Hindawi
Journal of Robotics
Volume 2022, Article ID 7682201, 15 pages
https://doi.org/10.1155/2022/7682201

Research Article

A Smooth Jump Point Search Algorithm for Mobile Robots Path Planning Based on a Two-Dimensional Grid Model

Zhen Yang, Junli Li, Liwei Yang, and Hejiang Chen

School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650093, China

Correspondence should be addressed to Junli Li; li_junli@stu.kust.edu.cn

Received 19 May 2022; Revised 30 June 2022; Accepted 16 July 2022; Published 31 August 2022

Academic Editor: Keigo Watanabe

To address the problems of the traditional A* algorithm in solving paths with many expansion nodes, high memory overhead, low operation efficiency, and many path corners, this paper improved the traditional A* algorithm by combining jump point search strategy and adaptive arc optimization strategy. Firstly, to improve the safety of our paths, the risk area of the obstacles was expanded. Then, the A* algorithm was combined with the jump point search strategy to achieve the subnode jump search, reducing the calculation scale and memory overhead, and improving search efficiency. Considering the influence of the density of obstacles on search efficiency, the heuristic function was enhanced according to the special effects of the density of obstacles. Finally, the redundant jump point and adaptive arc optimization strategies were used to shorten the path length further and enhance the initial path's smoothness. Simulation results showed that our algorithm outperforms traditional A ∗ and literature algorithms in path length, security, and smoothness, and then was further validated and applied in large-scale marine environments and realistic settings.

1. Introduction

With the continuous improvement of automation, mobile robots have been widely used in various scenarios, such as warehouse handling robots, service robots, disinfection robots, and uncrewed express delivery vehicles. Path planning is one of the critical technologies in mobile robots, and the purpose is to find a collision-free optimal or suboptimal path for the robot from the starting position to the target position [1].

Many algorithms for path planning have been proposed by scholars, such as the A ∗ algorithm [2], Dijkstra algorithm [3] as the representative of geometric model search algorithm and particle swarm optimization (PSO) algorithm [4], and ant colony (ACO) algorithm [5] as the representative of the intelligent optimization algorithm. The A ∗ algorithm has been widely studied and applied due to its relatively small computational effort, high search efficiency, and relatively optimal planning path compared with other algorithms [6].

In response to the problems of the traditional A ∗ algorithm, such as too many traversing nodes, large memory occupation, and poor path smoothness, some better improvement strategies have been proposed by researchers. Yang et al. [7] extended the search method of the traditional A ∗ algorithm from eight directions in eight neighborhoods to sixteen directions in sixteen neighborhoods at the expense of the search efficiency of the algorithm, which improved the smoothness and optimality of the path. Wang et al. [8] used a two-way search strategy to improve the A ∗ algorithm, and the simultaneous iterative search in both positive and negative directions improved the search efficiency, but it ignored the problem of large memory occupation in large-scale map paths. Zafar et al. [9] used the jump point search method to improve the node expansion method of the A ∗ algorithm, which solved the memory problem of high overhead and low search efficiency in large-scale maps [8], but the smoothness of the paths is not satisfactory yet. Tang et al. [10] used third-degree B-spline to smooth the paths planned by the A ∗ algorithm but ignored the constraint problem of the obstacle region, which is prone to cause the smoothed paths to traverse obstacles in a narrow area. Saeed et al. [11] solved the problem of Tang et al. [10] by inserting

additional path nodes between the original path points until they could satisfy and generate a smoothed path that does not occur to traverse obstacles. Song et al. [12] designed three path smoothers to optimize the paths, but the time complexity of this algorithm is vulnerable to the scale of the number of nodes. For the safety of robot movement in realistic environments, Zhang et al. [13] considered the distance of nodes from obstacles in the A $*$ expansion node, using the threat generation value as part of the evaluation of the node cost. That algorithm ensures path safety but does not apply to the large-scale complex map environments in a natural setting.

The above-improved algorithms [7–13] improve the search efficiency, path smoothing, and security of the traditional A $*$ algorithm to some degree. Considering the above issues comprehensively, the main contributions of this paper are as follows: firstly, the obstacles modeled by the grid method are puffed, and the risk areas are defined, and then, our improved A $*$ does not search the risk region during the search path so that we reduce the computation of the algorithm and ensure the safety of the planned path. Then, we integrate the jump point search strategy with the A $*$ algorithm to avoid expending unnecessary nodes in the search process of A $*$, so as to reduce the computational scale and the computer memory usage. In addition, we introduce the distribution density of obstacles to adaptively change the weights of the evaluation function of our enhanced A $*$, which further improves the search efficiency of the algorithm. Finally, we utilize the redundant jump point deletion strategy and the adaptive arc optimization strategy to secondary optimize the planning path to reduce the turning points and circularize the corner path to improve the path quality further.

2. Environmental Modeling

The grid method [14] is widely used to build mobile robots for various working environments as it has a simple structure and is easy to implement. Traditional grid modeling methods ignore the mobile robot's volume constraints and kinematic constraints, leading to some planned paths colliding with the edges of obstacles, which increases the risk of collision in reality [15].

To solve the above problems, we add the identification processing of risk area based on the modeling approach of Tsardoulias et al. [16]. Firstly, the actual presence of irregular obstacles in the environment is considered. In the process of environment modeling with the traditional (0,1) grid method, the obstacle areas that do not occupy the whole-cell grid need to be puffed and filled, and then are considered obstacle areas that are forbidden to pass, after which they are represented by "1" in the matrix map. The area without filling is the free passage area and is characterized by "0" in the matrix Map. Next, the outer grid area of the obstacle grid is determined to be the risk area, and the risk area is represented as "2' in the matrix Map. The steps for identifying risk areas are as follows:

Step 1 Iterate the coordinate positions of all obstacles (x_i, y_i), $i = 1, 2, \ldots, n$ in the grid matrix Map.

Step 2 Judge whether the obstacle grid is located in the nonboundary range according to equation (1). If it is satisfied, the free grid near it will be judged as the risk grid according to equation (2), and the corresponding matrix value will be changed from "1" to "2."

$$\begin{cases} 0, & < x_i, \quad < x_M, \\ 0, & < y_i, \quad < y_N, \end{cases} \tag{1}$$

$$\begin{cases} \widetilde{\text{Map}(Dir_i)} = 2, & \text{if Map}(Dir_i) = 0, \\ \text{Map}(Dir_i) = 1, & \text{others}, \end{cases} \tag{2}$$

where n represents the number of obstacles; x_M represents the maximum horizontal coordinate of the Map; y_N represents the maximum vertical coordinate of the Map; $\text{Map}(Dir_i)$ represents the original grid of obstacle i in the eight fields in the grid matrix Map; and $\text{Map}(Dir_i)$ is the grid matrix after satisfying the judgment condition, where $Dir_i = [1\,2\,3\,4\,5\,6\,7\,8]$.

Figure 1 is a schematic diagram of our grid method, and the orange triangular area in Figure 1(a) is an obstacle in the actual environment; the black area represents the expanded grid area that the robot cannot pass and corresponds to "1" in Figure 1(b); the gray area is the expanded risk area and corresponds to "2" in Figure 1(b); the white area is the passable area and corresponds to "0" in Figure 1(b).

While developing the improved A $*$ algorithm to generate path, we avoid extending the search nodes to the risk region, enhancing algorithm search efficiency and making the initially planned paths maintain a certain safety distance from the obstacles. We only consider the distance between the path and the original obstacle in the path optimization stage. Our method guarantees the optimality and safety of the path to a great extent compared with some methods in the literature and reduces the performance requirements of the mobile robot for dynamic obstacle avoidance.

