electronics

MDPI

Article

Research on Path Planning with the Integration of Adaptive A-Star Algorithm and Improved Dynamic Window Approach

Tianjian Liao, Fan Chen, Yuting Wu, Huiquan Zeng, Sujian Ouyang and Jiansheng Guan *

College of Electrical Engineering and Automation, Xiamen University of Technology, Xiamen 361024, China; 2122031378@s.xmut.edu.cn (T.L.); 2122031328@s.xmut.edu.cn (F.C.); 2122031353@s.xmut.edu.cn (Y.W.); 2122031327@s.xmut.edu.cn (H.Z.); 2122031351@s.xmut.edu.cn (S.O.)
* Correspondence: jsguan@xmut.edu.cn

Abstract: In response to the shortcomings of the traditional A-star algorithm, such as excessive node traversal, long search time, unsmooth path, close proximity to obstacles, and applicability only to static maps, a path planning method that integrates an adaptive A-star algorithm and an improved Dynamic Window Approach (DWA) is proposed. Firstly, an adaptive weight value is added to the heuristic function of the A-star algorithm, and the Douglas–Pucker thinning algorithm is introduced to eliminate redundant points. Secondly, a trajectory point estimation function is added to the evaluation function of the DWA algorithm, and the path is optimized for smoothness based on the B-spline curve method. Finally, the adaptive A-star algorithm and the improved DWA algorithm are integrated into the fusion algorithm of this article. The feasibility and effectiveness of the fusion algorithm are verified through obstacle avoidance experiments in both simulation and real environments.

Keywords: adaptive A-star algorithm; improved Dynamic Window Approach; path planning

Citation: Liao, T.; Chen, F.; Wu, Y.; Zeng, H.; Ouyang, S.; Guan, J. Research on Path Planning with the Integration of Adaptive A-Star Algorithm and Improved Dynamic Window Approach. *Electronics* **2024**, *13*, 455. https://doi.org/10.3390/electronics13020455

Academic Editor: Giuseppe Menga

Received: 22 December 2023
Revised: 16 January 2024
Accepted: 19 January 2024
Published: 22 January 2024

1. Introduction

With the development of automation control technology, mobile robots have been widely used in industrial production, e-commerce, logistics, and other fields. As an essential component of mobile robot automatic control, path planning algorithms have attracted a series of research studies by scholars in recent years. The commonly used global path planning algorithms for mobile robots include the Dijkstra algorithm [1,2], the A-star algorithm [3–5], the RRT algorithm [6–8], etc. The commonly used local path planning algorithms include the artificial potential field method [9,10], the Dynamic Window Approach [11], etc. Indeed, other scholars have made important contributions to the field of path planning by studying other algorithms. Reference [12] extracts the variability of travel time from the existing correlation between traffic flow and corresponding link time, conducts multi-objective analysis on the generated paths, and generates a Pareto optimal set for each pair of demand nodes in the network. This solves the problem of finding a set of non-dominated shortest paths in a stochastic traffic network. Reference [13] proposes a new fully hierarchical genetic algorithm method to solve the problem of bus network path planning design. This method effectively handles the multi-objective nature of the problem and has universality and adaptability for path planning on a practical scale. Among them, the A-star algorithm, as a classic path planning algorithm, performs well in calculating the global optimal path and plays a key role in robot navigation, game NPC pathfinding, and other fields [14–19].

However, in practical applications of many engineering projects, it has been found that the traditional A-star algorithm has shortcomings, such as having too many traversing nodes, uneven paths, and too many turning nodes in the path it finds. In order to obtain a better path, many scholars have studied some improvement methods. Reference [20] proposes an EBS-A* algorithm, which introduces extended distance, bidirectional searches,

and smoothing into path planning. This greatly improves the efficiency of path planning and reduces the number of key nodes and the number of right-angle turns. However, it requires both forward and backward searches, which require more computational resources. Reference [21] addresses the challenge of determining heuristic functions in the A* algorithm and compares the performance of four heuristic functions: Manhattan distance, Euclidean distance, Chebyshev distance, and diagonal distance. However, the author only focused on the selection of heuristic functions and did not consider other factors that may affect the performance of the A* algorithm. In Reference [22], the key point selection strategy is applied to eliminate redundant points and unnecessary turning points, and the turning points in the path are smoothed. This reduces the path length and results in smoother path planning, but it takes longer to plan paths in areas without obstacles. Reference [23] expands the search neighborhood from 3 × 3 to 5 × 5, and this reduces the number of inflection points, improves the turning angle, and removes redundant points but does not consider the smoothness of the path. Reference [24] proposes an A-star algorithm based on bidirectional search paths, which greatly reduces the length of the driving path. However, since it requires the simultaneous processing of searches in both directions, it also requires more computational resources and time.

By analyzing the advantages and disadvantages of the algorithms in the literature, it is found that these methods reduce the path length, improve the search efficiency, and reduce the number of turns compared to the traditional A-star algorithm. However, each of these algorithms has its own shortcomings and is only applicable to static maps. In response to this, this article introduces adaptive weights into the heuristic function of the A-star algorithm and integrates it with the improved DWA algorithm, thereby enabling the algorithm to have a good path search efficiency and dynamic obstacle avoidance capability. The flowchart of the algorithm improvement in this article is shown in Figure 1.

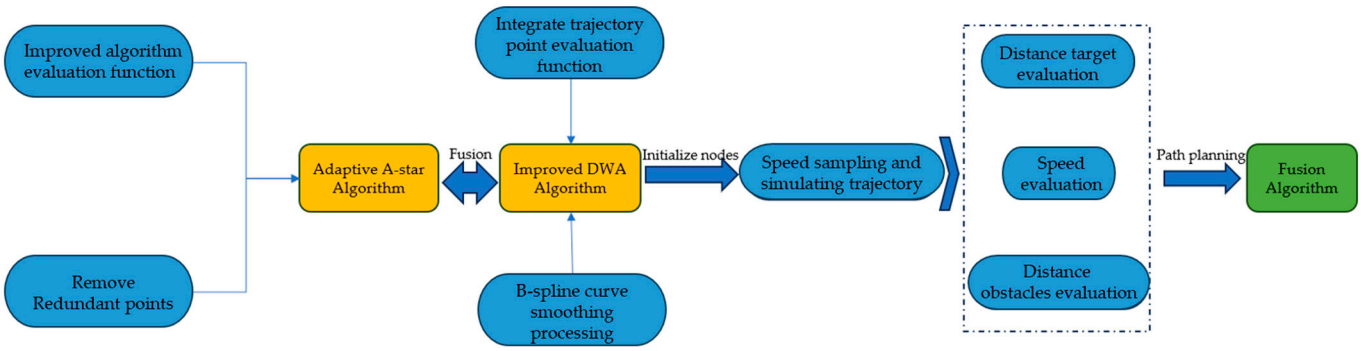


Figure 1. Improved algorithm flowchart.

The structure of this article is as follows: Sections 2 and 3 introduce the related algorithms and how this article improves upon existing algorithms. Section 4 validates the algorithm proposed in this article and analyzes the results in both simulated and real environments. Section 5 discusses the application scope, limitations of the algorithm in this article, and directions for future research improvements. Section 6 reveals the conclusion.

2. Principles and Improvements of A-Star Algorithm

2.1. Basic Principles of Traditional A-Star Algorithm

The A-star algorithm, as a commonly used path planning algorithm, has been widely applied in areas such as robot navigation and automatic path finding for characters in games. It mainly addresses the problem of low efficiency in the Dijkstra algorithm, which calculates the distance from a single source node to all nodes, traverses all nodes, and expands the nodes in all possible directions until finding the target node [25–30]. Unlike it, the A-star algorithm is a heuristic algorithm that greatly improves path search efficiency

by incorporating information about target points while maintaining optimal path search performance. The evaluation function is as follows:

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

where, $f(n)$ is the evaluation function of node n from the starting point to the target point; $g(n)$ represents the actual cost from the starting point to node n; $h(n)$ is the estimated cost from node n to the target point, also known as the heuristic function. Based on project requirements, the cost of distance is generally calculated using Euclidean distance, with the following formula:

$$\begin{cases} h(n) = \sqrt{\left(x_{goal} - x_n\right)^2 + \left(y_{goal} - y_n\right)^2} \\ g(n) = \sqrt{(x_n - x_{start})^2 + (y_n - y_{start})^2} \end{cases} \tag{2}$$

where (x_n, y_n), (x_{start}, y_{start}), and (x_{goal}, y_{goal}) represent the coordinates of node n, starting point s, and target point g, respectively;

The steps of the A-star algorithm are as follows:

1. Start the search from the starting point s, and add s to the "open_list" as a node to be searched.
2. Search all the nodes around the starting point s that can be traversed, and add them to the "open_list". Set the starting point s as the parent node of these nodes.
3. Remove the starting point s from the "open_list", and add it to the "close_list".
4. Calculate the estimated cost $f(n)$ for each node in the current "open_list", and select the node n_1 with the smallest $f(n)$ value. Remove n_1 from the "open_list" and add it to the "close_list".
5. Search all the nodes around node n_1. If these nodes are obstacles or nodes in the "close_list", they are not considered. If these nodes are not in the "open_list", add them to the "open_list" and calculate their estimated cost $f(n)$, setting n_1 as the parent node. If an adjacent node n_2 is already in the "open_list", calculate the new path cost from point s to node n_2 using $g(n)$. If the new $g(n)$ value is lower, set n_1 as the parent node and recalculate $f(n)$; if the new $g(n)$ value is higher, do not make any changes.
6. Continue to find the node with the smallest $f(n)$ value in the "open_list", remove it from the "open_list", add it to the "close_list", and find adjacent reachable nodes. Repeat this process until the target point g appears in the "open_list", indicating that a path has been found and the loop is ended.

