Hindawi
Security and Communication Networks
Volume 2020, Article ID 6371814, 13 pages
https://doi.org/10.1155/2020/6371814

Research Article

Warehouse-Oriented Optimal Path Planning for Autonomous Mobile Fire-Fighting Robots

Yong-tao Liu,[1,2] Rui-zhi Sun,[1,3] Tian-yi Zhang,[4,5] Xiang-nan Zhang,[1] Li Li,[1] and Guo-qing Shi[1]

[1]*College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China*
[2]*North China Institute of Science and Technology, East Yanjiao, Beijing 065201, China*
[3]*Scientific Research Base for Integrated Technologies of Precision Agriculture (Animal Husbandry), The Ministry of Agriculture, Beijing 100083, China*
[4]*Graduate School of Advanced Integration Science, Chiba University, Chiba 263-8522, Japan*
[5]*Wedo Electric Solutions Technology Co. Ltd., Beijing 100095, China*

Correspondence should be addressed to Rui-zhi Sun; sunruizhi@cau.edu.cn

Received 17 December 2019; Revised 30 January 2020; Accepted 7 February 2020; Published 20 June 2020

Guest Editor: Xiaolong Xu

In order to achieve the fastest fire-fighting purpose, warehouse autonomous mobile fire-fighting robots need to make an overall optimal planning based on the principle of the shortest time for their traveling path. A∗ algorithm is considered as a very ideal shortest path planning algorithm, but the shortest path is not necessarily the optimal path for robots. Furthermore, the conventional A∗ algorithm is affected by the search neighborhood restriction and the theoretical characteristics, so there are many problems, which are closing to obstacles, more inflection points, more redundant points, larger total turning angle, etc. Therefore, A∗ algorithm is improved in eight ways, and the inflection point prior strategy is adopted to compromise Floyd algorithm and A∗ algorithm in this paper. According to the criterion of the inflection point in this paper, the path inflection point arrays are constructed and traveling all path nodes are replaced by traveling path inflection points for the conventional Floyd algorithm backtracking, so it greatly reduces the backtracking time of the smooth path. In addition, this paper adopts the method of the extended grid map obstacle space in path planning safety distance. According to the relationship between the actual scale of the warehouse grid map and the size of the robot body, the different safe distance between the planning path and the obstacles is obtained, so that the algorithm can be applied to the safe path planning of the different size robots in any map environments. Finally, compared with the conventional A∗ algorithm, the improved algorithm reduces by 7.846% for the path length, reduces by 71.429% for the number of the cumulative turns, and reduces by 75% for the cumulative turning angle through the experiment. The proposed method can ensure robots to move fast on the planning path and ultimately achieve the goal of reducing the number of inflection points, reducing the cumulative turning angle, and reducing the path planning time.

1. Introduction

One of the core technologies of autonomous mobile robots is the ability of real time path planning. Path planning refers to, under the premise of following an optimal index (such as the shortest time, the optimal path, and the lowest energy consumption), the optimal path without collision from origin to termination is planned in the usage scenario [1], and the second path planning can be planned in the process of the original path planning in real time.

The premise of path planning needs to build environment map. There are generally two ways to build a map: one is that the robot automatically builds the map through SLAM; the other one is that the site CAD is manually imported into the robot system according to the specified format. Either way, the environmental layout is required to be as fixed as possible.

The object of this paper, warehouse-oriented autonomous mobile fire-fighting robot, is just suitable for this scenario. In the environment, multiple flame detection

probes with address codes are installed on the top. When a flame is detected by a certain probe, the probe code and alarm signal are sent to the robot through edge computing processing and wireless networking technology [2–7]. Here, we need to ensure the security and stability of wireless sensor network and reduce the network delay [8–11]. The robot completes autonomous path planning and reaches the ignition point in the shortest time, and it extinguishes fires early in the fire. The structural space in the scene is relatively stable. In order to adapt to the path planning of the fire-fighting robot, we should take the best rather than the shortest as the principle according to the optimization index path. Because the robot is limited by its own space structure, it is difficult to ensure high-speed travel when it is in a small space and the loss time is much longer than that of path planning. The optimal path is based on the principle of the shortest time, i.e., the robot can achieve the fastest traveling path, and the fire-fighting robot can reach the fire point in the shortest time to put out the fire. This requires that, under the premise of ensuring the shortest planning path, the path and obstacles (such as shelves) have enough safe distance, reserved robot turning action space, etc. And, the number of turning points is less; the total turning angle is lower.

After years of development, the path planning algorithm has achieved good results, but there are still some problems in specific application scenarios. The proposed method of the artificial potential field by Khatib [12] in 1994 falls into local minimum and oscillate near obstacles easily. In genetic algorithm (GA) [13], the range of coding length is large and the convergence is bad. In the complex map mode, the convergence speed of the algorithm is slow and the computation is large. In artificial neural network (ANN) [14], when the initial feedback information of the algorithm is insufficient, it needs to adjust the weight in a long time to achieve the optimization effect of the project approval. The search speed of the ant colony algorithm is slow and falls into local optimality easily [15]. The Dijkstra algorithm is a classical breadth prior algorithm [16], and it can always find the optimal path and conduct the overall search directly. However, it does not consider the target point information, and it has a long search time and low search efficiency because it cannot meet the need of rapid path planning. BFS optimal prior algorithm can quickly guide the search to the target node and greatly improve the search efficiency but cannot often get the shortest path.

A∗ algorithm, first proposed in 1968 by Hart et al. [17], is a heuristic search algorithm [18–20] combined with the advantages of the above algorithms. It uses heuristic information to guide the search direction, so as to reduce the search scope and improve the search efficiency. It is a typical heuristic path planning algorithm [21, 22], which has been successfully applied and verified in mobile robot path planning [23].

However, due to the computational characteristic of A∗ algorithm, the planned path points generally have many problems, such as many broken lines and large cumulative turning angle, and easily discard some points to generate suboptimal problems [24]. Therefore, many researchers have proposed improved algorithms in computational time [25, 26]. Gao et al. [27] proposed a bidirectional time-effective A∗ algorithm to find the path and adopted a multineighborhood grid distance computational scheme to achieve the effect of improving efficiency and smoothing the path, but it is not suitable for large-scale complex maps. In [28], an improved search A∗ algorithm combined with skip points is proposed, the speed of the algorithm has been greatly improved, but there are still many inflection points and close to obstacles. In [29], the size of the robot is considered, and a neighborhood matrix is proposed to improve the path safety of obstacle search. However, the path smoothing and length are not improved, so there are still many problems including more turning angles and large total turning angle. In [30, 31], proposed expand search neighborhood and dispose of quadratic smoothing of quintic polynomial reduce the length of search path and turning angle and greatly improve path smoothing. However, there are still some problems, such as closing to obstacles, and safety distance is not considered.