3. Improved A $*$ Algorithm Based on Jump Point Search (JPS-A $*$)

3.1. Standard A $*$ Algorithm. The A $*$ algorithm is a classical heuristic search algorithm with a wide range of applications in solving optimal paths in two-dimensional static environments [17]. The algorithm combines the heuristic idea of the Breadth-First Search algorithm and the shortest path search method of Dijkstra to select the optimal node sequentially by evaluating the cost of each extended node, and the evaluation function of the current node is as follows:

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

where $f(n)$ denotes the total cost of the current node, $g(n)$ denotes the actual cost from the starting node to the current node, $h(n)$ denotes the heuristic distance of the current node from the target node.the The heuristic function of the standard A$*$ algorithm uses the Manhattan distance $d_{M(n)}$ (4) and the Euclidean distance $d_{E(n)}$ of equation (5), respectively, which can be expressed as follows:

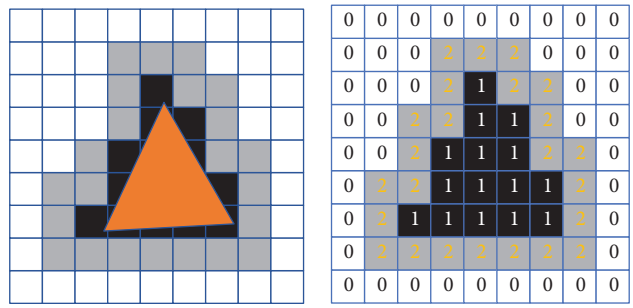


Figure 1: Improved grid method modeling. (a) Irregular obstacles. (b) Matrix map.

$$d_{M(n)} = \left|x_n - x_g\right| + \left|y_n - y_g\right|, \tag{4}$$

$$d_{E(n)} = \sqrt{\left(x_n - x_g\right)^2 + \left(y_n - y_g\right)^2}, \tag{5}$$

where (x_n, y_n) is the coordinates of the current node, and (x_g, y_g) is the coordinates of the target node. The A $*$ algorithm based on Manhattan distance can only achieve the search of four neighborhoods and four directions. The A $*$ algorithm based on Euclidean distance can investigate eight neighborhoods and eight directions, so in this paper, we use the Euclidean distance to calculate $h(n)$ to ensure the optimal path. As shown in Figure 2, both methods will inevitably search for multiple symmetric paths with the same start point, endpoint, and path length, which differ only in the order of the nodes, thus theoretically causing redundant computations in the algorithm search process [18].

The A $*$ algorithm creates two lists when searching the path: Open List for the current node's neighboring nodes and Close List for the visited nodes. Then, the algorithm starts from the starting point and expands to the neighboring nodes, adds the nodes that A $*$ considers for expansion to the Open List, calculates their generation values, sorts them according to their generation values, takes out the node with the smallest generation value as the next parent node, and adds the previous parent node to the Close List, then repeats the above search process until it reaches the target node. The search strategy of the traditional A $*$ algorithm makes a large number of nodes constantly maintained and accessed, which leads to the defects of high computation, serious memory consumption, and low efficiency when performing path planning in a large-scale map environment.

3.2. Improving A $*$ Algorithm Based on Jump Point Search Strategy. To solve the problems mentioned earlier, such as many extended nodes, high memory consumption, and low search efficiency in a large-scale map environment, we use the jump point search strategy [19] to improve the conventional A $*$ algorithm.

The jump point search strategy breaks the symmetry of paths and reduces the computational scale by giving up evaluating a large number of nodes during path search and only expanding the nodes that conform to the jump point rules. The jump point search strategy will prefilter out the non-natural neighbor nodes in the map and then search the suitable nodes from the remaining nodes as jump points. The following are the rules for defining non-natural neighbor nodes, forced neighbor nodes, and jump points in this paper.

3.2.1. Non-Natural Neighbor Nodes. Figure 3 shows the filtering rules for non-natural neighbor nodes, where $p(x)$ is the parent node of node x; the direction of the parent node $p(x)$ pointing to x is the direction of the algorithm extension node; and the yellow grid is the non-natural neighbor nodes that do not need to be considered in the search path process.

As shown in Figure 3(a), starting from the parent node $p(x)$, there are shortest or equivalent paths to all yellow nodes without going through node x. Therefore, all yellow nodes in the straight-line direction are nodes that do not need to be computed, and for such nodes in the straight-line path that satisfy equation (6), we define them as non-natural neighbor nodes. All the yellow nodes in Figure 3(b) are directly reachable by $p(x)$ with a minimum path score value, and the filtering rule for non-natural neighbor nodes in the diagonal direction is in equation (7)as follows:

$$\operatorname{len}\left(\frac{\langle p(x), \ldots, n\rangle}{x}\right) \leq \operatorname{len}\left(\langle p(x), x, n\rangle\right), \tag{6}$$

$$\operatorname{len}\left(\frac{\langle p(x), \ldots, n\rangle}{x}\right) < \operatorname{len}\left(\langle p(x), x, n\rangle\right), \tag{7}$$

where len represents the generation value of the path; $p(x), x, n$ represents the path from parent $p(x)$ to node n via node x; and $\langle p(x), \ldots, n/x\rangle$ denotes the path from parent $p(x)$ to node n directly without going through node x.

3.2.2. Forced Neighbor Nodes. For pathfinding with obstacle disturbances like Figure 4, there does not exist an optimal path from $p(x)$ without going through node x to node n directly in the both linear and diagonal directions. Therefore, node n is defined as a forced neighbor node of node x with the following filtering rule.

(1) Node n satisfies the definition of non-natural neighbor node.

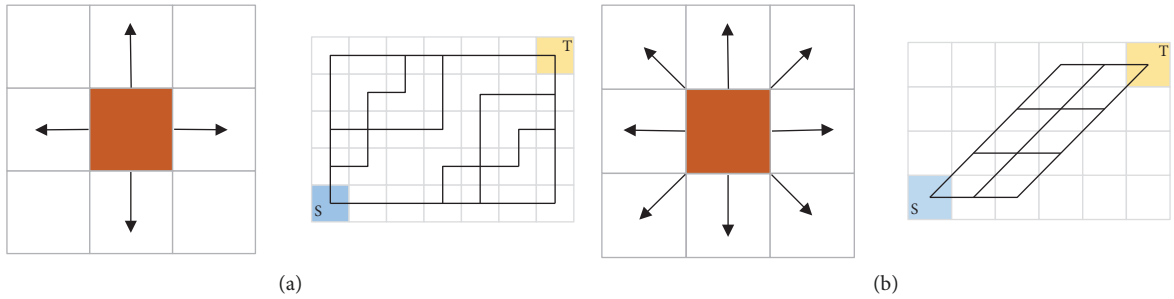


Figure 2: Symmetric path search. (a) Four-neighborhood search and symmetric path. (b) Eight-neighborhood search and symmetric path.

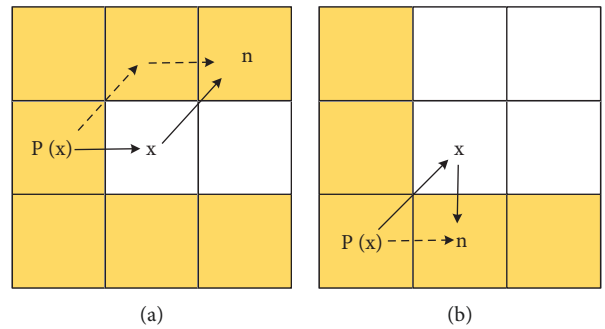


Figure 3: Non-natural neighbor nodes. (a) Linear direction. (b) Diagonal direction.

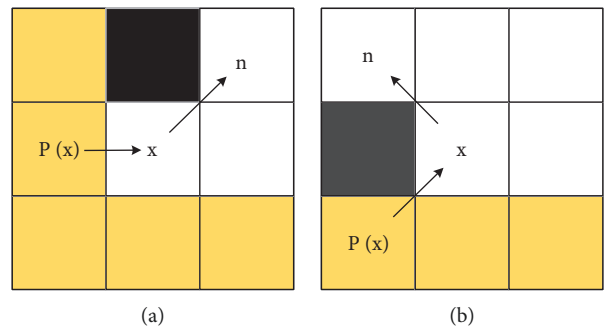


Figure 4: Forced neighbor node. (a) Straight line direction. (b) Diagonal direction.

(2) Node n satisfies the rule of equation (8).

$$\text{len}(\langle p(x), x, n\rangle) < \text{len}\left(\frac{\langle p(x), \ldots, n\rangle}{x}\right). \tag{8}$$

3.2.3. Jump Point. A node y is defined as a jump point of node x if there exists a minimum value k such that the formula $y = x + k\vec{d}$ holds and one of the following three conditions is satisfied [20].

(1) Node y is the target node.

(2) At least one forced neighbor node of node y.

(3) If $\vec{d}$ is the diagonal direction and there exists a node z at the same time ($\vec{d}$ is the diagonal horizontal and vertical decomposition vector direction), node y can move k steps in that direction to reach node z, while node z is the defined jump point by condition (1) or condition (2).