2.2. Improvement of A-Star Algorithm

2.2.1. Improved A-Star Algorithm Evaluation Function

To improve the search efficiency of A-star algorithm, an adaptive weight value is introduced on the basis of the traditional A-star algorithm evaluation function τ_t, then the evaluation function is as follows:

$$f(n) = g(n) + \tau_t * h(n) \tag{3}$$

The traditional A-star algorithm evaluation function has a weight of 1:1 for $g(n)$ and $h(n)$. When $\tau_t > 1$, the algorithm tends to the estimated cost $h(n)$ during path planning, which accelerates the speed of traversing nodes and improves search efficiency, but may miss the optimal path; When $\tau_t < 1$, the algorithm tends to lean towards the actual cost $g(n)$ during path planning, which tends to search for the optimal path and increases the search time. Therefore, when the mobile robot is closer to the starting point s, increase the adaptive weight value τ_t. Expand the algorithm search to the target point as soon as possible, reduce the number of nodes traversed, and improve search efficiency; When the mobile robot is closer to the target point g, reduce the adaptive weight value τ_t. The algorithm will slow

down the search speed and increase the search range to find the optimal path. This article will introduce adaptive weight values τ_t is represented as

$$\tau_t = \frac{\sqrt{\left(x_{goal} - x_n\right)^2 + \left(y_{goal} - y_n\right)^2}}{\sqrt{\left(x_n - x_{start}\right)^2 + \left(y_n - y_{start}\right)^2}} = \frac{h(n)}{g(n)} \tag{4}$$

When the mobile robot is closer to the starting point s, $g(n) < h(n)$, then $\tau_t > 1$; When the mobile robot is closer to the target point g, $g(n) > h(n)$, then $\tau_t < 1$; The new improved evaluation function is as follows:

$$f(n) = g(n) + \tau_t * h(n) = g(n) + \frac{h(n)}{g(n)} * h(n) = \frac{g(n)^2 + h(n)^2}{g(n)} \tag{5}$$

2.2.2. Redundant Point Removal

The path planned by the traditional A-star algorithm has many redundant nodes, resulting in unnecessary path turns and affecting the life of the motor. The Douglas–Pucker thinning algorithm can compress a large number of redundant points to extract key points. In general, the algorithm can preserve feature points on more frequently curved shapes, so it can accurately remove fixed points on consecutive small corners. The simulation environment that this article is going to build will add many obstacles, so this algorithm is well-suited for the complex multi-obstacle environment in which the experiment will be conducted.

The optimization effect of the Douglas–Pucker algorithm is shown in Figure 2. The path composed of black arrows represents the path before optimization, and the path composed of green arrows represents the path after redundancy point processing optimization. And the processing steps for path redundancy points are shown in Figure 3.

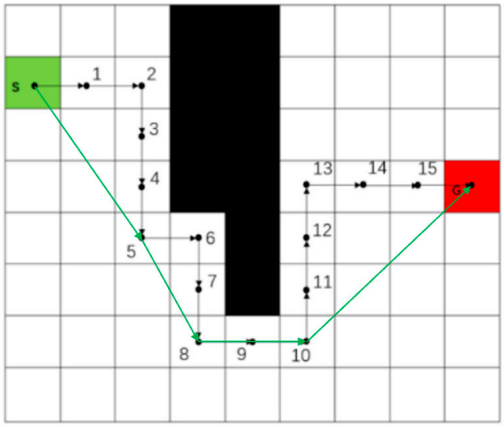


Figure 2. Effective handling of redundant points.

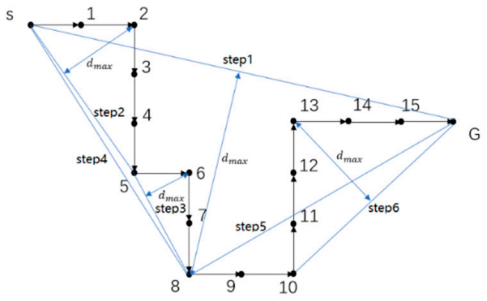


Figure 3. The Douglas–Pucker algorithm diagram.

The specific algorithm steps are as follows:

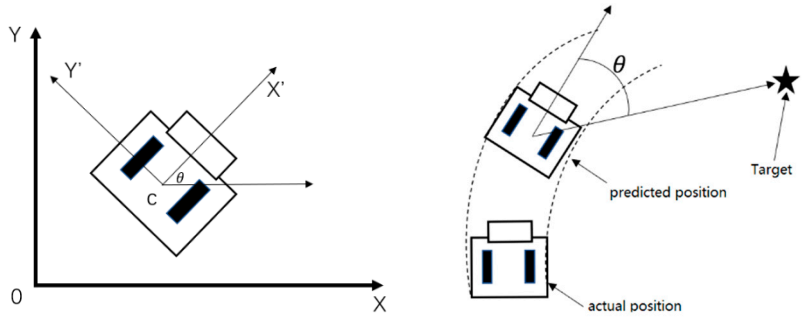
(1) Set the threshold $d_{\text{threshold}}$ for removing redundant points. The smaller the $d_{\text{threshold}}$, the higher the precision and the fewer the redundant points removed.
(2) Connect the start point S and end point G of the path with a straight line (as shown in Figure 3).
(3) Calculate the distances between all path points between the start and end points to this straight line, and find the maximum distance $d_{\max}$ and the corresponding path point (as shown in Figure 3, point 8).
(4) If $d_{\max} < d_{\text{threshold}}$, it means that the start and end points S and G can represent the characteristics of this path and remove all path points between them. The path processing is complete.
(5) If $d_{\max} \geq d_{\text{threshold}}$, it means that the start and end points S and G cannot fully reflect the characteristics of this path and thus need to be divided further. Split the path at the path point corresponding to $d_{\max}$ into two parts, and repeat step (2) again.
(6) Until all path points are processed, the redundant point optimization is completed.

3. Principles and Improvements of DWA Algorithm

3.1. Basic Principles of Traditional DWA Algorithm

3.1.1. Kinematics Model

This article selects the Blue Whale Xiaoqiang XQ-5 mobile robot as the experimental platform for experimental verification. This robot is a non-omnidirectional two-wheel differential mobile robot, and its kinematic model is shown in Figure 4a.



(**a**) (**b**)

Figure 4. Mobile robot model and azimuth angle: (**a**) Kinematic model of XQ-5 mobile robot; (**b**) Mobile robot azimuth.

Because mobile robots do not have omnidirectional motion, they can only move forward and rotate, with the direction of forward speed consistent with the heading angle of the robot. At the same time, the control variables are the acceleration to control forward motion and the angular acceleration to control steering. The kinematic model of the robot is as follows:

$$\begin{cases} x(t) = x(t-1) + v(t)\Delta t cos(\theta(t-1)) \\ y(t) = y(t-1) + v(t)\Delta t sin(\theta(t-1)) \\ \theta(t) = \theta(t-1) + \omega(t)\Delta t \end{cases} \quad (6)$$

where $x(t), y(t), \theta(t)$ is the position and orientation of the robot under the world coordinate axis at time t, $x(t-1), y(t-1), \theta(t-1)$ is the position and orientation of the robot under the world coordinate axis at the previous moment.

3.1.2. Speed Sampling

The speed sampling constraints of the DWA algorithm [31] are as follows:

(1) The sampling speed of a mobile robot should be controlled within the range between its maximum and minimum speeds, as shown in the following formula:

$$V_S = \{(v, \omega) | v \in [v_{min}, v_{max}] \wedge \omega \in [\omega_{min}, \omega_{max}]\} \tag{7}$$

where v is the linear velocity, ω is the angular velocity, V_S is the set of all vector velocities that the robot can reach.

(2) Due to limitations in the motor performance of a mobile robot, there is a range of acceleration, within which the maximum acceleration and deceleration are able to reach before a dynamic window is formed. This ensures that

$$V_d = \{(v, \omega) | v \in [v_t - a_{v_{max}} \Delta t, v_t + a_{v_{max}} \Delta t] \wedge \omega \in [\omega_t - a_{\omega_{max}} \Delta t, \omega_t + a_{\omega_{max}} \Delta t]\} \tag{8}$$

where v_t is the linear velocity at the current moment, ω_t is the current angular velocity, Δt is the time interval, $a_{v_{max}}$ and $a_{\omega_{max}}$ represent the maximum linear acceleration and the maximum angular acceleration of the mobile robot, respectively.

(3) Considering the safety of the mobile robot, to avoid collision with obstacles, the speed should meet the following requirements:

$$V_a = \left\{ (v, \omega) \middle| v \leq \sqrt{2dist(v, \omega)\dot{v}} \wedge \omega \leq \sqrt{2dist(v, \omega)\dot{\omega}} \right\} \tag{9}$$

where $dist(v, \omega)$ is the shortest distance from the current trajectory to the obstacle.

