mathematics

MDPI

Article

Mobile Robot Path Planning Based on Kinematically Constrained A-Star Algorithm and DWA Fusion Algorithm

Yanjie Liu, Chao Wang *, Heng Wu and Yanlong Wei

State Key Laboratory of Robotics and System, Department of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China; yjliu@hit.edu.cn (Y.L.); 20b908039@stu.hit.edu.cn (H.W.); yanlongwei@hit.edu.cn (Y.W.)
* Correspondence: 21b908026@stu.hit.edu.cn

Abstract: Path-planning research has been the key to mobile-robot-navigation technology. However, traditional path-planning algorithms have some shortcomings. To solve these problems, this paper proposes a fusion algorithm that combines the kinematical constrained A* algorithm with the Dynamic Window Approach (DWA) algorithm. The kinematical constrained A* algorithm can plan the global path, and then the DWA algorithm can plan the local path under the global path's guidance. Firstly, combined with robot kinematics, we improve the node-expansion method and heuristic-function model of the A* algorithm, which improves the search efficiency, reduces the number of path bends and lowers the computational cost so that the path generated by the A* algorithm better meets the needs of robot motion. Secondly, we optimize the trajectory-evaluation function of the DWA algorithm so that the local paths planned by the DWA algorithm are smoother and more coherent, which is easier for robot-motion execution. Finally, we extract the key nodes from the global path planned by the A* algorithm to guide the DWA algorithm for local path planning and dynamic-obstacle avoidance and to make the local path closer to the global path. Through simulation and practical experiments, the effectiveness of the fusion algorithm proposed in this paper is verified.

Keywords: path planning; mobile robot; A-star; DWA; path generation

MSC: 70E60

Citation: Liu, Y.; Wang, C.; Wu, H.; Wei, Y. Mobile Robot Path Planning Based on Kinematically Constrained A-Star Algorithm and DWA Fusion Algorithm. *Mathematics* **2023**, *11*, 4552. https://doi.org/10.3390/math11214552

Academic Editor: Daniel-Ioan Curiac

Received: 8 October 2023
Revised: 29 October 2023
Accepted: 3 November 2023
Published: 5 November 2023

1. Introduction

At present, whether they are indoor logistics-handling robots or outdoor substation-inspection robots, their autonomous movement cannot be separated from SLAM navigation [1,2]. In the SLAM navigation process, SLAM (Simultaneous Localization and Mapping) can help the robot determine its position in an unknown environment and build the corresponding environment map, and path planning can help the robot plan its movement route on the basis of SLAM. Path planning is the key for mobile robots to realize autonomous navigation, which mainly refers to the fact that in an environment with obstacles, mobile robots can find a suitable collision-free path from the starting point to the target point according to certain evaluation criteria [3,4]. Mobile robots are typically used in complex environments where static and dynamic obstacles are present, and the path-planning results have a direct impact on the efficiency and safety of the robot [5]. Mobile robots' path-planning techniques are mainly categorized into global path planning and local path planning [6].

Global path planning is the offline planning of a collision-free path from a start point to a goal point by a mobile robot in a known environment [7]. However, mobile robots cannot work in a completely known environment, and global path planning alone cannot deal with unknown obstacles and environmental information in real time, so it is necessary to help robots avoid collisions with dynamic obstacles with the help of local path planning [8]. Local path planning does not rely on environmental information and utilizes sensors to

obtain information about the robot's surroundings for path planning, which is highly in real time and enables mobile robots to perform dynamic-obstacle avoidance [9]. However, local path planning lacks the guidance of global information, which cannot guarantee that the planned path is the optimal path, and it is easy to fall into the local optimum [10].

Classical global path-planning methods include the Dijkstra algorithm [11], A* algorithm [12], Ant Colony Optimization (ACO) [13], Particle Swarm Optimization (PSO) [14], Rapidly Exploring Random Tree (RRT) [15] and so on. Among these algorithms, the A* algorithm is considered one of the most efficient algorithms for solving the shortest paths in static maps due to its search efficiency [16].

However, using the A* algorithm for path planning in complex environments with unknown static and dynamic obstacles requires more search nodes and generates folded paths, which reduce the efficiency of path planning [17]. In addition, the A* algorithm is unable to realize dynamic-obstacle avoidance. To solve these problems, this paper proposes a fusion algorithm that combines the kinematical constrained A* algorithm and the Dynamic Window Approach (DWA) algorithm. Global path planning is computationally more efficient with fewer turns. Local path planning is more satisfying regarding the robot's motion needs, and the local path is closer to the global path. In addition, the robot has a local dynamic-obstacle-avoidance capability.

In short, the fusion algorithm is proposed to achieve a mobile robot that can both travel through globally optimal paths and avoid local dynamic obstacles. We make three contributions compared to the existing research:

(1) Improve the A* algorithm node-expansion method and heuristic-function model to improve the A* algorithm's search efficiency, reduce the number of A* algorithm path bends and reduce the A* algorithm computational cost, so as to make the generated path closer to the robot kinematics.
(2) Optimize the trajectory-evaluation function of the DWA algorithm so that the local path planned by the DWA algorithm is more reasonable and more in line with the needs of robot motion.
(3) Extract key nodes from the global path planned by the A* algorithm and guide the DWA algorithm for local path planning and dynamic-obstacle avoidance so that the local path is closer to the global path and the robot has a dynamic-obstacle-avoidance function.

The outline of this paper is as follows. In Section 2, the classification of path planning and the main research methods in recent years are discussed. In Section 3, the principles of the traditional A* algorithm and the improvements we made to the A* algorithm are presented. In Section 4, the process of optimizing the parameters of our DWA trajectory-evaluation function is described. In Section 5, we validate the algorithm proposed in this paper through simulations and practical experiments. Finally, in Section 6, the research work of this paper is summarized and interpreted.

2. Related Work

In recent years, with the rapid development of robotics technology, the application areas of mobile robots are expanding [18]. At the same time, the increasing demand for complex tasks puts higher requirements on robot path-planning technology. However, the A* algorithm, which is the most widely used algorithm in the field of path planning, shows many shortcomings, such as a low search efficiency and inability to perform local dynamic-obstacle avoidance. An improved A* algorithm has been studied.

The bidirectional alternating A* algorithm expands both the initial and target nodes, thus increasing the search and convergence speed of the A* algorithm [19]. The jump point search algorithm retains the main framework of the A* algorithm, selectively expanding and accessing nodes to improve the search efficiency of the A* algorithm [20]. Zhang et al. reduced the total number of nodes traversed by the A* algorithm by using a bidirectional search strategy and an improved evaluation-function approach [21]. Gan et al. modified the original evaluation function of the A* algorithm through Q-Learning to enable the A*

algorithm to realize path planning in changing environments [22]. Fu et al. proposed an improved A* algorithm to shorten the path by determining whether there is an obstacle between the current node and the target node [23]. Yan et al. effectively reduced the number of nodes in the A* algorithm by determining whether there are obstacles between the two nodes before and after the current node and retaining or deleting the current node [24]. Tang et al. set the filter function to avoid a too-large turning angle of the path obtained by the A* algorithm and combined it with the cubic B-spline interpolarization algorithm to make the global path smooth [25]. Zhang et al. improved the heuristic function of the A* algorithm with distance information and obstacle information to make the path smoother [26].

The above studies successfully improved the A* algorithm. However, in the face of complex scenarios where static and dynamic obstacles do not exist on the environment map, the above studies are unable to guide the robot to perform dynamic-obstacle avoidance and do not take into account the actual kinematic requirements of the robot. A local path-planning algorithm is a method to plan a path for a robot in a dynamic environment. It guides the robot for dynamic-obstacle avoidance based on the robot's kinematic model, real-time environment information, obstacle distribution and other factors [27,28]. Currently, common local path-planning algorithms for mobile robots include the Dynamic Window Approach (DWA) [29], Temporal Elastic Band (TEB) [30] and Model Predictive Control (MPC) [31]. Among these algorithms, the DWA algorithm is widely used because of its better fit with the robot's kinematic performance, low computational cost and high flexibility [32]. However, in the case of a relatively denser distribution of obstacles, the local paths planned by the DWA algorithm are nonoptimal.

In order to guide the mobile robot to travel along the optimal path and at the same time have a dynamic-obstacle-avoidance capability, the research on the hybrid path-planning method of the A* algorithm and DWA algorithm has received much attention. Wu et al. smoothed the A* algorithm path and hybridized it with the DWA algorithm to help the robot perform better path planning and obstacle avoidance [33]. Li et al. used an adaptive adjustment step-length algorithm and cubic Bessel curve to solve the problems of many corner points and a large corner in the search path of the A* algorithm and used it in fusion with the DWA algorithm, which effectively shortened the path length and made the robot equipped with a dynamic-obstacle-avoidance function [34]. The above research methods have yielded good results. However, the above studies lacked consideration of the actual motion requirements of the robot and did not improve the robot path-search efficiency. To solve these problems, this paper proposes a fusion algorithm that combines the kinematical constrained A* algorithm and the DWA algorithm.

3. Global Path Planning Based on Improved A* Algorithm

3.1. Traditional A Algorithm*

The A* algorithm, as a heuristic graph-search algorithm, discovers and determines the search target by evaluating the function and is widely used in global path planning to accomplish the static path planning of a mobile robot from the start node to the target node. The traditional A* algorithm works as follows: in a grid map, the A* algorithm starts from the grid where the initial node is located, searches for nodes in eight directions around the initial node, finds the node with the lowest path-estimation cost and takes that node as the current node, continues to search for nodes in eight directions around the current node and repeats the above process until the target node becomes the current node. The evaluation function of the traditional A* algorithm is as follows:

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

In the above equation, n is the current node and $f(n)$ is the total cost of the current node. $g(n)$ is the cost of moving from the initial node to the current node. $h(n)$ is the estimated cost of moving from the current node n to the target node, also known as the heuristic function.