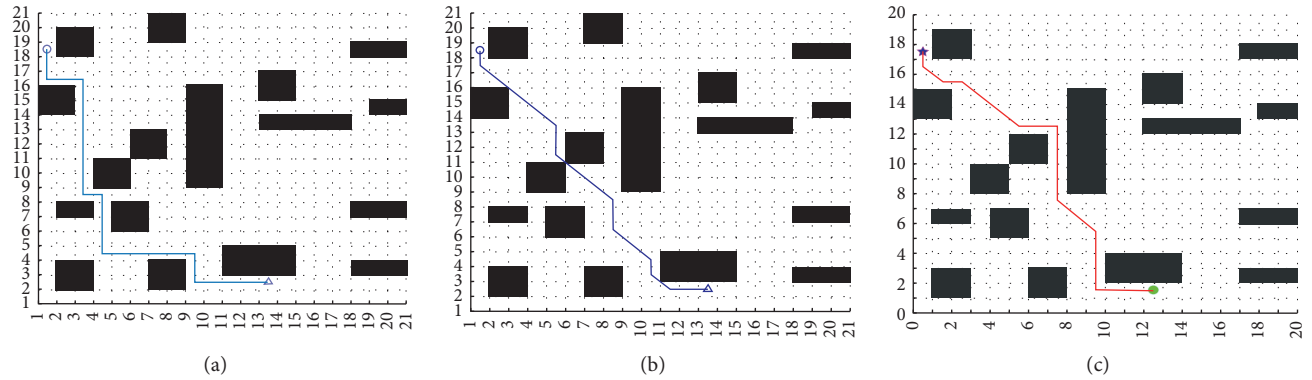
Aiming at the problems of usage scenarios of warehouse-oriented fire-fighting robots and the existing various optimizational A∗ algorithm, this paper proposes a comprehensive improved A∗ algorithm to achieve path smoothing and reduces broken lines and cumulative turning angle. And, according to the bulk of the robot and the scale of map, we set up safe distance with the obstacles to meet the path planning application of the robots.

The paper is organized as follows. In Section 2, we propose the improved A∗ algorithm and smoothen the optimal path. Then, we introduce the constraint conditions including robot's safety traveling distance and compare with the effect of the current latest algorithm. Section 3 introduces the practical application effect of the complete algorithm in this paper and verifies a great impact of safety distance on the path. Section 4 concludes the paper.

2. Improved A∗ Algorithm

The planning effect of the conventional four-way search A∗ algorithm is shown in Figure 1(a). There are a lot of right-angle points in the route, and the length of the route is not the shortest. On this basis, eight-way search A∗ algorithms are proposed; the planning effect is shown in Figure 1(b). It can be seen that the eight-way search A∗ algorithm overcomes the problem of right-angle inflection point in the path and plans the shortest path length under the algorithm. However, the algorithm has the problems of closing to the obstacles and crossing the diagonal vertex of the obstacles (Figure 1(a)); this path cannot be used as the traveling path of the robot obviously. Therefore, the conventional eight-way search A∗ algorithm is not suitable for robot path planning and it needs to improve the basic safety distance and the constraint.

2.1. Improved Eight-Way A∗ Algorithm. In view of the existing problems of conventional eight-way search A∗ algorithm, we make a basic improvement, i.e., when the eight-way search area is expanded and if the path forward direction is a slash, the constraint term is added, for example,



(a) (b) (c)

Figure 1: The comparisons of A∗ algorithm comparison. (a) Conventional four-way A∗ algorithm. (b) Conventional eight-way A∗ algorithm. (c) Improved eight-way A∗ algorithm.

in Figure 1(c), when the path is planned according to a lower right slash, then we judge whether there is any obstacle on the right and bottom sides of the parent-node. If there is, the path is not feasible and the subnode needs to be selected again. When the path is planned according to upper left slash, then we judge whether there is any obstacle on the left and upper sides of the parent-node. If there is, the path is not feasible, the subnode needs to be selected again. The planning path according to this method is shown in Figure 1(c).

The performance comparisons of the three algorithms in Figure 1 are shown in Table 1. It can be seen that the conventional four-way search A∗ algorithm has the longest path length and the slowest operation time. The conventional eight-way search A∗ algorithm has the shortest path length and the fastest operation time, but the path has the problems of closing to the obstacle vertex and crossing the obstacle vertex angle, so the path cannot be used by robots. On this basis the eight-way search, the A∗ algorithm is improved in this paper and the shortest path is realized under the algorithm. Compared with the conventional four-way search A∗ algorithm, the path length reduces by 12.53%, the operation time reduces by 46.9%, and the situation closing to the obstacle vertex and crossing the corner is effectively avoided.

2.2. Improved Floyd Path Smoothing Algorithm. In [30], traversing nodes are used for the smoothing process. In [32], Floyd algorithm is used for the smoothing process. These two methods need to trace back all path nodes one by one from the origin or termination to determine whether they intersect obstacles. If they do not intersect, the intermediate node will be discarded, and if they intersect, the previous node will be retained. These two methods can optimize the original path, but traversing all the path nodes significantly reduces the execution speed, and the straight path nodes in the original path can avoid traversing judgment completely. Therefore, this paper proposes the criterion of the prior inflection point, and Floyd algorithm is used to connect the backtracking method of the inflection point.

2.2.1. Criterion of the Planning Route Inflection Point. Before path smoothing is completed, the eight-way A∗ algorithm path planning has to be improved. Inflection criteria are added after planning; path inflection array is generated. For example, let point D in Figure 2 be the current node n, according to the A∗ algorithm formula:

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

where $g(n)$ indicates the length of the actual path from the origin to the current node and $h(n)$ is the distance of estimate cost function from the state node n to termination.

