


Research Article

Research on Static/Dynamic Global Path Planning Based on Improved A* Algorithm for Mobile Robots

Huifang Bao ¹, Jie Fang ¹, Chaohai Wang,² Zebin Li,¹ Jinsi Zhang,¹
and Chuansheng Wang¹

¹School of Electrical and Photoelectronic Engineering, West Anhui University, Lu'an 237012, China

²Anhui Wanxiang Power Equipment Co., Ltd., Lu'an 237012, China

Correspondence should be addressed to Huifang Bao; 1069484572@qq.com and Jie Fang; 63640193@qq.com

Received 11 October 2022; Revised 8 May 2023; Accepted 30 May 2023; Published 16 June 2023

Academic Editor: Weitian Wang

Copyright © 2023 Huifang Bao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In view of the problems of A* algorithm in path planning, such as collision risk, the path is not necessarily optimal, and there are numerous turning nodes. Therefore, this study proposes an improved A* algorithm to improve the quality of the planned path. First, the 8-neighborhood children of the parent node are generated, and the security of the planned path is improved by further investigating the properties of the neighbors of these children one by one to reasonably set virtual obstacles. Second, the heuristic function of A* algorithm is ameliorated to make it closer to the actual cost, so as to enhance the accuracy of the planned path; finally, the planned path is smoothed by using the cubic uniform B-spline curve to eliminate corner sharp points in the path. The simulation results show that the improved A* algorithm can not only ensure the safety and smoothness of the planned path but also obtain the shortest planned path in the static environment with different obstacle rates. In addition, we combine the improved A* algorithm with the dynamic window algorithm to enable mobile robots to realize real-time dynamic obstacle avoidance while ensuring the optimality of global path planning.

1. Introduction

Nowadays, mobile robots are becoming more and more significant in a variety of fields. No matter how the application scenario changes, one of the key problems that mobile robots must solve is path planning. Path planning refers to planning a collision-free, safe, and feasible optimal route from the starting point to the target point for a mobile robot in an obstacle environment while satisfying certain constraints [1]. Path planning can be broadly classified into two main approaches: traditional algorithm path planning and intelligent bionic algorithm path planning.

Traditional algorithms include Sunita and Garg [2], Floyd [3], RRT [4], A* [5], and D* [6]. In [7], in order to improve the mobile efficiency of robots in logistics warehouses and manufacturing workshop, a dynamic fusion routing algorithm (DFPA) based on Delaunay triangulation and improved A* is proposed. In [8], Lee proposed a path planning algorithm for range-constrained multirobot tasks based on hybrid tabu search and 2-opt path planning. In [9],

it is useful to enhance the classical Q-learning approach by the APF method. The proposed QAPF learning algorithm can increase the efficiency of learning using the combination of Q-learning and the APF method. Intelligent bionic algorithms include ant colony optimization [10], genetic [11], particle swarm optimization [12]. Orozco-Rosas et al. [13] proposed a hybrid path planning algorithm based on membrane pseudobacterial potential field (MemPBPF) to improve the effectiveness of obstacle avoidance and smoothness. In [14], the authors suggested an algorithm for hybrid path planning to increase the efficiency of heterogeneous mobile robots using the combination of CS and BSO. In [15], the authors combined membrane computing with a genetic algorithm and the artificial potential field method for path planning.

In addition, there are some suggestions for applying the A* algorithm to tackle path planning issues. The A* algorithm is a heuristic search algorithm [16], which has the advantages of short search time, fast running speed, and simple operation, so it is frequently used in mobile robot

path planning [17]. However, in practical applications, the classic A* algorithm is shown to have various drawbacks [18], for example, not always ideal planning paths, numerous turning nodes, near proximity to obstacles, and low security, which are not conducive to the robot arriving at the final point without incident [19].

Therefore, in order to obtain better planning paths, many enhanced A* algorithms have been proposed. The authors in [20] proposed the JPS (A*) algorithm, which used a jump point search strategy to shorten the algorithm search time, but most of the time, the path obtained by its planning was not optimal. Based on the parent node's significance to the path search, the authors in [21] revised the weight coefficient of the algorithm, which reduced the search time, but the planned path had the risk of collision and was not smooth. The authors in [22] reduced the search space by weighting the improved heuristic function in an exponential decay manner but ignored the safe gap between the movable robot and the obstacle. According to the study of [23], the state hazard coefficient should be established in the environment modeling to assure the safety of the intended path, and an optimization approach should be used to eliminate any unnecessary nodes, but the resulting path was not a smooth curve.

It can be seen from the abovementioned analysis, although numerous academics have improved the A* algorithm from different perspectives to enhance its performance, there are relatively few researches on achieving optimal planning on the basis of ensuring security and taking into account the smooth processing of the path. If the size of the robot body is not taken into account, it is easy to encounter obstacles in path search, and the optimal path cannot be obtained. Therefore, we propose a node search method to set virtual obstacles to avoid collisions between the robot and the obstacles and eventually obtain a smooth path through the B-spline curve. The algorithm's efficacy and security are confirmed by simulation in both static and dynamic obstacle settings. Before concluding this section, the following are the main contributions of the study:

- (1) An improved eight domain node search strategy is introduced
- (2) The heuristic function of A* algorithm is ameliorated to make it closer to the actual cost
- (3) B-spline curve is used to smooth the planned path
- (4) The improved A* algorithm and dynamic window algorithm are combined to realize real-time obstacle avoidance

2. Grid Environment Modeling

When planning the path of a mobile robot, the establishment of environment model is the basis of robot control. Because of its straightforward structure, easy implementation, and strong fault tolerance to sensors, the grid method is frequently employed in robot path planning. In this study, the grid map [24] is used to abstract the real environment information obtained by various sensors into a two-

dimensional plan, and it is divided into several square grids of equal size. Using numbers 0, 1, 2, and 3 to set the state of the grid based on environmental information, 0 represents the idle state, which is shown in white on the map, meaning that the mobile robot can traverse this region. 1 indicates the occupied state, which is represented by black on the map, this is the obstacle area, which means that the mobile robot cannot pass through; 2 and 3 represent the starting point and the goal point, respectively, which are shown in yellow and red on the map [25] as shown in Figure 1.

Suppose that the robot moves in a two-dimensional plane area A with a limited number of obstacles and takes the upper left corner of A as the coordinate origin, the horizontal direction is the X axis, and the vertical direction is the Y axis. A rectangular coordinate system X - Y is constructed as the robot motion space. The maximum values in the horizontal and vertical directions are x_{\max} and y_{\max} separately, and the single step length of the robot is ρ , so the number of grids in each row and column is $n_x = x_{\max}/\rho$ and $n_y = y_{\max}/\rho$.

For any grid in the two-dimensional plane area A, there are certain coordinates and linear indices corresponding to it, as shown in Figure 2. $S = \{1, 2, 3, \dots, n_x * n_y\}$ is defined as the grid linear index set, take the upper left corner of the coordinate area as the origin, set the linear index number of each grid from top to bottom and from left to right, and the relationship between the coordinates (x_i, y_j) and the index number i is as follows:

$$\begin{aligned} x_i &= \text{ceil}\left(\frac{i}{n_y}\right), \\ y_i &= (i - 1) \bmod n_y + 1. \end{aligned} \quad (1)$$

3. Algorithm Introduction

The A* algorithm is based on the Dijkstra algorithm. First of all, we should understand the principles of Dijkstra algorithm and A* algorithm. The Dijkstra algorithm applies the greedy algorithm method to resolve the shortest path problem [26] which is applicable to the shortest path planning problem for a single source, that is, the shortest path from a vertex to all other nodes. The basic idea is as follows: once the full network has been explored, the node closest to the starting point and not yet visited is selected as the parent of the subsequent cycle. Therefore, it can always plan the shortest path from the starting point to any designated target point; on the other hand, because it is a blind search method, the algorithm search process has a large amount of computation [27]. Especially for large networks, the algorithm will take a long time to traverse the entire network.

In A* algorithm, the distance from the current location to the target point is called the estimated cost, and then the total cost function is formed together with the actual cost from the starting node to the current node to determine the search direction [28]. In each cycle, the algorithm determines the movement costs of the 8-neighborhood

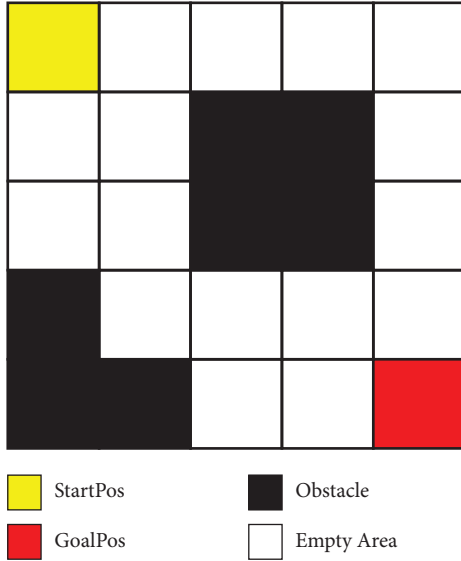


FIGURE 1: Grid map.

children around the parent node in the grid map and selects the child with the smallest cost value as the parent for the next cycle until the target node is searched. Since the algorithm does not need to traverse the entire network, its search efficiency is greatly improved compared with Dijkstra algorithm [29].

The total cost function of the A* algorithm is as follows:

$$f(m) = g(m) + h(m). \quad (2)$$

In formula (2), $g(m)$ is the actual cost from the starting node to the current node m , and $h(m)$ is the estimated cost from the current node m to the target node. $g(m)$ is equivalent to the shortest path from the starting node in Dijkstra's algorithm and is a known fixed value, so the value of the total cost $f(m)$ is determined by the size of the heuristic function $h(m)$. The key of the A* algorithm is to select a reasonable value for $h(m)$, which directly affects the search efficiency and solution quality of the algorithm.

