Comparison and Improvement of Local Planners on ROS for Narrow Passages

Huajun Yuan
School of Advanced Technology
Xi'an Jiaotong-Liverpool University
Suzhou 215123, China
Huajun.Yuan21@student.xjtlu.edu.cn

Hanlin Li
School of Advanced Technology
Xi'an Jiaotong-Liverpool University
Suzhou 215123, China
Hanlin.Li21@student.xjtlu.edu.cn

Yuhan Zhang
School of Advanced Technology
Xi'an Jiaotong-Liverpool University
Suzhou 215123, China
Yuhan.Zhang2102@student.xjtlu.edu.cn

Shuang Du
Department of Research and Development
Suzhou Inteleizhen Intelligent Technology Co., Ltd
Suzhou 215123, China
dushuang@irobotop.com

Limin Yu*
Department of Communications and Networking
Xi'an Jiaotong-Liverpool University
Suzhou 215123, China
Limin.Yu@xjtlu.edu.cn

Xinheng Wang*
Department of Mechatronics and Robotics
Xi'an Jiaotong-Liverpool University
Suzhou 215123, China
Xinheng.Wang@xjtlu.edu.cn

***Abstract*—Path planning is challenging in various extreme and complex environments due to the explosive growth of autonomous mobile robots. The widely used Dynamic Window Approach (DWA) and Time Elastic Band (TEB) local planners are integrated into Robot Operating System (ROS). A great quantity of their parameters has a significant impact on the actual performance of planning. Aiming at the scenario of robots passing through narrow passages, this work compared and optimized the typical DWA and TEB. The results of simulations showed that the improved DWA and TEB local path planners both have better performance with gentler fluctuations of velocity and smoother paths of travel in continuous narrow passages. In addition, the number of collisions was reduced to zero and the time consumption of completing navigation was decreased by an average of 15% and 7%, respectively. Moreover, the general strategies and suggestions for adjusting related parameters of DWA and TEB through narrow passages were given in this paper. Finally, the adjusted TEB local planner was applied to the customized robot in a real similar environment, and it accomplished the navigation stably with a smooth path. The results of real experiments proved the validity and reliability of our work further.**

Keywords— local path planning; dynamic window approach (DWA); time elastic band (TEB); robot navigation; simultaneous localization and mapping (SLAM)

I. INTRODUCTION

Benefiting from the ability to navigate in an environment without the demand of physical or electro-mechanical guidance devices, autonomous mobile robots have been widely used in industry, agriculture, service, and household scenarios to improve efficiency and save labor costs. These robots perform different tasks by carrying different tools and sensors. Latombe defined that the basis for accomplishing these tasks was the capability of a robot to move autonomously and adaptively [1]. Durrant, et al. proposed that path planning is one of the key technologies of autonomous mobile robots, and described it as finding a path between the start point and the target that was smooth and collision-free in the working environment [2].

However, as mobile robots are used in a wider range of scenarios and are given more tedious tasks, such as autonomous driving, smart logistics, smart home, emergency rescue, and space exploration, those complex and dynamic environments pose new challenges to the path planning ability of mobile robots. When the robot has a priori environment information in advance, a path from the starting point to the target will be created, which is called global path planning [2]. To solve path planning problems in various static environments, plenty of researchers have proposed and optimized many typical global path planning strategies and algorithms. H. Kang, et al. applied a particle swarm optimization algorithm to the Dijkstra path that was improved by graph search and obtained better performance [3]. M. Luo, et al. modelled and transformed the curved surface with Delaunay triangulation, and explored an extended Dijkstra algorithm applicable to finding an optimal path of a curved surface [4]. M. Gracia, et al. combined fuzzy cost function evaluation and ant colony optimization metaheuristic algorithm, which improved the speed of navigation by about 10% [5].