According to the eight-way extended A∗ algorithm, there are only two path lengths of adjacent nodes, i.e., L and $\sqrt{2}L$. If the distance from the current node n to its parent-node $(n-1)$ and sub-node $(n+1)$ is not equal, then the node in the planning path is the inflection point. The criteria are as follows:

$$L(X_{n-1}, X_n) \neq L(X_n, X_{n+1}). \tag{2}$$

For example, if $L(C, D) \neq L(D, E)$, then the planning path node D is the inflection point; if $L(D, E) = L(E, F)$, then the planning path node E is a normal node. In the judgment, all the noninflection nodes in the planning path are expressed as (x, y, 0); the inflection nodes are expressed as (x, y, 1); the origin of the robot path is defined as a common node; the termination is defined as an inflection. All the nodes are saved in the file-list array. In the last path backtracking, simply priorly connecting the inflection points can get the path smooth optimization.

2.2.2. Design of Floyd Algorithm. We combine Floyd algorithm and the improved A∗ algorithm; the length of the original planning path can be shorter; the number of inflection points can be less; the cumulative turning angle can be less; the path can be smoother.

In the smooth process, firstly, we need to get the file list of the inflection points of the planning path and directly connect the second inflection point in the backtracking path when backtracking from the termination. If it does not intersect the obstacle, then the third inflection point should be connected. If the connection encounters the obstacle, all nodes between the third inflection point and the second inflection point are traced back in turn until the node that

Table 1: The comparisons of performance efficiency for A∗ algorithm.

Origin termination	Algorithm	The length of the path	Turn times
(18, 1) (2, 13)	Four-way A∗	28.00	0.0678
	Eight-way A∗	22.14	0.0196
	Improved eight-way A∗	24.49	0.036

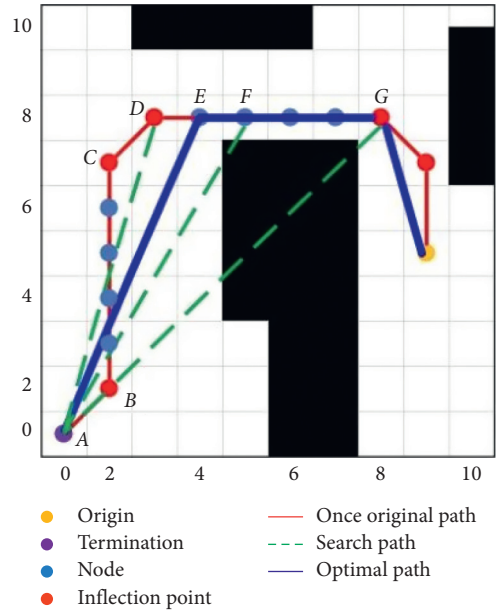


Figure 2: The principle of improved A∗ algorithm.

does not intersect the obstacle is found, and the intermediate node is discarded.

In Figure 2, if points A and B are adjacent inflection points, then they can be connected directly. The distance is written as $L(A, B)$. Judge $A \longrightarrow C$ and $A \longrightarrow D$ in turn; if they do not intersect obstacles when they are connecting, the original path can be changed to $A \longrightarrow D$. But, when $A \longrightarrow G$, they intersect obstacles, so the path length $L(A, G) = +\infty$:

$$L(A, C) < L(A, B) + L(B, C). \tag{3}$$

Then, the array of points A and C are retained, and the array of point B is deleted:

$$L(A, D) < L(A, C) + L(C, D). \tag{4}$$

Then, the array of points A and D are retained, and the array of point C is deleted.

Because $L(A, G) = +\infty$,

$$L(A, G) = L(A, D) + L(D, G). \tag{5}$$

At this time, assume the coordinate of point G is $G(i, j, 1)$, then the original path node between D and G will be traced back from point G, $G_1(i-1, j, 0)$, $G_2(i-2, j, 0)$, $G_3(i-3, j, 0)$, and $G_4(i-4, j, 0)$, where $G_3 = F$ and $G_4 = E$. When connected with point A, $L(A, G_1) = +\infty$, $L(A, G_2) = +\infty$, and $L(A, F) = +\infty$ are abandoned:

$$L(A, E) + L(E, G) < L(A, D) + L(D, G). \tag{6}$$

Then, the array of points A and E are retained, and the array of point D is deleted. The original path $A \longrightarrow B \longrightarrow C \longrightarrow D \longrightarrow$ G is optimized as $A \longrightarrow E \longrightarrow G$, so far, this section of path optimization is completed, and the next path optimization is carried out accordingly.

The specific implementation steps of the path smoothing algorithm are as follows:

(i) Step 1: take out the processed nodes of the above improved eight-way A∗ algorithm and carry out the secondary backtracking analysis from the termination.

(ii) Step 2: start the termination to connect all the inflection points one by one, $B \longrightarrow C \longrightarrow D$, and then the nodes of noninflection points, such as the points between $B \longrightarrow C$, are ignored during the connection process.

(iii) Step 3: when points A and D are directly connected, continue to search backward inflection point G. At this time, there will be a conflict between search path $A \longrightarrow G$ and obstacles. Then, search the nodes of noninflection point between $G \longrightarrow D$.

(iv) Step 4: if there is still a conflict with the obstacle when searching to node F, then continue to Step 3

until node E, that is, no conflict with the obstacle is found.

(v) Step 5: at this time, take the direct connection path $A \longrightarrow E$ as the fixed path and repeat step $2 \longrightarrow 3 \longrightarrow 4$ from node E until reaching the origin.

The flowchart of the algorithm is given in Figure 3.

Compared with [33], the method only makes redundant judgments for inflection points; it adds the backtracking function of common nodes between obstacle inflection nodes and makes the path distance shorter. Compared with [30, 32], the traversal connection judgment time of the smoothing path algorithm is significantly reduced. In the method, all node connection judgments are no longer performed, and only the inflection nodes in the eight-way A∗ algorithm planning nodes are preferentially connected twice. There is a conflict with obstacles, and then node by node from the inflection node back is connected; the number of connection judgments is reduced greatly; the path smoothing processing time becomes shorter.