4. Improved A* Algorithm

This section improves the A* algorithm by improving the 8 domain node search strategy and heuristic function and smoothing the path.

4.1. Virtual Obstacle Setting. When planning the path with the traditional A* algorithm, the robot is considered to be a moving particle, without considering the influence of factors such as robot volume, positioning accuracy, and distance error. Therefore, although an ideal path can be planned; there are certain potential security risks. For example, in Figure 3, the robot may cross the vertex of the obstacle in an oblique direction, resulting in a collision and damage to the robot; In Figure 4, the robot passes through the vertex between two obstacles. In actual road conditions, it is impossible to get through here [25].

To confirm that the robot can safely and reliably cross the obstacle area to the goal point. In [30], the A* algorithm is used to search discrete 8 neighborhoods and expand to infinity, which increases the search direction and improves the performance index of path smoothness. However, the increase in computational effort results in a significant decrease in search efficiency. The authors in [31] set virtual obstacles of a certain size around all obstacles to prevent possible collisions between robots and obstacles. Although this method is effective in avoiding collisions, the range of obstacle regions in the map is enlarged due to the oversetting of virtual obstacles, and there is a large gap with the actual map, which affects how accurately the next path will be planned. Therefore, we propose a modified virtual obstacle setting strategy in this paper.

When the current position of the robot is 5, it searches in the eight directions shown in Figure 5. If the grid point in this direction is a nonobstacle region, it can be considered as a feasible path, and the next step is to investigate it. However, in practical applications, not all nonobstacle regions are accessible because the robot has a certain volume. As shown in Figure 6, when judging whether a grid point 1 is reachable, firstly, we need to judge whether the point is an obstacle area. If it is a free area, we proceed to determine whether the adjacent points 4 and 2 are obstacle areas. As long as any of them is an obstacle region, grid point 1 is inaccessible and we set it as a virtual obstacle, denoted by cyan in the map. Similarly, the same strategy is used when deciding whether the other three diagonal directions are reachable or not.

4.2. Improvement of Heuristic Function. It is clear that the A* algorithm's search effectiveness and solution quality are determined by the $h(m)$, and the closer to the actual value, the higher the search efficiency. However, in a complex map environment, it is difficult for us to accurately estimate the value of $h(m)$. There are two main calculation methods for $h(m)$: Manhattan distance (equation (3)) and Euclidean distance (equation (4)).

$$h(m) = |x_m - x_n| + |y_m - y_n|, \quad (3)$$

$$h(m) = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}. \quad (4)$$

Manhattan distance is the heuristic function of the standard A* algorithm, which allows only four directions of movement up, down, left, and right. It is calculated as the total absolute wheelbase of two points in the standard coordinate system. The Euclidean distance, which can be moved at any angle, is the linear distance between two points. However, in the case of obstacles, there will be a big gap between the two calculation methods and the actual distance. The Euclidean distance is shorter than the actual distance, and the Manhattan distance is larger than the actual distance. In general, in order for the robot to reach the desired place quickly and safely, its motion pattern is generally a combination of oblique and horizontal motion as shown in Figure 7. Therefore, this study uses diagonal

	0	1	2	3	4	5	X
1	1	6	11	16	21		
2	2	7	12	17	22		
3	3	8	13	18	23		
4	4	9	14	19	24		
5	5	10	15	20	25		
Y							

FIGURE 2: Relationship between coordinates and linear index.

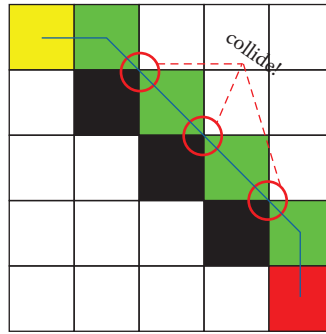


FIGURE 3: Path obliquely passes through obstacle vertices.

distance as a heuristic function $h(m)$ to estimate the mobile path cost [32].

The distance of oblique movement is as follows:

$$h_1(m) = \sqrt{2} * d_y. \quad (5)$$

The distance of its horizontal movement is as follows:

$$h_2(m) = d_x - d_y. \quad (6)$$

Therefore, the total moving distance of the mobile robot is as follows:

$$h(m) = h_1(m) + h_2(m) = \sqrt{2} * d_y + (d_x - d_y). \quad (7)$$

The abovementioned discussion is based on the example that the distance in the horizontal direction is greater than that in the vertical direction, which is generalized to the following general situations:

$$h(m) = \sqrt{2} * \min \{d_y, d_x\} + (d_y + d_x - 2 * \min \{d_y, d_x\}), \quad (8)$$

where $d_y = |y_m - y_n|$, $d_x = |x_m - x_n|$

(x_m, y_m) is the coordinate of the current node, and (x_n, y_n) is the coordinate of the target node. In the complex environment, the calculation result of the heuristic function $h(m)$ proposed is closer to the actual distance than the Manhattan distance and Euclidean distance. In addition, there are only some simple operations in the formula, and no

complex square and square root operations. The amount of calculation is small, which will not affect the efficiency of mobile robot path planning [33].

4.3. Path Smoothing Based on B-Spline Curve. After the abovementioned improvement of the A* algorithm, the quality of the planned path is greatly improved, but the path still lacks smoothness. Frequent turns can easily cause the robot to wobble, and the motion is not smooth. B-spline curve [34] has many excellent properties such as geometric invariance, convex hull, convexity preservation, local support, and so on. When the order is 3, the B-spline curve has the characteristics of second-order continuity at the node [35], which meets the requirements of the continuity of the acceleration and velocity of the robot movement. Therefore, the third-order B-spline curve is used for path smoothing in this study.

The definition of the k -th degree B-spline curve equation is as follows [36]:

$$P_{m,k}(t) = \sum_{i=0}^k P_{m+i} G_{i,k}(t), t \in [0, 1], \quad (9)$$

where $P_{m,k}(t)$ ($m = 0, 1, \dots, n - k$) is the k -th degree B-spline curve segment of the m -th segment, there are $n - k + 1$ curve segments in total, and the whole of these curve segments is called the k -th degree B-spline curve;

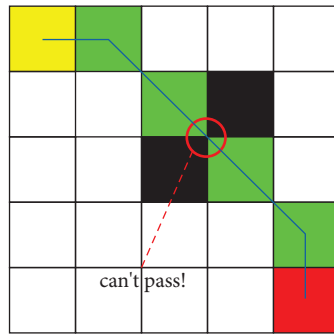


FIGURE 4: Path passes through two obstacle vertices.

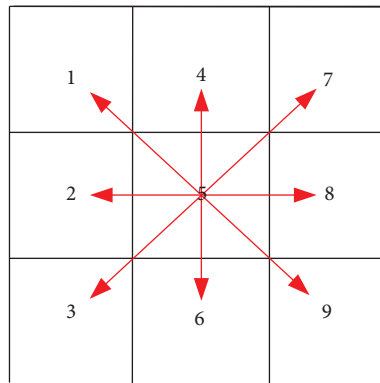


FIGURE 5: The direction of robot movement.

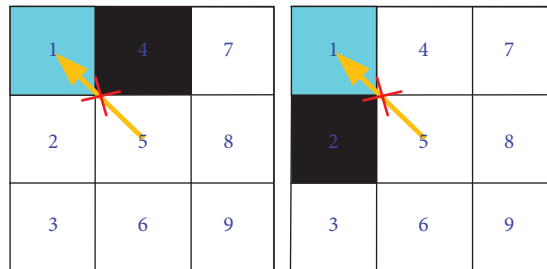


FIGURE 6: Strategies for setting virtual barriers.

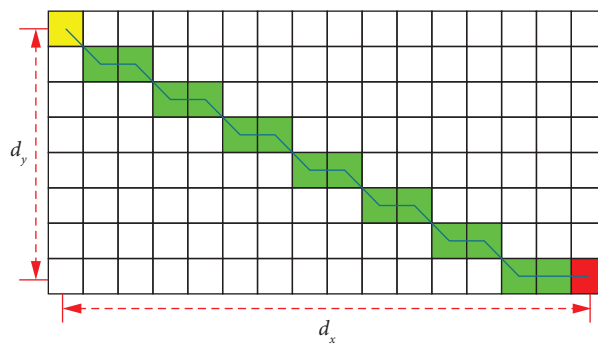


FIGURE 7: Diagonal movement.

P_{m+j} ($m = 0, 1, \dots, n - k$) is the vertex of the control polygon, there are $n + 1$ in total; $G_{i,k}(t)$ ($i = 0, 1, \dots, k$) is called the basis function of the k -th B-spline curve, and its expression is as follows:

$$G_{i,k}(t) = \frac{1}{k!} \sum_{j=0}^{k-i} (-1)^j C_{k+1}^j (t + k - i - j)^k \quad (10)$$

$$t \in [0, 1], i = 0, 1, \dots, k,$$

when $k = 3$, then:

$$\left\{ \begin{array}{l} G_{0,3}(t) = \frac{1}{6}(-t^3 + 3t^2 - 3t + 1), \\ G_{1,3}(t) = \frac{1}{6}(3t^3 - 6t^2 + 4), \\ G_{2,3}(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1), \\ G_{3,3}(t) = \frac{1}{6}t^3, \end{array} \right. \quad t \in [0, 1]. \quad (11)$$

Substituting (11) into (9), the equation of the 3-order uniform B-spline curve is as follows:

$$P_{m,3}(t) = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_m \\ P_{m+1} \\ P_{m+2} \\ P_{m+3} \end{bmatrix}, t \in [0, 1]. \quad (12)$$

By substituting the node coordinates of the planned path into the above formula, the spline curve equation corresponding to the control points can be determined [37]. With the continuous value of t , a smooth B-spline curve can be drawn, thereby realizing the smooth processing of the path [38]. As shown in Figure 8, it is a cubic B-spline curve controlled by five control vertices, which consist of three B-spline curve segments. Among them, each curve segment is controlled by four vertices.

