Research on Mobile Robot Path Planning Based on Improved A* and DWA Algorithms

Wei Qian[1], Jiansheng Peng[1,2](✉), and Hongyu Zhang[1]

[1] College of Automation, Guangxi University of Science and Technology, Liuzhou 545000, China
sheng120410@163.com

[2] Department of Artificial Intelligence and Manufacturing, Hechi University, Hechi 547000, China

Abstract. This paper explores the navigation of robots, which involves two key aspects: global path planning and local path planning. The current approach utilizes the A* algorithm for global planning and the DWA algorithm for local planning. However, the traditional A* algorithm often fails to consider obstacles, leading to impractical paths that robots struggle to navigate. As a result, success rates are narrow. To address these issues, we propose an enhanced A* algorithm that optimizes the search approach, heuristic function as well as path smoothing. This modification ensures that the generated paths align better with the robot's motion while still prioritizing the shortest distance. Additionally, we refine the evaluation function of the DWA algorithm to account for the robot's angular velocity at different linear velocities. This adjustment enables smoother steering through curves while maintaining a consistent linear velocity at the same time. The improved algorithm greatly improves the speed of the robot when cornering and has a more reasonable completion path, while the success rate of cornering is greatly improved. In our physical map, the robot with the improved algorithm consumes 33.95% less time than the robot with the traditional algorithm, and the robot's path is much better.

Keywords: DWA algorithm · A* algorithm · Mobile robot · ROS

1 Introduction

Robot navigation involves two main components: global path planning [1] and local path planning [2]. Global path planning determines the optimal path from the robot's starting point to its target point. It typically involves rasterizing and searching the map by using algorithms like A* [3] or Dijkstra [4] to find the best path. In this paper, we focus on improving the A* algorithm for global path planning and local path planning. On the other hand, it allows the robot to adjust its path based on its current state and surroundings while following the global path. This enhances the overall system's robustness. The robot collects information from its environment and itself to timely adjust its path and speed, ensuring that it stays on track and avoids disturbances. Common algorithms used for local path planning include DWA [5] and TEB [6], and in this paper, we aim to improve the DWA algorithm.

Traditional robot navigation algorithms have limitations when it comes to realistic and specific tasks. The four-wheel differential speed robot used in our experiment utilizes A* and DWA algorithms. However, the traditional A* algorithm doesn't fully align with the robot's motion principles. The DWA algorithm only calculates optimal routes for the next moment, resulting in sub-optimal performance during cornering. To address these issues, we propose modifications to the A* algorithm's neighborhood search strategy and heuristic function. These enhancements generate better curved paths for global planning. Additionally, we modify the evaluation function of the DWA algorithm to improve the robot's performance during bends. In summary, this paper focuses on improving global and local path planning algorithms for robot navigation. By adapting the A* algorithm and modifying the DWA algorithm, we aim to overcome the limitations of traditional approaches and achieve better performance in practical tasks.

2 Introduction to Traditional Algorithms

2.1 Traditional A* Algorithm

The traditional A* algorithm begins by searching for points in the surrounding area from the current point, as illustrated in Fig. 1. It completes this calculation through a systematic search process. Subsequently, the cost function evaluates the points reached by different search methods and selects the point with the lowest cost as the next target point. This iterative process continues until the final target point is reached [7].

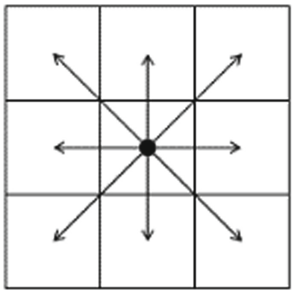

Fig. 1. Traditional A* algorithm search method

The A* algorithm uses heuristic search to get the best path to the destination, and it finds the optimal path by means of an estimation function, which is generally written as follows:

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

where g(n) represents the generation value from the starting node to the current node, using the Euclidean distance representation, and h(n) represents the generation value from the current node to the target node. In simple terms g(n) enables the robot to travel to the target point with a shorter path, and h(n) enables the robot to plan the path faster while limiting the path deviation.