Under the restriction of the three influencing factors mentioned above (v, ω) a dynamic window will be formed, with a speed range V_r that meets

$$V_r = V_S \cap V_d \cap V_a \tag{10}$$

3.1.3. Evaluation Function

The DWA algorithm uses a cost function to evaluate the predicted trajectory. The evaluation function is as follows:

$$G(v, \omega) = \sigma(\alpha * heading(v, \omega) + \beta * dist(v, \omega) + \gamma * velocity(v, \omega)) \tag{11}$$

Among them, $heading(v, \omega)$ is the angle deviation between the heading angle and the line connecting the robot and the target point is used to correct the robot's heading (as shown in Figure 4b). $dist(v, \omega)$ is the distance between the robot's trajectory and the nearest obstacle, used to keep the robot at a safe distance from the obstacle. $velocity(v, \omega)$ is the speed of the robot, used to make the robot reach the target point as soon as possible. α, β, γ are three parameters represent the weights, σ is the smoothing coefficient.

In order to prevent the weight of one evaluation function from being too large due to different evaluation criteria for each evaluation function, thus affecting the accuracy of the evaluation function, normalization is required for each evaluation function:

$$normal_{heading(i)} = \frac{heading(i)}{\sum_{i=1}^{n} heading(i)} \tag{12}$$

$$normal_{dist(i)} = \frac{dist(i)}{\sum_{i=1}^{n} dist(i)} \tag{13}$$

$$normal_{velocity(i)} = \frac{velocity(i)}{\sum_{i=1}^{n} velocity(i)} \tag{14}$$

where n is the total number of sampled trajectories, and i is the current trajectory to be evaluated.

3.2. Improvement of DWA Algorithm

3.2.1. Improved DWA Algorithm Evaluation Function

In the traditional DWA (Dynamic Window Approach) algorithm, the evaluation function mainly considers the distance to obstacles, path distance, and target distance. However, the traditional DWA algorithm sometimes deviates too much from local planning. This article introduces a new trajectory point estimation function: "*trajcost*", to the original evaluation function, which can more comprehensively assess the quality of the path. This makes the planned path more in line with the actual motion characteristics of the robot and reduces the impact of known obstacles on the path. The evaluation function is as follows:

$$G(v,\omega) = \sigma(\alpha * heading(v,\omega) + \beta * dist(v,\omega) + \gamma * velocity(v,\omega) + \theta * trajcost(v,\omega)) \tag{15}$$

The trajectory point estimation function takes four parameters: *traj* (the predicted trajectory), *goal* (the target position), *ob* (the position of the obstacle), and *R* (the radius of the obstacle). Firstly, *trajcost* is initialized to 0, and then each point in *traj* is traversed. For each point, it calculates the heading evaluation value *heading* and the distance evaluation value *dist*, and then adds these two values to *trajcost*. Finally, the value in *trajcost* is divided by the number of points in *traj* to obtain the average cost. The pseudocode is as follows:

```
trajcost(traj, goal, ob, R){
    trajcost = 0;
    for (each point in traj){
        heading = calculate_heading(point, goal);
        dist = calculate_distance(point, ob);
        trajcost += heading + dist;
    }
    return trajcost / traj.size();
```

Its complexity and scalability largely depend on the implementation of the calculate_heading and calculate_distance functions. If these two functions can handle various types of targets and obstacles, such as calculating the distance from a point to complex geometric shapes in the calculate_distance function, then this algorithm can be applied to more complex environments. This paper does not cover obstacle avoidance for complex obstacles, mainly to simplify the experimental process and facilitate the analysis of results.

3.2.2. Optimization of B-Spline Curve Path

While the Dynamic Window Approach (DWA) is an effective path planning method, the paths it generates may not be smooth, which may not be ideal for the actual movement of robots. Literature [22] uses the Bezier curve method to optimize the generated path, but due to its large computational load and inability to modify locally, it does not perform well in complex environments with multiple obstacles. The B-spline curve method compensates for these shortcomings, so this article uses the B-spline curve method to smooth and optimize the path.

The B-spline curve is a linear combination of B-spline basis functions with a recursive formula:

$$B_{i,k}(u) = \begin{cases} \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & other \end{cases} & , \quad k = 1 \\ \frac{u-u_i}{u_{i+k-1}-u_i} B_{i,k-1}(u) + \frac{u_{i+k}-u}{u_{i+k}-u_{i+1}} B_{i+1,k-1}(u) & , \quad k \ge 2 \end{cases} \tag{16}$$

There are $n + 1$ nodes: P_0, P_1, P_2,..., P_n. The definition formula of B-spline curve is

$$P(u) = \begin{bmatrix} P_0 & P_1 & \cdots & P_n \end{bmatrix} \begin{bmatrix} B_{0,k}(u) \\ B_{1,k}(u) \\ \vdots \\ B_{n,k}(u) \end{bmatrix} = \sum_{i=0}^{n} P_i B_{i,k}(u) \tag{17}$$

Among them, $B_{i,k}(u)$ is the i-th k-order B-spline basis function. The higher the degree of the B-spline basis function, the higher the number of derivative functions of the curve, which will generate many zeros. More derivative zeros will lead to more extreme values in the original curve, resulting in more peak and valley values in the curve. The lower the degree of B-spline basis functions, the better the approximation of the spline curve to the control point. On the other hand, the cubic B-spline curve can achieve second-order derivative continuity, so this article selects the cubic B-spline curve as the curve for trajectory planning for smooth optimization.

3.2.3. Algorithm Fusion

After the optimization strategy mentioned above, the adaptive A-star algorithm can complete global path planning and find the optimal path in a static working environment. However, when unknown obstacles suddenly appear in the map environment, it cannot perform dynamic obstacle avoidance and local path planning. The improved DWA (Dynamic Window Approach) algorithm can enable robots to avoid obstacles dynamically, but because it cannot perform global planning, it cannot guarantee obtaining the optimal path solution. To complement the advantages, the two algorithms are integrated. The DWA algorithm is used for local planning between two adjacent nodes in the path planned by the adaptive A-star algorithm. The next node in the current path is initialized, the speed is sampled, and the latest node is continuously updated to find the optimal path. This allows the robot to have both global path planning and local dynamic obstacle avoidance capabilities. The flowchart of the integrated algorithm is shown in Figure 5.

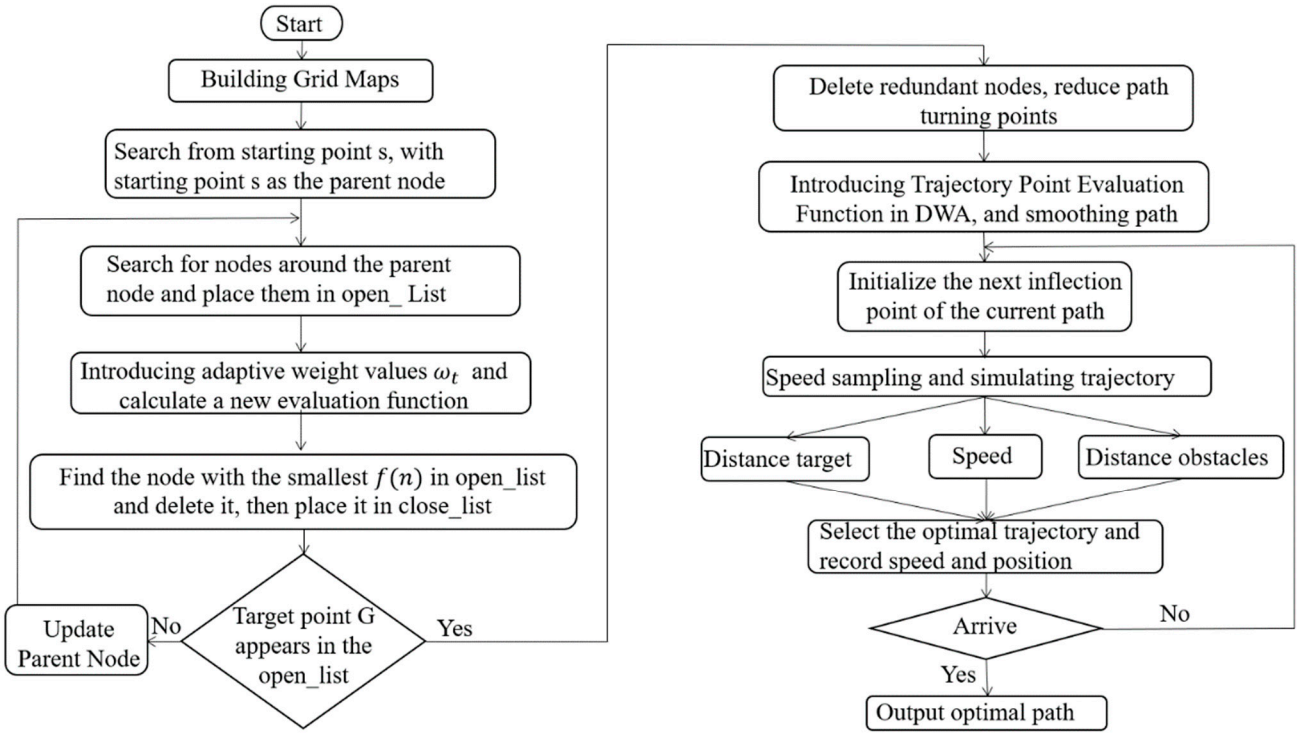


Figure 5. Flow chart of fusion algorithm.