4.4. Application of Improved A* Algorithm in Path Planning. The process of path planning using the improved A* algorithm is as follows:

Step 1: build the environment map and initialize parameters.

Step 2: define two lists: openList stores the nodes to be investigated, closedList stores the investigated nodes, and puts the starting node S into openList.

Step 3: the node in the openList with the least $f(m)$ value should be chosen as the parent, added to the close list, and eliminated from the openList.

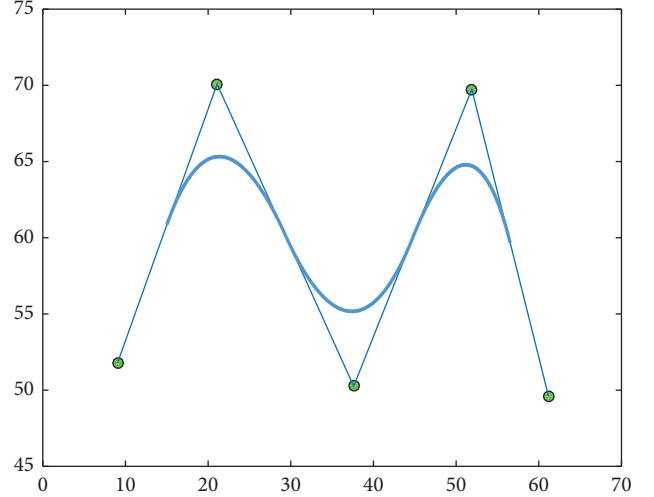


FIGURE 8: Example of B-spline curve.

Step 4: judge whether the parent node is the goal node. If yes, the search process ends and the planning path is generated; otherwise, the 8-neighborhood children of the parent are generated and the nodes in the obstacle region and those already present in the proximity list are excluded.

Step 5: further investigate the child nodes and set up virtual obstacles according to the strategy in Section 4.1.

Step 6: judge whether the remaining child node in the previous step is in the openList. If yes, recalculate the movement cost $f(m)$, compare it with the original value, keep the smaller value, and update the path; if not, append node m to openList, calculate its value of $f(m)$, and update the path.

Step 7: go back to Step 3 and enter the loop search until the goal point G is searched.

Step 8: smooth the planned path by using the cubic uniform B-spline curve.

Figure 9 shows the flow chart of the improved A* algorithm.

Algorithm 1 presents the improved A* algorithm pseudocode. The improved A* algorithm employs the next input parameters: the starting point, S, and the goal point, G, an openList of states for expansion, a closedList of states for completion.

5. Experimental Simulation and Result Analysis

For the purpose of verifying the abovementioned theoretical analysis and the effectiveness of the improvement of A* algorithm, the MATLAB 2018a experimental platform was used to conduct simulation experiments. In the static obstacle environment, the traditional A* algorithm (TAA), the A* algorithm with virtual obstacle (AAO), the improved A* algorithm (IAA), Dijkstra algorithm and the algorithm in reference [31] are, respectively, carried out four groups of simulation experiments. The grid map environment selects $30 * 20$ with a low obstacle rate, $50 * 40$ with a low obstacle

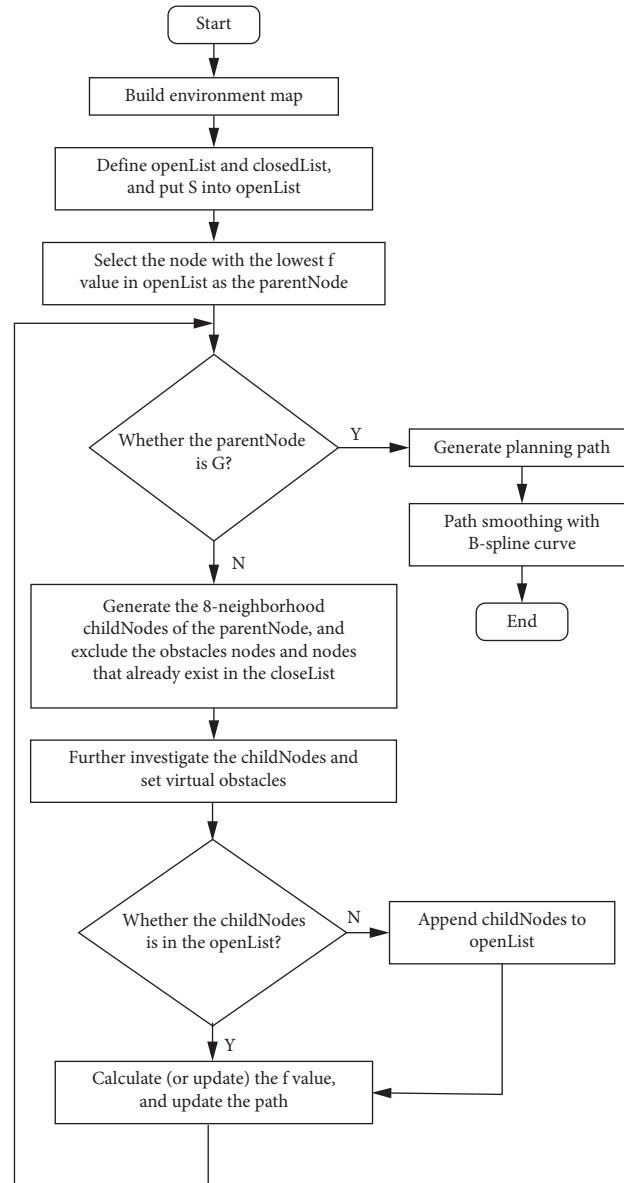


FIGURE 9: Flow chart of the improved A* algorithm.

rate and $50 * 40$ with a high obstacle rate, $100 * 100$ with a high obstacle rate respectively. A set of simulation experiments were implemented in a dynamic obstacle environment. The hardware platform is: Windows 10 operating system, Intel Core i5-7200U @2.4 GHz with 8 G RAM.

Algorithm description: in order not to affect the timeliness of mobile robot path planning, the heuristic function $h(m)$ of the traditional A* algorithm and A* algorithm with virtual obstacle is calculated as Manhattan distance; the improved A* algorithm is contrasted with the Dijkstra algorithm to further confirm its efficacy. Since the Dijkstra algorithm is a blind search, its planned path must be optimal. Moreover, for a better experimental comparison analysis, in this study, we add virtual obstacles to the Dijkstra algorithm to avoid collisions, and the path is also smoothed.

In order to exclude the influence of other factors on the experiment, we conducted 20 repeated experiments for each group of experiments, and the experimental results showed that the paths planned by each algorithm were consistent each time. Furthermore, when processing the experimental data, the time costs calculated by each algorithm were recorded, and their average values were taken to eliminate the interference of random errors.

*5.1. 30 * 20 Grid Environment with 15.8% Obstacle Rate.* First, create a grid map with dimensions of $30 * 20$. In the map, the starting point is shown in yellow, the goal point is shown in red, and the obstacle area is shown in black, and the proportion of obstacles is 15.8%, green for the planned

```

Input: S, G
Output: the optimal path
(1) initialize openList and closedList
(2) set Pn: the least costly node in the openList
(3) while true do
(4)   find the 8 neighborhood nodes of Pn
(5)   whether the neighborhood node is in the openList
(6)   if in the openList then
(7)     compute g(m), h(m) and f(m)
(8)   else add to the openList
(9)   end
(10)  repeat search the least costly node in the openList
(11)  if G in the closedList then
(12)    break
(13)  end
(14)  path-smoothing with B-spline curve
(15)  return the optimal path

```

ALGORITHM 1: A* pseudocode.

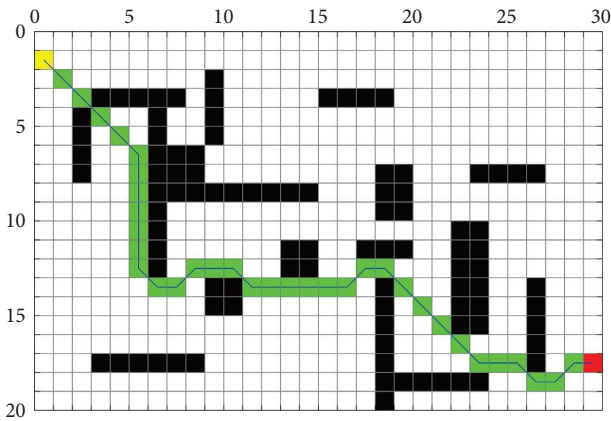


FIGURE 10: Path planned by TAA (30 * 20, 15.8%).

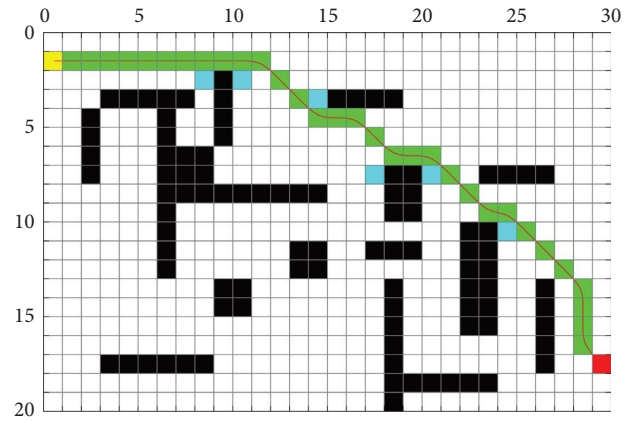


FIGURE 12: Path planned by IAA (30 * 20, 15.8%).

