaviour of the robot will be recorded.

B. Simulation on Simulation Environment B with Dynamic Obstacle

On top of that, to verify the performance of the algorithm in the avoidance of dynamic obstacles a simulation environment with dynamic obstacles was constructed in Gazebo as shown in Fig. 6. 2 types of dynamic obstacles are simulation in the environment, 1) obstacle that are moving the perpendicular to the direction of motion of the robot and 2) obstacle that are moving toward the obstacles shown in Fig. 9.

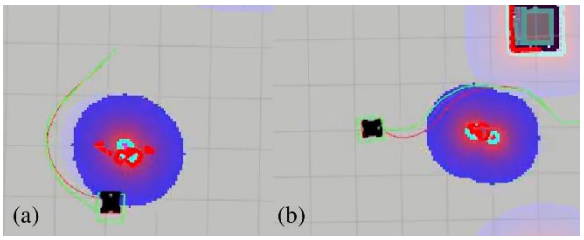

Fig. 9. Path planned when obstacle moving perpendicular the robot, red line indicate the local planner path, while green line indicate the global planner path (a). Path planned when obstacle moving toward the robot (b).

C. Implementation on Office Assistant Robot

Next, the selected algorithm will be implemented on the office assistant robot. The costmap parameters, inflation_dist was reduced to 0.1. The maximum forward velocity was reduced to 0.5 m/s, the maximum angular velocity was reduced to 0.5 rad/s and the min_obstacle_dist was reduced to 0.3 due to the size of the environment. Next, the robot is tested with the dynamic obstacles perpendicular and toward the robot in the room. The map generated was shown in Fig. 10.

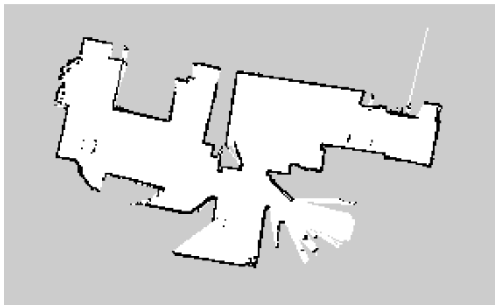

Fig. 10. Map of room.

D. Tuning of Parameters

Lastly, some of the parameters of TEB was tuned to investigate the performance of the robot when facing dynamics obstacles which are, weight_dynamic_obstacles, weight_dynamic_obstacles_inflation and dynamic_obstacles_inflation_dist. The parameters were increased to study the effect of modifying the parameters. The weight_dynamic_obstacles was increased from default value, 50 to 500, dynamic_obstacles_inflation_dist from 0.6 to 4.0 and weight_dynamic_obstacles_inflation from 0.01 to 1.0.

V. RESULTS ANALYSIS

A. Comparison of Global Planner Based on Simulation

Fig. 11 shown the path planned by the Dijkstra algorithm and A* algorithm. The computation time of the global planner algorithms are shown in Table Ⅰ and Table Ⅱ. Based on the results obtained, both global planner algorithms can generate the path to the goal. However, A* algorithm is 462.5% faster than Dijkstra algorithms when generating the path to 10 meter in front of the robot and 240.2% faster than Dijkstra algorithms when generating the path to the coordinate of (10,-25) of Simulation Environment A which is shown in

Fig. 12. This is because, A* has additional cost function, h(χ) which computes the distance between the nodes and the goal. The nodes further from the goal have higher cost in the A* algorithm and it eventually reduces the search space created when using the A* algorithm. Even though the computation time for A* is much faster, the time taken is in the range of milliseconds and only contribute to a small fraction of the total time in any given navigation process. It can be concluded that both global planners are suitable for use in this application. The global planner package is used with default parameters except cost_scaling_factor set to 0.1, old_navfn_behaviour set to true and use_dijkstra is set to default when using Dijkstra and false when using A* algorithm.

