Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2025.060204
ISSN 2790-0932 Vol. 6, Issue 2: 36-45

Dynamic Obstacle Avoidance Path Planning Algorithm Based on the Fusion of Improved A and DWA*

Meng Qin[1,a] , Qingyuan Xiao[1,b], Huiheng Suo[1,c], Guangjun Lai[1,d], Zuteng Chen[1,e], Jian Wu[1,f,*], Chenkai Zhang[2,g], Xie Ma [2,h], Yingping Bai[3,i], Weihong Zhong[3,j]

[1]Nanchang Hangkong University, Nanchang, China
[2]Ningbo University of Finance & Economics, Ningbo, China
[3]NingboTech University, Ningbo, China
[a]alphenqin@163.com,[b]xiaoqy0918@163.com, [c]suohuiheng@163.com, [d]l2647080898@163.com, [e]2293747855@qq.com, [f]flywujian@qq.com, [g]674060542@qq.com, [h]maxie88@163.com, [i]3459951916@qq.com, [j]zwh@nbt.edu.cn
[*]corresponding author

Keywords: Path planning; Mobile robot; A* algorithm; Dynamic Window Approach (DWA); Dynamic obstacle avoidance

Abstract: In dynamic environments, the traditional A* algorithm is apt to suffer from issues such as low search efficiency, unnecessary turning points, close proximity to obstacles, and no obstacle avoidance capability in global path planning. To address these issues, this paper presents a new path planning method that integrates an optimized A* algorithm and the Dynamic Window Approach (DWA) to enhance the mobile robot's navigation efficiency and obstacle avoidance ability in dynamic environments. During the global path planning process, multi-dimensional optimizations are incorporated into the A* algorithm. They include optimizing the cost function, taking multi-round polyline optimization approaches, and avoiding obstacle vertices, which successfully reduce unnecessary turning points in the path, reduce the frequency of turns, smooth the path curvature, and greatly enhance the search efficiency of the algorithm. In the local obstacle avoidance stage, the distance factor of dynamic obstacles is incorporated into the DWA evaluation function system to enhance the robot's ability to avoid dynamic obstacles. To verify the effectiveness of this method, a series of simulation experiments was conducted in a grid map environment built on the MATLAB platform. Comparative experiments of the classic A* algorithm, improved A* algorithm, and the proposed fusion algorithm were conducted under various dynamic obstacle collision threat scenarios. The experimental results show that this paper's fusion algorithm has higher path planning quality, obstacle avoidance success rate, and real-time performance than traditional algorithms, realizing stable and reliable navigation of paths.

1. Introduction

With the generalization and speeding-up of the development of artificial intelligence and

automation technology, applications of mobile robots[1] have also extended widely into industrial production, intelligent warehousing, and service robots. Path planning[2] is the core module of autonomous navigation of a mobile robot, and its quality decides the working efficiency and environmental adaptability of the robot. Since in complicated and ever-changing situations how effective coordination of global path planning and local obstacle avoidance may be achieved is a trending current matter under study, the traditional A* algorithm[3] has been widely used to perform path planning in static situations due to its heuristic search strategy and global planning capability. However, A* is not responsive to dynamic obstacles and has high computational complexity and is difficult to meet the requirement of real-time obstacle avoidance. The Dynamic Window Approach (DWA)[4] takes into account the kinematic constraints of robots and their outstanding local obstacle avoidance for static obstacles and is thus widely employed in real-time navigation tasks. However, DWA lacks global view of the entire path and is prone to being caught in local optima. Therefore, this paper proposes hybridizing the A* algorithm and DWA[5] to leverage A* global planning qualities and DWA local obstacle avoidance capabilities. This hybrid can fundamentally enhance the navigation ability of mobile robots in complex environments by enabling more robust path planning and obstacle avoidance.