Nevertheless, when a mobile robot performs tasks, a variety of dynamic factors in the environment will affect or restrict its stable movements, such as extreme climatic conditions [6], high-speed moving objects [7], complex terrain [8], and unknown obstacles [9]. Hence, modern mobile robots typically use global path planning as guidance but execute local path planning based on real-time information from multiple sensors. Dynamic Window Approach and Time Elastic Band algorithm are widely used as local path planning algorithms. D. FOX, et al. proposed the Dynamic Window Approach (DWA) algorithm that applied the optimal solution among all possible paths based on multiple evaluation functions [10]. Referring to the EB algorithm, Rösmann, et al. came up with an online collision avoidance method for multi-objective online trajectory optimization named Time Elastic Band (TEB) [11][12]. Both DWA and TEB are capable of avoiding obstacles and accomplishing navigation tasks in most cases [13]. But their ability to avoid local extrema and convergence speed are weakened when mobile robots encounter complex environments such as concave, narrow passages, cornered walls, and dynamic obstacles [1]. Research [14] concluded that TEB should be distinguished when it came to the speed of action and the sensitivity of obstacles, and DWA was better in terms of consistency and concentration of consecutive trials. I. Naotunna, et al. compared their

*Corresponding author

978-1-6654-9144-0/22/$31.00 ©2022 IEEE

performance for a heavy differential drive robot and considered applying the optimized TEB on a real large robot [15]. C.Z. Looi, et al. explored the effect of parameters for ROS motion planners, proposed that TEB was more feasible than DWA in avoiding dynamic obstacles through simulations, and implemented a differential drive office assistant robot successfully [16]. However, previous research did not put forward a general conclusion to solve the problem of path planning with DWA or TEB for autonomous robots in narrow environments.

This work aims to determine the effects of parameters of local planners integrated on ROS when robots move through narrow passages. The simulation results showed that the improved DWA and TEB had a better performance on smoother velocity fluctuations and smoother paths. Moreover, the time consumption for the navigation was reduced by an average of 15% and 7%, respectively. Based on the strategies of tuning related parameters concluded, the adjusted TEB local planner was applied to a customized robot to pass through multiple narrow passages, and successfully achieved good performance with a stable and smooth path.

The layout of this paper is as follows. Section II introduces the DWA and TEB local planners on ROS. Section III defines the experimental setup, including the descriptions of robots and environments used in simulations and real experiments. Then, section IV describes the methodologies and procedure of simulations with original and customized parameters to compare their performance through narrow passages. In addition, the selected local path planner is applied to the customized robot and tested in a real environment. Section V provides the analysis of the results. Finally, the conclusion is given in Section VI.

II. Local Path Planning Algorithms

A. Dynamic Window Approach (DWA)

One of the most common local path planning methods on ROS is Dynamic Window Approach (DWA). It samples multiple sets of velocities in velocity space (v, w) and simulates the possible trajectories based on all the combinations (v, w) for a certain time (*sim_period*). After this, those trajectories will be evaluated with multiple evaluation functions, and the combination corresponding to the best trajectory will be selected to drive the robot [10].

1) Motion model of the robot: The DWA algorithm assumes that the trajectory of a two-wheeled mobile robot is a set of circular arcs, which are represented by pairs of (v, w). When the robot is differential, it can only move forward and rotate. Therefore, while the rotational angular velocity w is not zero, the coordinates of the robot at t_{i+1} are [17]:

$$x_{t+1} = x_t - \frac{v_t}{w_t}\sin\theta_t + \frac{v_t}{w_t}\sin\left(\theta_t + w_t \Delta t\right) \quad (1)$$

$$y_{t+1} = y_t - \frac{v_t}{w_t}\cos\theta_t + \frac{v_t}{w_t}\cos\left(\theta_t + w_t \Delta t\right) \quad (2)$$

$$\theta_{t+1} = \theta_t + w_t \Delta t \quad (3)$$

2) Speed sampling: As mentioned above, there is an infinite number of possible combinations of velocities (v, w) in the two-dimensional space. But the hardware of robots will also limit the speed to a certain range.

a) Limitations on the maximum and minimum speed of a robot:

$$V_m = \{v \in [v_{min}, v_{max}], w \in [w_{min}, w_{max}]\} \quad (4)$$

b) Limitations of motor performance: The actual speed the robot can reach is,

$$V_{t+1} = \left\{ (v, w) \middle| \begin{array}{l} v \in \left[v_t - \dot{v}_d \Delta t, v_t + \dot{v}_a \Delta t\right] \\ w \in \left[w_t - \dot{w}_d \Delta t, w_t + \dot{w}_a \Delta t\right] \end{array} \right\}, \quad (5)$$

where $\dot{v}_a$ and $\dot{v}_d$ are the maximum linear acceleration and maximum linear deceleration; $\dot{w}_a$ and $\dot{w}_d$ are the maximum angular acceleration and maximum angular deceleration.

c) Collision avoidance: To stop before colliding an obstacle, the range of velocities under the maximum deceleration is,

$$V_t = \left\{ (v, w) \middle| \begin{array}{l} v \le \sqrt{2 \cdot dist(v, w) \cdot \dot{v}_d} \\ w \le \sqrt{2 \cdot dist(v, w) \cdot \dot{w}_d} \end{array} \right\}, \quad (6)$$

where $\dot{v}_d$ and $\dot{w}_d$ are the maximum linear deceleration and maximum angular deceleration. $dist(v, w)$ is the closest distance to the obstacle along the trajectory of V_t.