2.2 Traditional DWA Algorithm

The DWA algorithm samples multiple sets of velocity information in real-time local path planning (v, ω), and based on this sampling the trajectory of the next moment of action is simulated. The optimal path is selected among the predicted trajectories for navigation. The robot is first modeled in motion, and considering that the robot used cannot move omni directionally, the kinematic model yields a trajectory at moment t as

$$x = x + v * \Delta t * \cos(\theta_t) \tag{2}$$

$$y = y + v * \Delta t * \sin(\theta_t) \tag{3}$$

$$\theta_t = \theta_t + \omega * \Delta t \tag{4}$$

Considering that this trajectory model will be limited by the robot's own structure, three constraints are added to its velocity sampling. One of the most-valued constraints is:

$$V = \{v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\} \tag{5}$$

Acceleration and deceleration constraints are:

$$V_d = \{(v, \omega) \in [v_c - v_b * \Delta t, v_c + v_a * \Delta t], \omega \in [\omega_c - \omega_b * \Delta t, \omega_c + \omega_a * \Delta t]\} \tag{6}$$

The safety distance constraint is:

$$V_\omega = \{(v, \omega) | (v \leq \sqrt{2dist(v, \omega) \cdot v_b}) \wedge (\omega \leq \sqrt{2dist(v, \omega) \cdot \omega_b})\} \tag{7}$$

After taking to multiple sets of speed information, the optimal path selection is performed by using the evaluation function as follows:

$$G(v, \omega) = f(\alpha * heading(v, \omega) + \beta * dist(v, \omega) + \gamma * vel(v, \omega)) \tag{8}$$

The heading term is the azimuth evaluation function. The dist term is the distance evaluation function and the velocity term is the speed evaluation function [8].

3 A* Algorithm Improvement and Simulation

In this project, the A* algorithm is employed for global path planning. However, the traditional A* algorithm solely focuses on finding the shortest path, disregarding the smoothness of the route. Although interpolation is commonly used to enhance the smoothness of the robot's motion trajectory, it often struggles to handle areas with significant curves effectively [9]. Furthermore, the traditional A* algorithm limits its search and calculation to the immediate vicinity, resulting in unnecessary distance wastage on obstacle-free, straight roads.

To overcome these limitations, this chapter introduces an improvement to the A* algorithm tailored specifically for this project. The enhanced algorithm aims to optimize the robot's motion trajectory by considering both the shortest path and the smoothness of the route. Additionally, it expands the search range beyond the immediate surroundings to minimize distance wastage on flat, obstacle-free roads. The effectiveness of the proposed improvement is validated through simulations.

3.1 A* Algorithm Improvement

To account for the limitations of the non-omnidirectional motion capability of the robot and the absence of obstacles in the experimental site, this paper presents an innovative algorithm for searching neighborhood points, as depicted in Fig. 2. The primary objective is to enhance speed and reduce computational effort. By specifically targeting the search in the surrounding neighborhood, the robot can efficiently obtain a shorter final path, which proves highly effective in obstacle-free, straight-line environments.

This algorithmic improvement optimizes the robot's path by reducing the distance traveled for corner points when no obstacles are present. Subsequent physical experiments revealed that this modification also provides the robot with an increased safety margin during curves. This finding underscores the significance of the proposed algorithm, as it allows the robot to navigate more effectively, ensuring a greater level of safety while maximizing efficiency.

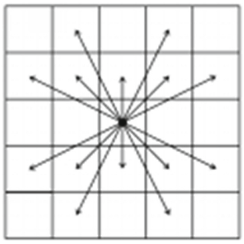

Fig. 2. Improved A* algorithm search method

Considering that the AKM chassis robot we utilize lacks direct lateral displacement capabilities, it bypasses the search for proximity points in the left and right directions. The conventional A* algorithm produces a path planning form as shown in Fig. 3. However, in practical scenarios, this trajectory fails to provide the robot with sufficient safety distance during sharp curves. Consequently, the robot is prone to getting stuck by obstacles, leading to navigation failures.

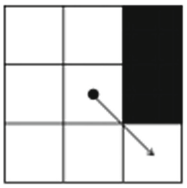

Fig. 3. Limitations of the traditional A* algorithm

To address this issue, we propose an alternative search method depicted in Fig. 4 when an obstacle is detected near the search point. In cases depicted in Fig. 5, only the search method shown in Fig. 4 is retained. This modified search method enables the robot to plan shorter paths in obstacle-free straight stretches and maintain an adequate safety distance when approaching curves. Compared to the traditional A* algorithm, this approach only requires searching six neighbors, significantly reducing computational effort.