Suppose there exist any two nodes $I(x_i, y_i)$ and $J(x_j, y_j)$ in a plane right-angle coordinate system, and the distance between the two nodes is noted as D. The heuristic-function representation of the traditional A* algorithm often uses the Euclidean distance, which is calculated as follows:

$$D = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

3.2. Improved A Algorithm*

3.2.1. Neighborhood Optimization

The search principle of the traditional A* algorithm is to add a start point to the open list and add that initial node as a parent to the closed list. Then, the child nodes corresponding to the surrounding eight directions are selected and expanded, as shown in Figure 1. Although the node expansion based on the eight-neighborhood approach seems reasonable, in practice, there are redundant directions that increase the computational cost of the A* algorithm.

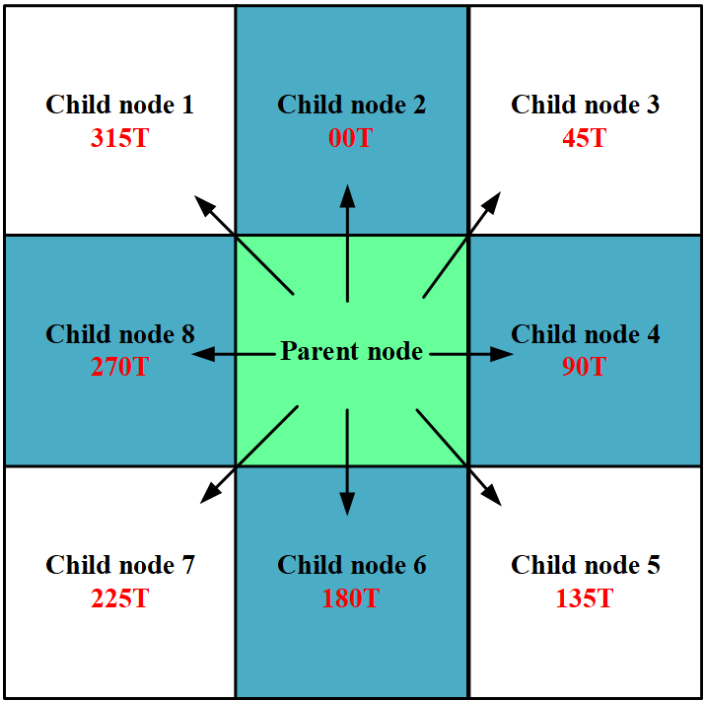


Figure 1. Node-search direction. The green grid represents the parent node, the blue grid represents the children in the base neighborhood, and the white grid represents the children in the extended neighborhood.

As shown in Figure 2, when the robot needs to reach the end point from the start point, there are always three directions that do not need to extend the node. For Figure 2, there is no need to search for child nodes 1, 7 and 8, so these three directions can be directly discarded when expanding and traversing the nodes. Summarizing the situation in Figure 2, it can be found that the robot does not need to traverse all eight directions of the neighborhood when it moves, and appropriately discarding the redundant three directions can reduce the computation amount of the algorithm. The three directions to be abandoned depend on the location of the start and end points. We use a parametric approach for the description; i.e., the A* algorithm search direction is classified according to the angle between the target point and the current point, as shown in Table 1.

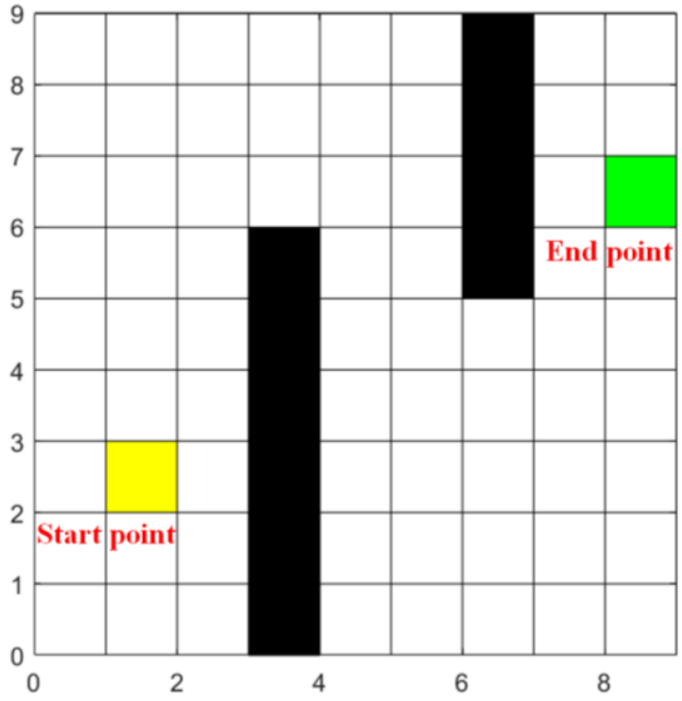


Figure 2. Neighborhood optimization example. The yellow grid indicates the initial point, the green grid indicates the target point, and the black grid indicates the obstacle.

Table 1. Neighborhood optimization strategy.

θ	Keep Directions	Abandon Directions
$[337.5°, 360^{\circ}) \cup [0°, 22.5°)$	00T,45T,90T,270T,315T	135T,180T,225T
$[22.5°, 67.5°)$	00T,45T,90T,135T,315T	180T,225T,270T
$[67.5°, 112.5°)$	00T,45T,90T,135T,180T	225T,270T,315T
$[112.5°, 157.5°)$	45T,90T,135T,180T,225T	270T,315T,00T
$[157.5°, 202.5°)$	90T,135T,180T,225T,270T	00T,45T,315T
$[202.5°, 247.5°)$	135T,180T,225T,270T,315T	00T,45T,90T
$[247.5°, 292.5°)$	180T,225T,270T,315T,00T	45T,90T,135T
$[292.5°, 337.5°)$	225T,270T,315T,00T,45T	90T,135T,180T

In Table 1, θ denotes the angle between the line connecting the direction of the initial node and the target node and the due north direction.

3.2.2. Heuristic-Function-Model Improvement

There is the following conclusion about the selection of the heuristic function: if $h(n)$ is zero, it means that only $g(n)$ plays a role in the evaluation function $f(n)$, and then the A* algorithm will become a Dijkstra algorithm, which is a single-source path search. If $h(n)$ is less than the actual cost, the algorithm can plan an optimal path, but $h(n)$ is not taken as small as possible as taking too small a value algorithm traverses a large number of nodes, and the search efficiency is low. If $h(n)$ is equal to the actual cost, the algorithm is in an ideal state as the optimal path can be planned quickly. If $h(n)$ is greater than the actual cost, at this time, the algorithm's search efficiency is fast, but the planned path is not guaranteed to be optimal, and $h(n)$ is much greater than $g(n)$, indicating that only the $h(n)$ of the evaluation function $f(n)$ comes into play, at which time the A* algorithm will become a breadth-first algorithm. This shows that the heuristic function has an important role in the A* algorithm, and the heuristic function determines the change trend and search performance of the A* algorithm.

The traditional A* algorithm chooses the Euclidean distance as the heuristic-function expression, which has a high computational complexity, leading to a larger number of

nodes visited by the A* algorithm, and the paths are likely to turn, which reduces the search efficiency of the A* algorithm. In this paper, we choose the Manhattan distance as the heuristic function representation of the A* algorithm, which is calculated as follows:

$$D = |x_i - x_j| + |y_i - y_j| \tag{3}$$

3.2.3. Heuristic-Function Weight Optimization

Since the heuristic function $h(n)$ in the A* algorithm affects the search performance, when there are obstacles in the environment, too many redundant nodes will be generated, making the number of node searches larger and leading to a decrease in the algorithm's search efficiency. When there are static obstacles in the environment, the most direct impact on the A* algorithm is that the environmental-distance-information changes, i.e., the distance of the current node from the initial node and the target node is constantly changing. Therefore, it is necessary to combine the changing environmental-distance information to dynamically adjust the A* algorithm heuristic function $h(n)$ and reduce the interference brought by the obstacles existing in the environment to the A* algorithm search.

When the traditional A* algorithm performs path planning, the extended nodes are selected by the moving cost $g(n)$ and heuristic function $h(n)$ without the dynamic coefficient weighting of $h(n)$ using the environmental-distance-scale information, resulting in locking the search space and search efficiency of the algorithm, which cannot be dynamically adjusted according to the position of the extended nodes in the grid map. If the current node is close to the initial node and far from the target node, the coefficient of $h(n)$ can be appropriately increased to make the coefficient value greater than the default value of one to improve the search speed of the algorithm and reduce the search space. On the contrary, if the current node is far from the initial node and close to the target node, the coefficient of $h(n)$ can be reduced to make the coefficient value close to the default value of one. If the heuristic function $h(n)$ can be dynamically adjusted according to the current node-distance-scale information, the search space of the A* algorithm can be greatly reduced and the search efficiency of the algorithm can be improved.