3) Evaluation functions: Each generated trajectory is evaluated by the following weighted cost function [18]:

$$C(t) = \alpha \cdot Obs + \beta \cdot Gdist + \gamma \cdot Pdist + \delta \cdot \frac{1}{\dot{x}^2}, \quad (7)$$

where Obs is the sum of the grid cells costs that the trajectory passes through; Gdist is the shortest distance from the end of the trajectory to the goal; Pdist is the shortest distance from the end to the optimal path; x^2 is the translational component of the velocity.

As a result, the path that is far from the obstacle, toward the target, near the optimal path, and drives fast will get the highest evaluation and be selected.

B. Time Elastic Band (TEB) Algorithm

The Time Elastic Band algorithm describes the path planning problem as a multi-objective optimization problem, in which the objectives are minimizing the execution time, maintaining a certain distance from obstacles, and adhering to motion dynamics constraints. Since most of the optimization objectives are local and only related to some successive states of the robot, it is an optimization of a sparse model [19].

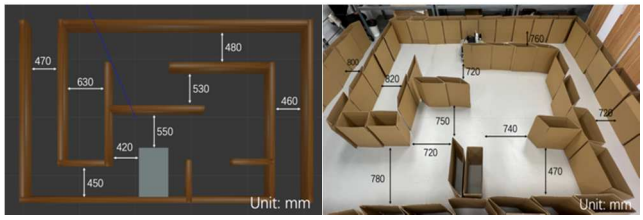
The novelty of TEB lies in the consideration of dynamic constraints and the addition of temporal information, such as representing the planning problem in a weighted multi-objective optimization framework by limiting the velocity and acceleration, which is also a bundle adjustment problem. As well, most objectives are local because they depend on some neighboring intermediate configurations [19][20]. This locality of TEB generates a sparse system matrix that allows large-scale optimization fast and efficiently. When the authors wrote their first paper, the support for sparse matrix optimization on ROS was imperfect, so the constraint was represented as a segmentally continuous and differentiable cost function. When it comes to their next paper in 2013, it was changed to a graph optimization method using g2o (the Livenberg method) [20].

III. Experimental Setup

To explore and optimize the performance of the original local path planners through narrow passages on ROS, it is necessary to build a proper scenario based on the dimensions of the robot selected. Moreover, since this work is a pre-study of an autonomous mobile robot applicable to the carriages on high-speed trains, a precise reconstruction of the whole scenario in the real world is essential for verifying the practical effectiveness of related works.

A. Description of Environments

In this work, local path planners had to be tested in a restricted environment. Therefore, a room with narrow passages, dead ends, and crowded corridors was designed in Gazebo, and there were two doors served as entrance and exit. It is important to mention that since the robot used in the simulations was different from the real one, the lengths of the cross-sections of these passages and corridors were adjusted accordingly to keep the reserved space consistent. Figure 1 shows the environments used in simulations and experiments. There is a unique path from the point in the top left to the point in the bottom right. The robot should pass through three longitudinal and four horizontal narrow passages, four C-bends, and two U-bends. The dimensions of each passage were marked in the figure. As shown in Fig. 1(a), the lengths of the vertical passages from left to right are 470mm, 630mm, 420mm, and 460mm, where the narrowest one only leaves 57mm for each side of the robot. From top to bottom, the lengths of the transverse passages are 450mm, 550mm, and 530mm, where the narrowest one only leaves 72mm for each side of the robot. In Fig. 1(b), the lengths of the vertical passages from left to right are 800mm, 820mm, 720mm, and 720mm, where the narrowest one only leaves 75mm for each side of the robot. From top to bottom, the lengths of the transverse passages are 780mm, 750mm, 720mm, and 760mm, where the narrowest one only leaves 75mm for each side of the robot.



(a) (b)

Fig. 1. Simulation environment & Real environment.