4. Simulation and Real Experiment Verification

4.1. Simulation Verification

4.1.1. Simulation Experiment of Adaptive A-Star Algorithm

To validate the effectiveness of the adaptive A-star algorithm, a simulation experiment was conducted in MATLAB R2022a using a 30 × 30 multi-obstacle map constructed by the grid method. The start point is (5, 2), the end point is (23, 25), black squares represent obstacles, white squares represent free space, and each grid side length is set to 1 m.

Simulation experiments were conducted on the traditional A-star algorithm, algorithm [22], algorithm [23], and the adaptive A-star algorithm proposed in this article, and the data were exported for comparison, as shown in Figures 6 and 7 and Table 1.

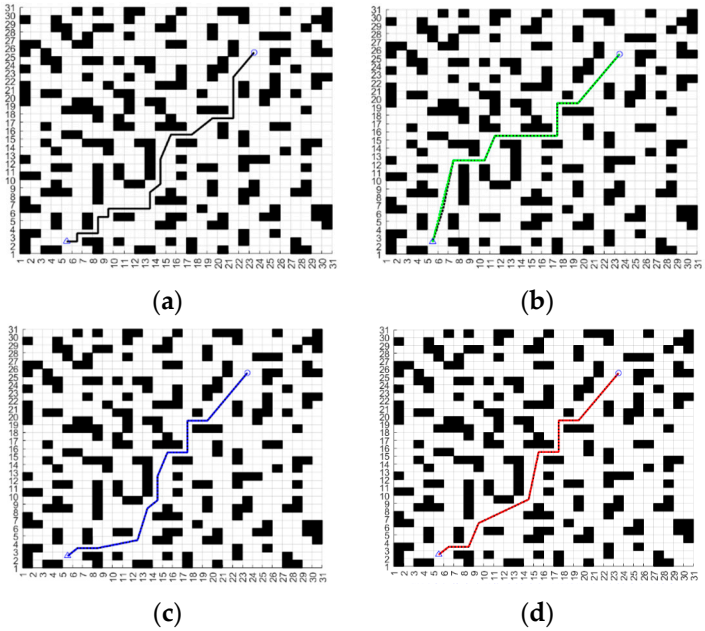


Figure 6. Comparison of algorithm simulation paths: (**a**) traditional A-star algorithm; (**b**) Reference [22] algorithm; (**c**) Reference [23] algorithm; (**d**) adaptive A-star algorithm.

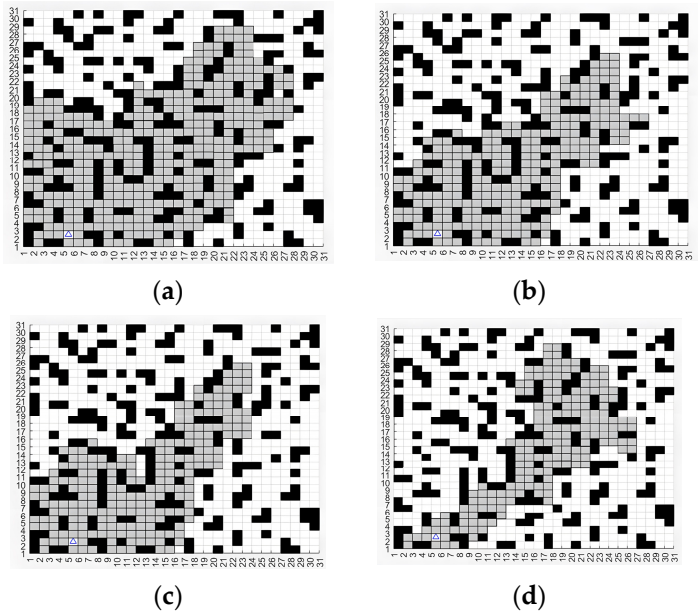


Figure 7. Comparison of algorithm simulation traverse nodes: (**a**) traditional A-star algorithm; (**b**) Reference [22] algorithm; (**c**) Reference [23] algorithm; (**d**) adaptive A-star algorithm.

From the simulation results, it can be seen that the adaptive A-star algorithm proposed in this article reduces the search time by 43.4% compared to the traditional A-star algorithm, reduces the traversed nodes by 51.8%, reduces the path length by 17.9%, reduces the turning angle by 53.5%, and reduces the number of path turning points by 46.7%. Compared to algorithm [22], the adaptive A-star algorithm proposed in this article reduces the search time by 34.6%, reduces the traversed nodes by 24%, reduces the path length by 8.2%, increases the turning angle by 22.2%, and increases the number of path turning points

by 33.3%. Compared to algorithm [23], the adaptive A-star algorithm proposed in this article reduces the search time by 4.1%, reduces the traversed nodes by 17.6%, reduces the path length by 5.1%, reduces the turning angle by 15.4%, and reduces the number of path turning points by 20%.

Table 1. Comparison of algorithm simulation experiment data.

Algorithm	Search Time	Search Nodes	Length	Turning Angle	Turning Point
Traditional A-star	2.51 s	369	41.05 m	1065°	15
Reference [22]	2.17 s	244	36.72 m	405°	6
Reference [23]	1.48 s	216	35.51 m	585°	10
Adaptive A-star	1.42 s	178	33.70 m	495°	8

4.1.2. Simulation Experiment of Improved DWA Algorithm

Section 3 introduces an improvement to the DWA algorithm by adding a fourth function, the trajectory point evaluation function (trajcost), to the original three evaluation functions (heading, dist, velocity). In actual operation, it is necessary to optimize and normalize the parameters (α,β,γ,θ) of these four evaluation functions. By comparing the time of the DWA algorithm's local path planning under different values of α,β,γ,θ, the purpose of comparative analysis can be achieved.

(1) Analysis of α and γ parameters

Based on engineering experience, set β = 0.3 and conduct MATLAB simulation experiments, respectively, in α = (0.02~0.1) and γ = (0.05~0.3). In the figure, the black squares represent obstacles, the circles represent the robot, and the blue curve represents the robot's trajectory. In the first experimental map, the starting point is (0, 0), and the target point is (10, 8); in the second experimental map, the starting point is (0, 0), and the target point is (10, 4). The simulation experiment results and the bar chart analysis are shown in Figures 8 and 9.

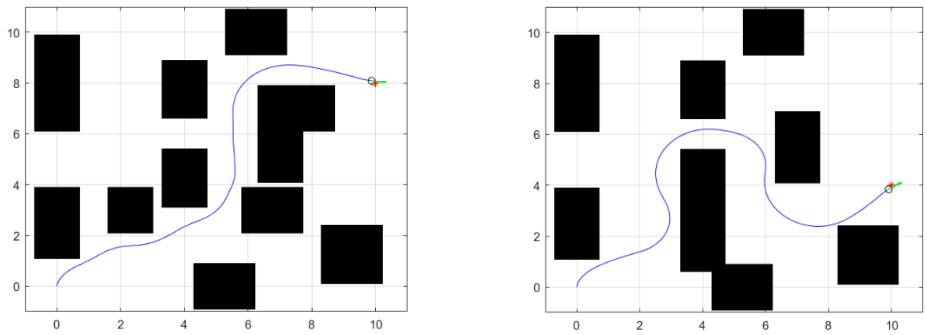


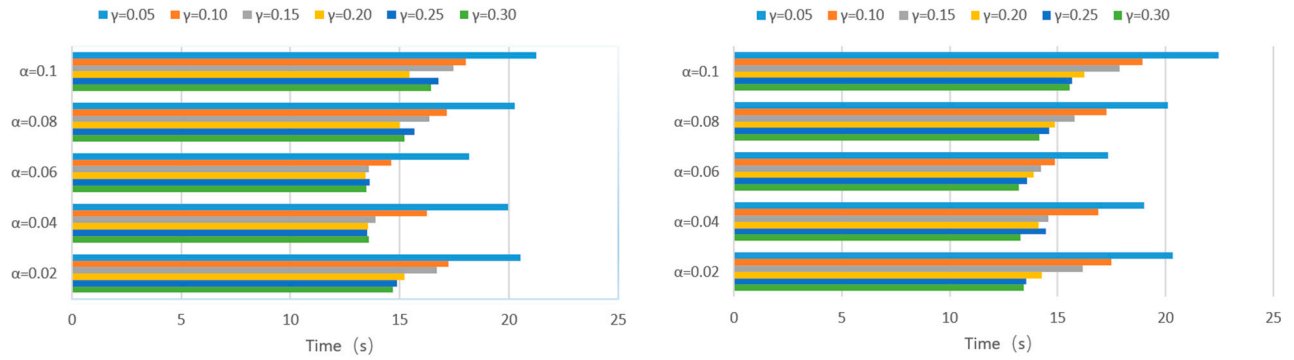
Figure 8. Two sets of experimental maps: (**a**) the first experimental map; (**b**) the second experimental map.