TABLE I. COMPUTATION TIME OF GLOBAL PLANNERS WITH COORDINATE 10 METER GOAL

Algorithm	Computation time, ms
A*	1.6
Dijkstra	7.4

TABLE II. COMPUTATION TIME OF GLOBAL PLANNERS WITH COORDINATE (10, -25) GOAL

Algorithm	Computation time, ms
A*	18.4
Dijkstra	44.2

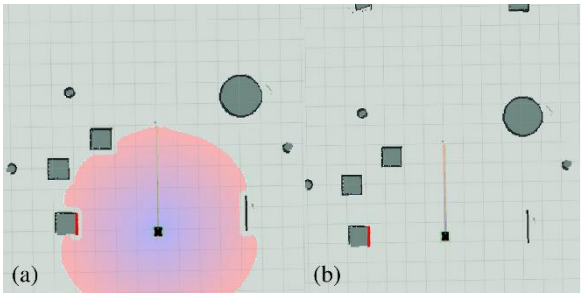

Fig. 11. Path planned by Dijkstra will form a circle of search space in which the robot act as a centre point (a), while A* will form a narrow search space toward the 10 meter goal (b)

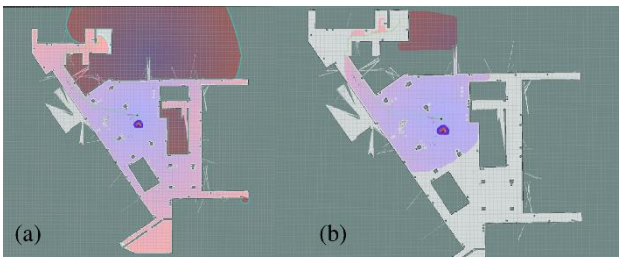

Fig. 12. Path planned by Dijkstra(a) with coordinate (10, -25) goal. Path planned by A* (b).

B. Comparison of Local Planner

Based on the results obtained, the robot is unable to move backwards using the default parameters of DWA. However, by tuning the min_vel_x parameter to a negative value, the minimum velocity of the robot is changed which allow the robot to move backwards. But this causes the robot to prefer backwards motion when the goal is located behind the robot. Unlike DWA, the robot with TEB can travel in backward motion when using the defaults parameter. By increasing the values of weight_kinematics_forward_drive parameters from 1 to 800, the robot will have a higher weight on travelling in forward motion and able to move backward when it is needed.

Both algorithms have bad performance with the default parameters when facing dynamic obstacles moving towards the robot. Based on Fig. 13 (a), the robot will only reduce its velocity when the obstacle is right in front of the robot and the angular velocity remain approximately constant. To improve the result of DWA algorithm, multiple parameters were tuned which shown in the Table Ⅲ.

TABLE III. PARAMETERS CHANGED TO IMPROVE DWA ALGORITHM

Parameters	Definition	Values
sim_time	The amount of time to simulate forward trajectories.	1.7 to 3.5
vx and vth_samples	Number of forward and angular sampling velocity.	(3 and 20) to (20 and 40)
ooccdist_scale	Weight of the cost to avoid obstacle	0.01 to 1.0

Although, the parameters had increased to improve the DWA algorithm on obstacle avoidance but it didn't improve the result on avoiding dynamic obstacles. The angular velocity of the robot remains approximate constant to 0 rad/s. Hence, the robot is unable to avoid the dynamic obstacles with DWA. The robot with TEB has slightly better performance than DWA algorithm facing the perpendicular dynamic obstacles. Based on the results shown in Fig. 13 (b), the robot will increase its angular velocity to approximate 0.75 rad/s the robot unlike the robot with DWA which remain approximately zero.

To improve the results of TEB algorithm, there are few parameters were tuned, which are include_dynamic_obstacles and costmap_converter_plugin with dynamic obstacle layer (DOL). Based on the results shown in Fig. 13 (c), the angular velocity increase 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 can predict the moving obstacles motion and able to stop and turn before hitting the dynamic obstacles, which shown in Fig. 13 (d).

According to [16] the working of algorithms starts with a background subtractor and blob detector to track and locate the obstacles. A new track will be generated if there are obstacles that are not tracked, and older tracked obstacles will be removed due to inactivity. The tracks are generated based on the current and past position of the obstacles. To solve the data acquisition problem, Hungarian algorithm was implemented. Lastly, the Kalman filter with the first constant velocity model was applied to estimate the velocity of the dynamic obstacles. In conclusion, based on the simulation results, TEB algorithm was chosen to be applied 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.