B. Description of Robots

The turtlebot3 Waffle Pi was used in the simulations. It is a differential drive robot with a 360° laser, and its URDF (Universal Robot Description Format) file is open source [21]. The robot used in the real experiment was customized based on the carriages on high-speed trains. Figure 2 shows the models of Turtlebot3 Waffle Pi and the customized robot. The former comes with a camera and a LiDAR located at the top center. While the latter is equipped with a depth camera at the center to provide basic visual information, and two LiDARs are located at the top left and bottom right. Table I lists the size, MCU/CPU, LiDAR, and other hardware properties of these two robots.

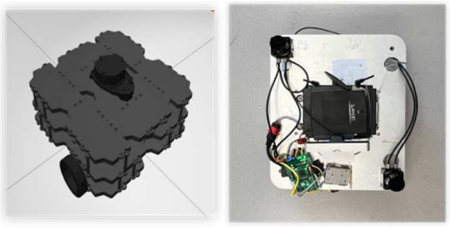


Fig. 2. Turtlebot3 Waffle Pi in Gazebo & customized robot.

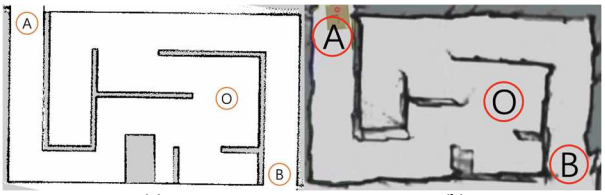
TABLE I. Properties of Turtlebot3 Waffle Pi and Customized Robot

Property	Turtlebot3 – Waffle Pi	Real Robot
Size (L x W x H)	281mm x 306mm x 141mm	550mm x 570mm x 200mm
MCU / CPU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)	Intel® Core™ i7-7600U CPU @ 2.80GHz x 4
Laser	360 Laser Distance Sensor LDS-01	WLR-716 x 2
Camera	Raspberry Pi Camera Module v2.1	LeTMC-520
IMU	Gyroscope 3 Axis, Accelerometer 3 Axis, Magnetometer 3 Axis	SC-AHRS-100D2

IV. Procedure of Simulation and Experiment

A. Construction of Maps

As mentioned earlier, this work aims to improve the performance of the local planners on ROS through narrow passages. To implement the path planning, two maps with a resolution of 0.01 pix/m were created with proper methods. The constructed maps are listed in Figure 3.



(a) (b)

Fig. 3. Constructed raster maps.

1) Constructing the map of the simulation environment with Gmapping: Gmapping can complete 2D raster map construction based on a 2D LiDAR with Rao-Blackwellized Particle Filters. It has less requirement of scanning frequency and computation but has a high mapping accuracy.

2) Constructing the map of the real environment with Cartographer: Cartographer is a mapping and positioning algorithm based on graph optimization. It is composed of global SLAM and local SLAM. The former is responsible for constructing submaps with data from the odometry, LiDAR, IMU and other sensors, while the latter is responsible for loop detection and back-end optimization to combine the submaps into a map.

B. Definition of Waypoints

To observe and record the motion data, navigation was performed between three waypoints A, B, and O, which were defined in the simulated map and the real map as Figure 4. In

every single test, the robot would move from the original point O to either A or B.

C. Implementation of Simulation

In the implementation part, Gazebo and Rviz were used for simulations to observe the motion of turtlebot3. For each group of tests of a single planner, the robot travelled 15 times along the paths of OA and OB under a certain set of parameters and configurations. The motion data was recorded by subscribing to related topics in rosbag and was visualized with PlogJuggler. After each group of tests, the parameters of each planner were modified. After that, previous processes were repeated to explore how they affected the performance through narrow passages.