The jump point search starts by finding jump points along the straight line and diagonal directions, adding the nodes that meet the definition of jump points to the OpenList, and then selecting the node with the lowest generated value as the new extended node for calculation up to the target point. As shown in Figure 5, it is the starting point, T is the target point, the green grids represent the nodes extended to in the path search, the red grids represent the forced neighbor nodes of the jump point search, and the solid black line is the final path. In the same map environment, both algorithms can plan a safe path.

However, jump point search prunes a large number of non-natural neighbor nodes in the process of expanding

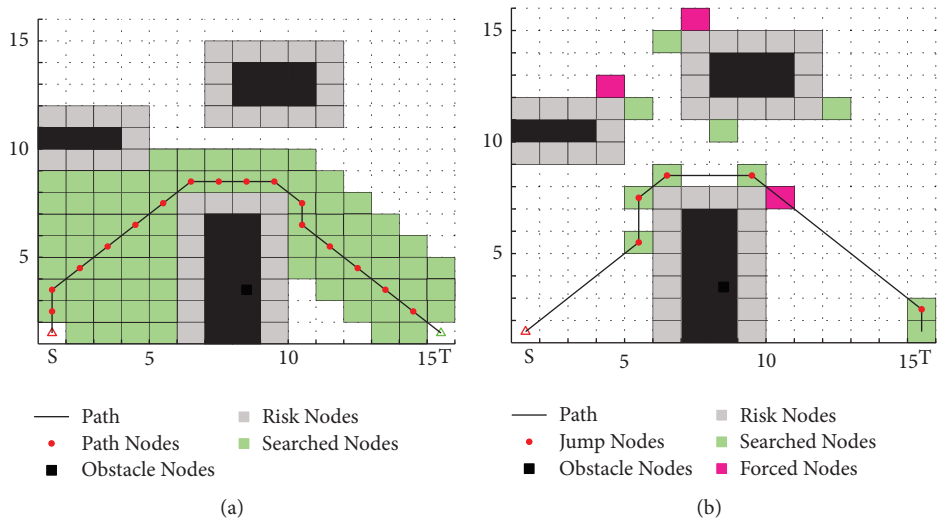


Figure 5: Comparison of path planning. (a) Traditional A* algorithm. (b) Improved A* algorithm based on jump point search

nodes and expands only natural neighbor nodes as well as forced neighbor nodes. Compared with the A ∗ algorithm, it dramatically reduces the computation of nodes and speeds up the search time. It is evident from Figure 5 that the number of green search grid points of our algorithm is small, and the jump point search can achieve a long-distance jump between two nonadjacent jump points according to the definition of the jump point, which means the computational scale is small. In summary, the improved algorithm can significantly reduce the computational complexity and thus the time scale. Also, the number of path nodes is further reduced, which is a more pronounced improvement in large-scale maps, as demonstrated in subsequent sections 5.1 and 5.2.

Our jump point search strategy is suitable for 2D static planar path planning. It considers only distance cost and greatly outperforms the traditional A ∗ algorithm since it removes many neighboring nodes with suboptimal spaces. Moreover, it does not apply to 2D environments considering nonflat terrain [13] because it focuses on the distance constraint of the node rather than other factors, such as the terrain height difference between that node and the surrounding nodes.

3.3. Improving the Heuristic Function of JPS-A ∗ Algorithm. The heuristic function of the traditional A ∗ algorithm is shown in equation (3), which does not consider the influence of obstacle density on search efficiency. Since there are usually many obstacles between the starting point and the target point, the algorithm will occupy ample memory space and reduce the efficiency during the search process, and generate more redundant nodes [21]. When the number of obstacles in the environment space is small, the value of the heuristic function of A ∗ is closer to the actual distance value, so the weight of the heuristic function can be increased appropriately to reduce the algorithm's search space and improve the search efficiency. When the number of obstacles in the region is more complex, the heuristic function value of the algorithm will be smaller than the actual distance value, so the weight of the heuristic function can be reduced appropriately to increase the number of expansion nodes of the algorithm, which will improve the search capability of the optimal path by expanding the search space.

To this effect, equations (9) and (10) are used to calculate the obstacle density P within the rectangular region composed of the current node and the target node, and the obstacle density P is introduced into the heuristic function for improving A ∗ .

$$P = \frac{N}{\left(\left|x_t - x_n\right| + 1\right) \times \left(\left|y_t - y_n\right| + 1\right)}, \tag{9}$$

$$f(n) = g(n) + (\ln(e + P)) \cdot h(n), \tag{10}$$

where (x_t, y_t) is the coordinates of the current node; N is the number of obstacles in the rectangular region composed of the current node and the target node; and x_n and y_n are the lengths of the boundaries in the horizontal and vertical axis directions of the map, respectively. The improved heuristic function enables the algorithm to adaptively adjust the weights of the heuristic function according to the obstacle rates in different regions.

To verify the effectiveness of the improved heuristic function, we use the heuristic function as the only variable. It is evident from Figure 6 that the number of extended nodes of the improved heuristic function is reduced to 28 compared with 41 under the traditional method for both planning path lengths of 20.9706 m. The 31.7% reduction in the expansion nodes improves the search speed of the algorithm by 32.8%. The main reason is that the algorithm searches in the region with a low obstacle rate by reducing the search space and thus improving the search efficiency.

4. Path Optimization Strategy

The improved A ∗ algorithm based on jump point search can significantly improve the search efficiency, but the planned path still has more turning points and the robot

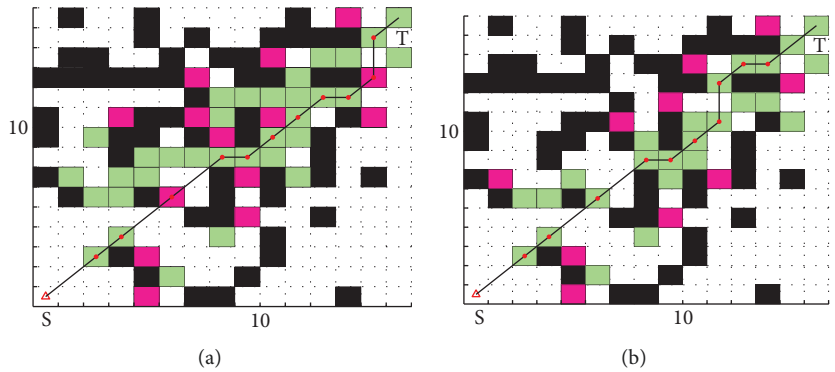


Figure 6: Heuristic function ablation experiment based on JPS-A*. (a) Traditional heuristic function with 4.994 ms. (b) Improved heuristic function with 3.355 ms.

needs to make several unnecessary attitude adjustments to track path [22]. In order to satisfy the nonholonomic constraints of mobile robots, we propose a redundant jump point deletion strategy and an adaptive arc optimization strategy to optimize the path further.

4.1. Redundant Jump Point Removal Strategy. The path generated by the traditional A ∗ algorithm is composed of a continuous grid containing more turning points. Jump point search can achieve long-distance jumps between nodes, but the smoothness of the path is not improved more significantly, so we propose a redundant jump point removal strategy to eliminate the redundant turning points.

4.1.1. Step 1 Extract key jump points. Extract three consecutive jump points Node_{i+1}, Node_i, and Node_{i-1} from the initial path compare the slope of the line formed by Node_{i+1} and Node_i with the slope of the line formed by Node_i and Node_{i-1}. If the slopes are equal, we delete jump point Node_i. We iterate through all jump points, delete all common jump points, and generate a path sequence containing only the start point, key jump points, and endpoint.

4.1.2. Step 2. Remove redundant jump points. The path sequence generated after Step 1 is $\{P_i, 1 \le i \le n\}$ and connects the point P_1 with the point P_3, and if the line segment P_1P_3 does not cross the obstacle area, then we continue to connect P_1P_4 until the line segment $\overline{P_1P_i}$ crosses the obstacle area, and then, we determine $P_2, P_3 \ldots P_{i-2}$, which are the redundant jump points, and the corresponding path sequence is deleted. We repeat the above operation from point P_2 again until all the jump points in the path have been detected.