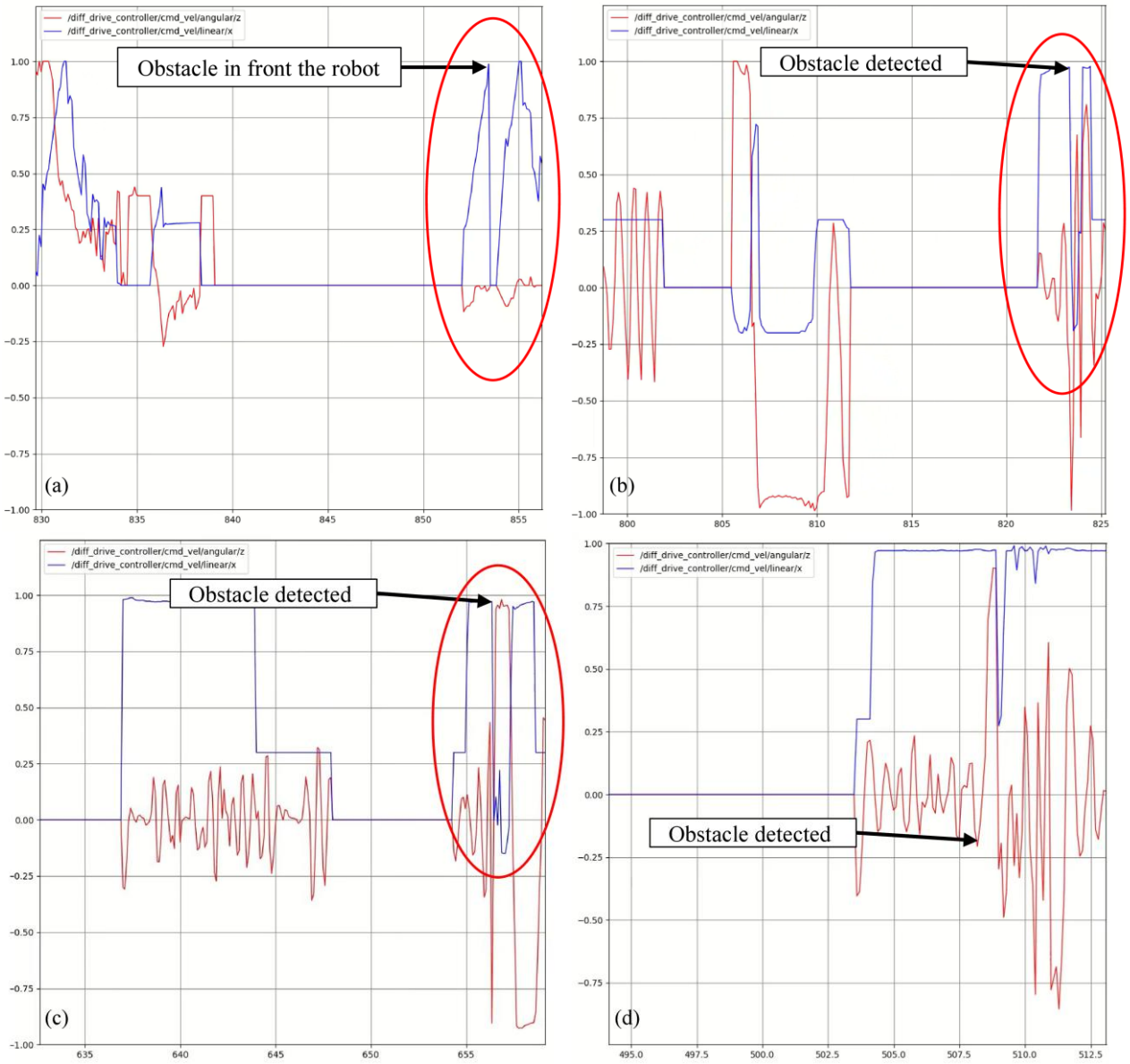

Fig. 13. Command velocity graph against time for perpendicular dynamic obstacle (DWA), while blue line indicate linear velocity and red line indicate angular velocity. (a). Command velocity graph against time for perpendicular dynamic obstacle (TEB) (b). Command velocity graph against time for perpendicular dynamic obstacle (TEB+DOL) (c). Command velocity graph against time when obstacle moving toward robot (TEB+DOL) (d).