Since the aisles were crowded, a very small excess angle change may also cause the robot to correct its pose later. On the other hand, because the trajectory generated by a planner was a circular arc, the longer the prediction was, the path was closer to a complete circle, which also made the robot turn unnecessarily. Therefore, the value of *sim_time* was increased appropriately to improve the path foresight. To prevent the robot from rotating near the target, the maximum angular velocity and target tolerance were adjusted to be larger. Moreover, the backward velocity (both in DWA and TEB) and backward weight (in TEB) were set to smaller to limit the backward behavior. To save computation, the robot radius was added in *min_obstacle_dists*. Since the robot is rectangular, the *footprint_model* was set to *line* instead of *point* or *polygon*. It should be noted that *weight_obstacle* cannot be set too high when applying this scheme, otherwise, the planner would determine a passable gap as not passable and terminate the planning in order not to violate the minimum distance constraint. An alternative is to set *footprint_model* to the realistic model of the robot and reduce *min_obstacle_dist* appropriately to improve crashworthiness and the possibility of passing through narrow passages. Moreover, to save computing resources, *enable_homotopy_class_planning* was set as *False* because the possible path was unique and parallel planning was not necessary. As the environment was relatively complex and a high-speed motion was not required, *feasibility_check_no_poses* should be reduced to one or even zero. It was helpful to avoid overly stringent trajectory detections that kept the robot re-planning stop-and-go paths constantly in narrow passages. In addition, based on the research of K. Zheng, the *costmap* decay curve with a relatively slow slope is preferably so that the robot tends to move in the middle of passages [22]. As well, most of the passages were narrow and the narrowest one only left about 5 cm of space on each side of the robot. Considering these two factors, the *inflation radius* and *cost scaling factor* in *global_costmap* and *local_costmap* were tuned accordingly to avoid the robot to be trapped in place.

At last, based on the control principle, the parameters of dynamics constraints of the robots and some physical configurations were set to the same. Table II lists parts of the values mentioned previously.

TABLE II. PART OF TUNED PARAMETERS OF LOCAL PLANNERS

Parameters	Description	Value
xy_goal_tolerance	the tolerance for the controller in x & y distance (in meters)	0.02→0.05
yaw_goal_tolerance	the tolerance for the controller in yaw & rotation (in radians)	0.10→0.18
clearing_rotation_allowed	whether or not attempting and rotation to clear out space	True→False
vtheta_samples	the number of samples to use when exploring the theta velocity space	20→40
max_vel_theta	the maximum rotational velocity (in radians/sec)	1.50→1.82
acc_lim_theta	the maximum rotational acceleration (in radians/sec^2)	2.0
controller_patiente	the amount of time to wait without receiving a valid control before giving up	5.0
inflation_radius	the radius to which the map inflates obstacle cost values (in meters)	0.55→0.15
csot_scaling_factor	the scaling factor to apply to cost values during inflation	10.0→3.0
occdist_scale	the weight for how much the robot should attempt to avoid obstacles	0.01→0.02
footprint_model	the robot footprint model type used for the optimization	line
min_turning_radius	the minimum turning radius of a carlike robot	0→0.1
min_obstacle_dist	the minimum desired separation from obstacles in meters	0.5→0.15
weight_kinematics_forward_drive	the weight of forcing robots to move ahead	1→100
feasibility_check_no_poses	The number of points used to determine whether the generated trajectories collide or not	5→1

D. Implementation of TEB on the real robot

The selected local planner was applied to the customized robot based on the results of previous simulations. The essential parameters were tuned further according to the robot model, dynamics constraints of hardware, the widths of passages and other features. The performance was tested in a real environment. Rviz was used to observe the motion and trajectories of the customized robot.

V. RESULT AND ANALYSIS

Figure 5, Figure 6, and Table III show all results of the simulations. The trajectory information was obtained by subscribing to the */odom* topic. The changes of x and y on coordinates contained in */odom/pose/position* were plotted into a continuous curve. As for the velocity information, they were contained in */odom/twist/angular* and */odom/twist/linear*.

Table III compares the time consumption of the robots with four different local path planners in simulations. It lists the time cost of the best performance and the worst performance, and the average value among 15 tests. As well, it indicates whether there was a collision during the planning. It can be found that although the typical DWA could complete navigation, collisions may occur and consume a long time. This is because the robot would correct its position by executing recovery behaviors and constantly reversing many times. The typical TEB has a better performance with no collisions and fewer recovery behaviors. The adjusted DWA and TEB both have better performance and there were

no more collisions. The time consumption to complete navigation was decreased by an average of 15% and 7%, respectively.