As shown in Figure 7, the path optimized by the redundant jump point removal strategy breaks through the limitation of eight neighborhood search directions and can realize the connection between nodes in noneight directions. The length of the path is further shortened, and the number of turning points is also reduced. But the same problem of an unsmooth path at the turning point still exists [23].

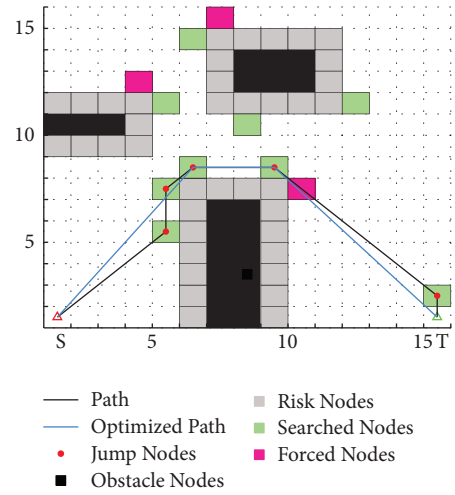


Figure 7: Redundant hop removal policy.

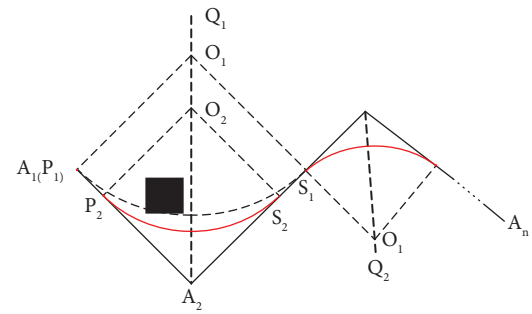


Figure 8: Adaptive arc optimization algorithm.

4.2. Adaptive Arc Optimization Strategy. As shown in Figure 8, the starting position of the mobile robot is $A_1(x_1, y_1)$, and the end position is $A_1(x_n, y_n)$. The algorithm starts from the starting point and circularizes the corner areas in the path until the endpoint [24], and the specific steps are as follows:

Step 1. Compare the lengths of the line segments $\overline{A_{i-1}A_i}$ and $\overline{A_iA_{i+1}}$, choose the short side of them as the initial tangent side, whose endpoint $P(x_p, y_p)$ is the initial tangent point,

and make a vertical line through the initial tangent point P to intersect with the angle bisector A_iQ_{i-1} of $\angle A_{i-1}A_iA_{i+1}$ at the point $O_{i-1}(x_0, y_0)$; then, the center of the circle of the tangent circle is as follows:

$$\begin{cases} x_0 = \dfrac{\left(x_p + k_{01}y_p + k_{01}\left(k_0x_2 - y_2\right)\right)}{1 + k_0k_{01}} \\ y_0 = k_0\left(x_0 - x_2\right) + y_2 \end{cases} \tag{11}$$

The radius of the tangent circle is as follows:

$$R = \sqrt{x_0^2 - 2x_0x_p + y_0^2 - 2y_0y_p + x_p^2 + y_p^2}. \tag{12}$$

The equation of the tangent circle is as follows:

$$\left(x - x_0\right)^2 + \left(y - y_0\right)^2 = R^2, \tag{13}$$

where k_{01} is the slope of the short side, and k_0 is the slope of the angle bisector.

Step 2. Judge whether the tangent circle intersects the long side at a point S. If yes, we execute Step (3); otherwise, we go to Step (4).

Step 3. Determine whether the arc $\widehat{PS}$ crosses the obstacle area. If yes, then execute Step (4); otherwise, we replace the fold line $\overline{A_{i-1}A_iA_{i+1}}$ with the arc $\widehat{PS}$ and go to Step (5).

Step 4. The tangent point $P(x_P, y_P)$ moves to the point $P_2(x_{p2}, y_{p2})$ along the line segment, and x_{p2} can be expressed as follows:

$$x_{p2} = x_p + \lambda\left|x_2 - x_p\right|, \quad \lambda \in (0, 1), \tag{14}$$

where λ is set according to the actual situation, while P_2 is set as the initial tangent point, and return to Step (1).

Step 5. Determine whether all the jump points in the path are traversed. If yes, the algorithm is finished; otherwise, we return to Step (1). The result of adaptive arc optimization is shown in Figure 9.

4.3. Algorithm Flow. The flow chart of the pathfinding optimization based on our improved A∗ algorithm is shown in Figure 10.

5. Simulation Experiment

5.1. Small- and Medium-Scale Map Simulation. To verify the effectiveness of our improved algorithm, the following experiments were conducted, and the 30×30, 50×50, and 100×100 grid environments of Li et al. [25] were used as test maps for simulation and compared with the traditional ACO, improved ACO [26], traditional A∗ algorithm, secure A∗ algorithm, bidirectional A∗ smoothing algorithm [25], and PRM, and experimental environment such as Windows 10 (64 bit), AMD Ryzen5-4600H, CPU 3Ghz, memory 16 GB, and Matlab 2019a. Simulation experimental results are shown in Table 1 and Figures 11–16.

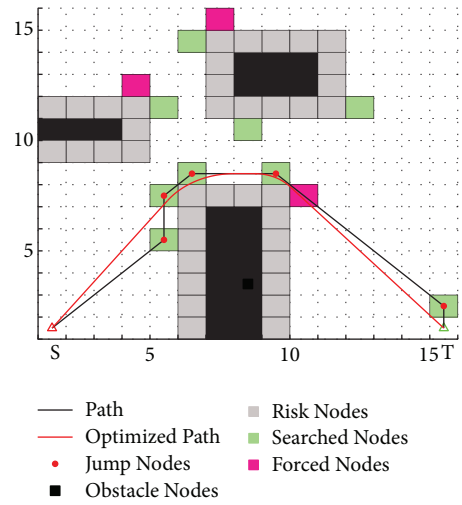


Figure 9: Adaptive arc optimization.

As shown in Figures 11 and 12, the paths planned by traditional ACO and traditional A∗ both collide with obstacles and do not meet the safety requirements for robot motion. Based on the environment modeling approach we proposed, the safe A∗ algorithm, the improved A∗ [25], PRM, and our algorithm can all plan safe paths as shown in Figures 13–16, and the paths obtained by our algorithm outperform the safe A∗ algorithm, the improved A∗ [25], and PRM in terms of length. As shown in Figures 14–16, in terms of smoothness, we combine adaptive circular arc optimization and achieve similar smoothness as Li et al. [25], who use Bézier curves for optimization, but our optimization in terms of distance length is significantly better. Since the paths planned by PRM have some turning points, the smoothness of its planned paths is worse. Besides, the time spent by our algorithm is significantly better than PRM and slightly better than the improved A∗ [25].

From the experimental results in Figure 11, the IACO of Yang et al. [26] also guarantees the path security, but the length of the planned paths, the smoothness, and the running time of that algorithm is not as good as ours in the 30×30, 50×50, and 100×100 environments. In addition, the number of nodes extended in this paper is significantly better than the traditional A∗ algorithm and slightly better than the improvedA∗ [25], so it is also the best among the compared algorithms in terms of computational efficiency and the algorithm search time.

Based on the statistical tests, we draw a conclusion that the comprehensive performance of our algorithm is better than that of the other algorithms in the experiments.