In summary, this paper gives a distance-scale parameter P to describe the distance-scale relationship between the current node, the initial node and the target node. Let the coordinates of the initial node of the A* algorithm be $S(x_s, y_s)$ and the coordinates of the target node be $G(x_g, y_g)$; then, the expression of the distance-scale parameter $N(x_n, y_n)$ of the current node P in the grid map is as follows:

$$P = \frac{|(x_n - x_g)| + |(y_n - y_g)|}{|(x_s - x_g)| + |(y_s - y_g)|}, P \in [0, 1] \tag{4}$$

The search space of the A* algorithm should change with the distance-scale parameter P. The improved A* algorithm introduces the distance-scale information P into the evaluation function, as shown in Equation (5), and changes the weights of the movement cost $g(n)$ and the estimated cost $h(n)$ according to the distance-scale parameter P, so as to improve the flexibility of the evaluation function $f(n)$ and promote the A* algorithm to plan the path quickly and accurately. Since the weight of the A* algorithm heuristic function cannot be too large, a new dynamic weighting parameter is designed in this paper in combination with the environmental-distance-scale parameter P. The improved A* algorithm evaluation function is as follows:

$$f(n) = g(n) + e^P \cdot h(n) \tag{5}$$

The improved A* algorithm differs from the traditional A* algorithm in that the improved A* algorithm is able to adaptively adjust the weights of the heuristic function according to the obstacle information in the environment. From Equations (3) and (5), it can be seen that after increasing the distance-scale parameter P, the A* algorithm can more accurately estimate the distance cost of the current node between the initial node and the

target node so that it can dynamically adjust the search strategy and find the better path at a higher operational efficiency.

4. Local Path-Planning Method and Fusion Algorithm Based on Optimized DWA

4.1. DWA Trajectory-Evaluation Function

The trajectory-evaluation function of the DWA consists of three parts, mainly including the heading angle of the target node, the value of linear velocity and the closest distance to the obstacle. The larger the value of the trajectory-evaluation function, the better the trajectory planned by the algorithm. The trajectory-evaluation function $G(v, w)$ of the DWA is shown in the following equation:

$$G(v, w) = \alpha \times vel(v, w) + \beta \times head(v, w) + \gamma \times dist(v, w) \tag{6}$$

In the above equation, N is the total number of simulated trajectories of the algorithm, i represents the i-th simulated trajectory and $\Delta\theta_i$ is the azimuthal deviation angle of the terminal attitude direction of the i-th simulated trajectory from the target node.

$vel(v, w)$ is the current speed value of the mobile robots, and the faster the mobile robots move, the higher the percentage of $vel(v, w)$ in the evaluation function and the greater the score:

$$vel(v, w) = \frac{v_i}{\sum_{i=1}^{N} v_i} \tag{7}$$

In the above equation, v represents the linear velocity in the travel direction of the mobile robots and w represents the angular velocity in the rotation direction of the mobile robots. α, β and γ are weighting coefficients, which can be set with different weights according to the actual needs.

$head(v, w)$ is the direction angle, which is usually evaluated by $\pi - \Delta\theta$, and $\Delta\theta$ is the deviation angle between the end attitude direction of the simulated trajectory and the target-node orientation. As the mobile robots gradually approach the target node along the forward direction, the smaller $\Delta\theta$ is, the larger the value of $\pi - \Delta\theta$ is, and the higher the percentage of $head(v, w)$ in the evaluation function is, the larger the score is:

$$head(v, w) = \frac{\pi - \Delta\theta_i}{\sum_{i=1}^{N} (\pi - \Delta\theta_i)} \tag{8}$$

In the above equation, v_i is the absolute value of the velocity at the end of the i-th simulated trajectory.

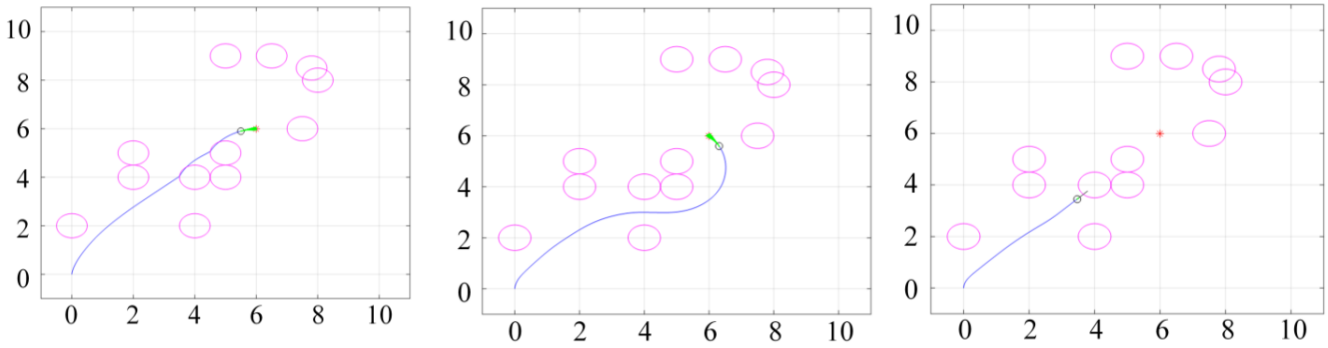
$dist(v, w)$ is the closest distance between the mobile robots and the obstacles; the farther the mobile robots are from the obstacles, the greater the value of d. The higher the percentage of $dist(v, w)$ in the evaluation function, the greater the score:

$$dist(v, w) = \frac{d_i}{\sum_{i=1}^{N} d_i} (d_i = \begin{cases} d_i, d_i \prec D \\ D, d_i \succ D \end{cases}) \tag{9}$$

In the above equation, d_i is the closest distance to the obstacle at the end of the i-th simulated trajectory and D is the constant set by the algorithm.

4.2. DWA Trajectory-Evaluation-Function Optimization

The weighting coefficients α, β and γ in the trajectory-evaluation function $G(v, w)$ of the traditional DWA algorithm affect the evaluation and selection of the optimal trajectory by the algorithm. However, there is no clear reference and explanation on how to choose the weighting coefficients α, β and γ. Fox D et al. believe that an α of 0.1, β of 0.8 and γ of 0.1 can bring more reasonable path planning for the algorithm [29]. In this paper, we build a specific simulation scenario in conjunction with the actual experimental environment, and the weighting coefficients α, β and γ of the trajectory-evaluation function of the DWA algorithm are optimized in the simulation scenario. At first, we tested the limits by taking 0.8 for each of the three parameters α, β and γ in conjunction with the conclusions of Fox D et al. The experimental results are shown in Figure 3.



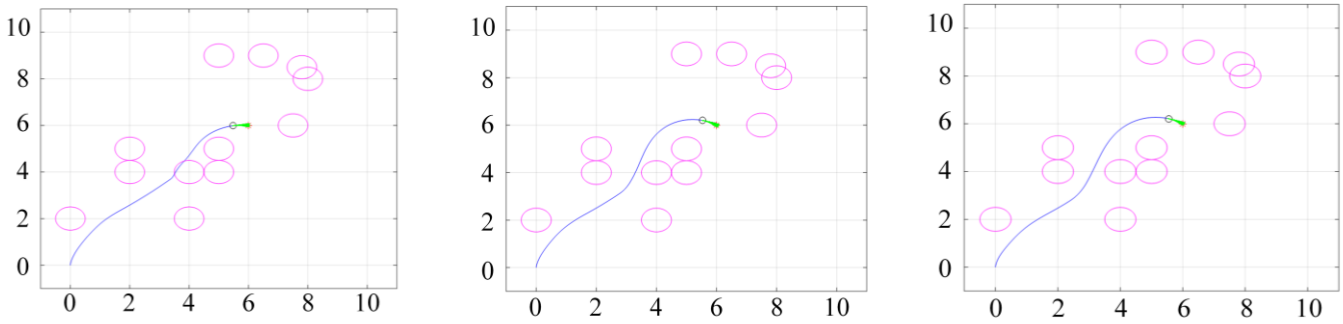
(**a**) α = 0.8, β = 0.1, γ = 0.1 (**b**) α = 0.1, β = 0.8, γ = 0.1 (**c**) α = 0.1, β = 0.1, γ = 0.8

Figure 3. Experiment on the limit value of the weight of the DWA trajectory-evaluation function. Pink circles indicate obstacles, blue trajectories indicate local paths of the robot, and red dots indicate target points.

It can be seen from Figure 3a that when the value of α is large and the values of β and γ are small, the mobile robot moves toward the target node but cannot bypass the obstacle, and the mobile robot is blocked by the obstacle that is in between itself and the line of the target point. It can be seen from Figure 3b that when the value of β is large and the values of α and γ are small, the mobile robot can reach the target node successfully, but the chosen path is not optimal and it is inefficient and unnecessary for the mobile robot to bypass the obstacles at a large angle. It can be seen from Figure 3c that when the value of γ is large and the values of α and β are small, the mobile robot will hit the obstacle on the line with the target point because the direction angle and speed of the mobile robot cannot be adjusted, although the obstacle-avoidance ratio is the largest.

As can be seen in Figure 3, the three parameters are ranked according to the magnitude of the impact that they have on the robot's path-planning results in the following order: β, α and γ. In other words, β should take a larger value, followed by α and finally γ. Fox D et al. proposed that γ can take a very small value, but not 0, because when the value of γ is 0, no obstacle-avoidance protection can be provided [29]. Kaiyu Zheng in the ROS Navigation Tuning Guide suggests that the algorithm obtains better-generalized results by setting α to 20.0, β to 32.0 and γ to 0.02 [35].

Combining the above theories, we set the value of γ to 0.02 and leave it unadjusted, and the total value of the parameters α and β is set to 52 (32 + 20) according to Kaiyu Zheng's proposal [35]. From Figure 3a,b, we know that the value of α should be reduced appropriately and the value of β can be larger. So, we set up three sets of control experiments and set the value of β to integer multiples of α, which were 1, 2 and 3 (the fractional parts of the data were rounded), and the experimental result is shown in Figure 4.