C. *TEB on Real-World Experiment*

Based on the results shown in Fig. 14, the real world experiment result is approximately the same as the simulation results. The robot is able to plan a path in order to avoid the dynamic obstacles.

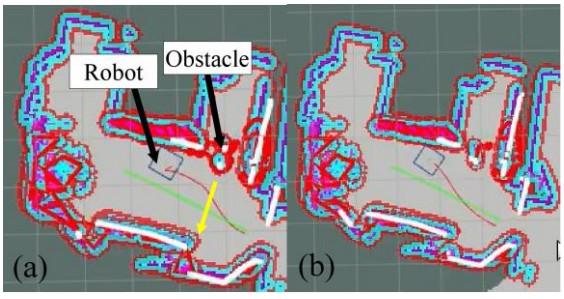

Fig. 14. Path planned when obstacle moving across the robot in real environment (TEB + DOL) (a). The path had bend to avoid the obstacle (b).

On top of that, based on the result obtained, the robot will slow down, stop and increase its angular velocity to avoid the obstacles moving toward and generate a path shown in Fig. 15. Besides, if the obstacles continue to move towards the robot without stopping, the robot will collide with the obstacles and the robot will move backwards afterwards. If the obstacles stop right before hitting the robot, the robot will rotate and follow the path planned.

According to the simulation and real-life results, it can conclude that the office assistant robot with TEB and DOL is able to avoid obstacles more efficient than TEB only. Therefore, TEB with DOL were implemented to the office assistant robot.

D. *Parameters Affecting Dynamic Obstacles*

Based on the results obtained. the path planned is further away from the moving obstacles when it moving toward the robot when weight_dynamic_obstacles = 500 which shown in Fig. 15.

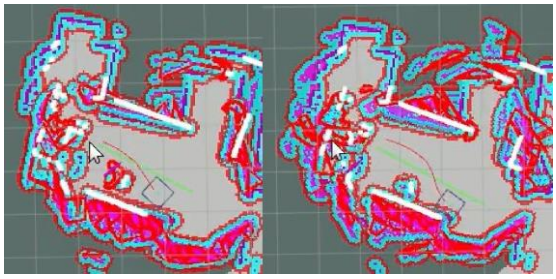

Fig. 15. Path planned with default parameters when obstacle moving toward the robot (TEB+DOL) (a). Path avoid further when weight_dynamic_obstacles = 500 (b).

Besides, based on the results shown in Fig. 16, when the dynamic obstacle is perpendicular to the direction of travel, the local path planned will be distorted. This is because, the higher the dynamic_obstacles_inflation_dist, the greater the local path planned will be distorted further. Moreover, based on the results obtained, the weight_dynamic_obstacles_inflation is not having any significant effect on the path planned by the TEB algorithms in simulation and room environment. The parameters changed for the office assistant robot was shown in Table Ⅳ.

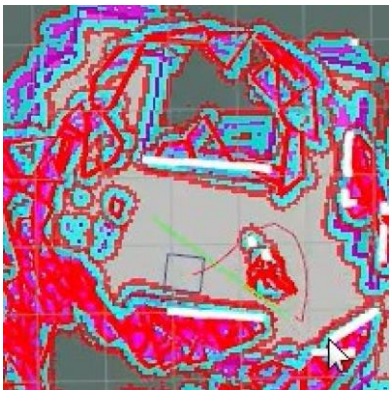

Fig. 16. Path planned distorted with dynamic_obstacles_inflation_dist = 4.0.

TABLE IV. PARAMETERS MODIFIED FOR OFFICE ASSISTANT ROBOT

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::CostmapToDynamic Obstacles
static_converter_plugin	costmap_converter::CostmapToPolygons DBSMCCH
weight_kinematics_forward_drive	800

VI. CONCLUSIONS AND FUTURE WORK

In conclusion, TEB local planner had implemented on the office assistant robot due to its excellent performance on avoiding dynamic obstacles and have better performance compare to DWA. Moreover, the TEB parameters had modified according to Table Ⅳ. The weight_forward_drive was increased to 800 to increase the tendency moving in forward motion. Besides, the dynamic constraints of the robot were reduced which correspond to the area of the environment.