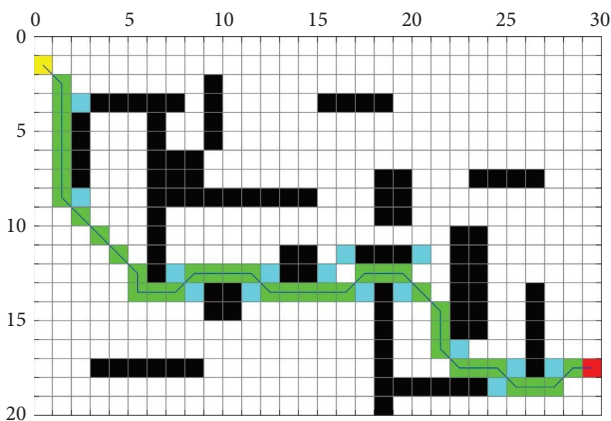


FIGURE 11: Path planned by AAO (30 * 20, 15.8%).

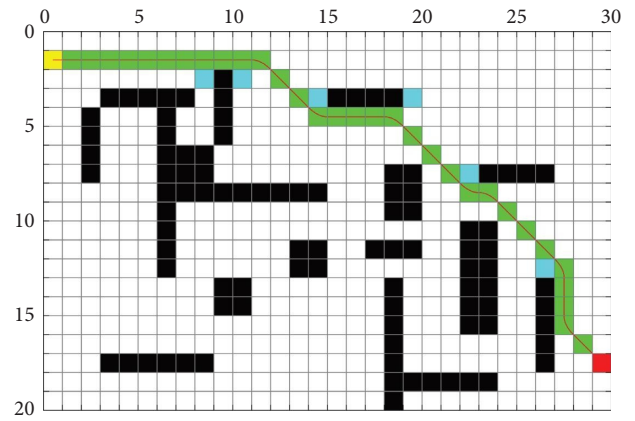


FIGURE 13: Path planned by Dijkstra (30 * 20, 15.8%).

path, and cyan for the virtual obstacle. The path planning results obtained by each algorithm are shown in Figures 10–14, respectively. At the same time, Table 1

displays the number of moving grids, path length, number of virtual obstacles, and search time of the five algorithms.

TABLE 1: Experimental results comparison of five algorithms (30 * 20, 15.8%).

Algorithms	Number of moving grids	Path length	Number of virtual obstacles	Search time (s)
TAA	36	41.6274	0	0.137
AAO	39	43.3848	15	0.176
IAA	33	38.2132	6	0.167
Dijkstra	33	38.2132	6	0.351
Reference [31]	40	42.3137	219	0.395

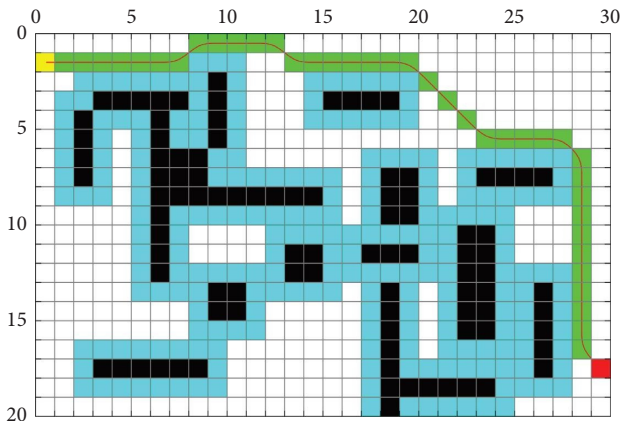


FIGURE 14: Path planned by reference [31] (30 * 20, 15.8%).

Figure 10 makes it evident that the path chosen by the traditional A* algorithm is in close proximity to the obstacle, there are multiple collisions, and in the actual map environment, there is a risk that some paths will not pass at all. Therefore, virtual obstacles are added on the basis of the traditional algorithm to avoid the occurrence of the above two phenomena, and Figure 11 displays the results of the path planning. The figure shows that the virtual obstacle is set so that the mobile robot is safely separated from the obstacle, which guarantees the safety of the planned path. However, this planned path has the problem of sharp turns (large changes in curvature) between straight lines. Thus, this paper uses cubic uniform B-spline curve to smooth the planned path, as shown in Figures 12 and 13. A path planning experiment that was completed using the enhanced A* algorithm from the reference [31] is shown in Figure 14.

Table 1 demonstrates that when compared to the traditional A* algorithm, the number of moving grids planned by the A* algorithm with virtual obstacles increases by 3, the path length increases by 4.02%, and the search time increases by 22.16%. This is primarily to guarantee the safety of path planning; the effect of increasing the path length caused by avoiding obstacles is unavoidable. Compared with the A* algorithm with virtual obstacles, the number of moving grids planned by the improved A* algorithm is reduced by 6, the path length is reduced by 11.92%, and the search time is basically the same. This is because the value of a heuristic function $h(m)$ in the improved A* algorithm is closer to the real cost than the Manhattan distance, which makes the search result closer to the optimal path. Moreover, by

comparing the improved A* algorithm with the Dijkstra algorithm, we find that the two algorithms produce paths with the same number of moving grids and length. We know that the Dijkstra algorithm uses blind search, so its planned path must be optimal. Therefore, so it is patently obvious that the improved A* algorithm can obtain the optimal path on this map scale, and its search time is 52.42% less than the Dijkstra algorithm. The algorithm in reference [31] not only increases the robot's security but also its running duration and path length.

5.2. 50 * 40 Grid Environment with 14.1% Obstacle Rate.

To further confirm the improved A* algorithm's viability in a large-scale map environment, a grid map with a size of 50 * 40 and an obstacle rate of 14.1% was used as the experimental environment for the experiment. The path planning results obtained by each algorithm are shown in Figures 15–19. Table 2 displays the data analysis.

From this experiment, we can see that in the environment of a large map with a low obstacle rate, the improved A* algorithm, the A* algorithm with virtual obstacles and the Dijkstra algorithm can all ensure the security of path planning. Table 2 shows that under the 50 * 40 map size, the search time of the five algorithms has increased to varying degrees, and the Dijkstra algorithm has the largest increase, which is 3.26 times that of the 30 * 20 map size. The length of the improved A* algorithm's planned path is equal to that of the Dijkstra algorithm, indicating that the improved A* algorithm can also achieve the optimal planned path for the 50 * 40 map size. Moreover, compared with the A* algorithm with virtual obstacles, the path length of the optimal path planned is reduced by 5.72%, and the search time is reduced by 71.51% compared with Dijkstra algorithm.

5.3. 50 * 40 Grid Environment with 38.9% Obstacle Rate.

Above, we have verified that the improved algorithm has achieved good performance under low obstacle rate grid maps of 30 * 20 and 50 * 40. In order to verify the universality of the improved algorithm in the high obstacle map environment, the map environment with the size of 50 * 40 and the obstacle rate of 38.9% is selected for experiment. Figures 20–24 display the results of the path planning, and Table 3 displays a comparison of the data.

Table 3 shows that in the high obstacle map environment, although the length of the planned path of each algorithm has increased compared with the previous group of

TABLE 2: Experimental results comparison of four algorithms (50 * 40, 14.1%).

Algorithms	Number of moving grids	Path length	Number of virtual obstacles	Search time (s)
TAA	60	69.7696	0	0.141
AAO	69	75.8701	8	0.197
IAA	63	71.5269	11	0.316
Dijkstra	63	71.5269	10	1.109
Reference [31]	66	73.2843	335	1.533

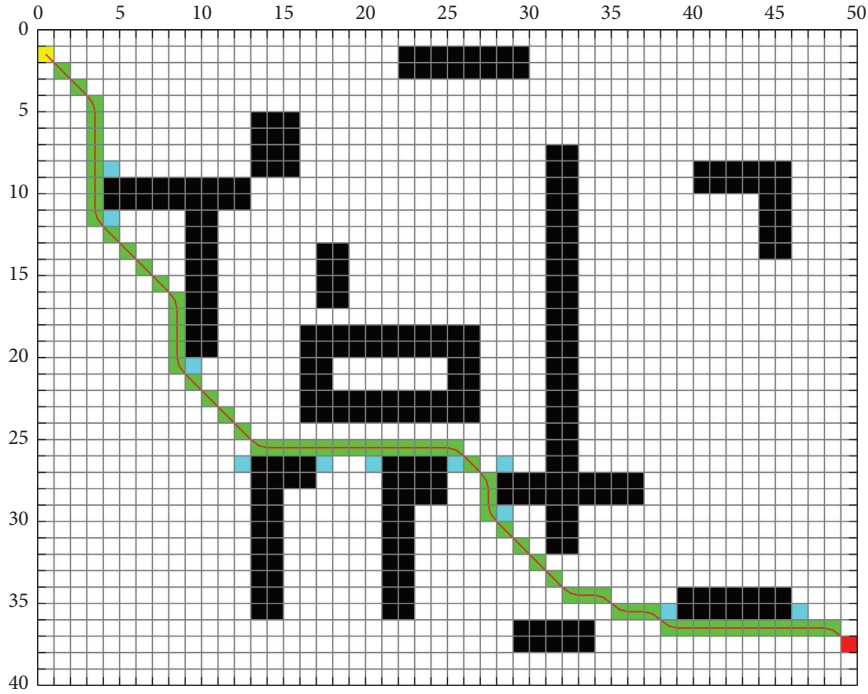


FIGURE 17: Path planned by IAA (50 * 40, 14.1%).

algorithm in the study of [31] cannot plan a proper path in this obstacle environment as shown in Figure 24.