By implementing this search method, our robot can navigate more efficiently. It obtains shorter paths in straight sections without obstacles while ensuring increased safety margins during curved sections. This improvement proves crucial in avoiding obstacles and achieving successful navigation.

In addition, considering that in the actual navigation process, the robot's path will be offset to a certain extent, and the h(n) term in the heuristic function is difficult to calculate accurately. Either using the Euclidean distance or the Manhattan distance will have unavoidable deviations, a compromise of the h(n) term is considered as follows:

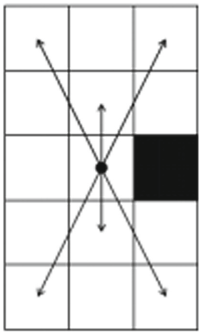

Fig. 4. Right neighborhood with obstacles

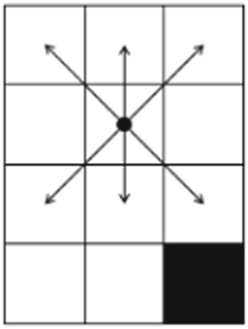

Fig. 5. Obstacles in the oblique front neighborhood

$$h(n) = 0.5 * k * \left(\sqrt{(N_x - P_x)^2} + \sqrt{(N_y - P_y)^2}\right) + 0.5 * (1 - k) * \left(\sqrt{(N_x - P_x)^2 + (N_y - P_y)^2}\right) \quad (0 \le k \le 1) \tag{9}$$

where N represents the target point, P represents the current point, and k is the weight that can be chosen according to the different values of the robot's environment.

In practice, h(n) and g(N) are not well balanced, and the robot tends to have some extreme cases, such as when h(n) is too large and g(n) accounts for a smaller percentage, the robot will tend to choose the shortest path but the path direction will deviate more; in the opposite case the robot will consume more resources to plan more paths, wasting computational resources and computational time. We want these two items to be user-determined in the valuation function, and therefore add weights to these two items as follows:

$$f(n) = \frac{g(n) + e^{-\omega} h(n)}{1 + e^{-\omega}} \tag{10}$$

By choosing appropriate values for ω based on practical applications, the proportion of g(n) and h(n) in the pairwise valuation function is adjusted to achieve a balance between improving the computational efficiency of the algorithm and obtaining the shortest path.

In order to obtain a smoother path, consider three spline interpolations of the final generated global path for smoothing, assuming that a total of n + 1 folds and a total of h steps are known, we can obtain [10]:

$$a_i = y_i \tag{11}$$

$$b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i m_i}{2} - \frac{h_i (m_{i+1} - m_i)}{6} \tag{12}$$

$$c_i = \frac{m_i}{2} \tag{13}$$

$$d_i = \frac{m_{i+1} - m_i}{6h_i} \tag{14}$$

The final smoothed curve is obtained by bringing in the equation:

$$g_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{15}$$

The flow chart of the improved algorithm is as followed in Fig. 6.

3.2 Simulation Results

To test the effectiveness of the algorithm, a simple map with large curves was created in Python. The focus was on comparing the improved algorithm's performance at the curves. The map consisted of squares, with black squares as obstacles, dark blue squares as starting points, and purple squares as ending points.

In Fig. 7, the yellow line represents the path planned by the traditional A* algorithm, while the blue line represents the path planned by the improved algorithm. The comparison mainly focused on the trajectory and distance traveled since the number of searched neighborhood points was similar. The improved algorithm showed better alignment with the robot's motion and provided a greater safety distance at the curves.

After smoothing, the improved algorithm had a slightly shorter overall distance traveled while maintaining better safety at the curves (red lines in Fig. 8). In open fields, the improved algorithm had significant advantages in terms of distance traveled. It prioritized allocating more paths for better safety at the curves, which is crucial in such environments (Table 1).