In recent years, scholars have proposed many improved algorithms to address the problems encountered in mobile robot path planning. Zhang D[6] improved the cost function and integrated the artificial potential field method, resulting in a significant reduction in the number of path turning points. Sang W[7] introduced the obstacle rate to optimize the heuristic function of the A* algorithm and dynamically adjusted the weights of the DWA evaluation function. Zhang C[8] logarithmically weighted the heuristic function of the A* algorithm and designed a DWA fuzzy controller, greatly reducing the path deviation. Song B[9] Proposed to integrate the improved Ant Colony Optimization (ACO) algorithm with the DWA algorithm to enhance the performance of global and local path planning in dynamic environments.

Although the existing research has made some achievements in improving the search efficiency of the A* algorithm, reducing the number of path turning points, improving the accuracy of global path planning, and integrating local obstacle avoidance algorithms, in dynamic complicated environments, issues such as slow response of the path, instability of obstacle avoidance, and sensitivity to local optimal solutions still dominate. To enhance the path planning capability of mobile robots in dynamic environments further, this paper presents an enhanced A* algorithm integrated with the Dynamic Window Approach (DWA) to create a path planning scheme. In the global planning stage, this approach increases search effectiveness, path quality, and safety of global paths. At the local planning stage, it takes the safety distance in static environments into consideration and uses particular avoidance strategies for dynamic ones. With such integration, mobile robots can achieve a global path with fewer turning points and sufficient safety, apart from fast and accurate avoidance of static and dynamic obstacles during movement.

2. Traditional A* and DWA Algorithms

2.1 Traditional A* Algorithm

Traditional A* algorithm is a heuristic best-first search method, and its core cost function is:

$$f(n) = g(n) + h(n) \#(1)$$

where f represents the actual cost from the start node to the current node, and h denotes the heuristic estimate from the current node to the goal. In this paper, to improve the accuracy and continuity of the planned path, both the actual cost and the heuristic function are defined using the Euclidean distance[10]:

$$g(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \#(2)$$

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \#(3)$$

The distance measurement method used in the A* algorithm can more accurately reflect the true spatial distance between nodes. The A* algorithm conducts path search by maintaining an Open List and a Closed List. Initially, the start node is added to the Open List, and its cost function *g* is set to 0. The heuristic function *f* is then calculated. Subsequently, the node *n* with the smallest *f* value in the Open List is selected as the current node to be expanded. If *n* is the goal node, it indicates that a path has been found, and the complete path can be constructed by backtracking through the parent nodes. Otherwise, *n* is moved to the Closed List, and all adjacent nodes are processed. If an adjacent node already exists in the Closed List, it is skipped. If an adjacent node is not in the Open List, it is added to the Open List, and its parent node is recorded as *n*. If the adjacent node is already in the Open List, the new *g* value is compared; if it is better, the corresponding *g* and *f* values for the parent node are updated. This process is repeated until either the goal node is expanded or the Open List becomes empty, thereby completing the path search.

2.2 Traditional Dynamic Window Approach (DWA)

The core idea of the Dynamic Window Approach (DWA) algorithm is the velocity space (*v*,*w*), sample multiple velocity combinations, predict the motion trajectories of each combination over a short period of time, and score these trajectories using a predefined evaluation function.[11] The optimal trajectory is then selected, and its corresponding velocity command (*v*, *w*) is used to drive the robot forward. The restrictions that the sampled velocities must satisfy include:

$$V_s = \{(v,w)|v \in [v_{min}, v_{max}] \wedge \omega \in [\omega_{min}, \omega_{max}]\} \#(4)$$

where V_s is the set of velocities that the robot can achieve, v_{min} and v_{max} represent the minimum and maximum linear velocities of the robot, and ω_{min} and ω_{max} represent the minimum and maximum angular velocities of the robot.