TABLE III. TIME CONSUMPTION OF TESTED LOCAL PLANNERS

Path		Simulation			
		DWA	***Adjusted DWA***	***TEB***	***Adjusted TEB***
O→ A	best[a]	69.067	48.452	50.359	49.667
	worst[b]	93.567	79.872	62.325	56.269
	Collision[c]	Y	N	N	N
	std-mean[d]	80.334	64.662	55.342	47.734
O→ B	best	51.251	42.259	43.335	39.235
	worst	61.151	55.264	49.235	46.930
	Collision	Y	N	N	N
	std-mean	55.338	49.903	45.246	43.235

[a.] Best performance with the shortest time

[b.] Worst performance with the longest time

[c.] Whether or not occurred collisions when turning a corner

[d.] Arithmetic mean of travel times

Fig. 4(a)(b)(c)(d) are the trajectories of OA and OB that were generated by DWA and adjusted DWA separately. It can be found that the smoothness of the path has been improved conspicuously. The times of correcting its direction by constantly reversing and rotating are significantly reduced as well. Fig. 4(e)(f)(g)(h) are the trends of angular velocity and linear velocity of forwarding direction. It can be found that the tuned DWA has less variation and smoother linear velocity overall. As for angular velocity, although it has the same high peak value, the steepness is reduced relatively. This indicates that the robot no longer changes its direction continuously and frequently when it passes a bend with a large angle but tends to perform multi-segment smooth paths.

Fig. 5(a)(b)(c)(d) are the trajectories of OA and OB that were generated by TEB and adjusted TEB separately. It can be found that TEB already has a good performance with very few reversing behaviors. However, its angular velocity and linear velocity fluctuate greatly because TEB constantly adjusts its direction at a very fast frequency during the mission, which can be found in Fig. 5(e)(f)(g)(h). This is because TEB calculates the linear velocity and angular velocity in a control cycle by the distance, time, and angle difference between every two states. After optimization, the smoothness of the paths is further improved, and the frequency of fluctuations in linear velocity and angular

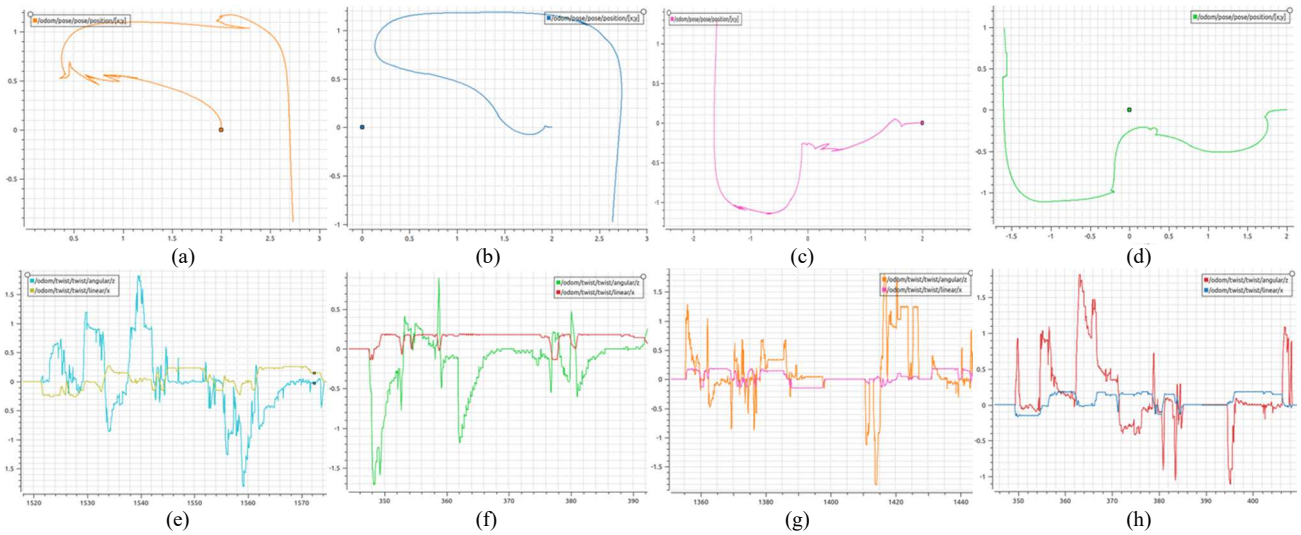


Fig. 4. Comparisons of typical DWA and Imp.DWA. (a)(c) Trajectories of OA and OB generated by DWA. (b)(d) Trajectories of OA and OB generated by Imp.DWA. (e)(g) Angular and linear velocity of DWA for OA and OB. (f)(h) Angular and linear velocity of Imp.DWA for OA and OB.