5.2. Large-Scale Marine Environment Simulation. To verify the performance of the improved algorithm in the superscale map scenario, Figure 17(a) of the area around Zhubu Island (7 km*7 km) in Qingdao, Shandong Province, China, and Figure 18(a) of the area around Changhai County (64 km*48 km) in Liaoning Province, China, were selected as the sea maps for testing [27]. The satellite map of the area was first acquired, binarized, and then modeled the ultra-

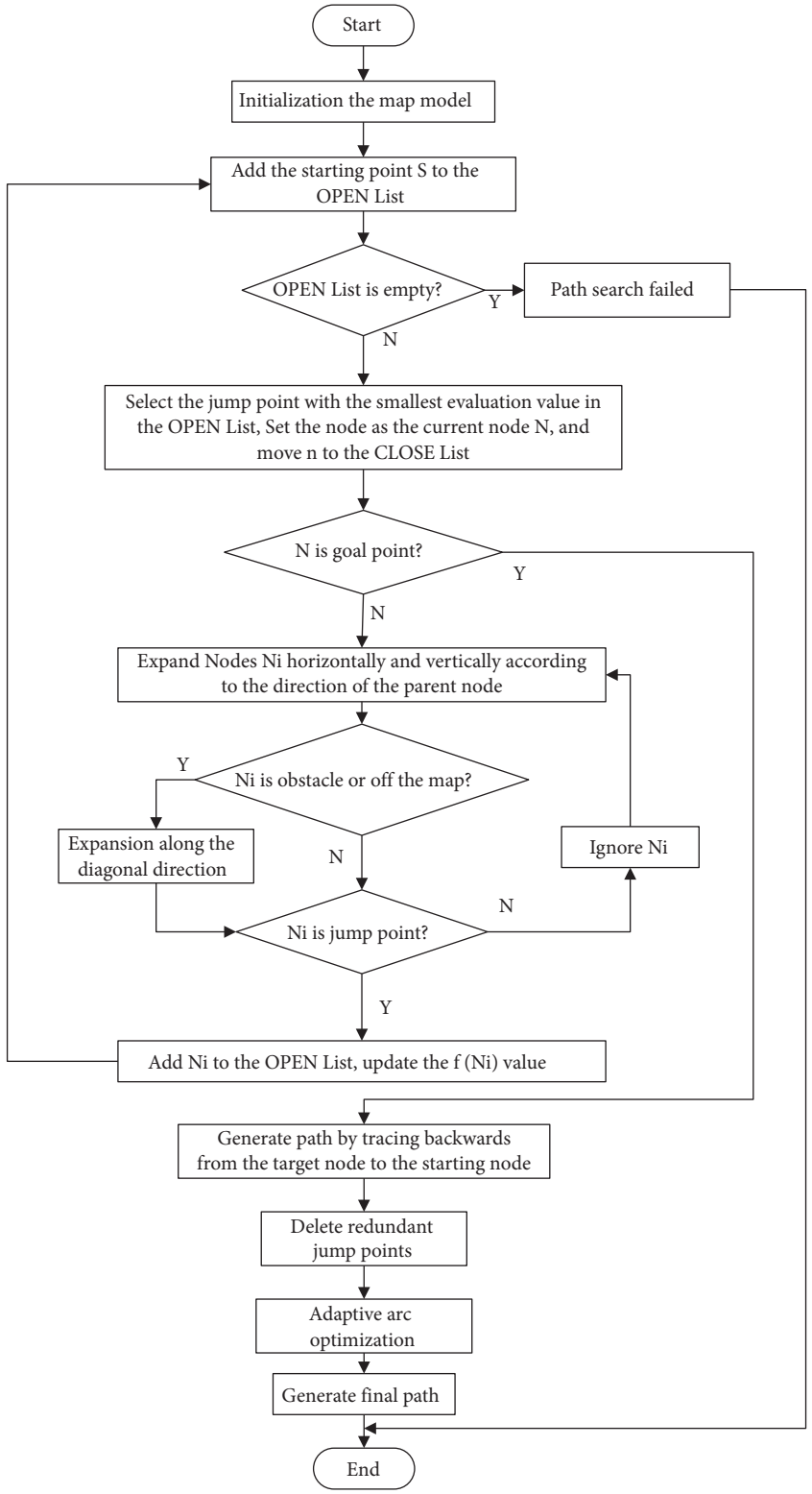


Figure 10: Flow chart of the improvement algorithm.

large-scale grid map with our method, where the accuracy is 10 m. The two areas' binarization maps are shown in Figure 17(b) and 18(b). Due to the limited performance of our computer, this experiment again divided three groups of maps: Map_1 with a simple obstacle environment but a more extensive map size of 7 km*7 km, and Map_2 and Map_3 with a slightly smaller scale of 5 km*5 km. Three environments are in increasing order of complexity, and we compared our algorithm with the A∗. The experimental results and data are shown in Figures 19 and 20 and Tables 2

Table 1: Three sets of static map simulation data.

Map	Algorithm	Path length/m	Path length optimization ratio	Number of turning points	Turning point optimization ratio	Number of extended nodes/ number	Path safety	Algorithm search time/ ms	Path convergence or not
1	Traditional ACO	43.9411	—	5	—	—	*N*	107.87	*Y*
	Improved ACO [26]	47.4558	—	2	—	—	*Y*	98.71	*Y*
	Traditional A ∗	43.9411	—	5	—	346	*N*	9.47	—
	Safety A ∗	45.6985	—	11	—	370	*Y*	9.18	—
	Improved A ∗ [25]	43.10	4.9%	0	100%	40	*Y*	6.82	—
	PRM	43.3658	—	4	—	—	*Y*	15.44	—
	This paper	42.4743	7.0%	0	100%	52	*Y*	6.20	—
2	Traditional ACO	77.6396	—	18	—	—	*N*	266.32	*N*
	Improved ACO [26]	76.4392	—	6	—	—	*Y*	180.54	*Y*
	Traditional A ∗	72.2254	—	7	—	504	*N*	13.48	—
	Safety A ∗	76.9117	—	13	—	772	*Y*	19.96	—
	Improved A ∗ [25]	71.69	5.7%	0	100%	64	*Y*	8.700	—
	PRM	72.1034	—	4	—	—	*Y*	25.67	—
	This paper	71.3966	7.1%	0	100%	47	*Y*	8.37	—
3	Traditional ACO	352.374	—	>30	—	—	*N*	458.62	*N*
	Improved ACO [26]	173.9828	—	5	—	—	*Y*	82.6	*Y*
	Traditional A ∗	148.2048	—	9	—	2527	*N*	85.82	—
	Safety A ∗	151.7229	—	9	—	2750	*Y*	85.70	—
	Improved A ∗ [25]	143.35	5.5%	0	100%	120	*Y*	15.50	—
	PRM	148.4120	—	3	—	—	*Y*	35.27	—
	This paper	143.0791	6.0%	0	100%	22	*Y*	14.29	—

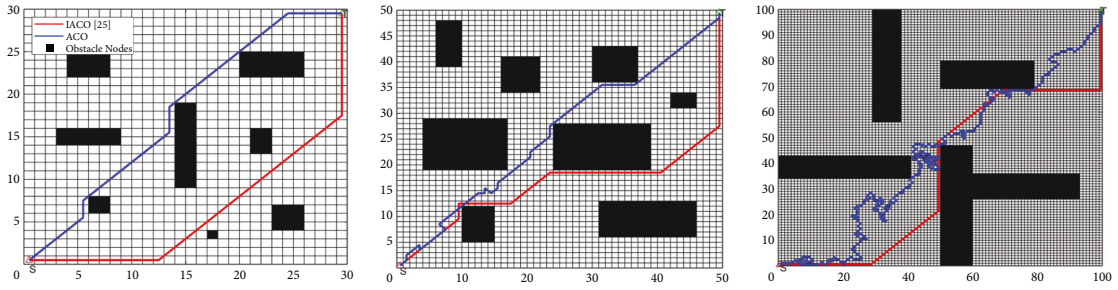
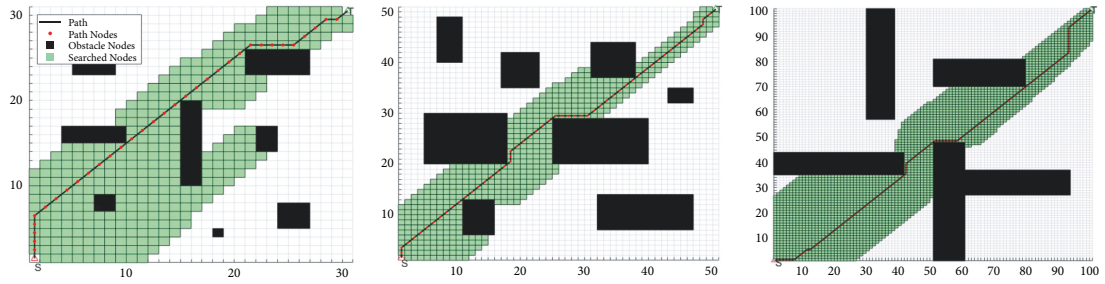


Figure 11: Traditional grid method environment modeling + traditional ACO and improved ACO [26]. (a) 30*30. (b) 50*50. (c) 100*100.