From the analysis of the experimental results in Figure 9, it can be seen that when β = 0.3 and α is certain, the path planning time decreases as γ increases from 0.05 to 0.15, and the path planning time is very close when γ is between 0.15 and 0.30. When γ is certain, the path planning time decreases as α increases from 0.02 to 0.06 and increases as α increases from 0.06 to 0.10.

(2) Analysis of β parameters

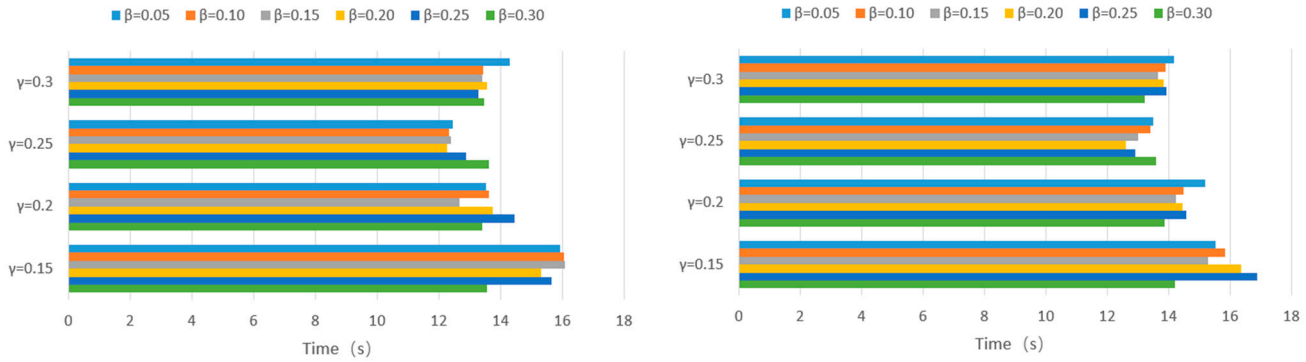
Based on the conclusions drawn from the analysis of α and γ parameters, that is, when α = 0.06 and γ is in the range of 0.15 to 0.3, we analyze the impact of changes in β (ranging from 0.05 to 0.3) on the path planning time of the DWA algorithm. The experimental

maps still use the two sets of maps in Figure 9, and the experimental results are shown in Figure 10.



(**a**) (**b**)

Figure 9. Analysis of α and γ parameters: (**a**) results of the first set of experiments; (**b**) results of the second set of experiments.



(**a**) (**b**)

Figure 10. Analysis of β parameters: (**a**) results of the first set of experiments; (**b**) results of the second set of experiments.

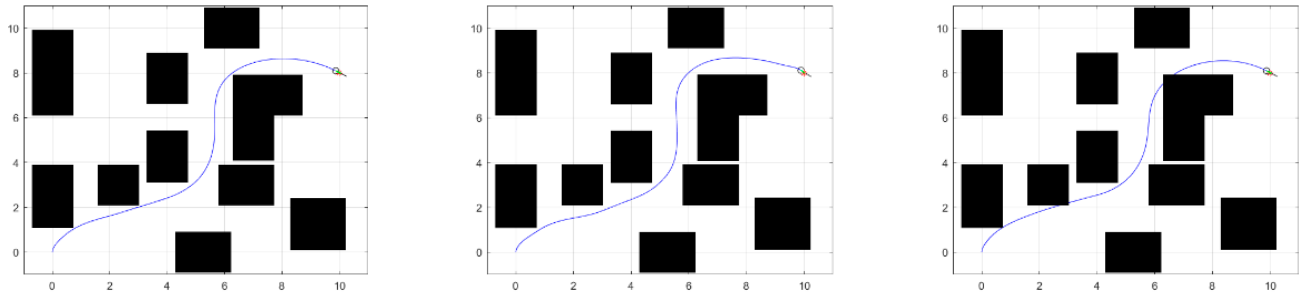
As can be seen from Figure 10, experiments were conducted under the conditions of α = 0.06, β ranging from 0.05 to 0.3, and γ ranging from 0.15 to 0.3. The navigation efficiency is optimal when α = 0.06, β = 0.20, and γ = 0.25.

(3) Analysis of θ parameters

We introduce a new trajectory point evaluation function to the original evaluation functions of the DWA algorithm. Hence, it is also necessary to analyze the appropriate value of the parameter θ. Still using the two sets of maps from the previous experiments, based on the results of the previous two sets of analysis experiments, when α = 0.06, β = 0.20, and γ = 0.25, we analyze the impact of the parameter θ on the path planning and navigation time of the DWA algorithm.

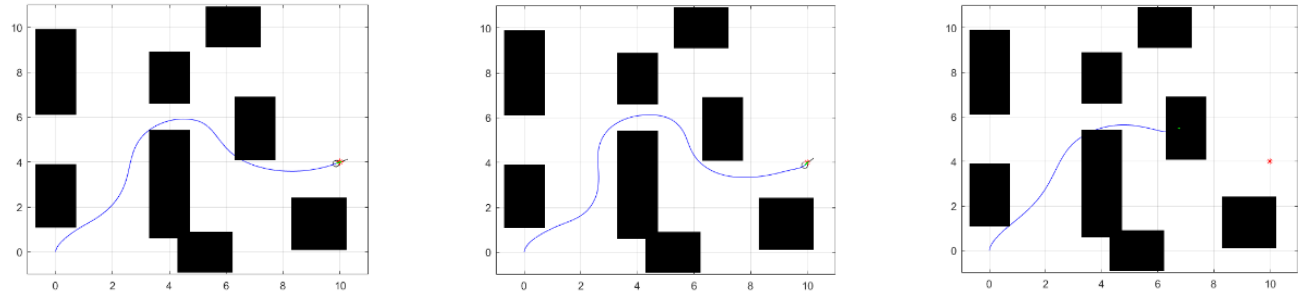
Experiments were conducted on two sets of maps, comparing and analyzing the impact of the parameter θ on the robot's path planning. The results are shown in Figures 11 and 12.

From the analysis of Figures 11 and 12 and Table 2, it can be seen that under two different sets of maps, when θ = 2, the robot's travel time is the shortest, and no collisions occurred, ultimately finding the destination. To further verify the effectiveness of the newly added trajectory point evaluation function in the original DWA evaluation function, experiments were conducted before and after the addition of the trajectory point evaluation function when α = 0.06, β = 0.20, and γ = 0.25. The results are shown in Figures 13 and 14.



(**a**) (**b**) (**c**)

Figure 11. Analysis of parameter θ in the first set of map experiments: (**a**) results of $\theta = 1$; (**b**) results of $\theta = 2$; (**c**) results of $\theta = 3$.

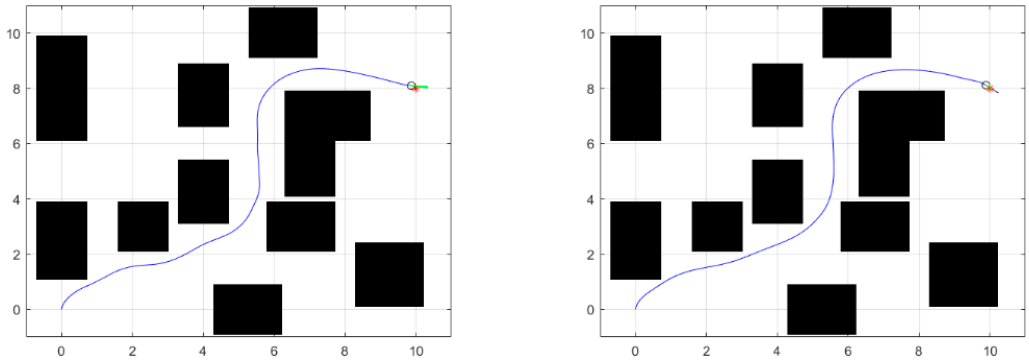


(**a**) (**b**) (**c**)

Figure 12. Analysis of parameter θ in the second set of map experiments: (**a**) results of $\theta = 1$; (**b**) results of $\theta = 2$; (**c**) results of $\theta = 3$.

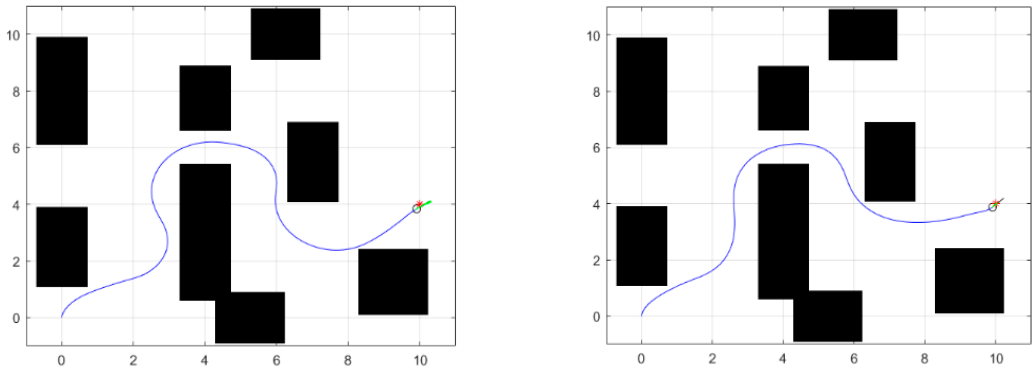
Table 2. Comparison of Simulation Experiment Data for θ Parameter Analysis.