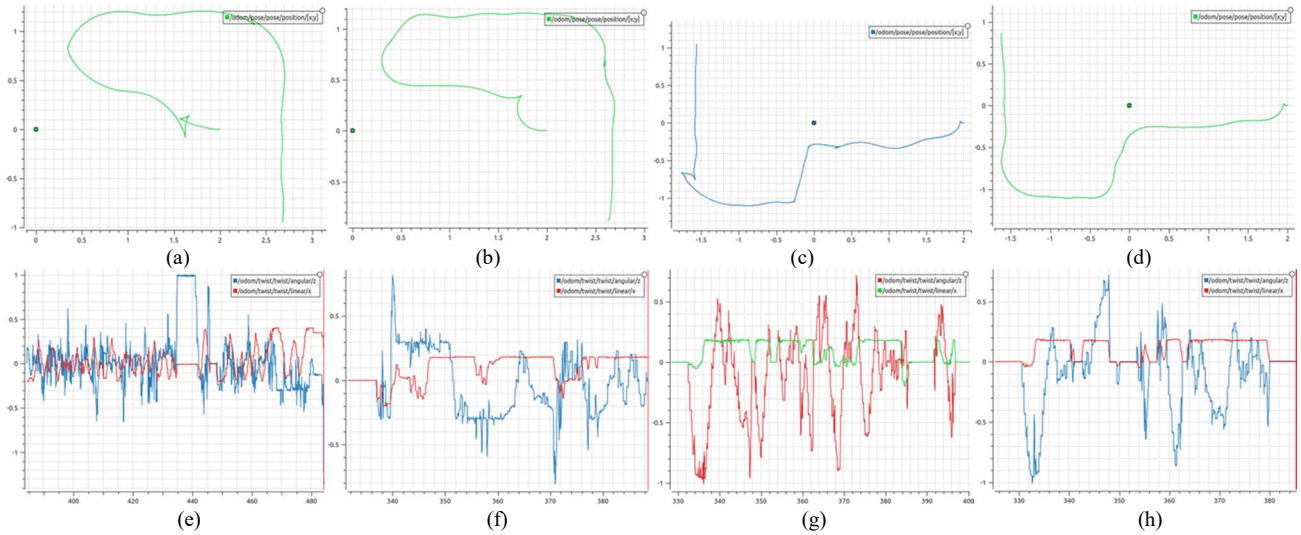


Fig. 5. Comparisons of typical TEB and Imp.TEB. (a)(c) Trajetories of OA and OB generated by TEB. (b)(d) Trajectories of OA and OB generated by Imp.TEB. (e)(g) Angular and linear velocity of TEB for OA and OB. (f)(h) Angular and linear velocity of Imp.TEB for OA and OB.

velocity is notably reduced, which helps the robot keep good stability when navigating in a more complex environment.

Finally, considering the better performance and the adaptability to a robot with an Ackerman model or a carlike model, TEB was applied to the customized robot. In addition, based on previous experience and strategies, the original TEB was adjusted and tested in the real environment as Fig. 1(b) and Fig. 3(b). The results showed the customized robot with adjusted TEB was able to complete the navigation with a stable velocity and a smooth trajectory.

VI. Conclusion and Future Work

In conclusion, after adjustments, both DWA and TEB can accomplish navigation through narrow passages with a gentler velocity and a smoother trajectory. In addition, the time consumption was decreased by 15% and 7%, and the number of collisions was reduced to zero. Moreover, the strategies and experience for tuning essential parameters of local planners were concluded for further research and solving similar problems. Furthermore, due to the features of being suitable for a robot with a carlike model, stronger foresight, shorter time cost, and better performance on narrow passages, the adjusted TEB local planner was implemented on the customized robot. Finally, the navigation stack with the A* algorithm as the global planner and adjusted TEB as the local planner was implemented on a customized robot. It successfully navigated in the real environment with multiple narrow passages. Future work will focus on how to deeply customize the TEB algorithm to reduce the possible backward behavior further to make the path smoother. In addition, this research is a pre-study of the robots on high-speed trains. Considering the presence of many potential irregular movements of passengers in the carriage, it is necessary to improve the timeliness and performance of TEB to avoid dynamic objects.

Acknowledgment

I would like to appreciate the suggestions for academic writing from Tongpo Zhang. As well, I am grateful for the support from Yue Zhang, Taoyu Wu, and Haocheng Zhao.

References

[1] J. Latombe, *Robot motion planning*. Boston: Kluwer, 2010.

[2] H. Durrant-Whyte, "Where am I? A tutorial on mobile vehicle localization", *Industrial Robot: An International Journal*, vol. 21, no. 2, pp. 11-16, 1994. Available: 10.1108/eum0000000004145 [Accessed 22 August 2022].