An array of once planned path node *int a* [3] [6] is shown as

$$\left\{\begin{array}{l} 0,0,0, \\ 2,2,1, \\ 2,3,0, \\ 2,4,0, \\ 2,5,0, \\ 2,6,0, \\ 2,7,1, \\ 3,8,1, \\ 4,8,0, \\ 5,8,0, \\ 6,8,0, \\ 7,8,0, \\ 8,8,1, \\ 9,7,1, \\ 9,6,0, \\ 9,5,1 \end{array}\right\}. \tag{7}$$

It can be seen that there are five inflection nodes besides origin [0, 0, 0] and termination [1, 5, 9] in once planned path, and the following five inflection nodes are used as the prior traversal point:

$$\left\{\begin{array}{l} 2,7,1, \\ 3,8,1, \\ 8,8,1, \\ 9,7,1, \\ 9,5,1 \end{array}\right\}. \tag{8}$$

After executing the optimized algorithm based on Floyd algorithm, the final path contains only 3 nodes and the noninflection points in the once path. The optimal path node array *int a* [3] [3] is shown as follows:

$$\left\{\begin{array}{l} 4,8,0, \\ 8,8,1, \\ 9,5,1. \end{array}\right\}. \tag{9}$$

2.2.3. Simulation Analysis of the Confluent Improved Algorithm. To verify the effect of the proposed confluent improved Floyd and improved A∗ algorithms in this paper, comparisons are made between the basic improved A∗ algorithm and the proposed prior moving route algorithm in [34]. The optimized path in [34] is shown in the red path in Figure 4(a). The length of the path is actually 21.4852 grids, but the path has the problem of closing to the apexes of the obstacles and the possibility of crossing the top angle of the obstacles, so the path is not suitable as a robot planning path.

This paper basically improves the path planning of the A∗ algorithm and adds grid constraints to avoid closing to obstacle paths. The planned path is shown in Figure 4(b).

Finally, path smoothing process is added based on the path planning of the improved A∗ algorithm, and the processing effect is shown in the blue path in Figure 4(c).

In the case that the initial orientation of the robot is not considered in the three paths in Figure 4, the obtained data graph by the experimental simulation is shown in Table 2. Compared with the improved eight-way A∗ algorithm, in this paper, the path length of the confluent algorithm reduces by 7.846%, the number of cumulative turns reduces by 71.429%, and the cumulative turning angle reduces by 75%. Compared with the improved A∗ algorithm in [34], in this paper, the path length of the smoothing algorithm reduces by 0.02%; the length optimization is not obvious. But the number of the cumulative turns reduces by 66.67%, and the cumulative turning angle reduces by 57.91%. The advantage of the part is obvious, the path is more conducive to the execution of the robot.

So far, although the proposed confluent algorithm in this paper has obvious advantages in cumulative turning points and cumulative turning angles, there are still cases crossing the vertices of obstacles, so it needs to further optimize the path safety distance considering the actual situation of the robot body space.

2.3. Optimization Algorithm of the Path Planning Safety Distance. For the reasonable traveling path of the robot, we should fully consider its own space structure and a sufficient safe distance from the obstacles should be reserved in the path planning. In this way, the robots can reduce the detection and processing time required to avoid obstacles when it actually follows the path in the later stage. Although this paper achieves the goal of the smoothing path by improving the Floyd algorithm, the optimized path still has the situation closing to the obstacles. Therefore, this paper continues to lead into the concept of path safety space in the design for algorithm optimization.

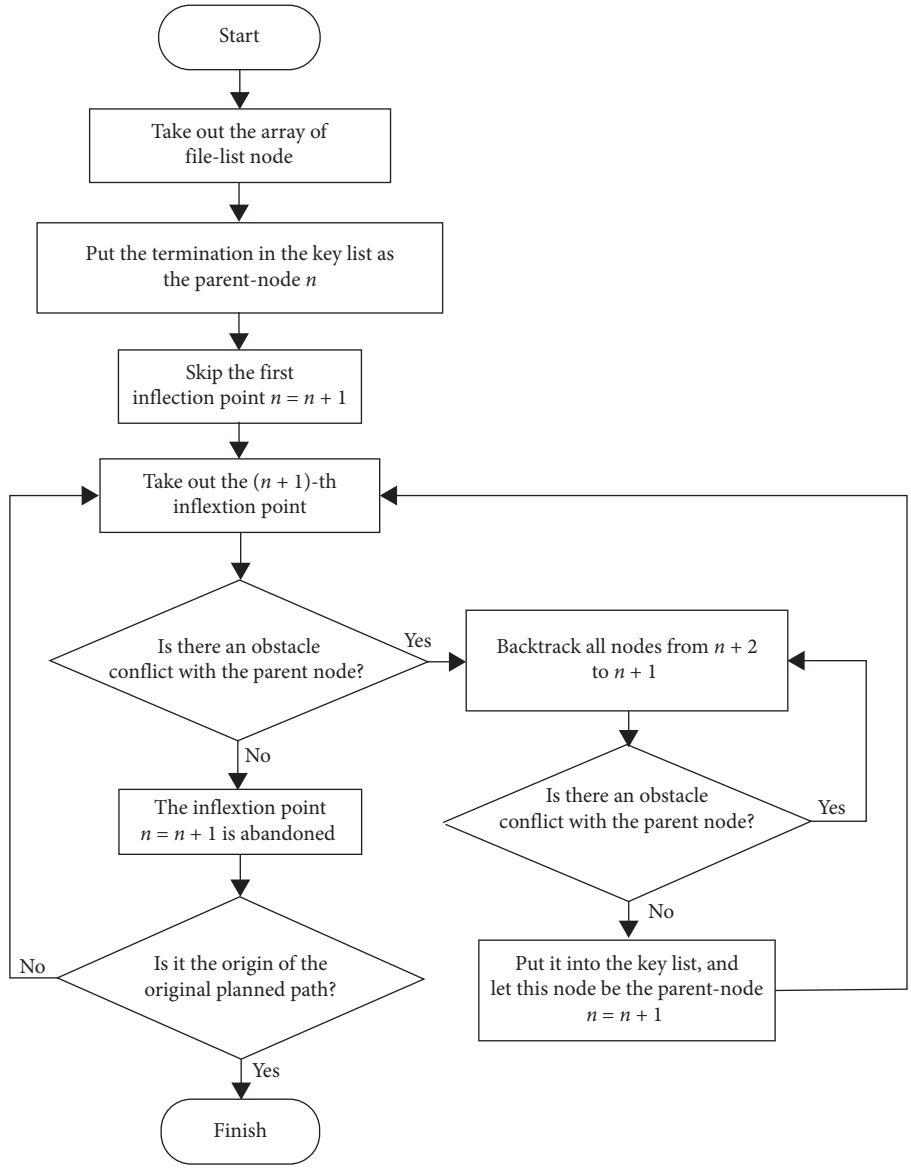


Figure 3: Algorithm flowchart.