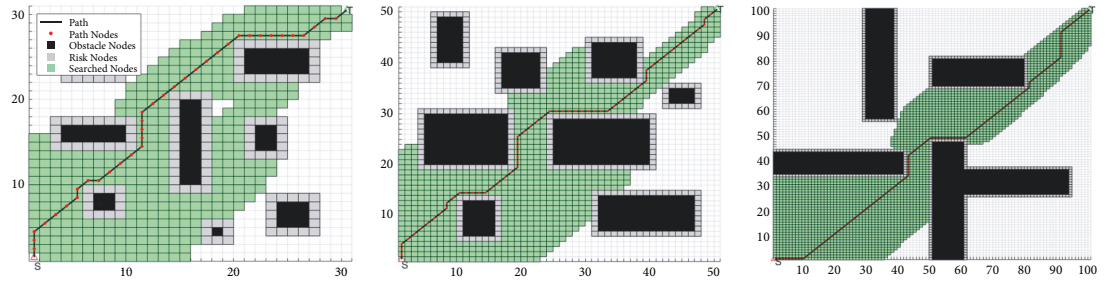
and 3, where the yellow grid area in Figure 19 is the expanded node area of A ∗ .

In Map_1, both algorithms can successfully plan paths, and our improved algorithm achieves significant results in this type of large-scale map. The algorithm in this paper is 771.752 m shorter and 1268 times less than the A ∗ algorithm in terms of path length and the number of extended nodes, respectively, and the number of nodes visiting the obstacle is also significantly better than A ∗ . In Map_2 and Map_3, which have higher complexity, both our algorithm and A ∗ perform three sets of path planning. From Tables 2 and 3, we can see that our algorithm still has



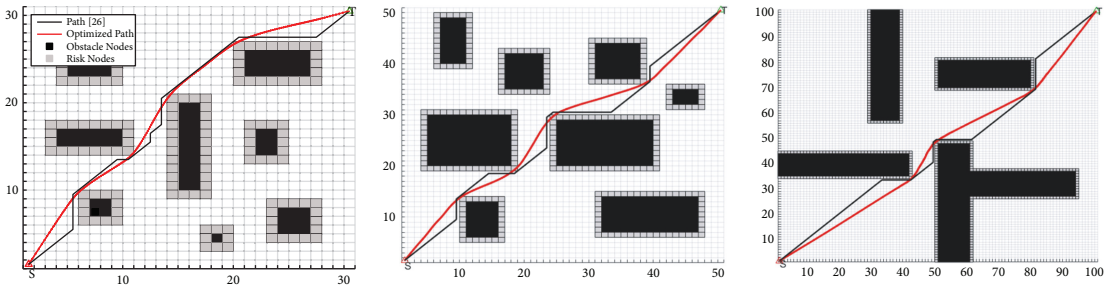
(a) (b) (c)

Figure 12: Traditional grid method environment modeling + traditional A ∗ . (a) 30*30. (b) 50*50. (c) 100*100.



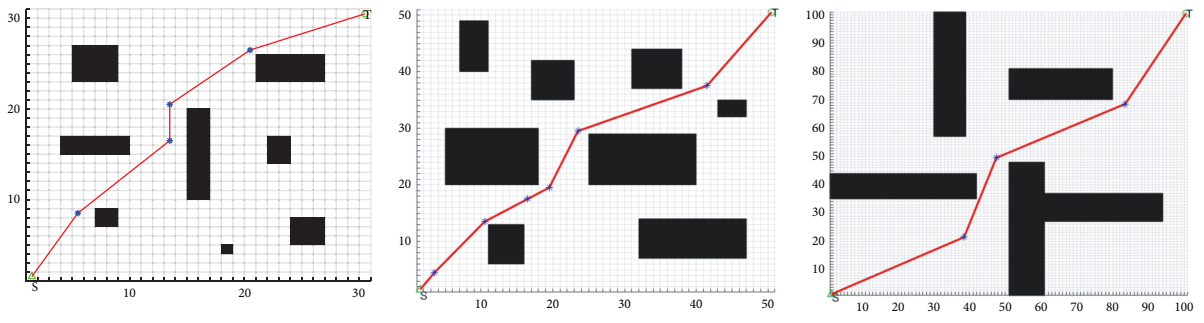
(a) (b) (c)

Figure 13: This paper grid method environment modeling + traditional A ∗ (security A ∗). (a) 30*30. (b) 50*50. (c) 100*100.



(a) (b) (c)

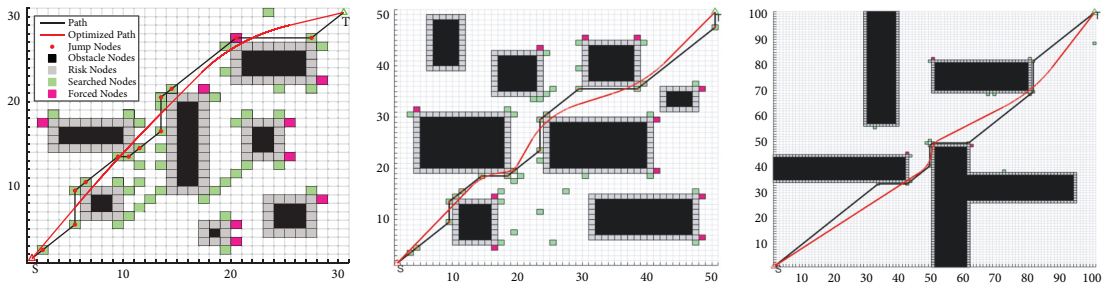
Figure 14: Improved A ∗ algorithm [25]. (a) 30*30. (b) 50*50. (c) 100*100.



(a) (b) (c)

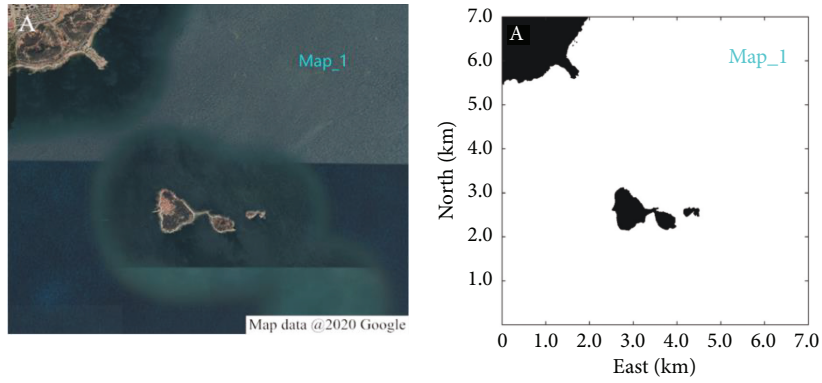
Figure 15: PRM algorithm. (a) 30*30. (b) 50*50. (c) 100*100.

apparent advantages in path length, the number of extended nodes, and the number of nodes visiting the obstacle. The paths of our algorithm in path_1 of Map_2 are smoother when traversing narrow areas, and the comparative indexes are more prominent. We do not consider the secondary path optimization in the experiments in Figure 20 due to the



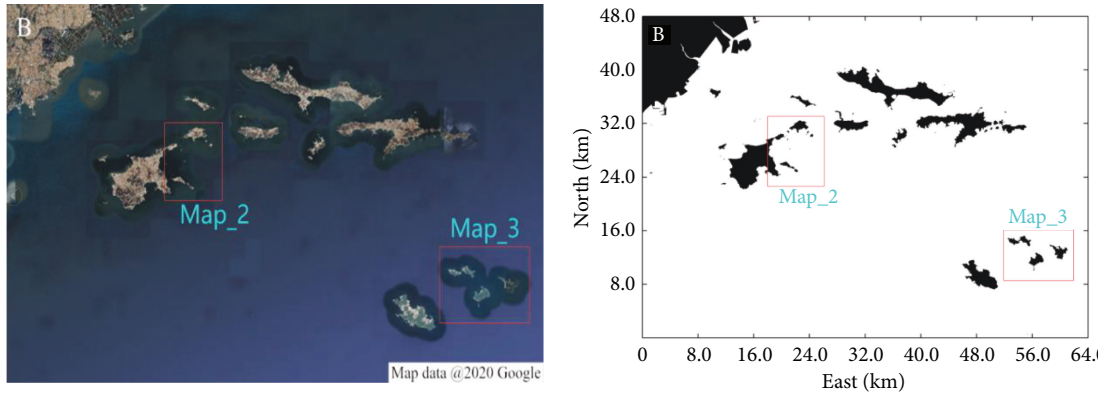
(a) (b) (c)

Figure 16: The improved A ∗ algorithm in this paper. (a) 30*30. (b) 50*50. (c) 100*100.