[3] H. Kang, B. Lee and W. Jang, "Path Planning Method Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm", *Journal of Korean Institute of Intelligent Systems*, vol. 18, no. 2, pp. 212-215, 2008. Available: 10.5391/jkiis.2008.18.2.212 [Accessed 23 August 2022].

[4] M. Luo, X. Hou and J. Yang, "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm", *IEEE Access*, vol. 8, pp. 147827-147838, 2020. Available: 10.1109/access.2020.3015976 [Accessed 23 August 2022].

[5] M. Garcia, O. Montiel, O. Castillo, R. Sepúlveda and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation", *Applied Soft Computing*, vol. 9, no. 3, pp. 1102-1110, 2009. Available: 10.1016/j.asoc.2009.02.014 [Accessed 23 August 2022].

[6] J. -T. Chen, A. Yousefi, S. Krishna, B. Sliney and P. Smith, "Weather avoidance optimal routing for extended terminal airspace in support of Dynamic Airspace Configuration," 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), 2012, pp. 3A1-1-3A1-16, doi: Available: 10.1109/dasc.2012.6382301 [Accessed 23 August 2022].

[7] E. Frazzoli, M. Dahleh and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles", *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116-129, 2002. Available: 10.2514/2.4856 [Accessed 23 August 2022].

[8] Y. Singh, S. Sharma, R. Sutton, D. Hatton and A. Khan, "A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents", *Ocean Engineering*, vol. 169, pp. 187-201, 2018. Available: 10.1016/j.oceaneng.2018.09.016 [Accessed 23 August 2022].

[9] N. Melchior and R. Simmons, "Particle RRT for Path Planning with Uncertainty", *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007. Available: 10.1109/robot.2007.363555 [Accessed 23 August 2022].

[10] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance", *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23-33, 1997. Available: 10.1109/100.580977 [Accessed 23 August 2022].

[11] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control", *[1993] Proceedings IEEE International Conference on Robotics and Automation*. Available: 10.1109/robot.1993.291936 [Accessed 23 August 2022].

[12] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1-6 [Accessed 23 August 2022].

[13] D. Chikurtev, "Mobile Robot Simulation and Navigation in ROS and Gazebo", *2020 International Conference Automatics and Informatics (ICAI)*, 2020. Available: 10.1109/icai50593.2020.9311330 [Accessed 28 August 2022].

[14] B. Cybulski, A. Wegierska and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot", *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019. Available: 10.1109/romoco.2019.8787346 [Accessed 28 August 2022].

[15] I. Naotunna and T. Wongratanaphisan, "Comparison of ROS Local Planners with Differential Drive Heavy Robotic System", *2020 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2020. Available: 10.1109/icamechs49982.2020.9310123 [Accessed 28 August 2022].

[16] C. Looi and D. Ng, "A Study on the Effect of Parameters for ROS Motion Planer and Navigation System for Indoor Robot", *International Journal of Electrical and Computer Engineering Research*, vol. 1, no. 1, pp. 29-36, 2021. Available: 10.53375/ijecer.2021.21 [Accessed 28 August 2022].

[17] G. Lucas, "A Tutorial and Elementary Trajectory Model for The Differential Steering System", Rossum.sourceforge.net, 2001. [Online]. Available: http://rossum.sourceforge.net/papers/DiffSteer/. [Accessed 27 Aug 2022].

[18] B. P. Gerkey, and K. Konolige. "Planning and control in unstructured terrain." *ICRA workshop on path planning on costmaps*. 2008. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.330.2120&rep=rep1&type=pdf. [Accessed 27 Aug 2022].

[19] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," ROBOTIK 2012; 7th German Conference on Robotics, 2012, pp. 1-6. [Accessed 27 Aug 2022].

[20] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann and T. Bertram, "Efficient trajectory optimization using a sparse model," 2013 European Conference on Mobile Robots, 2013, pp. 138-143, doi: 10.1109/ECMR.2013.6698833. [Accessed 27 Aug 2022].

[21] "ROBOTIS e-Manual", *ROBOTIS e-Manual*, 2022. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/. [Accessed 31 Aug 2022].

[22] K. Zheng, "ROS Navigation Tuning Guide", *Studies in Computational Intelligence*, pp. 197-226, 2021. Available: 10.1007/978-3-030-75472-3_6 [Accessed 31 August 2022]