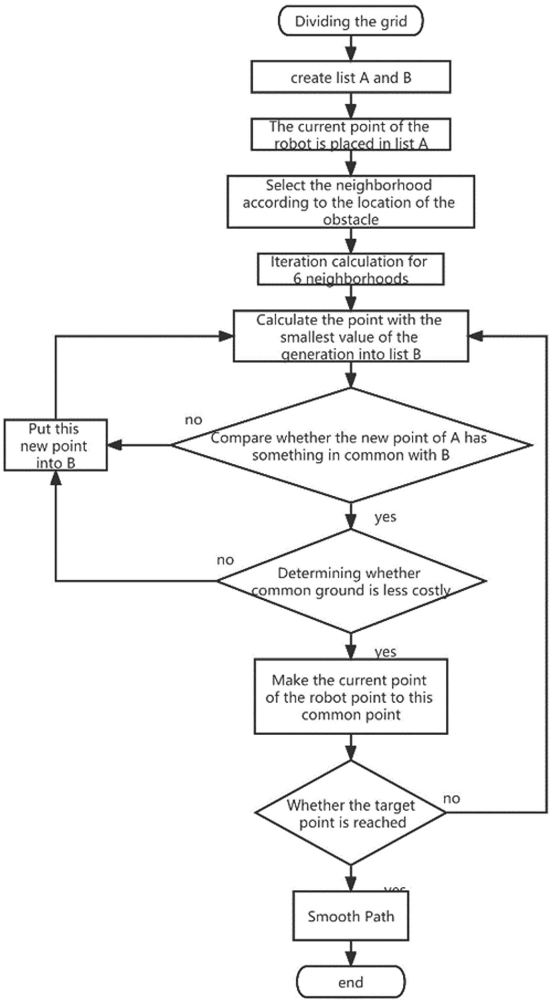
Dividing the grid
create list A and B
The current point of the robot is placed in list A
Select the neighborhood according to the location of the obstacle
Iteration calculation for 6 neighborhoods
Calculate the point with the smallest value of the generation into list B
Compare whether the new point of A has something in common with B
no
Put this new point into B
yes
Determining whether common ground is less costly
no
yes
Make the current point of the robot point to this common point
Whether the target point is reached
no
yes
Smooth Path
end

Fig. 6. Flow chart of the improved A* algorithm

Table 1. Comparison of the two algorithms

	Route	Curve safety
Traditional A* algorithm	32.48	Poor
Improved A* algorithm	32.26	Better

4 Improvement of DWA Algorithm

The traditional DWA algorithm has limitations that affect robot performance in curves and practical applications. It only predicts the next moment and calculates the next best path, which can lead to poor performance [11]. To address this issue, we made improvements for bending situations in this project. We adjusted the robot's velocity function to overcome the limitations of the DWA algorithm. By removing prediction planning, we ensured that the robot's actual direction aligns better with the planned path in curves. To enhance the evaluation function, we introduced constraints on the robot's angular velocity relative to its linear velocity. This ensures the robot has enough angular velocity for steering during deflection while preventing excessive steering on flat routes with high linear velocity. These modifications aim to improve the robot's performance in curves and enhance its steering capabilities in various scenarios.

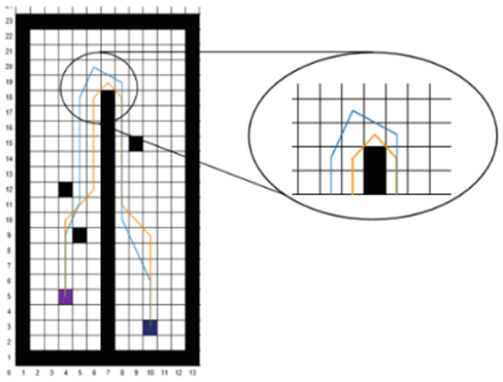

Fig. 7. Simulation results for improving A*

$$r(v) = \kappa * \left(\frac{e^{|v|} - e^{-|v|}}{e^{|v|} + e^{-|v|}}\right) + b \tag{16}$$

This equation evolves from the hyperbolic tangent function (tanh). This design holds a dual purpose: firstly, maximizing the robot's linear velocity to ensure sufficient speed, and secondly, preventing excessive velocities that could lead to instability during turns. The choice of this function brings forth unmistakable benefits. It allows for rapid growth in linear velocity when it is small, while still keeping it bounded within a desirable range. By adopting a linear function form, we gain greater control over its value range, providing us with more versatile means of regulation. And this function shows in Fig. 9.