$$V_d = \{(v,w)|v \in [v_c - \dot{v}_b \vartriangle t, v_c + \dot{v}_a \vartriangle t] \wedge \omega \in [\omega_c - \dot{\omega}_b \vartriangle t, \omega_c + \dot{\omega}_a \vartriangle t]\} \#(5)$$

where V_d is the set of velocities that are limited by the motor's acceleration and deceleration capabilities, $\dot{v}_a$ and $\dot{v}_b$ represent the maximum acceleration and maximum deceleration, and $\dot{\omega}_a$ and $\dot{\omega}_b$ represent the maximum angular acceleration and minimum angular deceleration.

$$V_a = \left\{(v,\omega)\middle| v \le \sqrt{2 \cdot dist(v,\omega) \cdot \dot{v}_b}, \omega \le \sqrt{2 \cdot dist(v,\omega) \cdot \dot{\omega}_b}\right\} \#(6)$$

where V_a is the set of velocities that are limited by obstacle constraints. The DWA actually uses the intersection of three sets, $v_{d\omega} = v_s \cap v_d \cap v_a$. Only velocities that simultaneously satisfy physical constraints, motor performance constraints, and collision safety constraints are considered valid velocity candidates. Based on the trajectory predicted by the kinematic model[12], an evaluation function is introduced to score and select the optimal trajectory. The evaluation function is as follows:

$$G(v,\omega) = \sigma\big(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)\big) \#(7)$$

where the purpose of the normalization function σ is to unify the scoring scale, making various evaluation metrics numerically comparable to select the optimal speed command. The heading(*v*, *ω*) function serves as the evaluation function for the azimuth angle, used to assess the angular difference between the predicted trajectory endpoint orientation and the direction of the target point for a robot given velocity (*v*, *ω*). The dist(*v*, *ω*) function is a distance rating function that indicates

the distance between the robot and the nearest obstacle when it moves to the end of the current predicted trajectory. The vel(v,ω) function acts as a velocity rating function reflecting the magnitude of the current linear velocity. This set of functions collectively contribute to the comprehensive evaluation of the robot's movement status, thereby ensuring the selection of the most suitable path and speed parameters to achieve efficient and safe navigation towards the target position.

3. Global Path Planning Based on an Improved A* Algorithm

3.1 Improved Cost Function

In the traditional A* algorithm, the heuristic function $h(n)$ typically considers only the distance between the current node and the goal, often neglecting the probabilistic distribution of obstacles in the environment. As a result, in complex environments with densely distributed obstacles, the planned path may traverse high-risk areas. To address this issue, an obstacle probability P is introduced into the path cost calculation to adjust path selection. The obstacle probability refers to the likelihood of encountering obstacles along the path from the current node to the goal. The regional obstacle probability is defined as:

$$P_{obs} = \frac{Number\ of\ Obstacle\ Points}{The\ number\ of\ grid\ cells\ in\ the\ region} \#(8)$$

Furthermore, given a node $n(x_n, y_n)$ and the target node $n(x_{target}, y_{target})$, the search area for path planning is defined as a rectangular region, the formula for the district boundary is as follows:

$$\begin{cases} x_{min} = min(x_n, x_{target}) - 0.8 \\ x_{max} = max(x_n, x_{target}) + 0.8 \end{cases} \#(9)$$

$$\begin{cases} y_{min} = min(y_n, y_{target}) - 0.8 \\ y_{min} = max(y_n, y_{target}) + 0.8 \end{cases} \#(10)$$

The total number of grids within the region is calculated using the following formula:

$$\begin{cases} N_x = |x_n - x_{target}| + 1 \\ N_y = |y_n - y_{target}| + 1 \end{cases} \#(11)$$

By incorporating obstacle probability, the heuristic function takes into account not only the path length but also the safety of the path. The improved heuristic function introduces a weighting of the heuristic cost by incorporating the probability of obstacles. The formula is as follows:

$$F(n) = g(n) + (1 + e^{-p*P_{obs}}) \cdot h(n) \#(12)$$

where p is a parameter that adjusts the influence weight of obstacle probability. The coefficient preceding the heuristic term renders the heuristic function sensitive to obstacle probability. When P_{obs} is high, $e^{-p*P_{obs}}$ decreases, causing the coefficient to approach 1. This reduces the weight of the heuristic term, compelling the algorithm to prioritize actual cost and thereby facilitate avoidance of high-risk regions.