5.4. 100 * 100 Grid Environment with 30% Obstacle Rate. In order to verify the universality of the improved algorithm in the complex map environment, a map environment with the size of 100 * 100 was selected for the experiment. Figures 25–29 display the results of the path planning, and Table 4 displays a comparison of the data.

The results can clearly be seen, and the improved A* algorithm has a good optimization effect on the path and can also effectively improve the safety quality of the path and generate smooth paths in complex environments. The algorithm in the study of [31] is only suitable for path planning in low obstacle rate environments. A high obstacle rate can lead to mission failure as shown in Figure 29.

5.5. Dynamic Obstacle Environment. In practice, there are not only known static obstacles, but also unknown dynamic obstacles that hinder the movement of the robot, which leads to the failure of path planning. Aiming at the dynamic path planning of robots in complex environment, the improved

A* algorithm and dynamic window algorithm (DWA) [39] are combined to realize dynamic obstacle avoidance.

DWA converts the position constraint of the mobile robot into a velocity constraint, establishes speed sampling space based on constraints and predicts its running trajectory in the next period of time. Finally, the trajectory with the highest score is selected as the robot's moving path based on the evaluation function. The DWA consists of three parts: kinematics model, velocity sampling space, and evaluation function.

5.5.1. Kinematics Model. Assuming that v_t, ω_t respectively represent the linear velocity and angular velocity of the robot at time t , and it approximately moves in a uniform straight line within the sampling period Δt , then its kinematics model can be expressed as follows:

$$\begin{cases} x_{t+1} = x_t + v_t \Delta t \cos \theta_t, \\ y_{t+1} = y_t + v_t \Delta t \sin \theta_t, \\ \theta_{t+1} = \theta_t + \omega_t \Delta t. \end{cases} \quad (13)$$

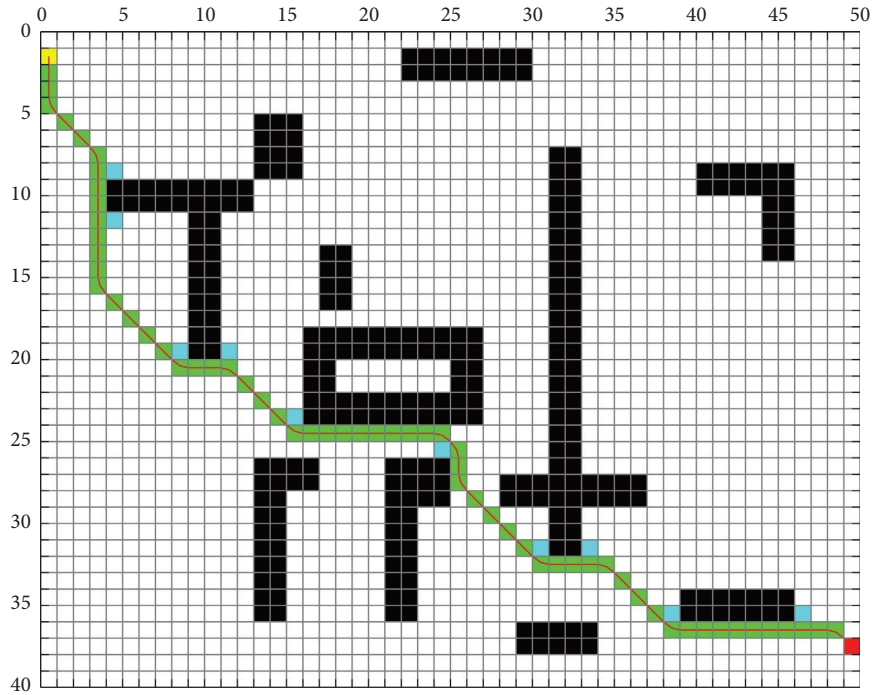


FIGURE 18: Path planned by Dijkstra (50 * 40, 14.1%).

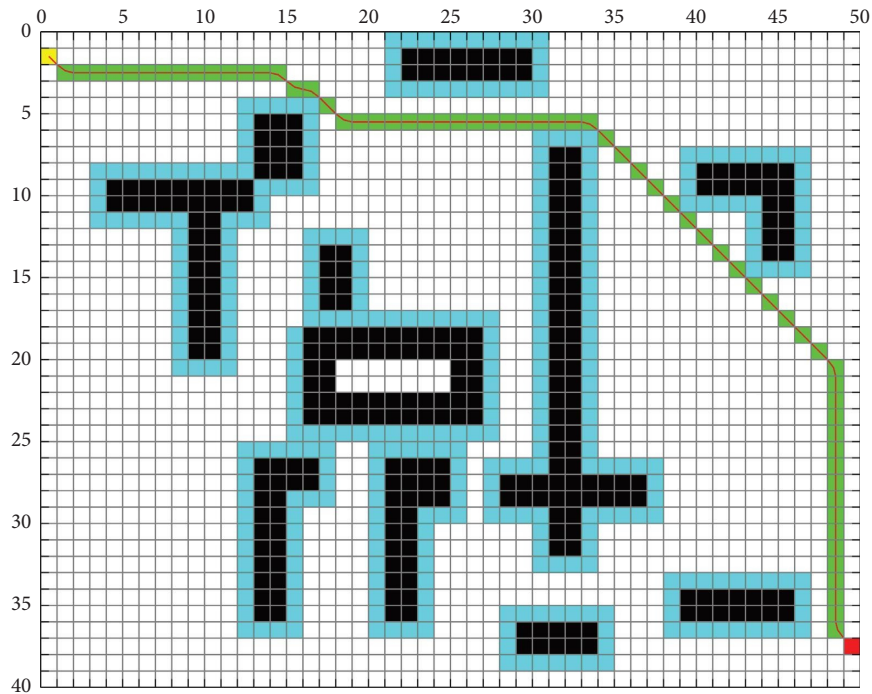


FIGURE 19: Path planned by reference [31] (50 * 40, 14.1%).

5.5.2. Sampling Space (Dynamic Window)

Constraint 1. The movement speed of the robot

$$V_m = \{(v, \omega) | v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\}, \quad (14)$$

where v_{\min}, ω_{\min} are the minimum linear velocity and angular velocity; v_{\max}, ω_{\max} are the maximum linear velocity and angular velocity.

Constraint 2. Acceleration/deceleration of the motor

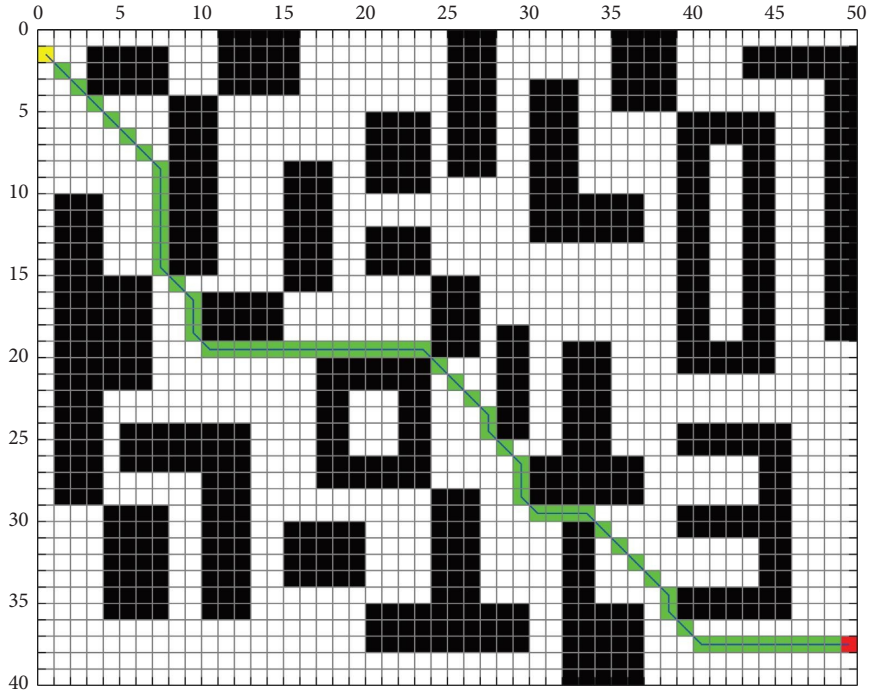


FIGURE 20: Path planned by TAA (50 * 40, 38.9%).