By this means, the angular velocity of the robot during steering is proportional to the linear velocity, while the angular velocity of the robot is not too large due to the properties of the function (Fig. 9). Also add the angular velocity evaluation term for the evaluation function $anvel(v, \omega)$:

$$anvel(v, \omega) = \frac{r(v) * \omega}{v} \tag{17}$$

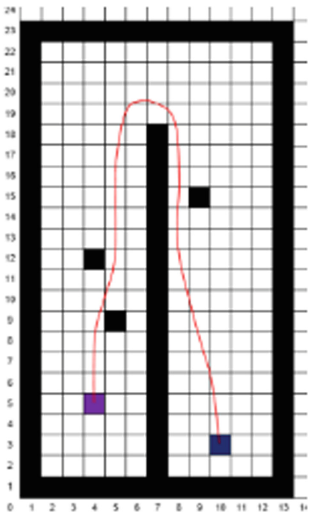

Fig. 8. Trajectory after smoothing process

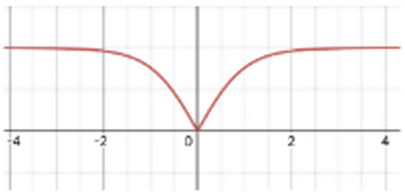

Fig. 9. tanh(|x|)

By employing a linear function for the velocity-related component, we attain the ability to effortlessly manipulate its value range. Furthermore, our objective is to endow the robot with enhanced angular agility and reduced linear velocity. This entails not only capping the upper limit of linear speed to prevent unwieldy accelerations but also imposing penalties when linear velocity escalates, while offering generous rewards for heightened angular velocity. Thus, the resulting design embodies these thoughtful considerations.

This evaluation term will comprehensively evaluate the relationship between linear velocity and angular velocity, and the score of this term will be lower when the angular velocity is too small and the linear velocity is too large, which happens to be the case when the robot is out of control in a curve, and the improved evaluation function is

$$\begin{aligned} G(v, \omega) = &f(\alpha * heading(v, \omega) \\ &+ \beta * dist(v, \omega) + \gamma * vel(v, \omega) + \lambda * anvel(v, \omega)) \end{aligned} \tag{18}$$

5 Experimental Results

The experimental robot is AKM robot, using two motors into the rear drive, the main control is Jetson nano, using ubuntu18.04 operating system, real-time map building and navigation using ROS (melodic) system to complete, radar using the Ledon D300, IMU using MPU6050. This experiment using LIDAR and using gmapping algorithm to complete the map construction. After getting the complete static map, the target point selection for navigation is performed in RViz. This experiment uses multi-point navigation to complete the pile wrapping test. Since the physical map is not empty and generates a lot of steering, a huge steering is generated at one side of the map end, which is a great challenge for the robot's navigation.

The experimental map and equipment are shown in Fig. 10. In the navigation process rqt_plot will record the indicators of the trolley in real time, we also wrote the same node file to listen to the robot map coordinates in real time and print on RViz to get the trajectory of the robot walking, so as to compare the trajectory performance and speed of the same robot under different algorithms, in the case of the same map and close to the starting and ending points. The maximum linear velocity of the robot was set to 2.5 m/s and the maximum angular velocity was set to 3 rads/s.

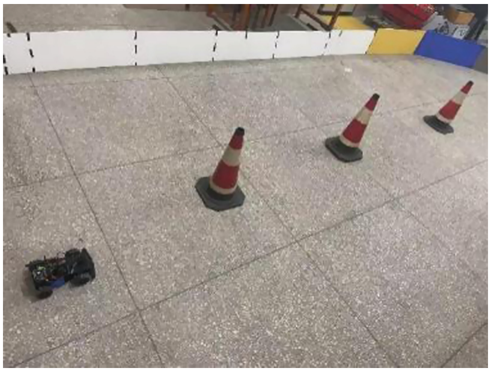

Fig. 10. Experimental environment and equipment

Comparing the trajectories in Figs. 11 and 12 reveals that the optimized algorithm outperformed the traditional algorithm, especially in the final sharp curve. Figure 14 illustrates that the optimized algorithm maintained a high angular speed throughout the curve, enabling the robot to take shorter paths and achieve higher linear speeds during cornering. In contrast, the common algorithm (Fig. 13) had limited adjustments in angular velocity, often resulting in delayed increases until the bend was nearly impossible to navigate. This led to wasted distance, and in practical use, the traditional algorithm had a low success rate in 180° bends.