(**a**) $\alpha = 26$, $\beta = 26$, $\gamma = 0.02$ (**b**) $\alpha = 17$, $\beta = 35$, $\gamma = 0.02$ (**c**) $\alpha = 13$, $\beta = 39$, $\gamma = 0.02$

Figure 4. Experiment on the weighting of the DWA trajectory-evaluation function. Pink circles indicate obstacles, blue trajectories indicate local paths of the robot, and red dots indicate target points.

In Figure 4a, we can see that when α and β take the same value, the mobile robot is still unable to go around the obstacle on the line with the target point. In Figure 4b, we can see that when β is twice the value of α, the mobile robot can bypass the obstacle on the line with the target point according to the path planned by the DWA, but there is a risk of hitting the obstacle. As can be seen in Figure 4c, when β is three times as large as α, the mobile robot can bypass the obstacles on the line with the target point according to the path planned by the DWA, and the planned path is optimal at this time. According to the experimental results in Figure 4b,c, we designed a set of intermediate values for testing, i.e., the value of β was 2.5 times the value of α. The experimental result is shown in Figure 5.

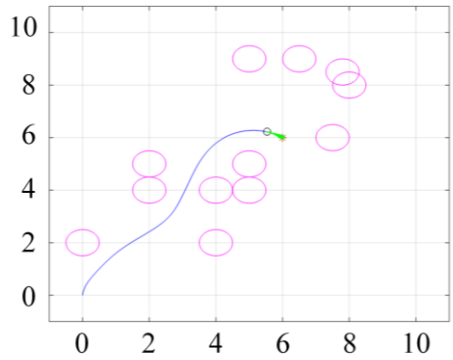


Figure 5. Median value test. In Figure 5, $\alpha = 15$, $\beta = 37$, $\gamma = 0.02$. Pink circles indicate obstacles, blue trajectories indicate local paths of the robot, and red dots indicate target points.

It can be seen in Figure 5 that when β is 2.5 times the value of α, the mobile robot can bypass the obstacles on the line with the target point according to the path planned by the DWA, and the planned path is also optimal at this time.

Finally, combining the above trends in the path-planning results, we chose the parameter values corresponding to Figure 5 for subsequent experimental studies. That is, α takes the value 15, β takes the value 37 and γ takes the value 0.02.

4.3. Algorithm Fusion

Initially, the A* algorithm plans an optimal global collision-free path and extracts the critical nodes from the global path. Then, starting from the initial position, the DWA algorithm is used to go through the critical nodes on the global path to realize local path planning and obstacle avoidance, and the flowchart of the fusion algorithm is shown in Figure 6.

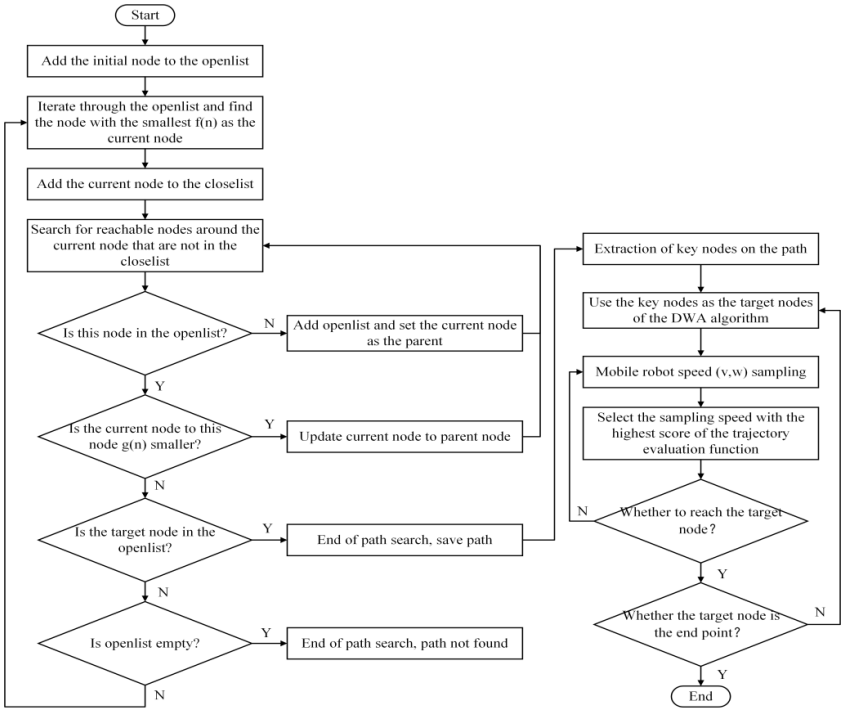


Figure 6. Flow chart of fusion algorithm.

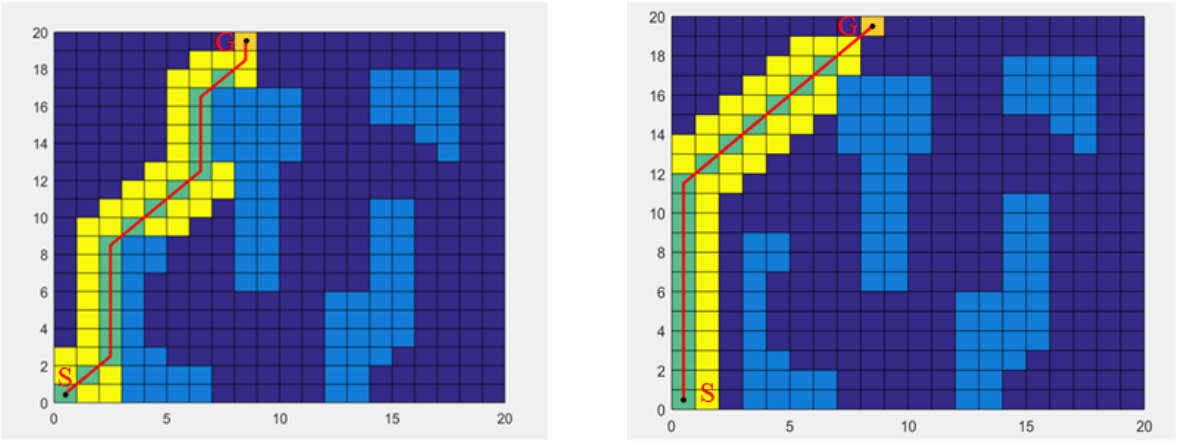
5. Experimental Results

We verified the feasibility and effectiveness of the improved A* algorithm, the improved DWA algorithm and the fusion algorithm step by step through a series of simulations and then programmed the fusion algorithm and practically applied it in the ROS system.

5.1. Simulation Experiments

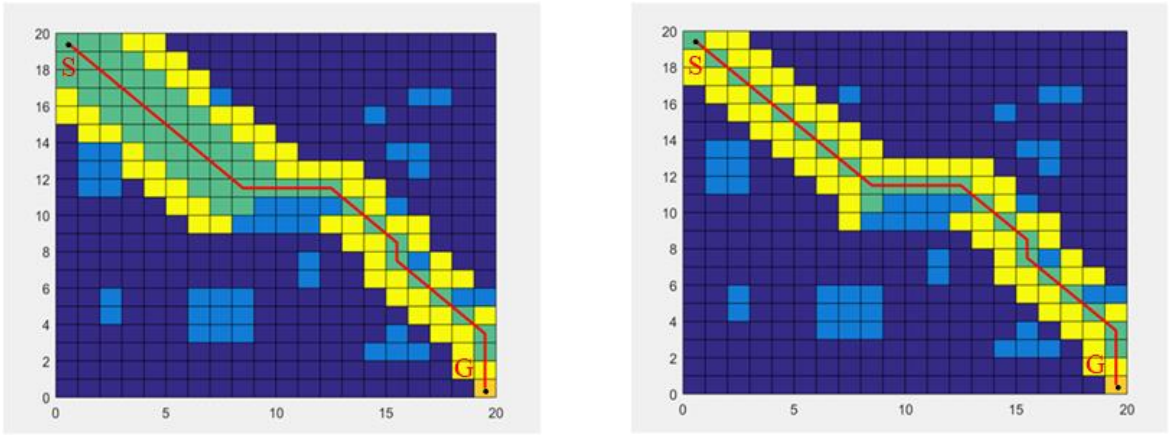
5.1.1. Improved A* Algorithm Simulation Experiments

To verify the effectiveness of the improved A* algorithm, we conducted two sets of experimental tests in different grid maps. Then, given the start and target points in the grid map, the traditional A* algorithm and the modified A* algorithm were executed, respectively. The results of the two sets of experiments are shown in Figures 7 and 8.



(**a**) Traditional A* algorithm (**b**) Improved A* algorithm

Figure 7. First set of experimental results. In Figure 7, S denotes the start point, G denotes the goal point, the green grid denotes the list of nodes to be extended the blue grid indicates obstacles, and the yellow grid denotes the searched neighborhood range.



(**a**) Traditional A* algorithm (**b**) Improved A* algorithm

Figure 8. Second set of experimental results. In Figure 8, S denotes the start point, G denotes the goal point, the green grid denotes the list of nodes to be extended the blue grid indicates obstacles, and the yellow grid denotes the searched neighborhood range.

From Figures 7 and 8, it can be seen that compared with the traditional A* algorithm, the improved A* algorithm can effectively reduce the number of path bends and narrow the node-search range, thus improving the algorithm's search efficiency and the quality of path planning and making it more in line with robot kinematics. We performed 20 replications for each group of experiments and counted the experimental data, as shown in Table 2. In Table 2, the values of the average calculation time are reported as the mean ± standard deviation for 20 executions.

Table 2. Neighborhood optimization experimental data.