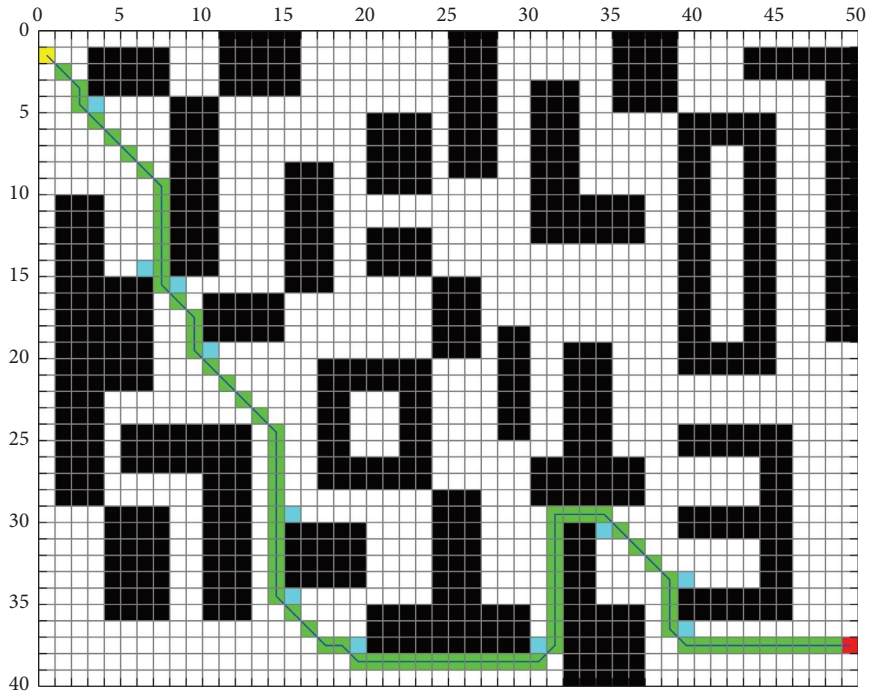


FIGURE 21: Path planned by AAO (50 * 40, 38.9%).

$$V_d = \left\{ \begin{array}{l} (v, \omega) | v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t], \\ \omega \in [\omega_c - \dot{\omega}_b \Delta t, \omega_c + \dot{\omega}_a \Delta t] \end{array} \right\}, \quad (15)$$

where v, ω represent current speed; $\dot{v}_a, \dot{\omega}_a$ are maximum acceleration; $\dot{v}_b, \dot{\omega}_b$ are maximum deceleration.

Constraint 3. Safety braking distance

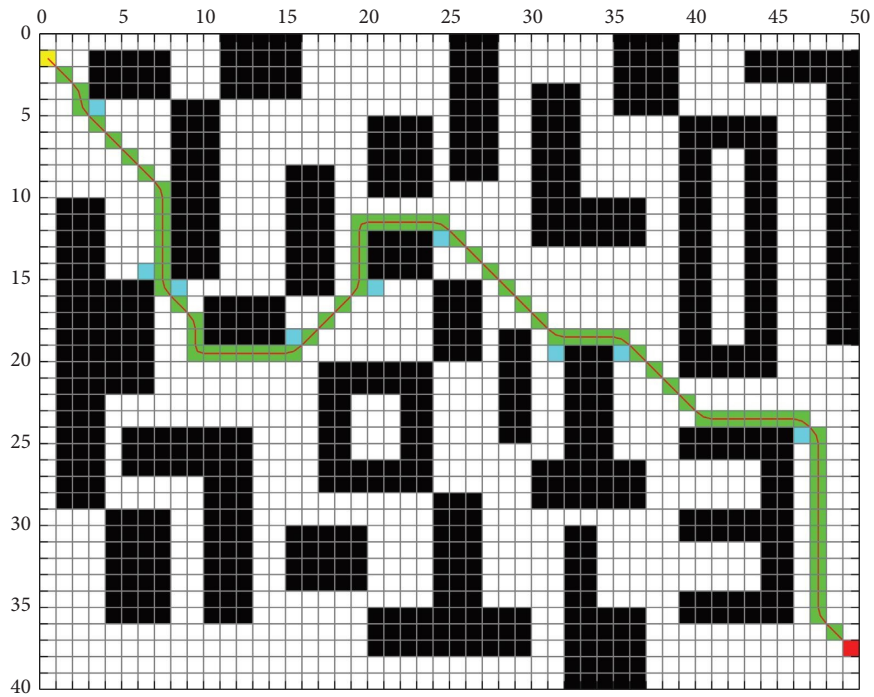


FIGURE 22: Path planned by IAA (50 * 40, 38.9%).

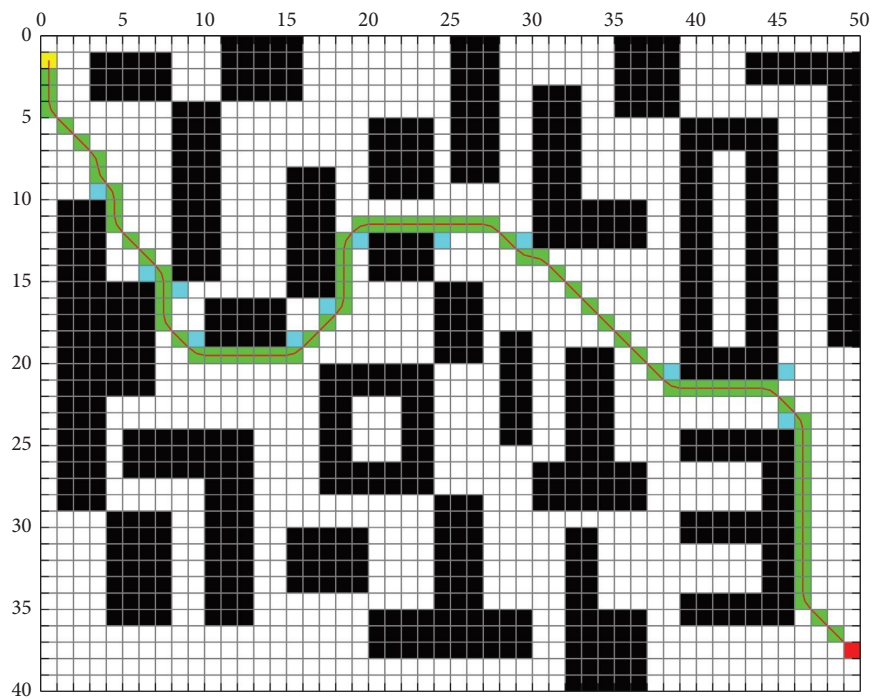


FIGURE 23: Path planned by Dijkstra (50 * 40, 38.9%).

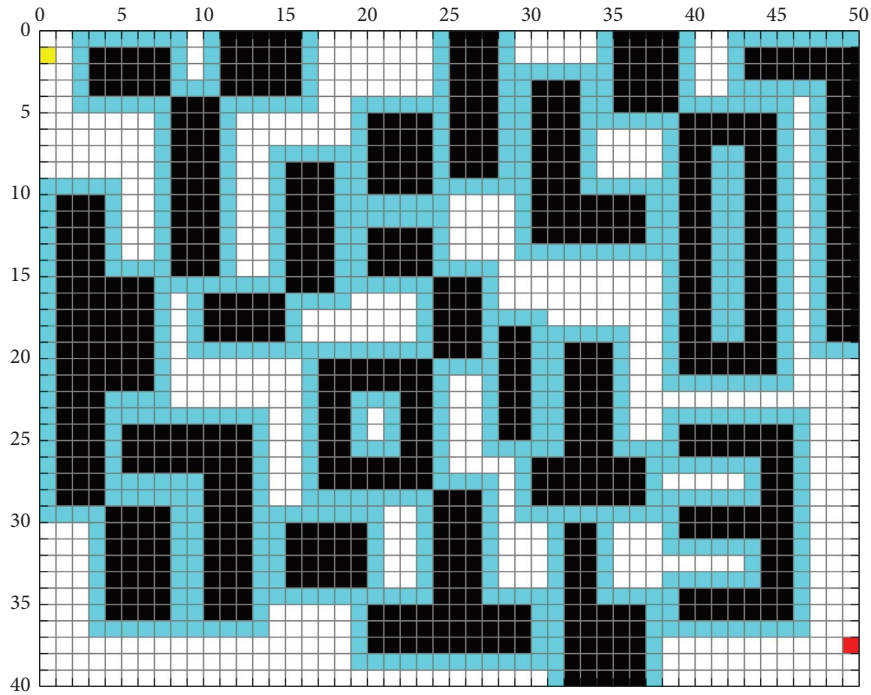


FIGURE 24: Path planned by reference [31] (50 * 40, 38.9%).

TABLE 3: Experimental results comparison of four algorithms (50 * 40, 38.9%).

Algorithms	Number of moving grids	Path length	Number of virtual obstacles	Search time (s)
TAA	62	70.9411	0	0.226
OAA	80	88.9411	11	0.424
IAA	74	84.5983	9	0.592
Dijkstra	74	84.5983	12	1.205
Reference [31]	—	Failed	—	—

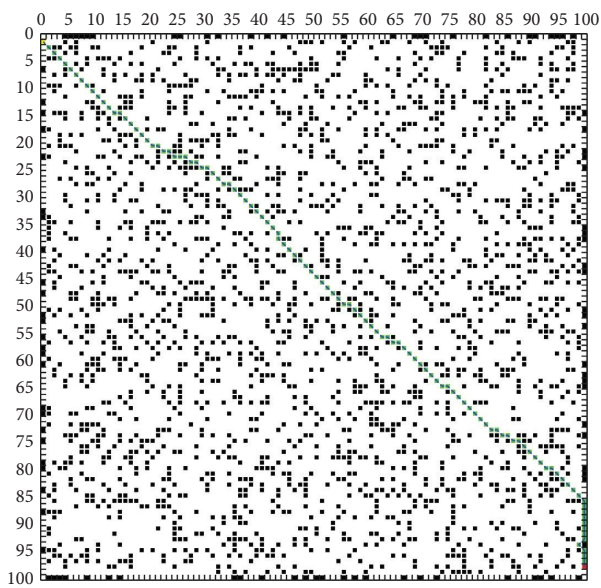


FIGURE 25: Path planned by TAA (100 * 100, 30%).