2.3.1. Optimization Algorithm of the Safety Distance Based on Extended Obstacle Boundary. In order to ensure the space safety distance of the path, there are generally two methods. One is to expand the pixels occupied by the robot body within the map pixels and make the robot body space consistent with the actual map pixel ratio. The cross-obstacle search algorithm and the *m*-shaped obstacle search algorithm are given in [29]. The algorithm obtains different safety distances by changing the size of the search matrix to ensure the safety of the different robots in different map environments. The other one is the method that is to extend the obstacle boundary. This method combining with the size of the robot can set the safety distance arbitrarily and intuitively without increasing the difficulty of the algorithm, and it can effectively reduce the number of traversal grids in the search area and the search time.

Note: in the following path planning diagrams, the red path is the planned path of the basic improved eight-way A∗ algorithm; the blue path is the smoothed planned path, which is the final path; the colored grid is all the traversal grids in the path planning process.

The planned route is the planned route obtained in [29] using the 12-neighborhood search algorithm in Figure 5(a). The method can improve the search efficiency, and it also leads to the increase of the path length. The algorithm is designed with the path length as the primary priority and the search time as the secondary priority. In Figure 5(b), the red path is the improved eight-way A∗ algorithm planned path in this paper. The path is linked by the grid center point. The planned path can also achieve a safety distance to a certain degree. However, it is not flexible enough and cannot be set arbitrarily according to the robot map ratio.

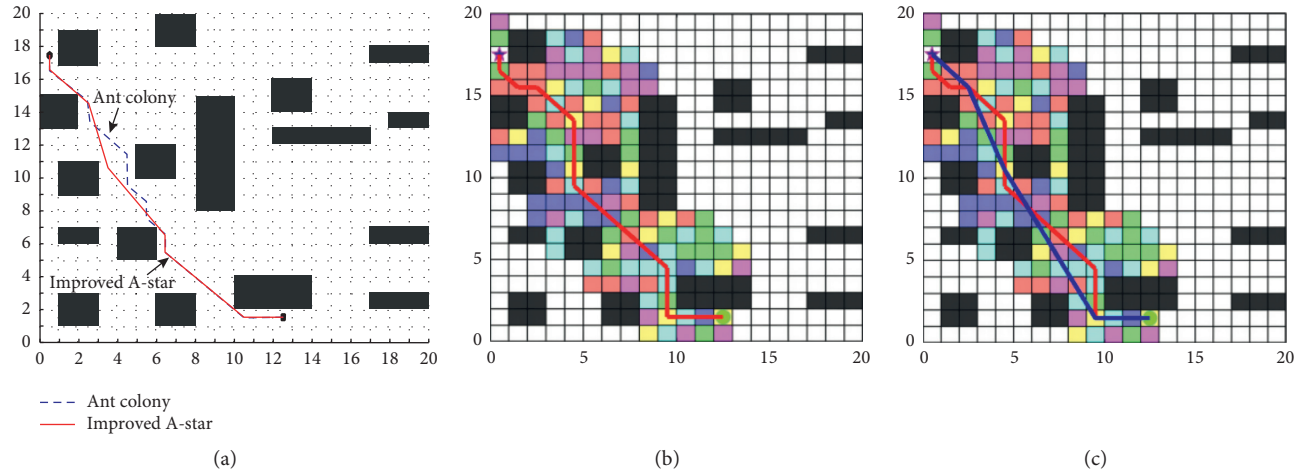


Figure 4: The improved path smoothing algorithm process. (a) The improved algorithm in [34]. (b) The improved algorithm in this paper. (c) The confluent path smoothing algorithm.

Table 2: Comparisons of key parameters for the algorithmic effectiveness.

Algorithm	The length of path	Inflection points	Cumulative turning point (°)
Improved eight-way A∗	23.31	7	360.00
Algorithms in [25]	21.4852	6	213.84
Smoothing algorithm in this paper	21.4809	2	90.00

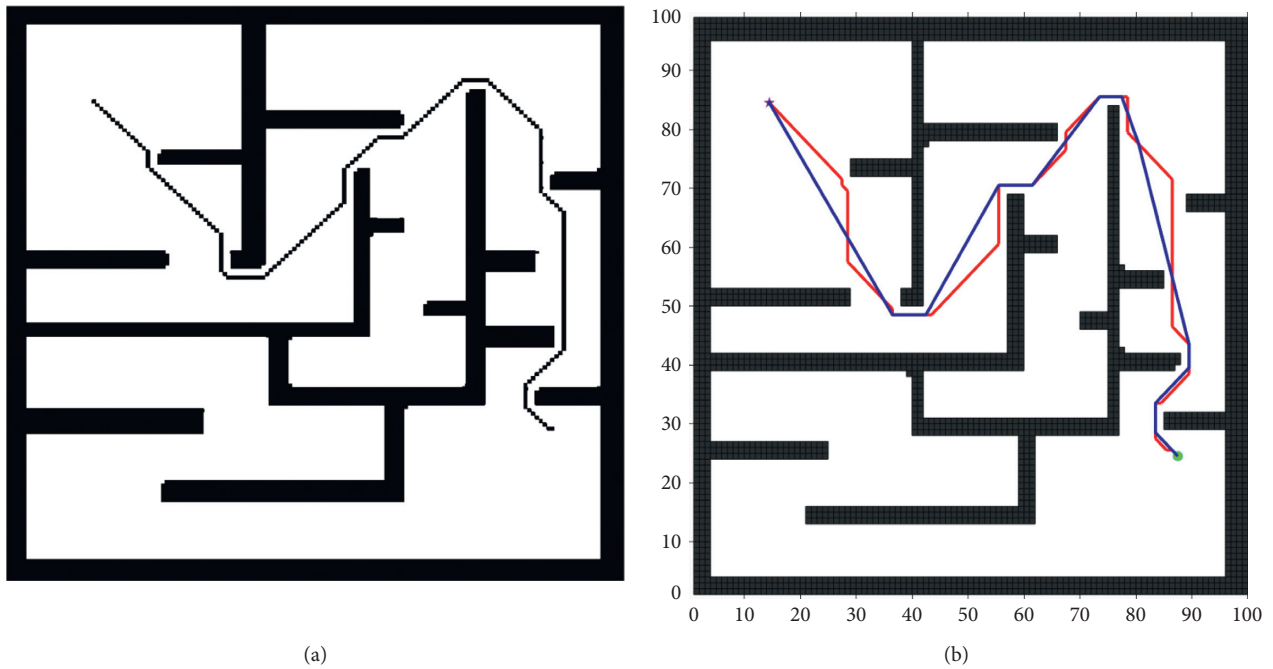


Figure 5: The comparison of the safety distance solutions. (a) Cross barrier search in [29]. (b) Extended obstacle boundary method in this paper.