In most situations, traditional algorithms struggle to navigate through this track successfully in the given experimental platform and environment. They often encounter difficulties, such as pausing or even colliding with walls during the challenging 180-degree turns. This observation is supported by monitoring the algorithm's output for linear and angular velocities. It is commonly observed that these velocity values fluctuate around the zero point, causing the robot to move in a circular path and pause multiple times to adjust its direction before completing the turn. This behavior is visually reflected in the graph, where the angular velocity tends to follow the changes in linear velocity. The main reason behind this phenomenon is that the robot cannot fully execute the algorithm's commands, while the algorithm attempts to rectify the robot's motion. Consequently, the robot experiences frequent pauses and takes wider turns, resulting in significant time wastage. However, when the robot operates with the improved algorithm, these pauses are eliminated. The robot becomes more adept at following the algorithm's motion commands, particularly noticeable at the sharpest turn, where the commands exhibit minimal fluctuations. As a result, the robot smoothly completes the 180-degree turn with little to no deceleration in real-world scenarios. The difference between the two algorithms at the curve section is illustrated in Fig. 15 (Table 2).

Table 2. Comparison of the effects of the two algorithms

	Time consumption(s)	Success rate
Before improvement	40.62	Frequent failures
After improvement	26.83	Hardly ever fails

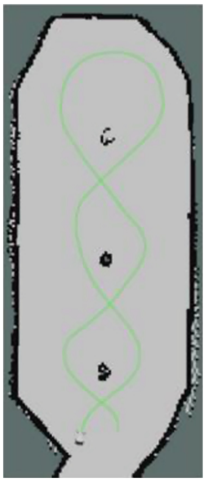

Fig. 11. Physical trajectory of the traditional algorithm

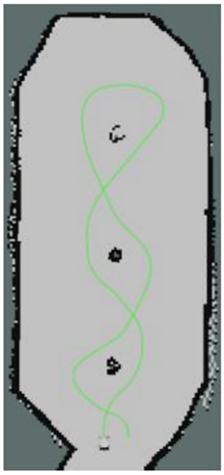

Fig. 12. Physical trajectory of the improved algorithm

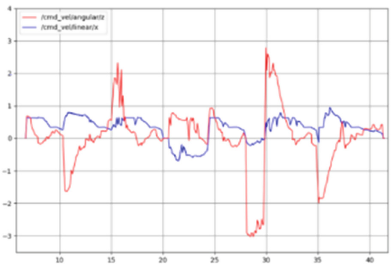

Fig. 13. Speed graph of traditional algorithm. Traditional algorithm speed graph: From 25 to 30 s, during the 180-degree turn, traditional algorithms decelerate, causing abrupt angular and linear velocity changes, and a lower overall cornering speed.

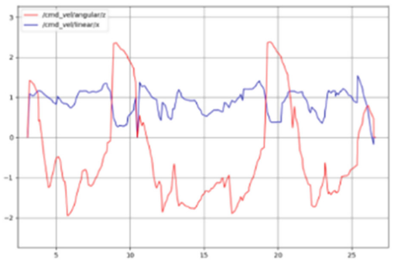

Fig. 14. Speed graph of improved algorithm.Between 25 and 30 s, the platform turns 180 degrees. The enhanced algorithm plans for a larger initial steering angle, leading to smoother angular and linear velocities, resulting in a higher overall cornering speed.

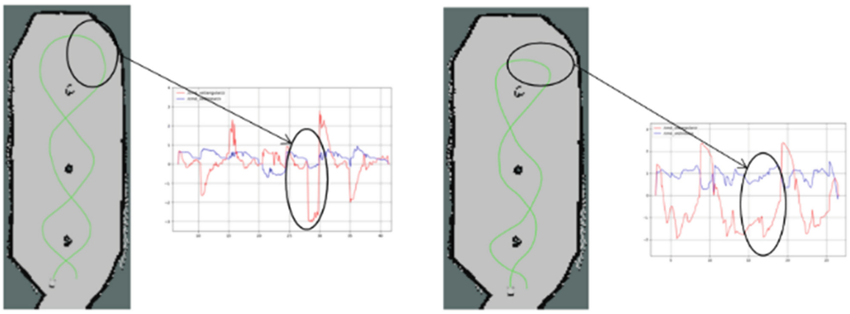

Fig. 15. Comparison of the two algorithms at the curve section.The improved algorithm has a clear advantage in trajectory and speed around curves.