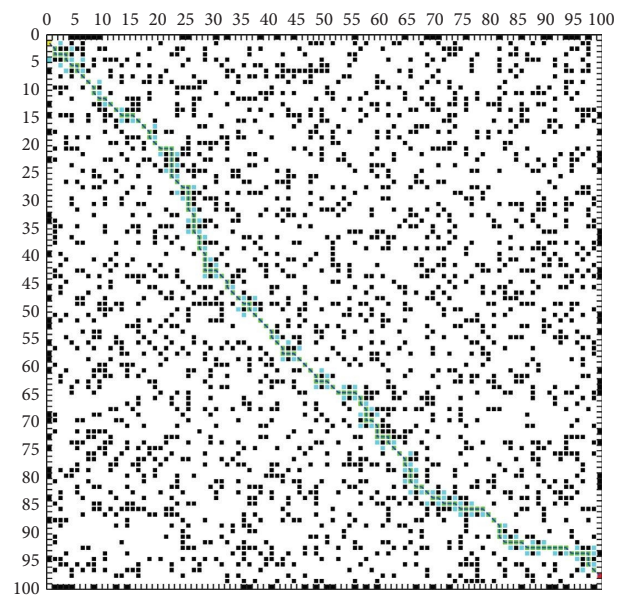


FIGURE 26: Path planned by AAO (100 * 100, 30%).

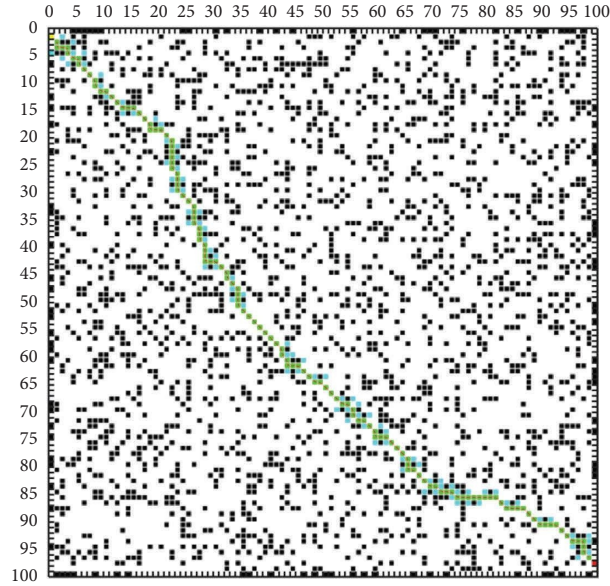


FIGURE 27: Path planned by IAA (100 * 100, 30%).

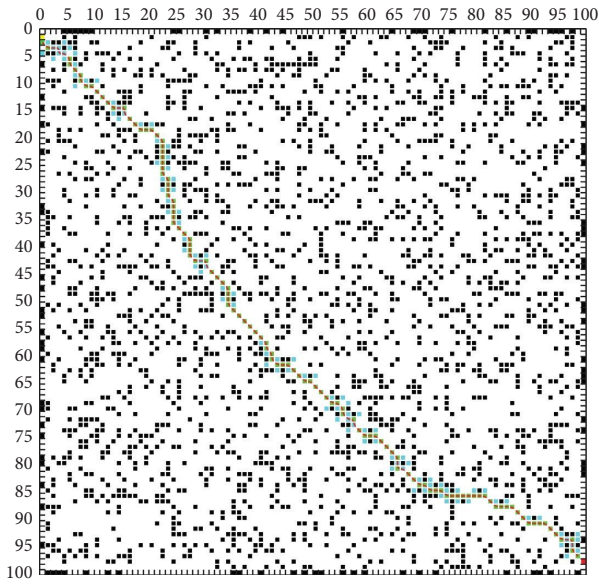


FIGURE 28: Path planned by Dijkstra (100 * 100, 30%).

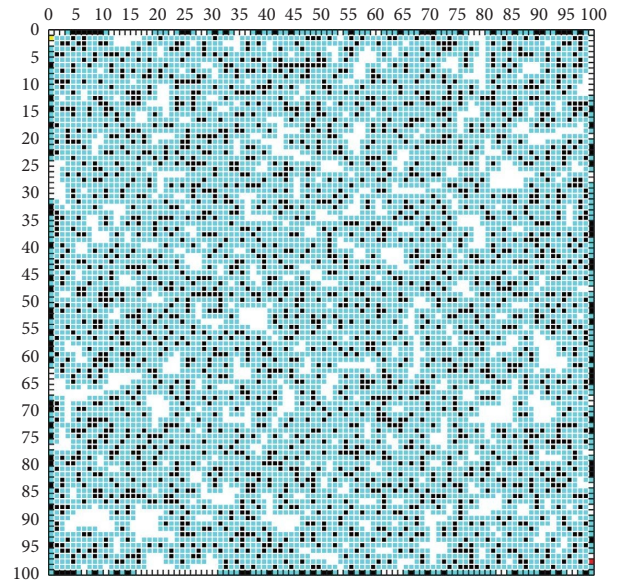


FIGURE 29: Path planned by reference [31] (100 * 100, 30%).

$$V_a = \left\{ (v, \omega) \mid v \leq \sqrt{2\text{dist}(v, \omega)v_b}, \omega \leq \sqrt{2\text{dist}(v, \omega)\omega_b} \right\}, \quad (16)$$

where $\text{dist}(v, \omega)$ is the closest distance from the current trajectory to the obstacle.

5.5.3. Evaluation Function

$$G(v, \omega) = \sigma(\text{ahead}(v, \omega) + \beta\text{dist}(v, \omega) + \gamma\text{vel}(v, \omega)), \quad (17)$$

where $\text{head}(v, \omega)$, $\text{dist}(v, \omega)$, $\text{vel}(v, \omega)$ are the evaluation functions for direction angle, distance, and velocity,

respectively; σ is the smoothing coefficient; α, β, γ are the weighting coefficients for these three items.

The improved A* algorithm can obtain the optimal solution of global path planning in static environment, but it cannot avoid the unknown obstacles in time, but the DWA has good local real-time obstacle avoidance ability. Therefore, this paper gathers the advantages of both, firstly, the improved A* algorithm is used to carry out global path planning. When a dynamic obstacle is detected, a dynamic window approach is involved to perform local path planning, which enables real-time dynamic obstacle avoidance based on ensuring optimal global paths.

In order to verify the feasibility of the improved A* algorithm in a dynamic environment, a path planning

TABLE 4: Experimental results comparison of four algorithms (100 * 100, 30%).

Algorithms	Number of moving grids	Path length	Number of virtual obstacles	Search time (s)
TAA	113	147.2082	0	0.462
AAO	140	162.1960	129	0.761
IAA	133	158.0955	111	4.812
Dijkstra	133	158.0955	108	15.232
Reference [31]	—	Failed	—	—

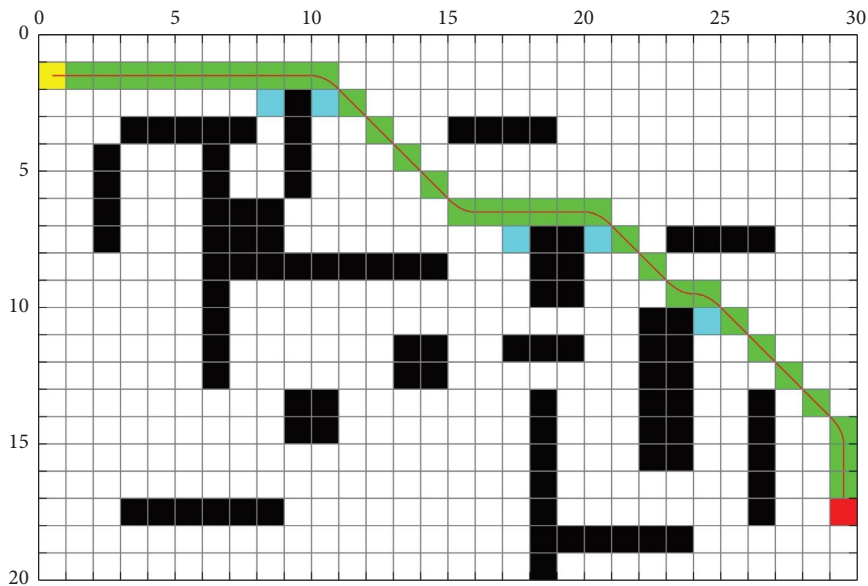


FIGURE 30: Static obstacle environment.

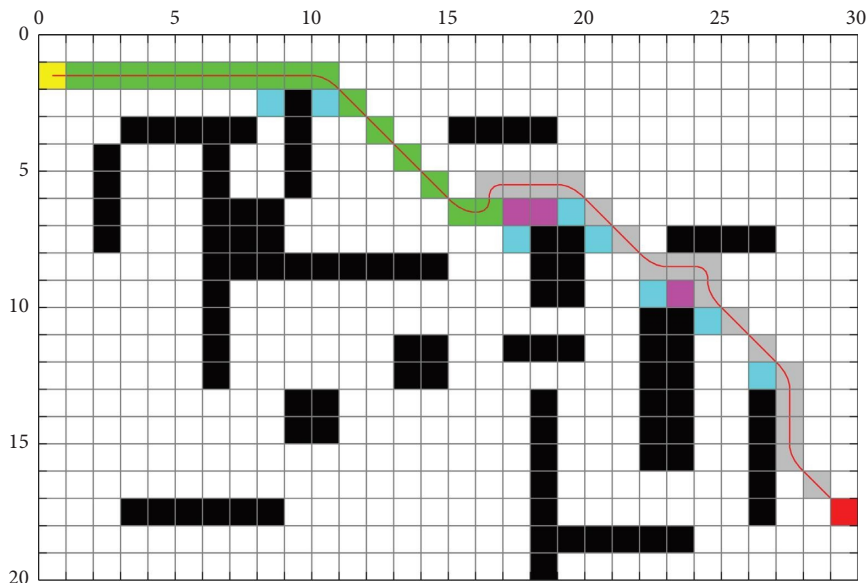


FIGURE 31: Dynamic obstacle environment.

simulation experiment based on a dynamic environment was conducted, as shown in Figures 30 and 31. Figure 30 shows the path planning diagram of the improved A* algorithm in a static obstacle environment. Figure 31 shows

the path planning diagram of the improved A* algorithm in a dynamic obstacle environment. Here, pink represents dynamic obstacles and gray represents the planned path after encountering dynamic obstacles. It is patently obvious that

the improved fusion algorithm can realize local path planning, avoid dynamic obstacles, and obtain smooth and safe planned paths.