It can be seen in Table 3 that the improved A∗ smoothing algorithm adopts the search method of expanding obstacle boundary in this paper; compared with [29], the path length is shorter and the number of traversed grids is smaller. The number of the path inflection points that is a key indicator for evaluating the robot's traveling path has also decreased 44.4%; the corresponding cumulative turning angle has decreased significantly; the path smoothness advantage is very obvious.

Table 3: The comparisons of two safety distance search methods.

Algorithm	The length of path	Inflection points	Ergodic grid number
12 neighborhood cross search algorithms in [29]	224	18	7957
Extended barrier boundary method	169.81	10	3739

2.3.2. Impact Analysis of the Path Safety Distance. Although the constraint condition of safety distance is required in path planning, the design path is more suitable for robot walking. On the surface, the path designed by the algorithm should be as far away from the obstacles as possible, but when this safety distance is set too large, the path cannot be planned. As shown in Figure 6, this paper has tested and analyzed the situations of the different safety distances.

In Figure 6, the path planning test was performed on a map of 25×25 grid pixels for each without safety distance constraint, 0.6 grid safety distance, and 1.0 grid safety distance.

According to the analysis in Table 4, it can be found that when there is no safety distance, the planned path is the shortest; the number of inflection points is centered; the number of traversed grids is the largest; and the planning time is the longest. At 0.6 grid safety distance, the path length is 1.2% longer than the shortest path; the number of inflection points reduces by 50%; the number of traversal grids reduces by 62.6%; and the planned time reduces by 37.79%. 1.0 grid safety distance has the longest path length, the largest number of inflection points, and the shortest planned time. If the safety distance grid is set to 2.0, path planning cannot be completed.

Under the condition that the robot path planning guarantees a safe distance, the time spent in path planning can be ignored compared with the time saved by robot turning and fast passing. Therefore, it can be ensured that the second shortest path with the least number of inflection points and the smallest cumulative turning angle of the robot is the optimal path of the robot.

3. Autonomous Mobile Robot Warehouse Center Path Planning

After the algorithm design is completed, it needs to be verified in the actual map environment. Because the robot application environment is a structural layout space, the verification uses the warehouse center as the verification object, and the warehouse map is derived from Baidu Gallery. Robot path planning must first establish an accurate two-dimensional map of the environment space and then perform path planning.

3.1. Construction of Environmental Map for Warehouse Center. The map construction in the normal A$*$ algorithm uses the method that directly creates a two-dimensional array in the program, as shown in Figure 7(a), and the produced grid map is shown in Figure 7(b). "0" in the array indicates that the open grid is in white, and "1" indicates that the obstacle is in black. Each "0" or "1" represents a grid, and the number of rows and columns of the two-dimensional array corresponds to the number of rows and columns of the grid map.

This representation method is flexible to change the map and is suitable for low-resolution map construction. For large-space and high-resolution maps, the two-dimensional array is huge and it is difficult to accurately match the actual space. For institutional spaces such as warehouse centers, accurate CAD drawings are generally available during construction, as shown in Figure 8 (The Figure 8 from the Internet, the website is: http://www.51w2c.com/details_id_1347.html). The actual positions and absolute coordinates of various items are accurately marked on the drawings, and the positions will not basically change easily. Low-resolution grid maps in this scenario will cause large errors and cannot accurately reflect the precise coordinate positions of obstacles and feasible paths, and it is easy to make the planned path deviates from the actual path, causing the robot to excessively rely on obstacle avoidance function when it is traveling.

Aiming at the above problems, in order to improve the practicability of the system, this paper designs a drawing program by using Matlab's composition conversion function. When the software is imported, we must first draw the warehouse center (Figure 8) into one of the three formats bmp, jpg, and png according to the size proportion and position, as shown in Figure 9. Select the imported image in the GUI interface designed by Matlab, set the required horizontal axis resolution of the grid image, and complete the conversion according to the horizontal and vertical proportions of the original image.

According to this step, the actual warehouse center shown in Figure 8 is converted into the binary map of Figure 9 for the application scenario and imported into Matlab R2017a for experimental verification.

In the scenario, the actual space of the CAD drawing of the warehouse center is 4476×4000 cm. The two-dimensional space of the robot is 60×70 cm. The two-dimensional size of a single shelf is 60×200 cm. The road width between the shelves is 150 cm.

The higher the resolution during the construction of the grid map, the smaller the error will be, but the time consumption with the system path planning will increase greatly. Therefore, there are three preliminary methods for setting the resolution:

(1) Take the two-dimensional space of the robot as the grid point size. The robot adopted a two-wheel differential turning method, and the width of the forward direction needs to be 60 cm. Taking this as the minimum resolution of the grid map, the resolution of the warehouse center grid map is 43×48.

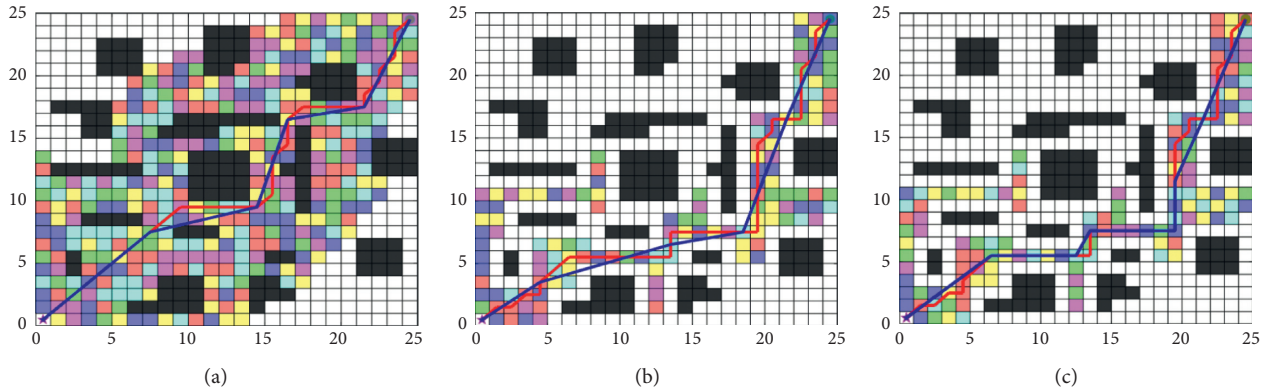


Figure 6: The effect of path planning at different safety distances. (a) No safety distance. (b) Safety distance 0.6 grid. (c) Safety distance 1 grid.