Experiment	Average Calculation Time (s)		Number of Nodes	
	Traditional A*	Improved A*	Traditional A*	Improved A*
1	2.005 ± 0.010	1.925 ± 0.014	56	56
2	6.123 ± 0.020	2.560 ± 0.012	113	87

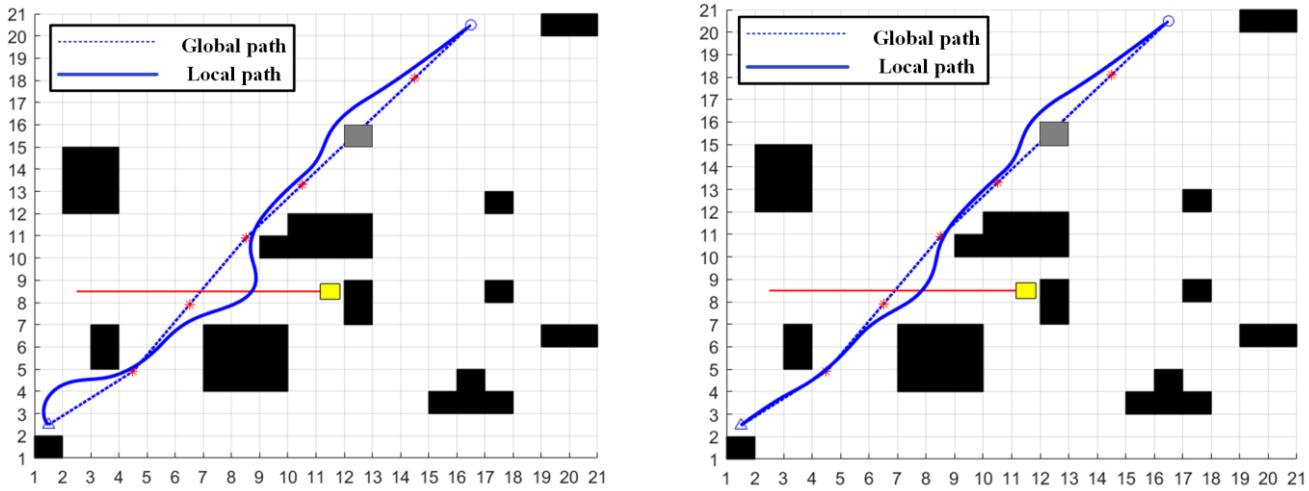
Combined with Table 2, it can be seen that the improved A* algorithm takes less time to plan the path and searches fewer nodes than the traditional A* algorithm. When the number of search nodes is the same, the improved A* algorithm plans paths with fewer bends, which is more in line with robot kinematics.

5.1.2. Improved DWA Algorithm and Fusion-Algorithm-Simulation Experiments

In order to verify the effectiveness of the improved DWA algorithm, we conducted a comparison experiment on the grid map. We gave the same target nodes to guide the DWA algorithm for path planning and then analyzed the experimental result. The experimental results are shown in Figure 9.

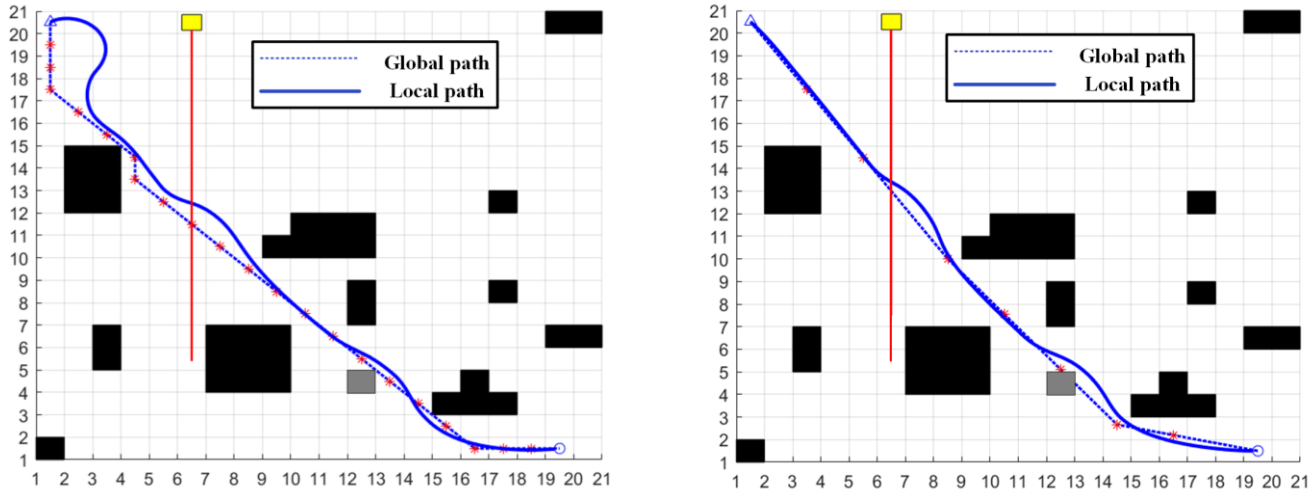
Comparing Figure 9a,b, it can be seen that the local paths planned by the improved DWA algorithm are closer to the global paths than the traditional DWA algorithm in the case of the same start and goal points and motion nodes.

After validating the improved DWA algorithm, we carried out an experimental validation of the fusion algorithm. We conducted comparative experiments between the traditional A* and traditional DWA hybrid algorithms as well as the fusion algorithm designed in this paper. The experimental results are shown in Figure 10.



(**a**) Traditional DWA algorithm (**b**) Improved DWA algorithm

Figure 9. Comparison of local path planning. The black squares represent the original obstacles in the static grid map, the gray square represents the subsequently added static obstacles and the yellow square represents the dynamic obstacles.



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

Figure 10. Global path planning and local path planning. (**a**) is the path-planning result of the hybrid algorithm of traditional A* and traditional DWA. (**b**) is the path-planning result of the fusion algorithm of kinematically constrained A* and DWA.

It can be seen from Figure 10 that compared with the hybrid algorithm of the traditional A* and traditional DWA, the global path planned by the fusion algorithm of the kinematically constrained A* and DWA is more satisfying regarding the actual motion requirements of the robot, and the local path is closer to the global path. The results of the two algorithms are shown in Table 3.

Table 3. Comparison of experimental data of two algorithms.

Map/m^2	Algorithm	Path Length (m)	Number of Turns	Path Time (s)
20 × 20	Traditional A*+ Traditional DWA	29.2294	4	91.0430
	Improved A*+ Improved DWA	27.4067	1	79.4570

The data in Table 3 show that the fusion algorithm of the kinematically constrained A* and DWA effectively shortens the path length by 6.24%, reduces the number of path bends by 75% and saves 12.73% of the planning time compared to the hybrid algorithm of the traditional A* and traditional DWA.

The robot-motion-velocity data published by the hybrid algorithm of the traditional A* and DWA as well as the kinematically constrained fusion algorithm of the A* and DWA are shown in Figure 11.