3.2 Multi-round Polyline Optimization

The preliminary path generated by the conventional A* algorithm often contains numerous redundant nodes and frequent turns, which increases the path length while compromising its executability and continuity. To address these limitations, this paper proposes a multi-stage polyline optimization approach to eliminate redundant nodes, reduce the number of turns, shorten the overall path length, and enhance path smoothness.

3.2.1 Redundant Point Removal

Given the original path point sequence $P = \{P_1, P_2, \dots, P_n\}$, if a group of three consecutive points P_{i-1}, P_i, P_{i+1} are collinear, the middle point P_i can be removed. The criterion for determining collinearity is as:

$$\left| \frac{\vec{A} \cdot \vec{b}}{||\vec{a}|| \cdot \left||\vec{b}|\right|} \right| < \epsilon \#(13)$$

Then the three points are approximately collinear, and P_i can be removed.

3.2.2 Connectivity Check

Building upon the initial optimization, we further consider any pair of path points P_i and a subsequent $P_j (j > i)$. If the straight-line segment connecting them is obstacle-free, all intermediate points can be safely eliminated. The obstacle detection employs the point-to-line distance metric:

$$D = \frac{\left|(y_j - y_i)x_o - (x_j - x_i)y_o + x_j y_i - y_j x_i\right|}{\sqrt{(y_j - y_i)^2 + (x_j x_i)^2}} \#(14)$$

The direct path between nodes is validated as obstacle-free when the perpendicular distance d from every obstacle point $O = (x_o, y_o)$ to line segment $P_i P_j$ maintains d > D (safety boundary).

3.2.3 Reverse Path Re-optimization

Since the polyline optimization operates in a locally greedy manner, it may only achieve unidirectional optimality due to its sequential traversal. To eliminate suboptimal nodes caused by processing order, we apply the connectivity-based optimization strategy again to the reversed path. This bidirectional optimization approach ultimately yields a path with significantly improved turning points and reduced total length.

3.3 Avoidance of Obstacle Vertices

In the traditional A* algorithm, the search process often generates paths that closely follow obstacle vertices, thereby reducing the safety margin and increasing the risk of collision during actual execution. To address this issue, this paper proactively excludes nodes adjacent to obstacle vertices during the node expansion phase, ensuring that the generated path maintains a safe distance from obstacles. The core idea is expressed as follows:

$$\mathcal{N}_{safe} = \left\{ n \in \mathcal{N}_{expend} \,\middle|\, \forall v \in \mathcal{V}_{obs},\ \| n - v \|_\infty > 1 \right\} \#(15)$$

where $\mathcal{N}_{expend}$ is the set of candidate nodes for expansion, $\mathcal{V}_{obs}$ is the set of obstacle vertices, and $\| n - v \|_\infty$ denotes the Chebyshev distance between node n and obstacle vertex v. A node is added to the safe expansion set $\mathcal{N}_{safe}$ only if its distance from all obstacle vertices exceeds 1. This method effectively increases the safety margin of the path and reduces the likelihood of edge-hugging paths.

4 Local Path Planning by Integrating Improved A and Enhanced DWA

4.1 Improved Evaluation Function

In this paper, we enhance the traditional Dynamic Window Approach (DWA) evaluation function by introducing a consideration for the distance from dynamic obstacles. The addition of a dynamic obstacle component enables robots to proactively avoid obstacles that may move into their planned path, thus preventing collisions. The extended evaluation function now includes four components: heading angle towards the goal, distance from static obstacles, velocity magnitude, and distance from dynamic obstacles. All components are normalized across all candidate velocity combinations to eliminate the influence of units and scale differences, specifically as:

$$\tilde{F}_k(i) = \frac{f_k(i)}{\sum_{i=1}^{N} f_k(I)} \#(16)$$

where k denotes the type of component, i represents the i-th velocity combination, and N is the total number of alternative velocity combinations. After normalization, each component is weighted and summed to obtain the final evaluation score. The specific form of the evaluation function is as follows:

$$F_i = \alpha \cdot \tilde{f}_{heading}(i) + \beta \cdot \tilde{f}_{clearance}(i) + \gamma \cdot \tilde{f}_{velocity}(i) + \delta \cdot \tilde{f}_{dynamic}(i) \#(17)$$

$$f_{dynamic} = \min_i \left|\left|P_{end} - P_{dynamic,j}(t_{end})\right|\right| \#(18)$$

where Equation 18 represents the minimum distance between the endpoint of the predicted trajectory P_{end} and the position of the j-th dynamic obstacle at the prediction time $P_{dynamic,j}(t_{end})$. In this formula, F_i stands for the overall evaluation value of the i-th velocity combination, with the highest scoring i being selected as the control output for the current moment. The parameters α, β, γ and δ represent the weights assigned to each respective component after normalization.

4.2 Integrated Algorithm

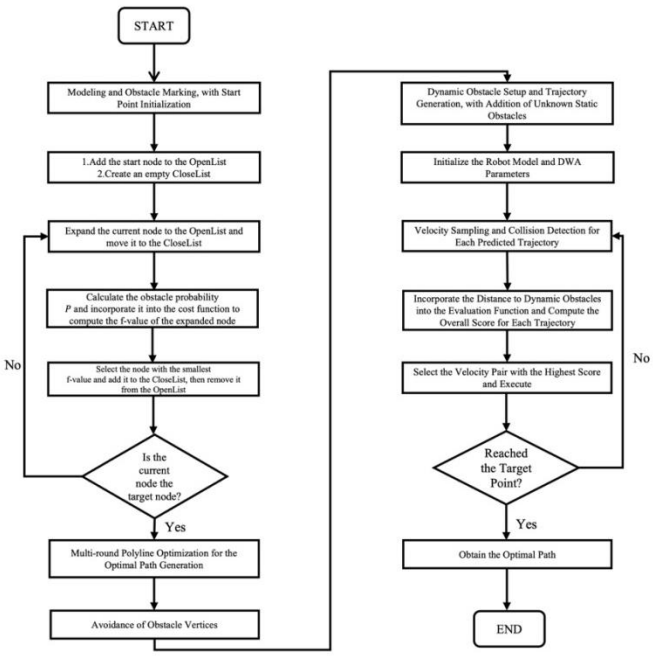

Figure 1: Flowchart of the Improved A and DWA Fusion Algorithm

5 Verification of the Algorithm through Simulation

To verify the path planning performance of the proposed improved A* algorithm and its dynamic obstacle avoidance effect after integration with the DWA, a simulation environment was built based on the MATLAB 2024b platform, and multiple sets of comparative experiments were designed. The improvements of the improved A* algorithm in terms of path length, planning efficiency, and path smoothness were evaluated. Subsequently, the improved A* algorithm was integrated with the DWA that uses the improved evaluation function to achieve coordinated control for global path planning and local dynamic obstacle avoidance.

5.1 Comparison of the A* Algorithm Before and After Improvement

A grid map with a resolution of 1×1 was constructed in MATLAB. The map uses different shapes and colors to represent specific functions: triangles represent the start node, circles represent the goal node, gray grids represent obstacles, and white grids represent free-moving areas.

In *Figure 2* and *Figure 3*, the comparison results of running the traditional A* algorithm and the improved A* algorithm on a 20×20 grid map are shown, with no path smoothing applied. The traditional A* algorithm took 0.025 seconds, with a turning degree of 315, 7 turns, a path length of 29, and 207 nodes expanded. In contrast, the improved A* algorithm took 0.02 seconds, with a turning degree of 263, 4 turns, a path length of 30, and 85 nodes expanded. The specific data can be found in

Table 1. As seen, the improved A* algorithm reduced the time by 20%, the number of turns by 42%, and the number of expanded nodes by 58.9%.