Table 4: Effect analysis of path planning for different safe distances.

Algorithm	The length of path	Discount points	Ergodic grid number	Planning time (S)
No safe distance	37.17	4	361	0.4424
0.6 grid safe distance	37.62	2	135	0.2749
1.0 grid safe distance	39.98	5	135	0.2272

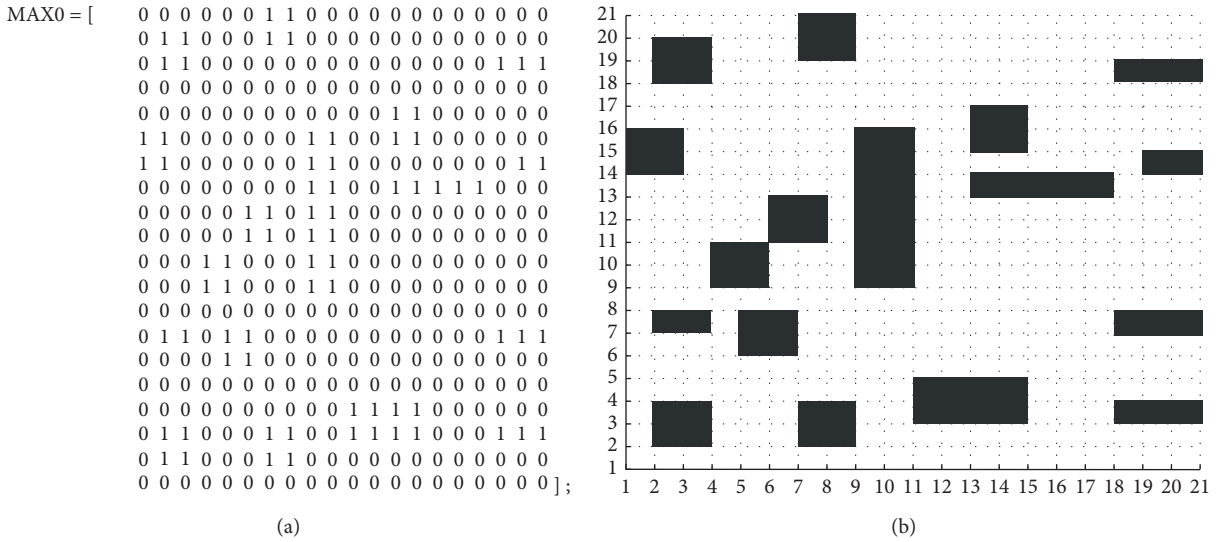


Figure 7: The grid map creation. (a) Map array. (b) Corresponding grid map.

(2) Use shelf width as a reference and consider most path widths. It can be seen from the CAD drawing that the width of most of the auxiliary passages between the shelves is 150 cm, so the grid size of the storage center is considered to be 30 cm, and the resolution of the warehouse center grid map is 132×149.

(3) According to the CAD map, the grid is drawn according to a certain proportion and the grid size is determined by the path planning time.

The Matlab image conversion program design process is shown in Figure 10.

3.2. Path Planning Verification

3.2.1. Impact of Different Resolutions on Planning Time. Path planning experiments were performed on the grid maps designed to be constructed at multiple resolutions. The obtained data are shown in Table 5.

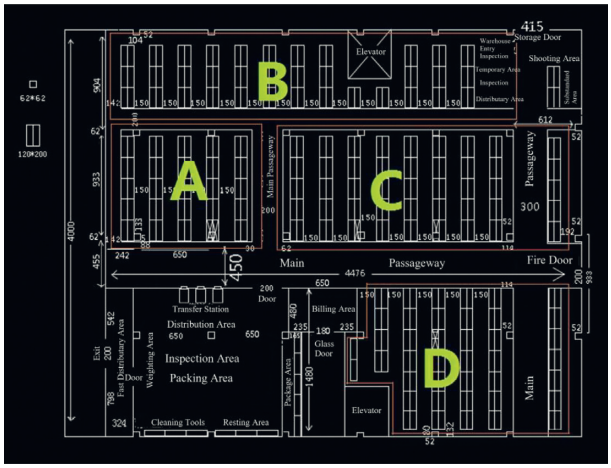


Figure 8: Actual layout of a warehouse center.

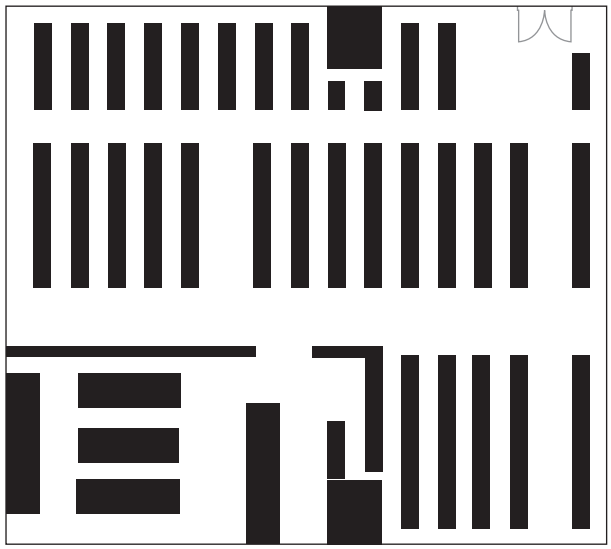


Figure 9: Binary map for warehouse center CAD.

It can be seen that the larger the number of grids on the same map, the higher the resolution, the smaller the spatial error of the map expression, but the longer the path planning time. When the number of grids is 132, the grid map can basically and accurately express the spatial layout of shelves and aisles in the warehouse center, and the planning time can be completed within 0.5S.