	θ	Time (s)	Obstacle Collision	Reach Destination
Map 1	1	15.33	No	Yes
	2	11.51	No	Yes
	3	21.19	Yes	Yes
Map 2	1	19.27	Yes	Yes
	2	11.89	No	Yes
	3	No data	Yes	No



(**a**) (**b**)

Figure 13. Comparison of simulation experiments before and after the improvement of the DWA algorithm in the first map: (**a**) traditional DWA; (**b**) improved DWA.



(**a**) (**b**)

Figure 14. Comparison of simulation experiments before and after the improvement of the DWA algorithm in the second map: (**a**) traditional DWA; (**b**) improved DWA.

As can be seen from Table 2 and Figures 13 and 14, the improved DWA algorithm, which adds a new trajectory point evaluation function to the original evaluation functions of the DWA algorithm, has a smoother path and shorter navigation time compared to before the improvement, with an average reduction of 6%.

4.1.3. Simulation Experiment of Fusion Algorithm

To verify the local dynamic obstacle avoidance ability of the fusion algorithm, 1–4 obstacles were added to the 30 × 30 grid map constructed in the previous section. New obstacles were represented by gray squares, the triangle represented the starting point, the circle represented the end point, and the obstacle avoidance of the mobile robot was observed.

Figure 15 shows the running situation of the mobile robot using the fusion algorithm proposed in this article and the Reference [24] fusion algorithm when 1–4 obstacles were added. Table 3 lists the parameters of the DWA algorithm, and Table 4 compares the simulation experimental data of the fusion algorithm proposed in this article and the Reference [24] fusion algorithm under different random obstacle numbers.

Table 3. DWA algorithm parameter settings.

Parameter	Value
$v_{max}/(\mathrm{m/s})$	1
$\omega_{max}/(\mathrm{rad/s})$	2π
$a_v/(\mathrm{m/s})$	0.03
$a_\omega/(\mathrm{rad/s})$	$\pi/24$
$a_{v_{max}}/(\mathrm{m/s^2})$	0.4
$a_{\omega_{max}}/(\mathrm{rad/s^2})$	5π
$\Delta t/\mathrm{s}$	0.1

Table 4. Comparison of fusion algorithm simulation experimental data.

Obstacle	Algorithm	Travel Time	Length	Angle	Collision
One obstacle	This article	199.66 s	29.52 m	216°	no
	Reference [24]	209.89 s	29.49 m	243°	yes
Two obstacles	This article	201.06 s	29.67 m	248°	no
	Reference [24]	214.43 s	29.63 m	275°	no
Three obstacles	This article	204.71 s	29.97 m	281°	no
	Reference [24]	218.39 s	29.68 m	308°	no
Four obstacles	This article	206.49 s	30.36 m	309°	no
	Reference [24]	222.01 s	29.74 m	332°	no

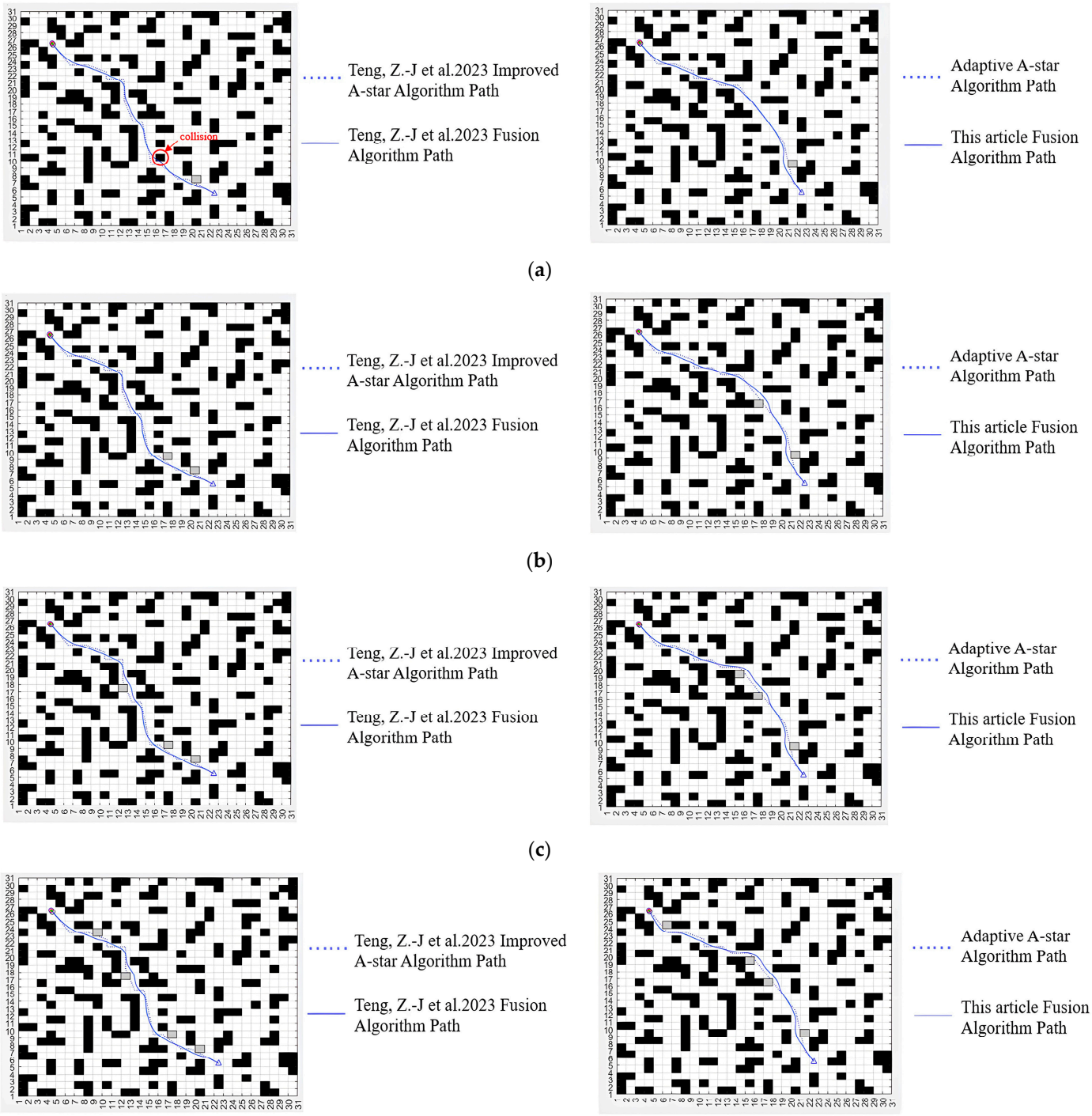


Figure 15. Comparison between the fusion algorithm in Reference [24] and the simulation path of the fusion algorithm in this article after adding obstacles: (**a**) comparison of paths between Reference [24] and this article when randomly adding an obstacle; (**b**) comparison of paths between Reference [24] and this article when randomly adding two obstacles; (**c**) comparison of paths between Reference [24] and this article when randomly adding three obstacles; (**d**) comparison of paths between Reference [24] and this article when randomly adding four obstacles.

The simulation results in Table 4 show that the path length traveled by the algorithm in Reference [24] is slightly shorter, and with the addition of obstacles one by one, the path length does not increase significantly. However, when an obstacle is added, there

is occasional collision during travel, and the travel time is longer. Although the fusion algorithm in this article has a slightly longer path after adding obstacles, its turn angle is smaller than that of the algorithm in literature [24], resulting in an average travel time reduction of 6.1% compared to the algorithm in literature [24] without collision. This indicates that the fusion algorithm in this article has better path search performance and can quickly plan the adequate travel path in the presence of random obstacles, ensuring better safety for the robot.

4.2. Real Experiment Verification

In order to verify the global path planning and dynamic obstacle avoidance effect of the fusion algorithm in a real environment, a Blue Whale Xiaoqiang XQ-5 mobile robot was selected as an experimental platform for experimental verification. The physical structure and experimental site of the robot are shown in Figure 16. The hardware infrastructure on the robot includes a host with a basic configuration of Intel I7 4560U, 64 GB of solid-state hard drive, 8 GB of memory, a pre-installed remote control, and an Ubuntu 16.04 operating system. The host also includes two drive motors, a 1080p USB camera, a lidar, two ultrasonic sensors, a bottom drive board, and a 9-axis sensor with mpu9250. To facilitate the debugging of the robot, on the premise of ensuring that the robot and the local remote control end are in a unified local area network, a VNC Server 6.7 remote desktop software is installed on the local PC end to enable remote control of the robot. The kinematics parameters of the robot are shown in Table 5.

(**a**) (**b**)

Figure 16. Robot physical object and experimental site map: (**a**) robot physical image; (**b**) experimental site map.

Table 5. Robot kinematic parameters.

Coordinate Axis	v_{max}/(m·s^{-1})	ω_{max}/(rad·s^{-1})	Sampling Time/(s)
X	0.4	$\pi/6$	0.3
Y	0.4	$\pi/6$	0.3

Open the VNC remote control software on the local PC end, input the IP address of the robot's current local area network to connect, start Gazebo, open the Rviz visualization tool, and open a new terminal on the local end, input "rosrun teleop_twist_keyboard teleop_twist_keyboard.py" to opens the keyboard control node, controls the mobile robot through the keyboard for SLAM mapping, and uses the map_ The save function package saves the map and exports the experimental site map as shown in Figure 17.