Additionally, *Figure 4* and *Figure 5* show the comparison results on a 30×30 grid map, with detailed data in Table 1. In this scale, the improved A* algorithm reduced the time by 11%, the turning degree by 82%, the number of turns by 44%, and the number of expanded nodes by 46%. It is worth noting that the improved A* algorithm did not exhibit the phenomenon of vertex diagonal crossing obstacles on both map sizes. Overall, the improved A* algorithm demonstrated superior performance in path optimization.

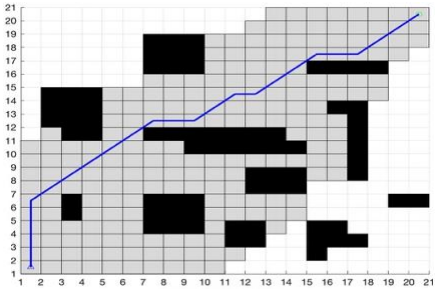

Figure 2: Path of the Traditional A Algorithm on a 20×20 Map*

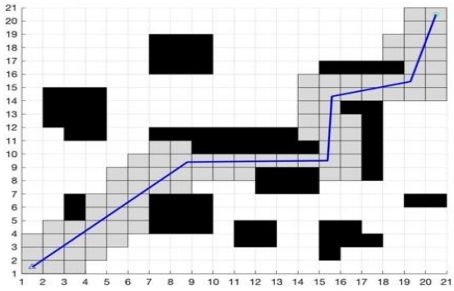

Figure 3: Path of the Improved A Algorithm on a 20×20 Map*

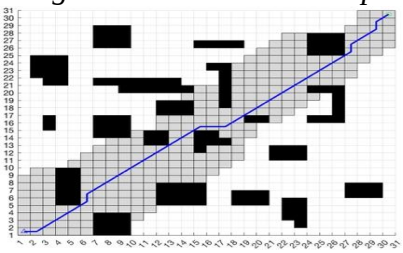

Figure 4: Path of the Traditional A Algorithm on a 30×30 Map*

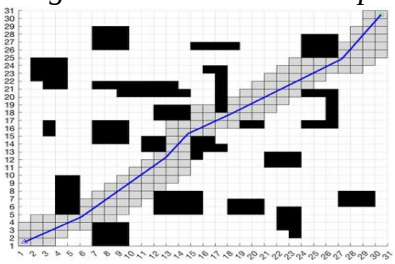

Figure 5: Path of the Improved A Algorithm on a 30×30 Map*

Table 1: Comparison Between Traditional and Improved A Algorithms*

	Grid Size	Planning Time(s)	Total Turning Angle	Number of Turns	Path Length(m)	Number of Visited Nodes
Traditional A*	20*20	0.025	315	7	29	207
Improved A*	20*20	0.02	263	4	30	85
Traditional A*	30*30	0.009	405	9	42	251
Improved A*	30*30	0.008	72	5	41	134

5.2 Dynamic Environment Path Planning with Improved A* Algorithm Integrated with DWA

This experiment investigates the obstacle avoidance response of a mobile robot in dynamic environments by setting up three typical interaction scenarios: intersection, crossing, and head-on. The goal is to validate the obstacle avoidance ability of the improved A* algorithm integrated with Dynamic Window Approach (DWA) in dynamic environments. As shown in the figures below, *Figure 6* demonstrates the path trajectories of the mobile robot and dynamic obstacles in a grid map during the crossing scenario. It is clearly observed that the robot effectively avoids the dynamic obstacles during the crossing phase. *Figure 7* shows the changes in the robot's linear velocity and angular velocity during this process, where the blue curve represents the linear velocity and the red curve represents the angular velocity. The results show that as the robot approaches the crossing point, its linear velocity accelerates to a certain value, rapidly decelerates when encountering obstacles, and then accelerates again. The angular velocity alternates between positive and negative, indicating that the robot performs multiple steering maneuvers during the obstacle avoidance process. *Figure 8* shows the changes in the robot's attitude angle. During the crossing avoidance phase, the attitude angle fluctuates slightly between 0.9 and 1.1, with small peak values. Overall, the experimental results show that the proposed improved A* algorithm integrated with DWA enables the mobile robot to make timely steering and detour maneuvers when encountering dynamic obstacles, demonstrating good dynamic obstacle avoidance ability.