6. Conclusion

This paper proposes a new improved A* algorithm to solve the path planning problem of mobile robots by setting up virtual obstacles, improving the heuristic function and smoothing the path. In addition, we performed comprehensive simulation experiments and tested it in different static and dynamic scenarios. The results verified the accuracy, security and superiority of the improved A* algorithm.

The future direction of the work is as follows:

- (1) Test the proposed A* algorithm on a real mobile robot
- (2) Shorten the time of path planning and reduce the length of path on the basis of path security

Data Availability

The labeled data set used to support the findings of this study is available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Huifang Bao performed the data analyses and wrote the manuscript; Jie Fang designed research schemes and feasibility analysis of research schemes; Chaohai Wang collated documents and participated in writing papers; Zebin Li and Jinsi Zhang designed research plans and conducted simulation experiments; Chuansheng Wang completely revised the grammar errors in the manuscript and conducted experiments on dynamic path planning.

Acknowledgments

This work was supported by West Anhui University Project (Grant nos. WXZR202015 and WXZR202120), Anhui Provincial Education Department Project (Grant no. 2022AH051675), and projects entrusted by enterprises and institutions (Development of AGV for String Rod Steel Coil Stacking).

References

- [1] B.-K. Patle, D. Parhi, A. Jagadeesh, and S. K. Kashyap, "Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot," *Computers & Electrical Engineering*, vol. 67, pp. 708–728, 2018.
- [2] D. G. Sunita and D. Garg, "Dynamizing Dijkstra: a solution to dynamic shortest path problem through retroactive priority queue," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 3, pp. 364–373, 2021.
- [3] D. Lyu, Z. Chen, Z. Cai, and S. Piao, "Robot path planning by leveraging the graph-encoded Floyd algorithm," *Future Generation Computer Systems*, vol. 122, no. 7, pp. 204–208, 2021.
- [4] Y. Chai, Y. Zhao, and C. Yin, "Indoor top mobile service robot based on improved RRT algorithm," *Journal of Physics: Conference Series*, vol. 2383, Article ID 012091, 2022.
- [5] S. Chen, J. Zhu, M. Wang et al., "Comparison of the therapeutic effects of adipose-derived and bone marrow mesenchymal stem cells on erectile dysfunction in diabetic rats," *International Journal of Molecular Medicine*, vol. 44, no. 3, pp. 1006–1014, 2019.
- [6] F. A. Raheem and U. I. Hameed, "Heuristic D* algorithm based on particle swarm optimization for path planning of two-link robot arm in dynamic environment," *Al-Khwarizmi Engineering Journal*, vol. 15, no. 2, pp. 108–123, 2019.
- [7] Z. H. Liu, H. B. Liu, Q. Zeng, and Z. Lu, "A dynamic fusion pathfinding algorithm using delaunay triangulation and improved A-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.
- [8] M. T. Lee, B. Y. Chen, and Y. C. Lai, "A hybrid tabu search and 2-opt path programming for mission route planning of multiple robots under range limitations," *Electronics*, vol. 9, no. 3, p. 534, 2020.
- [9] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022.
- [10] X. W. Wang, H. Shi, and C. Zhang, "Path planning for intelligent parking system based on improved ant colony optimization," *IEEE Access*, vol. 8, pp. 65267–65273, 2020.
- [11] H. Ding, "Motion path planning of soccer training auxiliary robot based on genetic algorithm in fixed-point rotation environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 12, pp. 6261–6270, 2020.
- [12] Q. Xiong, H. Zhang, and Q. Rong, "Path planning based on improved particle swarm optimization for AUVs," *Journal of Coastal Research*, vol. 111, no. 1, 2020.
- [13] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, no. 1, pp. 156787–156803, 2019.
- [14] D. Chen, Z. Wang, G. Zhou, and S. Li, "Path planning and energy efficiency of heterogeneous mobile robots using cuckoo-beetle swarm search algorithms with applications in UGV obstacle avoidance," *Sustainability*, vol. 14, no. 22, Article ID 15137, 2022.
- [15] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied Soft Computing*, 2019.
- [16] J. Z. Shi, Y. F. Su, C. G. Bu, and X. Fan, "A mobile robot path planning algorithm based on improved A*," *Journal of Physics: Conference Series*, vol. 1486, no. 3, Article ID 032018, 2020.
- [17] T. Rainer and N. Uchiyama, "Probabilistic robust path planning for nonholonomic arbitrary-shaped mobile robots using a hybrid A* algorithm," *IEEE Access*, vol. 99, p. 1, 2021.
- [18] H. Wang, X. Qi, S. Lou, J. Jing, H. He, and W. Liu, "An efficient and robust improved A* algorithm for path planning," *Symmetry*, vol. 13, no. 11, Article ID 2213, 2021.
- [19] Z. Xu and W. Yuan, "Mobile robot path planning based on fusion of improved A* algorithm and adaptive DWA algorithm," *Journal of Physics: Conference Series*, vol. 2330, 2022.

- [20] F. Duchon, A. Babinec, M. Kajan et al., "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.
- [21] M. Lin, K. Yuan, and C. Shi, "Path planning of mobile robot based on improved A* algorithm," in *Proceedings of the 2017 29th Chinese Control and Decision Conference*, pp. 3570–3576, IEEE, Piscataway, NJ, USA, August 2017.
- [22] W. Wang, P. Dong, and F. Zhang F, "The shortest path planning for mobile robots using improved A* algorithm," *Journal of Computer Applications*, vol. 38, no. 5, pp. 1523–1526, 2018.
- [23] H.-M. Zhang, M.-L. Li, and L. Yang, "Safe path planning of mobile robot based on improved A* algorithm in complex terrains," *Algorithms*, vol. 11, no. 4, pp. 44–18, 2018.
- [24] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.
- [25] X.-Q. Shen, Z.-P. Ma, and Q.-Z. Sun, "Improved A* algorithm in path planning of mobile robot," *Journal of Heilongjiang University of Science & Technology*, vol. 31, no. 4, pp. 494–499, 2021.
- [26] Y.-D. Rosita, E.-E. Rosyida, and M.-A. Rudiyanto, "Implementation of Dijkstra algorithm and multi-criteria decision-making for optimal route distribution," *Procedia Computer Science*, vol. 161, pp. 378–385, 2019.
- [27] Y.-Z. Chen, S.-F. Shen, T. Chen, and R. Yang, "Path optimization study for vehicles evacuation based on Dijkstra algorithm," *Procedia Engineering*, vol. 71, pp. 159–165, 2014.
- [28] H.-J. Jiang and Y. Sun, "Research on global path planning of electric disinfection vehicle based on improved A* algorithm," *Energy Reports*, vol. 7, pp. 1270–1279, 2021.
- [29] S. Jaiswal and S. Soumya, "Low-cost path planning in 2D environment using A* algorithm by considering slope of the obstacle," *IFAC-PapersOnLine*, vol. 55, no. 1, pp. 783–788, 2022.
- [30] Y. Xin, H. Liang, and M. Du, "An Improved A* Algorithm for Searching Infinite Neighborhoods," *Robot*, vol. 36, pp. 627–633, 2014.
- [31] H.-J. Chai, J.-J. Li, and M. Yao, "Improved A* algorithm for path planning of mobile robot," *Chinese Journal of Electron Devices*, vol. 44, no. 2, pp. 362–367, 2021.
- [32] B. Fu, L. Chen, Y.-T. Zhou et al., "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robotics and Autonomous Systems*, vol. 106, pp. 26–37, 2018.
- [33] Red Blob Games, "Heuristics from amit's thoughts on pathfinding," 2016, <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html#heuristics-for-grid-maps>.
- [34] Z.-R. Li, L. Xiong, D.-Q. Zeng, Z. Fu, B. Leng, and F. Shan, "Real-time local path planning for intelligent vehicle combining tentacle algorithm and B-spline curve," *IFAC-PapersOnLine*, vol. 54, no. 10, pp. 51–58, 2021.
- [35] D. Chiaravalli, F. Califano, L. Biagiotti, D. De Gregorio, and C. Melchiorri, "Physical-consistent behavior embodied in B-spline curves for robot path planning," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 306–311, 2018.
- [36] C. Sun, M. Liu, and S. Ge, "B-spline curve fitting of hungry predation optimization on ship line design," *Applied Sciences*, vol. 12, no. 19, p. 9465, 2022.
- [37] X. Ma, "Path planning of mobile robot based on improved PRM based on cubic spline," *Wireless Communications and Mobile Computing*, 2022.
- [38] J. F. Lian, W. T. Yu, K. Xiao, and W. Liu, "Cubic spline interpolation-based robot path planning using a chaotic adaptive particle swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2020, no. 3, Article ID 1849240, 20 pages, 2020.
- [39] P. Inigo-Blasco, F. Diaz-Del-Rio, and S. V. Diaz, "The shared control dynamic window approach for non-holonomic semi-autonomous robots," *International Symposium on Robotics*, 2014.