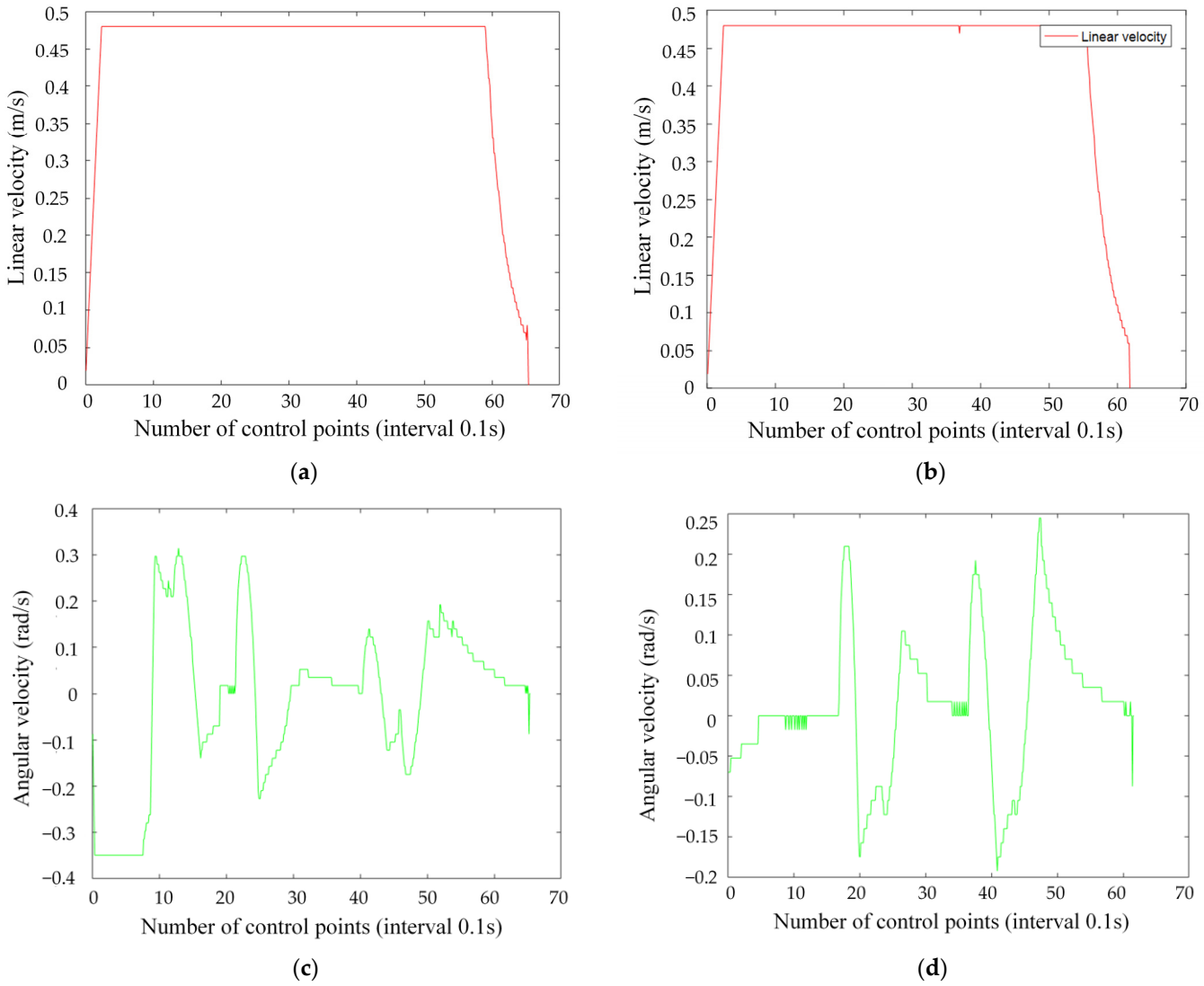


Figure 11. Robot-motion speed. (**a**) is the robot-line speed posted by the hybrid algorithm. (**b**) is the robot-line speed published by the fusion algorithm. (**c**) is the angular velocity of the robot published by the hybrid algorithm. (**d**) is the angular velocity of the robot published by the fusion algorithm. The red line indicates linear velocity and the green line indicates angular velocity.

It can be seen in Figure 11 that the fusion algorithm issues a smaller angular velocity of the robot compared to the traditional hybrid algorithm, i.e., the robot runs more smoothly. In addition, the fusion algorithm has a shorter local motion time, i.e., a higher quality of path planning.

5.2. Practical Experiments

In this section, we conduct a practical experimental study of the global and local paths planned by the fusion algorithm. We build an autonomous mobile robot experimental platform, which is called Mir-5, as shown in Figure 12. The mobile controller is a computer and is equipped with an i5-11400 processor, NVIDIA GTX1650s graphics card, which uses the melodic version of the Robot Operating System (ROS) in Ubuntu 18.04, and the algorithms involved in this experiment are based on C++ implementation.

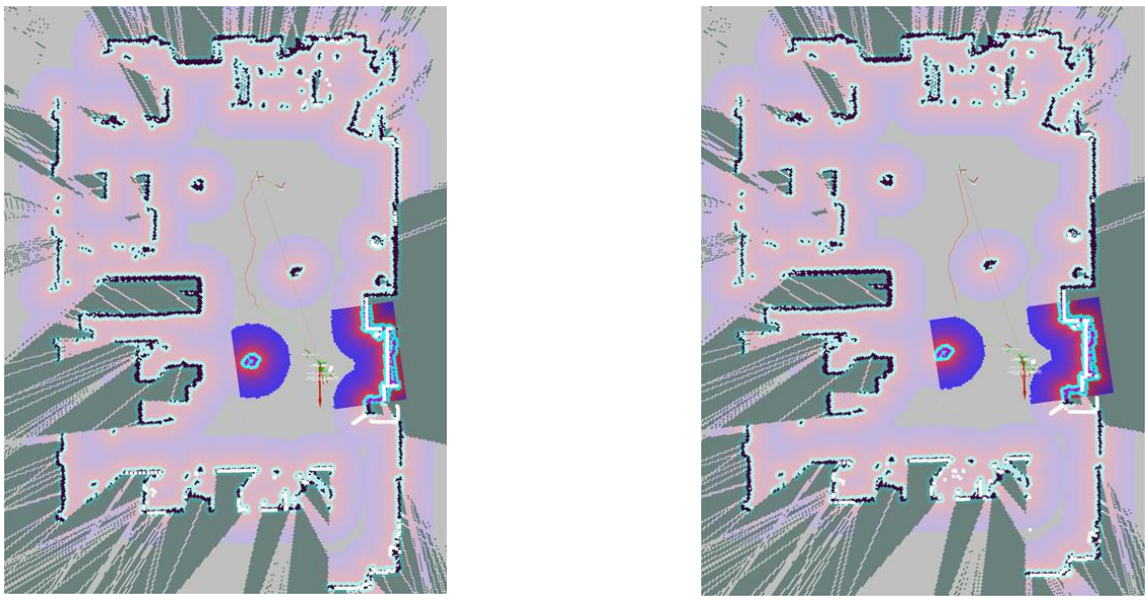
Figure 12. Mobile robot platform. The platform is equipped with a Kinect V2 sensor for visual inspection, a UR5 robotic arm for object grasping and two SICK S300 LIDARs on the chassis that can be used to acquire point cloud information about the surrounding environment for navigation research.

5.2.1. Static Global Path Planning

In order to verify the global path-planning performance of the fusion algorithm, we build a static obstacle-distribution experimental environment, as shown in Figure 13. Then, the global path planning is performed by using the traditional A* and DWA hybrid algorithm and the kinematically constrained A* and DWA fusion algorithm.

Figure 13. Distribution of static obstacles.

The global path-planning results of the two algorithms are shown in Figure 14.



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

Figure 14. Distribution of static obstacles. (**a**) is the global path-planning result of the traditional hybrid algorithm. (**b**) is the global path-planning result of the fusion algorithm proposed in this paper.

It can be seen from Figure 14a that the global path-planning result of the traditional hybrid algorithm has more unnecessary turns, which greatly increases the path redundancy length and is not conducive to the smooth motion of the robot. As can be seen in Figure 14b, the global path planned by the fusion algorithm proposed in this paper has fewer turns, does not increase the length of redundancy and is more suitable for the smooth motion of the robot. The global path-planning data of the two algorithms are shown in Table 4.

Table 4. Comparison of global path-planning data.

	Path Length (m)	Planning Time (s)	Number of Turns
Hybrid Algorithm	6.6748	0.03374	8
Fusion Algorithm	6.2410	0.02524	2
Comparison results	6.5% down	25.20% down	75% down

It can be seen from Table 4 that compared to the traditional hybrid algorithm, the fusion algorithm proposed in this paper reduces the global path length by 6.5%, the path-planning time by 25.2% and the number of turns by 75%.

5.2.2. Dynamic Local Path Planning

In order to verify the local path-planning performance of the fusion algorithm, we added moving obstacles based on the static experimental environment, as shown in Figure 15. Then, the dynamic local path planning was performed by using the traditional A* and DWA hybrid algorithm and the kinematically constrained A* and DWA fusion algorithm.

Figure 15. Dynamic experimental scenario.

The local path-planning results for both algorithms are shown in Figure 16.

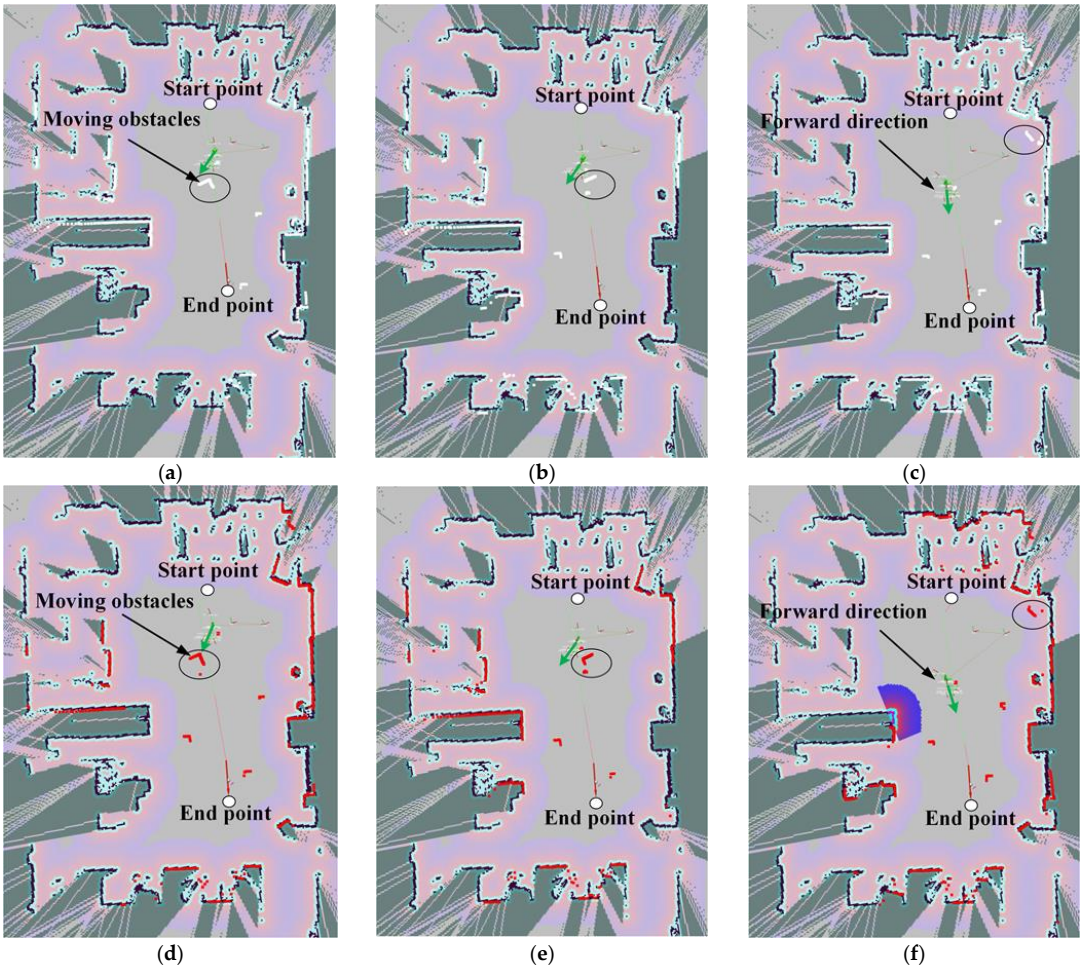


Figure 16. Dynamic-obstacle-avoidance process. (**a**) demonstrates the hybrid algorithm meeting dynamic obstacles. (**b**) is the hybrid algorithm avoiding dynamic obstacles. (**c**) is the hybrid algorithm returning to the global path. (**d**) is the fusion algorithm meeting dynamic obstacles. (**e**) is the fusion algorithm avoiding dynamic obstacles. (**f**) is the fusion algorithm returning to the global path.