Besides, TEB can avoid dynamic obstacles by implementing with DOL. It can track and predict the dynamic obstacles and avoid it. Dynamic_obstacles_inflation_dist and weight_dynamic_obstacle also affect the behaviour of the 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. In global planner, although, A* has faster computation time than Dijkstra algorithm, but both global planners are able to achieve the requirements in this application. Lastly, navigation stack with Dijkstra as global planner and TEB as global planner were implemented on the differential drive office assistant robot to successfully navigate in the room and simulation environment with obstacle avoidance. Further research will focus on evaluating the performance of the robot navigation system in crowded environment.

ACKNOWLEDGMENT

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.

REFERENCES

[1] *Robots and robotic devices* — Vocabulary, ISO 8373,2012.
[2] A. Koubâa et al., "Turtlebot at Office: A Service-Oriented Software Architecture for Personal Assistant Robots Using ROS," *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2016, pp. 270-276, doi: 10.1109/ICARSC.2016.66.
[3] A. Kouno, D. Takayama and E. Suzuki, "Predicting the State of a Person by an Office-Use Autonomous Mobile Robot," *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2012, pp. 80-84, doi: 10.1109/WI-IAT.2012.183.
[4] P. Marin-Plaza, A. Hussein, D. Martin, and A. d. La Escalera, "Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles," *Journal of Advanced Transportation*, vol. 2018, pp. 1–10, 2018, doi: 10.1155/2018/6392697.
[5] L. Tianyu, Y. Ruixin, W. Guangrui and S. Lei, "Local Path Planning Algorithm for Blind-guiding Robot Based on Improved DWA Algorithm," *Chinese Control and Decision Conference (CCDC)*, 2019, pp. 6169-6173, doi: 10.1109/CCDC.2019.8833259.
[6] B. Choi, B. Kim, E. Kim and K. W. Yang, "A modified dynamic window approach in crowded indoor environment for intelligent transport robot," *12th International Conference on Control, Automation and Systems*, 2012, pp. 1007-1009.
[7] Z. Yongzhe, B. Ma and C. K. Wai, "A Practical Study of Time-Elastic-Band Planning Method for Driverless Vehicle for Auto-parking," *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, 2018, pp. 196-200, doi: 10.1109/ICoIAS.2018.8494025.
[8] M. Pittner, M. Hiller, F. Particke, L. Patino-Studencki and J. Thielecke, "Systematic Analysis of Global and Local Planners for Optimal Trajectory Planning," *50th International Symposium on Robotics*, 2018, pp. 1-4.
[9] B. Cybulski, A. Wegierska and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," *12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 104-111, doi: 10.1109/RoMoCo.2019.8787346.
[10] Jordi Ferrer S´anchez, "Implementation and comparison in local planners for Ackermann vehicles," pp. 3–7, 2018.
[11] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," *7th German Conference on Robotics*, 2012, pp. 1-6.
[12] J. S. Gill, Setup and Configuration of the Navigation Stack on a Robot. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup (accessed: Accessed: 02-Sep-2020).
[13] W.A.S. Norzam, H. F. Hawari, and K. Kamarudin, "Analysis of Mobile Robot Indoor Mapping using GMapping Based SLAM with Different Parameter," *IOP Conference Series: Materials Science and*

Engineering, vol. 705, p. 12037, 2019, doi: 10.1088/1757-899X/705/1/012037.

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

[15] D. Fox, W. Burgard, F. Dellaert, S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," *AAAI Proceedings*, 1999, pp. 343-349.

[16] F. Albers, C. Rösmann, F. Hoffmann, and T. Bertram, "Online Trajectory Optimization and Navigation in Dynamic Environments in ROS," *Studies in Computational Intelligence, Robot Operating System (ROS)*, Cham: Springer International Publishing, 2019, pp. 241–274.