(a) (b)

Figure 17: 700*700 (7 km*7 km) scale sea area map. (a) Original map. (b) Binarized map.



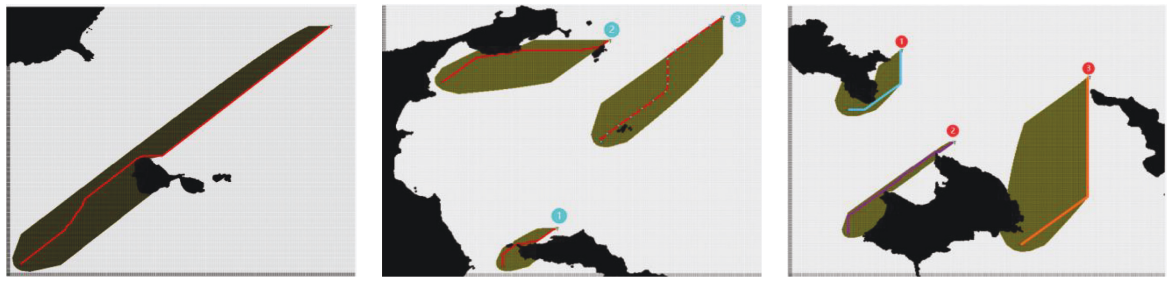
(a) (b)

Figure 18: 6400*4800(64 km*48 km) scale sea area map. (a) Original map. (b) Binarized map.

broader sea area, and the number of path turns is less, but it is still better than A ∗ from the results. To reflect the effect of adaptive arc optimization, we optimize path_3 with the path length of 3472.792 m in Map_3 as the target and get a smooth path with the length of 3406.679 m, as shown in Figure 21.

5.3. ROS Cart Experiment. To verify the applicability of the improved algorithm in a natural environment, a two-wheel differential speed mobile robot was composed of a McNamee wheel chassis, an STM32 underlying control board, a Lidar, and equipped with an Ubuntu 16.04 system including ROS-kinetic. The ROS cart experiment was conducted in an environment 1.5 m long and 0.5 m wide, and the experimental conditions are shown in Figure 22.

Figure 23 shows our observation of the ROS vehicle in motion, in which the red color is the outline of the obstacle scanned by lidar; the black area is the risk area extension; the white box is the lidar detection range; and the solid green



(a) (b) (c)

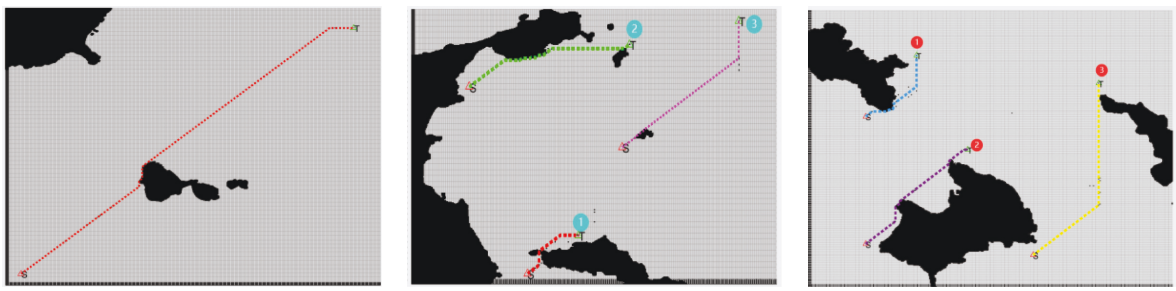
Figure 19: A ∗ algorithm path planning result. (a) Map_1. (b) Map_2. (c) Map_3.

Table 2: A ∗ algorithm for large-scale map path planning.

Map number	Route	Start point coordinates	End point coordinates	Path length/m	Number of extension nodes	Number of nodes visiting the obstacle
Map_1 (700*700)	1	(160.5 20.5)	(230.5 90.5)	9020.012	60955	88431
Map_2 (500*500)	1	(160.5 20.5)	(230.5 90.5)	1124.680	2608	41794
	2	(80.5 360.5)	(300.5 437.5)	2518.944	13699	52596
	3	(290.5 250.5)	(450.5 480.5)	2962.742	16072	54706
Map_3 (500*500)	1	(80.5 310.5)	(150.5 420.5)	1518.823	4308	39775
	2	(80.5 80.5)	(220.5 250.5)	2309.188	4809	40116
	3	(310.5 60.5)	(400.5 370.5)	3472.792	24460	59488

Table 3: Our algorithm for large-scale map path planning.

Map number	Route	Start point coordinates	End point coordinates	Path length/m	Number of extension nodes	Number of nodes visiting the obstacle
Map_1 (700*700)	1	(160.5 20.5)	(230.5 90.5)	8791.764	48	31378
Map_2 (500*500)	1	(160.5 20.5)	(230.5 90.5)	1099.080	42	42016
	2	(80.5 360.5)	(300.5 437.5)	2415.358	29	39487
	3	(290.5 250.5)	(450.5 480.5)	2807.652	37	39494
Map_3 (500*500)	1	(80.5 310.5)	(150.5 420.5)	1478.851	38	38302
	2	(80.5 80.5)	(220.5 250.5)	2220.133	26	35850
	3	(310.5 60.5)	(400.5 370.5)	3472.792	40	35865



(a) (b) (c)

Figure 20: The path planning result of the algorithm in this paper. (a) Map_1. (b) Map_2. (c) Map_3.

line is the global route planned by our A ∗ algorithm. From Figure 23, it can be seen that a collision-free global path is preplanned for the ROS cart by our algorithm, and then, the cart tracks the global path to move. From the experimental data in Figure 24, it can be seen that due to the expansion of the risk region and the optimization strategy of adaptive arc, the position change and speed change amplitude of the trolley in the driving process are relatively gentle, and the ROS cart finally avoids the obstacles safely and smoothly to reach the target position.

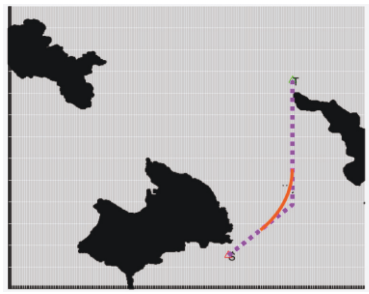
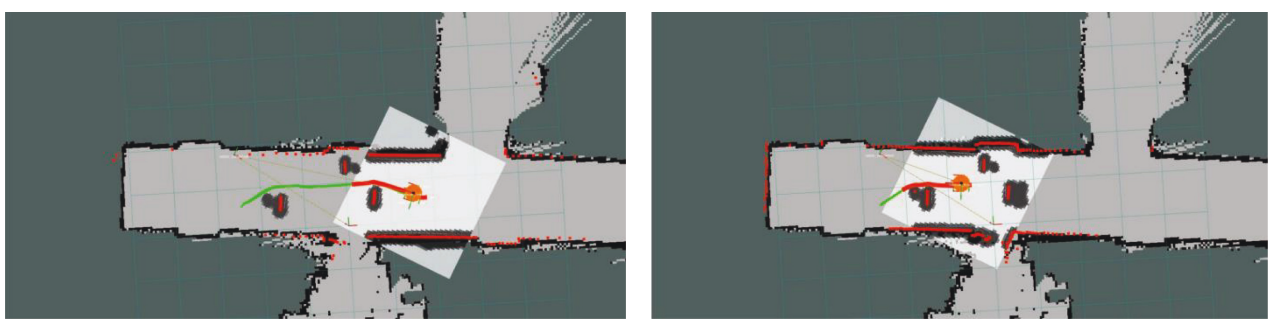


Figure 21: Smoothing optimization results of the algorithm in this paper.