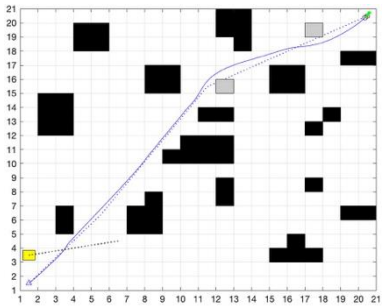

Figure 6: Path Trajectories of the Mobile Robot and Dynamic Obstacles in the Grid Map

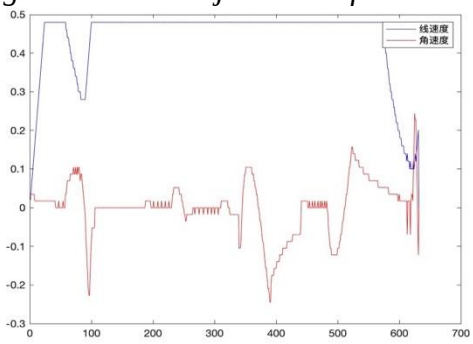

Figure 7: Linear and Angular Velocity

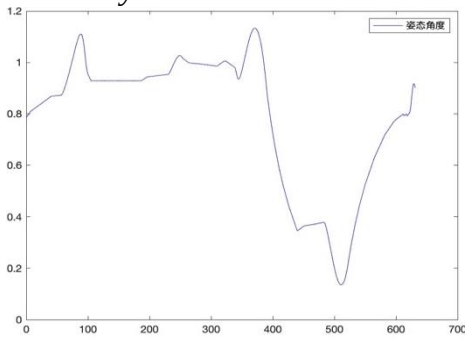

Figure 8: Attitude Angle

Figure 9 and *Figure 10* respectively show the trajectory of the mobile robot in the head-on and crossing dynamic obstacle scenarios, further reflecting the robot's path planning and obstacle avoidance performance under different dynamic obstacle interaction modes. The specific experimental data for each scenario are summarized in *Table 2*.

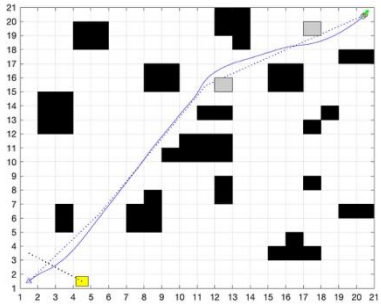

Figure 9: Crossing Path Map

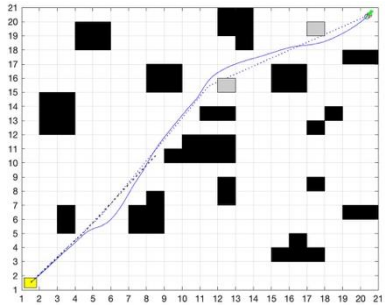

Figure 10: Head-on Path Map

Table 2: Data Comparison of the Mobile Robot in Different Collision Environments

	Time Duration (s)	Path Length (m)
Cross Collision	148.83	28.08
Crossing Collision	128.23	27.97
Head-on Collision	132.03	27.87