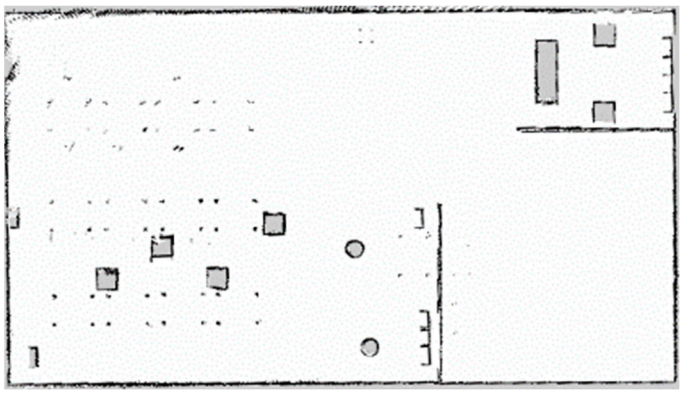
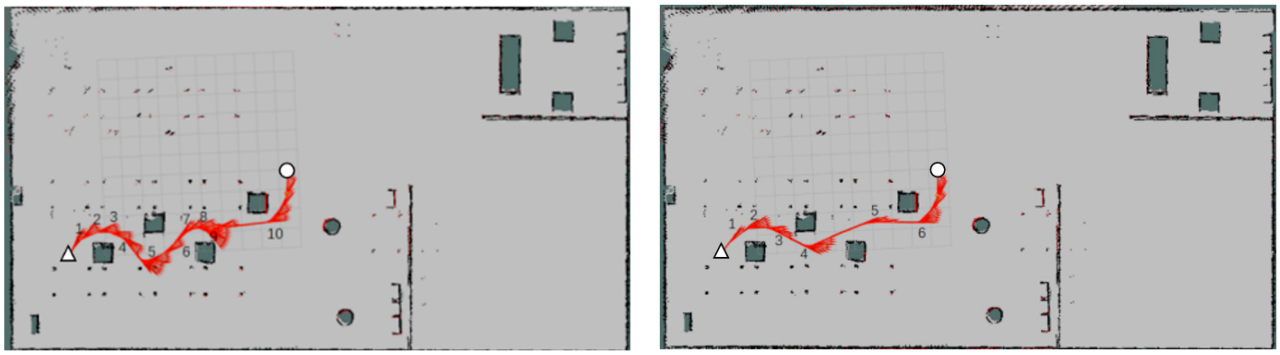


Figure 17. Map of the experimental site.

4.2.1. Experimental Verification of Adaptive A-Star Algorithm Navigation

In the experiment, the robot's navigation is completely autonomous, which is determined by the fusion algorithm of this article. The robot's navigation is controlled by a PID controller based on the path calculated by the algorithm. A position feedback mechanism based on the odometer is used, which accurately measures the robot's displacement through the encoder and feeds this information back to the controller for precise path tracking.

Run "map_server" function pack to read the map, start the chassis control node, post topics, and open the Rviz visualization interface to select target points. Experimental verification of path planning is conducted on the traditional A-star algorithm and the adaptive A-star algorithm, respectively. Figure 18 shows a comparison of navigation experimental paths between two algorithms, with a white triangle as the starting point and a white circle as the target point, and Table 6 compares the average experimental data from repeating the experiment five times with the traditional A-star algorithm and the adaptive A-star algorithm.



(**a**) (**b**)

Figure 18. Navigation Effect of traditional A-star algorithm and adaptive A-star algorithm: (**a**) traditional A-star algorithm navigation experiment; (**b**) adaptive A-star algorithm navigation experiment.

From the comparison of navigation data in Table 6, it can be seen that the traditional A-star algorithm has many corners and large turning angles, and the path is relatively long and not smooth. By repeating the experiment five times and compared to the traditional A-star algorithm, the adaptive A-star algorithm average reduces the path length by 22%, the number of turns by 43.1%, the steering angle by 35.7%, and the driving time by 23.7%, verifying the effectiveness of the proposed adaptive A-star algorithm.

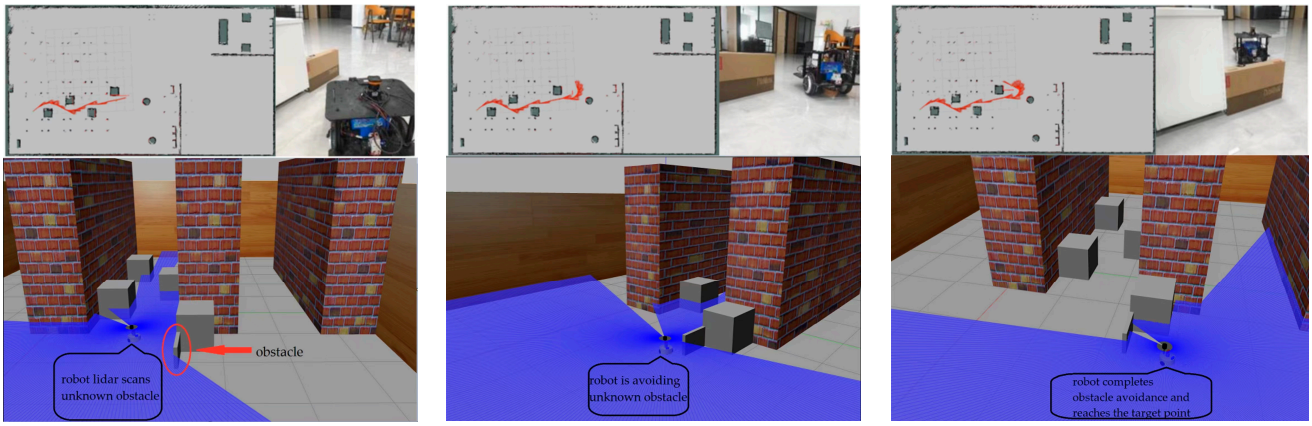
Table 6. Comparison of navigation data between A-star algorithm and adaptive A-star algorithm.

	Length	Turning Point	Turning Angle	Travel Time
Traditional A-star	15.39 m	10	415.8°	51.7 s
	14.91 m	9	370.9°	46.1 s
	15.46 m	10	416.2°	51.9 s
	16.38 m	11	433.5°	53.6 s
	16.26 m	11	431.6°	53.2 s
Average data	15.68 m	10.2	413.6°	51.3 s
Adaptive A-star	12.65 m	6	274.9°	40.6 s
	11.54 m	5	250.8°	37.0 s
	12.45 m	6	269.3°	39.8 s
	12.52 m	6	271.1°	40.1 s
	11.99 m	6	262.9°	38.5 s
Average data	12.23 m	5.8	265.8°	39.2 s

4.2.2. Experimental Verification of Fusion Algorithm Navigation

Based on the above experiments, while ensuring that the starting point and target point remain unchanged, add an unknown obstacle on the planned path, as shown in Figure 19. Additionally, establish a Gazebo simulation environment in the ROS system to simulate the experimental site map settings as closely as possible and verify the feasibility of the simulation and actual experiment in real time. Start the fusion algorithm, and the results are shown in Figure 20.

Figure 19. Adding unknown obstacle.



(**a**) (**b**) (**c**)

Figure 20. Fusion algorithm navigation effect: (**a**) robot lidar scans unknown obstacle; (**b**) robot is avoiding unknown obstacle; (**c**) robot completes obstacle avoidance and reaches the target point.

From Figure 20, it can be seen that the fusion algorithm can automatically plan new paths to avoid unknown obstacles and achieve dynamic obstacle avoidance while ensuring that the re-planned path is close to the static optimal path of the adaptive A-star algorithm.

5. Discussion

This algorithm effectively improves the shortcomings of the A-star algorithm, reduces the number of traversal nodes and running time, shortens the path length and turning points, and improves running efficiency. Moreover, it integrates with the improved DWA algorithm to achieve autonomous obstacle avoidance navigation for mobile robots. However, the application scope of this algorithm has limitations; that is, the fusion algorithm in this article is only suitable for path planning of AGVs in factory environments, and its application in small space home environments, such as sweeping robots, is not effective. Because in a factory environment, due to the relatively spacious space and constant environment, algorithms can accurately locate and navigate using prior map information. In smaller home environments, due to the small and complex space, the robot moves relatively slowly, and the length of the path planned each time is not very large, making it difficult to reflect the effect of the improved algorithm in this paper on optimizing navigation efficiency. Moreover, the algorithm in this paper does not take into account the avoidance of dynamic obstacles and is not suitable for areas with moving obstacles. Future research can improve the robot's perception capabilities by optimizing algorithm parameters, adding multiple sensors, and including experiments on dynamic obstacle avoidance to broaden the application range of the algorithm.