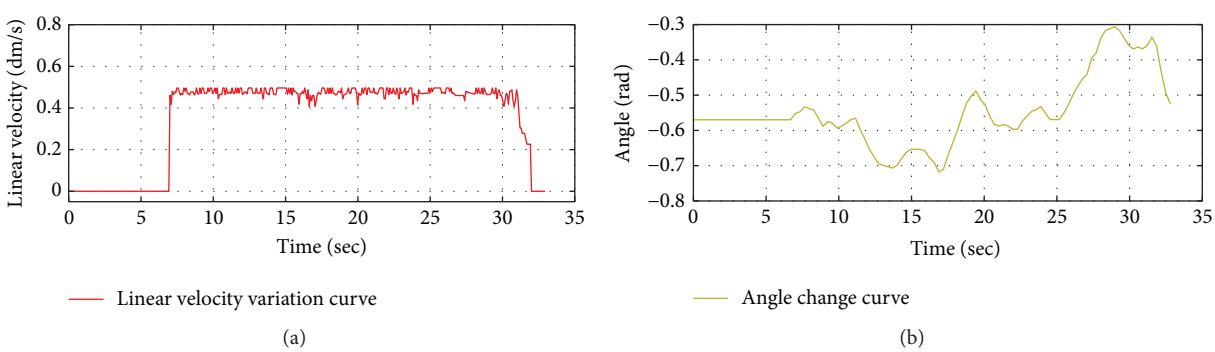
(a) (b)

Figure 22: Experimental scenario. (a) ROS trolley. (b) Dormitory hallway environment.



(a) (b)

Figure 23: ROS simulation results. (a) Path planning. (b) Obstacle avoidance.



(a) (b)

Figure 24: Parameter output results. (a) Linear velocities. (b) Attitude angle.

6. Conclusion

To solve the problems of smoothness, safety, and memory occupation of the path planned by the traditional A ∗ algorithm, we improve it by combining the jump point search strategy and the adaptive arc optimization strategy. To improve the safety of the planned paths, we expand the risk area of the obstacles. Combining the efficient search method of jump points, we improve the node expansion strategy of A ∗ and improve the heuristic function by combining the obstacle density effects of the environment to enhance the execution efficiency further and reduce the computer memory requirement. Considering the actual motion requirements of the mobile robot, the redundant jump point optimization strategy and the adaptive arc optimization strategy are used to shorten the path length and improve the smoothness. Comparative experiments with the traditional A ∗ algorithm and other algorithms [25,26] in small- to medium-scale environments validate the effectiveness of our algorithm in optimizing path length, smoothness, and safety. Simulation tests in mega-island environments demonstrate the advantages of our algorithm in terms of path metrics and low memory usage under large-scale maps. Finally, path planning experiments for mobile robot in a natural environment based on the ROS platform further verify that our improved algorithm can design a safe and smooth route, and meet the actual robot motion requirements.

Data Availability

No data were used in this study.

Conflicts of Interest

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

Authors' Contributions

Z.Y. and L.Y. conceptualized the data; Z.Y. performed methodology; Z.Y. contributed to software; Z.Y. and L.L. validated the document; H.C. performed resources; L.Y. visualized the study; Z.Y. wrote original draft; and L.L. and L.Y. reviewed and edited the manuscript. All authors have read and agree to the published version of this manuscript.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61163051).

Supplementary Materials

The following supporting information can be downloaded. (1) We expose the code for the adaptive arc optimization algorithm: https://github.com/akuyh/Adaptive-arc-optimization, Figure 9. (2) The large-scale sea area map we use is from Wang et al. [27], and we disclose the map data Map_1, Map_2, and Map_3 used in this paper for other scholars to study in their research: https://github.com/akuyh/map, Figures 19(a)–19(c). (*Supplementary Materials*)

References

[1] B. K. Patle, A. Pandey, D. R. Parhi, and A. Jagadees, "A review: on path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[2] O. O. Martins, A. A. Adekunle, O. M. Olaniyan, and B. O. Bolaji, "An Improved multi-objective a-star algorithm for path planning in a large workspace: design, Implementation, and Evaluation," *Scientific African*, vol. 15, Article ID e01068, 2022.

[3] L. S. Liu, J. F. Lin, J. X. Yao et al., "Path planning for smart car based on Dijkstra algorithm and dynamic window approach," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 8881684, 12 pages, 2021.

[4] S. Tian, Y. Li, Y. Kang, and J. Xia, "Multi-robot path planning in wireless sensor networks based on jump mechanism PSO and safety gap obstacle avoidance," *Future Generation Computer Systems*, vol. 118, pp. 37–47, 2021.

[5] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Computers & Industrial Engineering*, vol. 156, Article ID 107230, 2021.

[6] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of A ∗ pathfinding," *Procedia Computer Science*, vol. 179, pp. 507–514, 2021.

[7] J. M. Yang, C. M. Tseng, and P. S. Tseng, "Path planning on satellite images for unmanned surface vehicles," *International Journal of Naval Architecture and Ocean Engineering*, vol. 7, no. 1, pp. 87–99, 2015.

[8] 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, p. 2213, 2021.

[9] M. A. Zafar, Z. Zheng, and Y. Wenkai, "Mobile robots path planning based on A ∗ algorithm improved with jump point search," in *Proceedings of the 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, Islamabad, Pakistan, 2021.

[10] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-star algorithm: an improved A-star algorithm for AGV path planning in a port environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021.

[11] R. A. Saeed, D. R. Recupero, and P. Remagnino, "A boundary node method for path planning of mobile robots," *Robotics and Autonomous Systems*, vol. 123, Article ID 103320, 2020.

[12] R. Song, Y. Liu, and R. Bucknall, "Smoothed A ∗ algorithm for practical unmanned surface vehicle path planning," *Applied Ocean Research*, vol. 83, pp. 9–20, 2019.

[13] 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, p. 44, 2018.

[14] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *Journal of Intelligent and Robotic Systems*, vol. 84, no. 1-4, pp. 829–858, 2016.

[15] L. Xie, S. Xue, J. Zhang, M. Zhang, W. Tian, and S. Haugen, "A path planning approach based on multi-direction A algorithm for ships navigating within wind farm waters," *Ocean Engineering*, vol. 184, pp. 311–322, 2019.

[16] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A dynamic fusion pathfinding algorithm using delaunay triangulation and

improved a-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.
[17] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.
[18] G. Chen, S. Chen, X. Li, P. Zhou, and Z. Zhou, "Optimal seamline detection for orthoimage mosaicking based on DSM and improved JPS algorithm," *Remote Sensing*, vol. 10, no. 6, p. 821, 2018.
[19] B. Zhang and D. Zhu, "A new method on motion planning for mobile robots using jump point search and Bezier curves," *International Journal of Advanced Robotic Systems*, vol. 18, no. 4, 2021.
[20] K. Zhou, L. Yu, Z. Long, and S. Mo, "Local path planning of driverless car navigation based on jump point search method under urban environment," *Future Internet*, vol. 9, no. 3, p. 51, 2017.
[21] X. Bai, H. Jiang, J. Cui, K. Lu, P. Chen, and M. Zhang, "UAV Path Planning Based on Improved A and DWA Algorithms," *International Journal of Aerospace Engineering*, vol. 2021, Article ID 4511252, 12 pages, 2021.
[22] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid path planning based on safe A ∗ algorithm and adaptive window approach for mobile robot in large-scale dynamic environment," *Journal of Intelligent and Robotic Systems*, vol. 99, no. 1, pp. 65–77, 2020.
[23] J. Zhang, J. Wu, X. Shen, and Y. Li, "Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star," *International Journal of Advanced Robotic Systems*, vol. 18, no. 5, 2021.
[24] H. Wang, C. Hao, P. Zhang, M. Zhang, P. Yin, and Y. Zhang, "Path planning of mobile robots based on A ∗ algorithm and artificial potential field algorithm," *China Mechanical Engineering*, vol. 30, no. 20, pp. 2489–2496, 2019.
[25] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, "Global path planning based on a bidirectional alternating search A ∗ algorithm for mobile robots," *Computers & Industrial Engineering*, vol. 168, Article ID 108123, 2022.
[26] L. W. Yang, L. X. Fu, Q. Wang, L. H. Du, and P. Li, "Multilayer optimal ant colony algorithm for mobile robots path planning study," *Journal of Electronic Measurement and Instrument*, vol. 35, no. 09, pp. 10–18, 2021.
[27] D. Wang, J. Zhang, J. Jin, D. Liu, and X. Mao, "Rapid global path planning algorithm for unmanned surface vehicles in large-scale and multi-island marine environments," *PeerJ Computer Science*, vol. 7, p. e612, 2021.