6 Conclusion

This paper addresses path planning and obstacle avoidance for mobile robots in complex dynamic environments by developing an optimal path planning scheme, fusing the improved A* algorithm with Dynamic Window Approach (DWA). During the global path planning step, the traditional A* algorithm is adopted as the foundation, with a weight adjustment mechanism of the heuristic function, an obstacle probability model, multi-round polyline optimization approach, and obstacle vertex avoidance rules being added. These steps efficiently remove a series of problems in traditional A* paths, such as high-density turning points, redundancy of nodes, and inability to

avoid obstacle vertices. Meanwhile, the efficiency of searching is significantly improved and the generated path is adapted to better suit the kinematic characteristics of mobile robots. With dynamic obstacles and unknown static obstacles coexisting in actual environments as the target, the DWA algorithm is applied in the local obstacle avoidance stage, and a dynamic obstacle distance factor is embedded into the evaluation function so that mobile robots are able to perform dynamic obstacle avoidance. Simulation experiments in the MATLAB environment verify that the integrated approach presented here can generate stable and robust path planning and dynamic avoidance of obstacles in mixed environments with static and dynamic obstacles. Experimental results show that this scheme not only avoids dynamic obstacles but also shows remarkable improvements with respect to path length, search efficiency, turns, and path safety. However, the scheme has not yet been experimented with. I will continue to expand the application of the system in path planning for physical robots, self-map building, and real-time path replanning, and study the deployment and engineering usage of the scheme on embedded hardware platforms in the future work for its further extensive application in practical applications such as autonomous intelligent logistics and service robots.

Acknowledgements

This paper is supported by Projects of major scientific and technological research of Ningbo City (2021Z059, 2022Z090(2022Z050), 2023Z050(the second batch)), Projects of major scientific and technological research of Beilun District, Ningbo City(2021BLG002, 2022G009), Projects of scientific and technological research of colleges student's of China(202313022036, 202413001008).

References

[1] Liu L, Wang X, Yang X, et al. Path planning techniques for mobile robots: Review and prospect[J]. Expert Systems with Applications, 2023, 227: 120254.

[2] Qin H, Shao S, Wang T, et al. Review of autonomous path planning algorithms for mobile robots[J]. Drones, 2023, 7(3): 211.

[3] Zhang D, Chen C, Zhang G. AGV path planning based on improved A-star algorithm[C]//2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2024, 7: 1590-1595.

[4] Yan X, Ding R, Luo Q, et al. A dynamic path planning algorithm based on the improved DWA algorithm[C]//2022 Global Reliability and Prognostics and Health Management (PHM-Yantai). IEEE, 2022: 1-7.

[5] Li X, Hu X, Wang Z, et al. Path planning based on combinaion of improved A-STAR algorithm and DWA algorithm[C]//2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM). IEEE, 2020: 99-103.

[6] Zhang D, Chen C, Zhang G. AGV path planning based on improved A-star algorithm[C]//2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2024, 7: 1590-1595.

[7] Sang W, Yue Y, Zhai K, et al. Research on AGV Path Planning Integrating an Improved A Algorithm and DWA Algorithm[J]. Applied Sciences, 2024, 14(17): 7551.*

[8] Zhang C, Yang X, Zhou R, et al. A path planning method based on improved A and fuzzy control dwa of underground mine vehicles[J]. Applied Sciences, 2024, 14(7): 3103.*

[9] Song B, Tang S, Li Y. A new path planning strategy integrating improved ACO and DWA algorithms for mobile robots in dynamic environments[J]. Mathematical Biosciences and Engineering, 2024, 21(2): 2189-2211.

[10] Hajizadeh R ,Nazari F .Manifold learning through locally linear reconstruction based on Euclidean distance[J].Multimedia Tools and Applications,2024,84(1):1-21.

[11] Hu Y, Long H, Chen M. The analysis of pedestrian flow in the smart city by improved DWA with robot assistance[J]. Scientific reports, 2024, 14(1): 11456.

[12] Yin X, Cai P, Zhao K, et al. Dynamic path planning of AGV based on kinematical constraint A algorithm and following DWA fusion algorithms[J]. Sensors, 2023, 23(8): 4102.*