As can be seen from Figure 16, both algorithms can realize dynamic-obstacle avoidance, but the fusion algorithm proposed in this paper has better foresight and local path planning and fits the global path better. The dynamic local path-planning data for the two algorithms are shown in Figures 17 and 18, respectively.

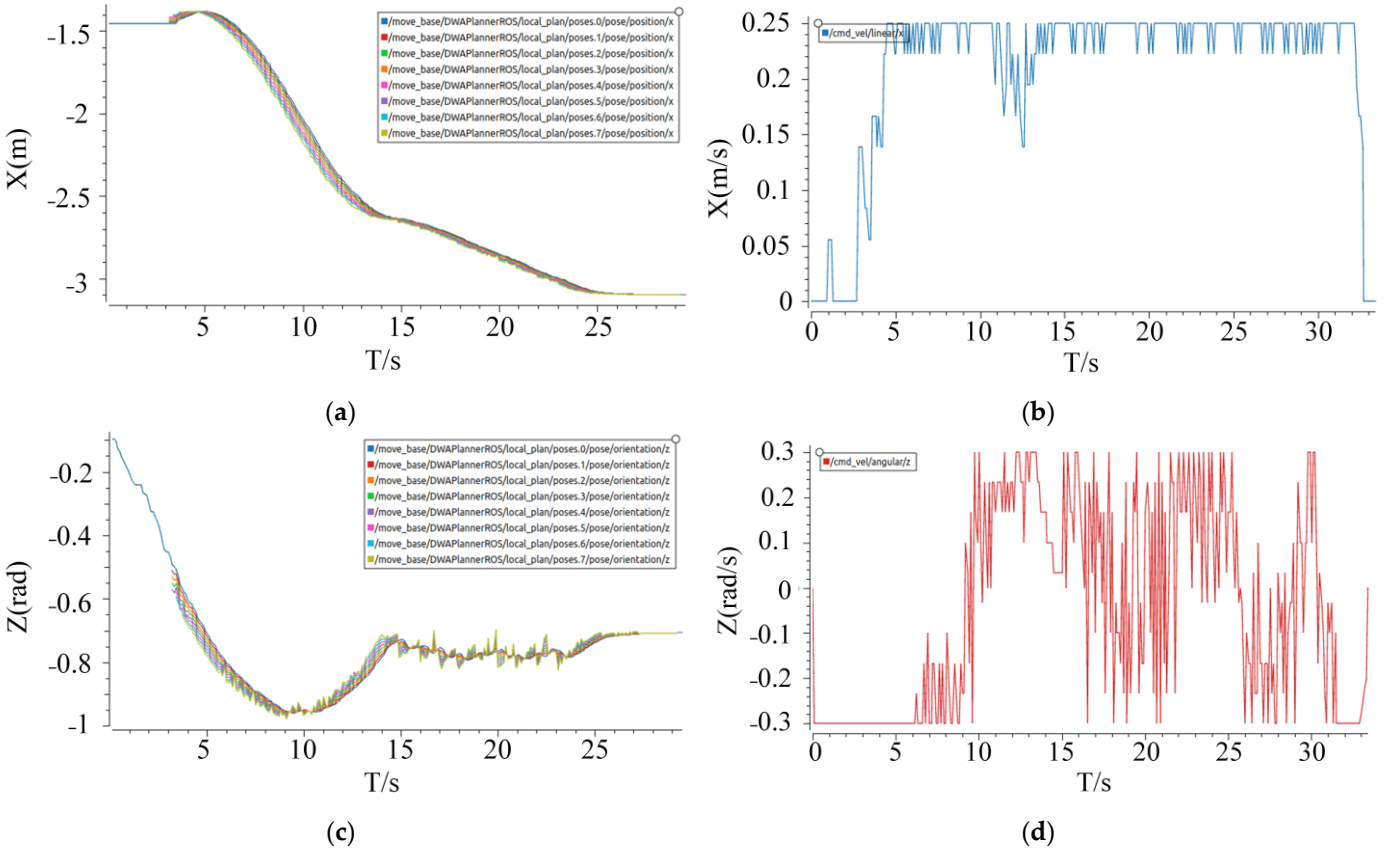


Figure 17. Traditional hybrid algorithm's local path-planning data. (**a**) is the traditional hybrid algorithm, whereby the *X*-axis is the local path-planning result. (**b**) is the linear motion speed of the robot planned by the traditional hybrid algorithm. (**c**) is the traditional hybrid algorithm, whereby the *Z*-axis is the local path-planning result. (**d**) is the robot angular velocity planned by the traditional hybrid algorithm.

As can be seen in Figure 17, the local path in the *X*-axis direction planned by the traditional hybrid algorithm is not smooth enough, resulting in discontinuous line speeds issued to the robot. Meanwhile, the local paths planned by the traditional hybrid algorithm have unstable *Z*-axis angle changes, resulting in discontinuous angular speeds issued to the robot.

As can be seen in Figure 18, the local path in the *X*-axis direction planned by the fusion algorithm in this paper is smoother, and the linear velocity issued to the robot is continuous and stable. Meanwhile, the local paths planned by the fusion algorithm in this paper have stable *Z*-axis angle variations, and the angular velocity issued to the robot is more continuous.

Comparing Figures 17 and 18, it can be seen that compared with the traditional hybrid algorithm, the local paths planned by the fusion algorithm proposed in this paper are smoother and more coherent, which is easier for robot-motion execution.

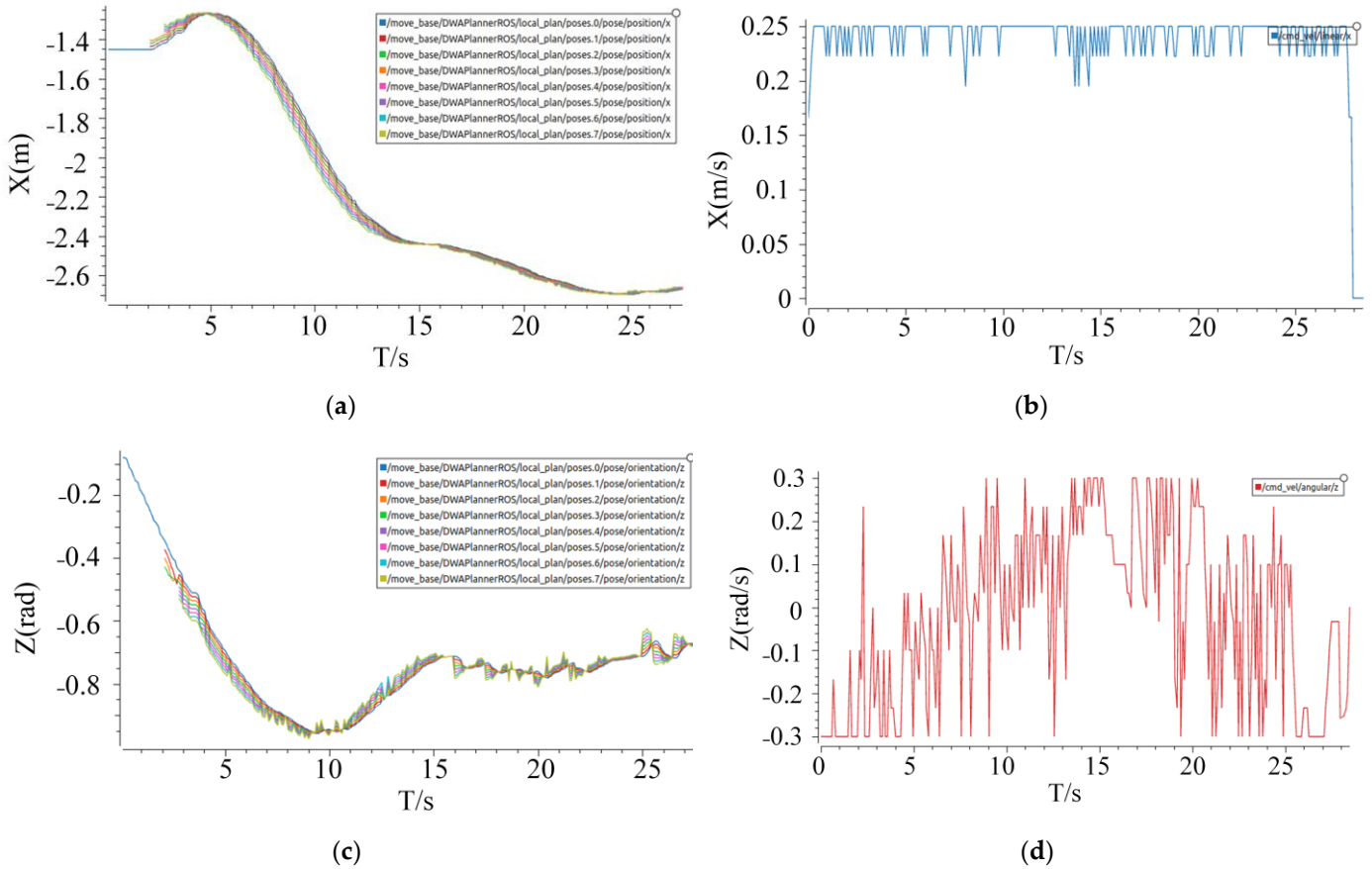


Figure 18. The fusion algorithm in this paper's local path-planning data. (**a**) is the fusion algorithm, whereby the *X*-axis is the local path-planning result. (**b**) is the robot linear motion speed planned by the fusion algorithm. (**c**) is the fusion algorithm, whereby the *Z*-axis is the local path-planning result. (**d**) is the robot angular velocity planned by the fusion algorithm.