6. Conclusions

This article proposes a path planning method that integrates an adaptive A-star algorithm with an improved DWA algorithm to address the issues of the traditional A-star algorithm, such as excessive node traversal, long search time, unsmooth paths, close proximity to obstacles, and applicability only to static maps. The adaptive A-star algorithm is based on adaptively weighting the heuristic function of the traditional A-star algorithm and introducing the Douglas–Pucker thinning algorithm to eliminate redundant points. Furthermore, a trajectory point evaluation function is added to the evaluation functions of the DWA algorithm, and the B-spline curve method is introduced for path smoothing optimization. Finally, the adaptive A-star algorithm and the improved DWA algorithm are integrated to implement dynamic obstacle avoidance. Through simulation experiments, the adaptive A-star algorithm reduces the search time by 43.4%, the number of traversed nodes by 51.8%, the path length by 17.9%, the turning angle by 53.5%, and the number of turning nodes by 46.7% compared to the traditional A-star algorithm. By comparing the effects of real experiments, the adaptive A-star algorithm reduces the path length by 22%, the number of turning nodes by 43.1%, and the turning angle by 35.7% compared to the traditional A-star algorithm. This algorithm effectively improves the shortcomings of the A-star algorithm, reduces the number of traversed nodes and running time, shortens the path length and the number of path turning points, and improves the running efficiency. Overall, this article provides a promising and effective solution for enhancing the navigation efficiency of autonomous mobile robots, which has potential implications for robotics applications.

Author Contributions: T.L. is responsible for literature review, experimental design, and writing of this article; J.G. served as the mentor, providing guidance on techniques and writing methodologies; T.L., Y.W., F.C., H.Z. and S.O. collaborated in conducting experiments and recording data. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Key Project of Natural Science Foundation of Fujian Province No. 2020J02048, and Xiamen Municipal Natural Science Foundation 3502Z20227215, and Xiamen Ocean and Fisheries Development Special Fund Youth Science and Technology Innovation Project 23ZHZB043QCB37.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, C.; Cheng, C.; Yang, D.; Pan, G.; Zhang, F. Path Planning in Localization Uncertaining Environment Based on Dijkstra Method. *Front. Neurorobot.* **2022**, *16*, 821991. [CrossRef] [PubMed]
2. Gong, H.; Ni, C.; Wang, P.; Cheng, N. A Smooth path planning method based on Dijkstra algorithm. *J. Beijing Univ. Aeronaut. Astronaut.* **2023**, 1–10. [CrossRef]
3. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 172988142096226. [CrossRef]
4. Tang, G.; Tang, C.; Claramunt, C.; Hu, X.; Zhou, P. Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. *IEEE Access* **2021**, *9*, 59196–59210. [CrossRef]
5. Zhang, J.; Wu, J.; Shen, X.; Li, Y. Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 172988142110427. [CrossRef]
6. Fu, Q.; Lan, X.-H.; Ren, F.-H.; Wu, J.-H. Research on Route Planning Algorithm of Bidirectional Target RRT and Dijkstra. *Comput. Simul.* **2023**, *40*, 447–454+461.
7. Zhang, L.; Shi, X.; Yi, Y.; Tang, L.; Peng, J.; Zou, J. Mobile Robot Path Planning Algorithm Based on RRT_Connect. *Electronics* **2023**, *12*, 2456. [CrossRef]
8. Zhou, L.; Wu, N.; Chen, H.; Wu, Q.; Lu, Y. RRT*-Fuzzy Dynamic Window Approach (RRT*-FDWA) for Collision-Free Path Planning. *Appl. Sci.* **2023**, *13*, 5234. [CrossRef]
9. Luo, J.; Wang, Z.X.; Pan, K.-L. Reliable Path Planning Algorithm Based on Improved Artificial Potential Field Method. *IEEE Access* **2022**, *10*, 108276–108284. [CrossRef]
10. Zheng, L.; Yu, W.; Li, G.; Qin, G.; Luo, Y. Particle Swarm Algorithm Path-Planning Method for Mobile Robots Based on Artificial Potential Fields. *Sensors* **2023**, *23*, 6082. [CrossRef]
11. Xiong, Y.; Ping, C.; Kangwen, Z.; Zhang, Y.; Zhou, Q.; Yao, D.-J. Dynamic Path Planning of AGV Based on Kinematical Constraint A-Star Algorithm and Following DWA Fusion Algorithms. *Sensors* **2023**, *23*, 4102. [CrossRef]
12. Owais, M.; Alshehri, A. Pareto Optimal Path Generation Algorithm in Stochastic Transportation Networks. *IEEE Access* **2020**, *8*, 58970–58981. [CrossRef]
13. Owais, M.; Osman, M.K. Complete Hierarchical Multi-Objective Genetic Algorithm for Transit Network Design Problem. *Expert Syst. Appl.* **2018**, *114*, 143–154. [CrossRef]
14. Yang, D.; Xu, B.; Rao, K.; Sheng, W. Passive Infrared (PIR)-Based Indoor Position Tracking for Smart Homes Using Accessibility Maps and A-Star Algorithm. *Sensors* **2018**, *18*, 332. [CrossRef]
15. Wang, P.; Liu, Y.; Yao, W.; Yu, Y. Improved A-star algorithm based on multivariate fusion heuristic function for autonomous driving path planning. *Inst. Mech.Eng. Part D J. Automob. Eng.* **2023**, *237*, 1527–1542. [CrossRef]
16. Pereira, F.U.; Brasil, P.M.d.A.; Cuadros, M.A.D.S.L.; Cukla, A.R.; Junior, P.D.; Gamarra, D.F.T. Analysis of Local Trajectory Planners for Mobile Robot with Robot Operating System. *IEEE Lat. Am. Trans.* **2022**, *20*, 92–99. [CrossRef]
17. Muhammad, A.; Ali, M.-A.-H.; Turaev, S.; Abdulghafor, R.; Shanono, I.-H.; Alzaid, Z.; Alruban, A.; Alabdan, R.; Dutta, A.-K.; Almotairi, S. A Generalized Laser Simulator Algorithm for Mobile Robot Path Planning with Obstacle Avoidance. *Sensors* **2022**, *22*, 8177. [CrossRef]
18. Bulut, V. Optimal path planning method based on epsilon-greedy Q-learning algorithm. *J. Braz. Soc. Mech. Sci. Eng.* **2022**, *44*, 106. [CrossRef]
19. Da-Silva-Costa, L.; Tonidandel, F. DVG+A-star and RRT Path-Planners: A Comparison in a Highly Dynamic Environment. *J. Intell. Robot. Syst.* **2021**, *101*, 58. [CrossRef]
20. Wang, H.; Lou, S.; Jing, J.; Wang, Y.; Liu, W.; Liu, T. The EBS-A* Algorithm: An Improved A* Algorithm for Path Planning. *PLoS ONE* **2022**, *17*, e0263841. [CrossRef]
21. Jing, X.; Yang, X. Application and Improvement of Heuristic Function in A* Algorithm. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; Volume 7, pp. 2191–2194. [CrossRef]
22. Li, S.-D.; Zhen, Y.-F.; Lu, N.; Li, S.-Y.; Qi, Y.-F. Smooth Path Planning based on Directed Search A-star Algorithm. *J. Dalian Jiaotong Univ.* **2022**, *43*, 5. [CrossRef]
23. Wei, Y.; Jin, F.; Dong, K.-F.; Song, J.-L.; Mo, W.-Q.; Hui, Y.-J. Improved Global Path Planning A* Algorithm Based on Node Optimization. *Comput. Meas. Control* **2023**, *31*, 143–148. [CrossRef]
24. Teng, Z.-J.; Mao, J.-Z. Research on Path Planning Algorithm Combining Improved A-Star and Dynamic Window Approach. *Mech. Sci. Technol. Aerosp. Eng.* **2023**, 1–13. [CrossRef]
25. Wang, X.; Ma, X.; Li, Z. Research on SLAM and Path Planning Method of Inspection Robot in Complex Scenarios. *Electronics* **2023**, *12*, 2178. [CrossRef]
26. Xiao, Y.-X.; Hai, T.-M.; Yuan, B.-Y.; Peng, Y.-W. Application improvement of A-star algorithm in intelligent vehicle trajectory planning. *Math. Biosci. Eng.* **2021**, *18*, 1–21. [CrossRef]

27. Liu, Z.; Liu, H.; Lu, Z.; Zeng, Q. A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-Star for Mobile Robots. *IEEE Access* **2021**, *9*, 20602–20621. [CrossRef]
28. Zhang, L.; Zhang, Y.; Li, Y. Mobile Robot Path Planning Based on Improved Localized Particle Swarm Optimization. *IEEE Sens. J.* **2021**, *21*, 6962–6972. [CrossRef]
29. Yu, X.; Jiang, C.; Duan, S.-R.; Deng, Q.-R. Improving path planning with A-star algorithm and APF algorithm. *J. Syst. Simul.* **2023**, 1–12. [CrossRef]
30. Hong, Z.; Sun, P.; Tong, X.; Pan, H.; Zhou, R.; Zhang, Y.; Han, Y.; Wang, J.; Yang, S.; Xu, L. Improved A-Star Algorithm for Long-Distance Off-Road Path Planning Using Terrain Data Map. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 785. [CrossRef]
31. Liu, H.; Zhang, Y. ASL-DWA: An Improved A-Star Algorithm for Indoor Cleaning Robots. *IEEE Access* **2022**, *10*, 99498–99515. [CrossRef]