6 Conclusion

In this paper, the traditional A* algorithm and DWA algorithm are optimized and improved for robot cornering. Firstly, an innovative 6-neighborhood A* algorithm is proposed, its heuristic function is optimized and its final trajectory is interpolated and smoothed, and finally the simulation verifies its effect so that its final cornering effect is better than that of the traditional A* algorithm. The evaluation function of the DWA algorithm is modified to allow the robot to obtain a larger angular speed and limit the linear speed in large bends, so that the robot can use a relatively faster speed to complete the bends safely. In the final physical experimental session, the robot's speed and operational stability were greatly improved. Such improvements can make the autonomous navigation mobile robot perform better in urban, logistics, industrial, and other map environments with multiple curves.

Acknowledgement. The authors are highly thankful to the National Natural Science Foundation of China (NO. 62063006), the Natural Science Foundation of Guangxi Province (NO. 2023GXNSFAA026025), to the Innovation Fund of Chinese Universities Industry-University-Research (ID: 2021RYC06005), to the Research Project for Young and Middle-aged Teachers in Guangxi Universities (ID: 2020KY15013), and to the Special research project of Hechi University (ID: 2021GCC028). This research was financially supported by the project of outstanding thousand young teachers' training in higher education institutions of Guangxi, Guangxi Colleges and Universities Key Laboratory of AI and Information Processing (Hechi University), Education Department of Guangxi Zhuang Autonomous Region.

References

1. Abdallaoui, S., Aglzim, E.H., Chaibet, A., et al.: Thorough review analysis of safe control of autonomous vehicles: path planning and navigation techniques. Energies **15**(4), 1358–1377 (2022)
2. Qin, H., Shao, S., Wang, T., et al.: Review of autonomous path planning algorithms for mobile robots. Drones **7**(3), 211–248 (2023)
3. Wu, B., Chi, X., Zhao, C., et al.: Dynamic path planning for forklift AGV based on smoothing A* and improved DWA hybrid algorithm. Sensors **22**(18), 7079–7098 (2022)

4. Luo, M., Hou, X., Yang, J.: Surface optimal path planning using an extended Dijkstra algorithm. IEEE Access **8**(1), 147827–147838 (2020)
5. Mohammadpour, M., Zeghmi, L., Kelouwani, S., et al.: An investigation into the energy-efficient motion of autonomous wheeled mobile robots. Energies **14**(12), 3517–3536 (2021)
6. Andreasson, H., Larsson, J., Lowry, S.: A local planner for accurate positioning for a multiple steer-and-drive unit vehicle using non-linear optimization. Sensors **22**(7), 2588–2606 (2022)
7. Tyagi, N., Singh, J., Singh, S.: A review of routing algorithms for intelligent route planning and path optimization in road navigation. Rec. Trends Prod. Des. Intell. Manuf. Syst. **16**(14), 851–860 (2022)
8. Öztürk, Ü., Akdağ, M., Ayabakan, T.: A review of path planning algorithms in maritime autonomous surface ships: navigation safety perspective. Ocean Eng. **251**(1), 111010–111031 (2022)
9. Sánchez-Ibáñez, J.R., Pérez-del-Pulgar, C.J., García-Cerezo, A.: Path planning for autonomous mobile robots: a review. Sensors **21**(23), 7898–7914 (2021)
10. Gul, F., Mir, I., Abualigah L., et al.: A consolidated review of path planning and optimization techniques: technical perspectives and future directions. Electronics **10**(18), 2250–2268 (2021)
11. Xiao, X., Liu, B., Warnell, G., et al.: Appld: Adaptive planner parameter learning from demonstration. IEEE Robot. Autom. Lett. **5**(3), 4541–4547 (2020)