6. Conclusions

In this paper, we improve the traditional A* algorithm to speed up the algorithm search efficiency, reduce the algorithm computational cost and decrease the number of path turns. Then, we optimized the parameters of the trajectory-evaluation function of the DWA algorithm to make its path-planning results more in line with practical needs. Finally, the improved A* algorithm was fused with the optimized DWA algorithm to complement the kinematic deficiencies of the A* algorithm and at the same time provide the robot with a dynamic-obstacle-avoidance capability.

Although the method designed in this paper achieved better results, there are still some issues that need to be further investigated. The DWA algorithm uses actions to generate robot trajectories, prompting the controller to directly publish the angular and linear velocities of the robot motion. When the angular or linear velocities released by the controller vary too much, the DWA algorithm can cause the robot to run very unevenly due to the lack of consideration of acceleration and braking transients of the robot chassis drives. Therefore, it is necessary to adjust the velocity posted by the controller to make it more suitable for robot-motion execution, which is more like trajectory tracking that the DWA algorithm does not have. Furthermore, how to add vision sensors [36] to assist the robot in performing better dynamic-obstacle avoidance and target tracking is also an issue worthy of in-depth research.

In conclusion, the main goal of this paper is to help the robot plan the shortest passable route in a short period of time, and the proposed research method is also applicable to outdoor conventional scenarios, but the method still needs to be further investigated for

outdoor operation scenarios where there is a narrow passageway or where the road network information needs to be included for long-distance planning.

Author Contributions: Project administration, Y.L.; writing—original draft preparation, C.W.; supervision, H.W.; formal analysis, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Heilongjiang Province Key Research and Development Program Project (NO. 2022ZX01A01) and the Self-Planned Task (NO. SKLRS201813B) of the State Key Laboratory of Robotics and System (HIT).

Data Availability Statement: Not applicable.

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

References

1. Zhang, Y.; Zhou, Y.; Li, H.; Hao, H.; Chen, W.; Zhan, W. The Navigation System of a Logistics Inspection Robot Based on Multi-Sensor Fusion in a Complex Storage Environment. *Sensors* **2022**, *22*, 7794. [CrossRef] [PubMed]
2. Hayajneh, M.; Al Mahasneh, A. Guidance, Navigation and Control System for Multi-Robot Network in Monitoring and Inspection Operations. *Drones* **2022**, *6*, 332. [CrossRef]
3. Zhang, H.; Zhuang, Q.; Li, G. Robot Path Planning Method Based on Indoor Spacetime Grid Model. *Remote Sens.* **2022**, *14*, 2357. [CrossRef]
4. Cao, Y.; Fang, X. Optimized-Weighted-Speedy Q-Learning Algorithm for Multi-UGV in Static Environment Path Planning under Anti-Collision Cooperation Mechanism. *Mathematics* **2023**, *11*, 2476. [CrossRef]
5. Qin, H.; Shao, S.; Wang, T.; Yu, X.; Jiang, Y.; Cao, Z. Review of autonomous path planning algorithms for mobile robots. *Drones* **2023**, *7*, 211. [CrossRef]
6. Yu, Z.; Yuan, J.; Li, Y.; Yuan, C.; Deng, S. A path planning algorithm for mobile robot based on water flow potential field method and beetle antennae search algorithm. *Comput. Electr. Eng.* **2023**, *109*, 108730. [CrossRef]
7. Ma, Z.; Chen, J. Adaptive path planning method for UAVs in complex environments. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *115*, 103133. [CrossRef]
8. Zhang, T.; Xu, J.; Wu, B. Hybrid Path Planning Model for Multiple Robots Considering Obstacle Avoidance. *IEEE Access* **2022**, *10*, 71914–71935. [CrossRef]
9. Yin, X.; Cai, P.; Zhao, K.; Zhang, Y.; Zhou, Q.; Yao, D. Dynamic Path Planning of AGV Based on Kinematical Constraint A* Algorithm and Following DWA Fusion Algorithms. *Sensors* **2023**, *23*, 4102. [CrossRef]
10. Yu, J.; Yang, M.; Zhao, Z.; Wang, X.; Bai, Y.; Wu, J.; Xu, J. Path planning of unmanned surface vessel in an unknown environment based on improved D* Lite algorithm. *Ocean Eng.* **2022**, *266*, 112873. [CrossRef]
11. Dijkstra, E.W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; ACM: New York, NY, USA, 2022; pp. 287–290.
12. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
13. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
14. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
15. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
16. Ajeil, F.H.; Ibraheem, I.K.; Sahib, M.A.; Humaidi, A.J. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Appl. Soft Comput.* **2020**, *89*, 106076. [CrossRef]
17. Liu, H.; Zhang, Y. ASL-DWA: An Improved A-Star Algorithm for Indoor Cleaning Robots. *IEEE Access* **2022**, *10*, 99498–99515. [CrossRef]
18. Liu, Y.; Wang, C.; Wu, H.; Wei, Y.; Ren, M.; Zhao, C. Improved LiDAR Localization Method for Mobile Robots Based on Multi-Sensing. *Remote Sens.* **2022**, *14*, 6133. [CrossRef]
19. Wu, X.; Xu, L.; Zhen, R.; Wu, X. Bi-directional adaptive A* algorithm toward optimal path planning for large-scale UAV under multi-constraints. *IEEE Access* **2020**, *8*, 85431–85440. [CrossRef]
20. Zhou, K.; Yu, L.; Long, Z.; Mo, S. Local path planning of driverless car navigation based on jump point search method under urban environment. *Future Internet* **2017**, *9*, 51. [CrossRef]
21. Zhang, H.; Tao, Y.; Zhu, W. Global Path Planning of Unmanned Surface Vehicle Based on Improved A-Star Algorithm. *Sensors* **2023**, *23*, 6647. [CrossRef]
22. Gan, X.; Huo, Z.; Li, W. DP-A*: For Path Planing of UGV and Contactless Delivery. *IEEE Trans. Intell. Transp. Syst.* **2023**, 1–13. [CrossRef]

23. Fu, B.; Chen, L.; Zhou, Y.; Zheng, D.; Wei, Z.; Dai, J.; Pan, H. An improved A* algorithm for the industrial robot path planning with high success rate and short length. *Robot. Auton. Syst.* **2018**, *106*, 26–37. [CrossRef]
24. Zhang, Y.; Li, L.L.; Lin, H.C.; Ma, Z.; Zhao, J. Development of path planning approach using improved A-star algorithm in AGV system. *J. Internet Technol.* **2019**, *20*, 915–924.
25. 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]
26. 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*, 17298814211042730. [CrossRef]
27. Wang, Z.; Li, G.; Ren, J. Dynamic path planning for unmanned surface vehicle in complex offshore areas based on hybrid algorithm. *Comput. Commun.* **2021**, *166*, 49–56. [CrossRef]
28. Guo, N.; Li, C.; Wang, D.; Song, Y.; Liu, G.; Gao, T. Local path planning of mobile robot based on long short-term memory neural network. *Autom. Control Comput. Sci.* **2021**, *55*, 53–65. [CrossRef]
29. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
30. Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012, Proceedings of the 7th German Conference on Robotics, Munich, Germany, 21–22 May 2012*; VDE: Frankfurt am Main, Germany, 2012; pp. 1–6.
31. Islam, M.; Okasha, M.; Sulaeman, E. A model predictive control (MPC) approach on unit quaternion orientation based quadrotor for trajectory tracking. *Int. J. Control Autom. Syst.* **2019**, *17*, 2819–2832. [CrossRef]
32. Kobayashi, M.; Motoi, N. Local path planning: Dynamic window approach with virtual manipulators considering dynamic obstacles. *IEEE Access* **2022**, *10*, 17018–17029. [CrossRef]
33. Wu, B.; Chi, X.; Zhao, C.; Zhang, W.; Lu, Y.; Jiang, D. Dynamic path planning for forklift AGV based on smoothing A* and improved DWA hybrid algorithm. *Sensors* **2022**, *22*, 7079. [CrossRef]
34. Li, Y.; Jin, R.; Xu, X.; Qian, Y.; Wang, H.; Xu, S.; Wang, Z. A mobile robot path planning algorithm based on improved a* algorithm and dynamic window approach. *IEEE Access* **2022**, *10*, 57736–57747. [CrossRef]
35. ROS Navigation Tuning Guide. Available online: https://kaiyuzheng.me/documents/papers/ros_navguide.pdf (accessed on 27 October 2023).
36. Liu, Y.; Wu, H.; Wang, C.; Wei, Y.; Ren, M.; Feng, T. Real-Time Dense Construction with Deep Multiview Stereo Using Camera and IMU Sensors. *IEEE Sens. J.* **2023**, *23*, 19648–19659. [CrossRef]