3.2.2. Impact of Different Safety Distances on Planned Paths. At the same grid resolution, different safety distance settings will have a greater impact on path planning.

The warehouse maps are set at different security distances and at the same 132 grid resolution, the obtained path planning results are shown in Figures 11(a) and 11(b), and the specific experimental data are shown in Table 6.

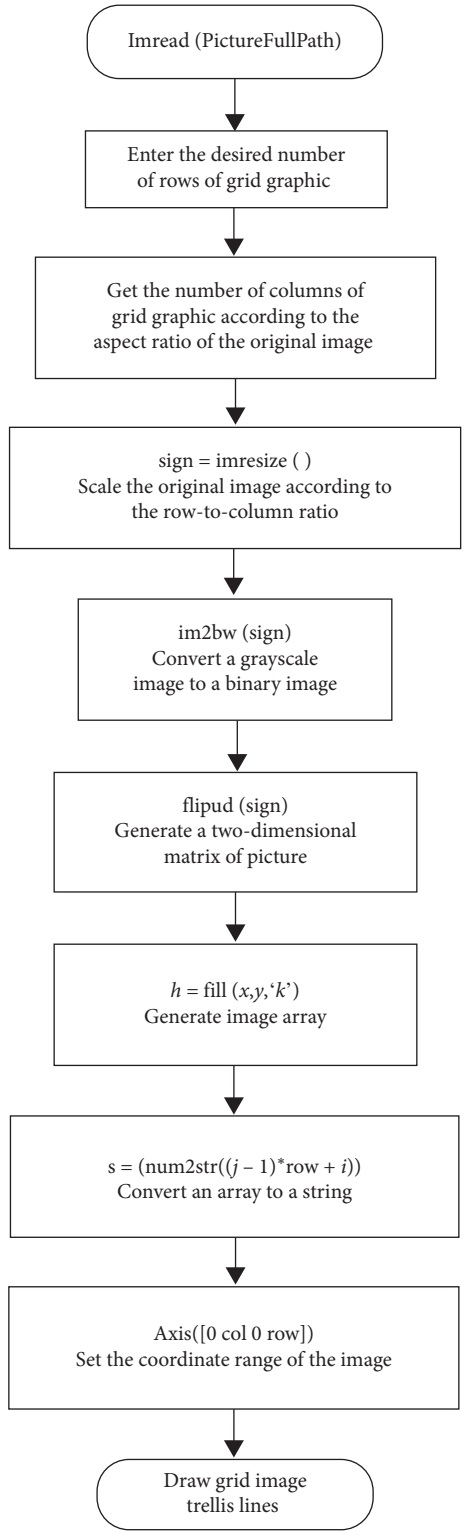


Figure 10: Grid map creation process.

Figure 11(a) shows the path planning without the safety distance. There are many cases where the path is close to the obstacles. The path planning passed the narrow section. The

Table 5: Path planning time at different grid resolutions.

Safe distance	Planning time/(S)	The length of path	Discount points	Ergodic grid number
Body width	0.4976	194.445	7	7807
15 cm	0.2711	205.5228	4	6034

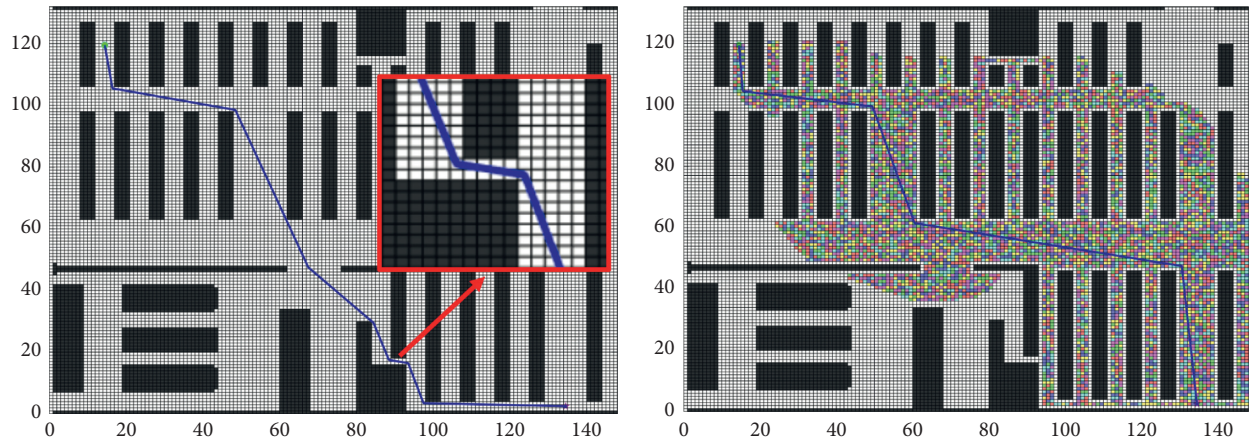


Figure 11: 132 grid maps with different safety distance paths. (a) No safety space path. (b) 10 cm safety space path.

Table 6: Effects of different safety distances on grid path planning.

Algorithm	Grid size (cm)	Safe distance	Planning time (S)
66	60.0	Body width	0.0975
132	30.0	10 cm	0.3421
200	20.0	10 cm	0.6429
400	10.0	10 cm	8.1772
800	5.0	10 cm	103.3154

road width is only 2 grids, and the actual width is 60 cm; it is shown in the enlarged path of Figure 11(a); it is exactly the same as the width of the robot body. Obviously, the section of the robot is very difficult to pass or even unable to pass. The path planning has 7 inflection points, and the robot needs to perform steering actions multiple times. Both of these problems greatly affect the traveling time of robot.

In Figure 11(b), a grid closing to the obstacle is placed in the closed-list by expanding the obstacle area, and it is set to prohibit traversal. The method reserves a 15 cm safety distance between the robot path and the obstacles. Due to the introduction of the safety distance, the planned path has changed greatly compared with that in Figure 11(a). Because it is difficult for the robot to travel at high speeds in narrow sections, increasing the safety distance ensures that the robot travels on the secondary shortest path, but it ensures that the robot can travel at full speed and reduces turns. Compared with the path without safety distance, the number of turns of the path reduces by 42.8% and the planning time reduces by 44.91%.

It can be seen that the correct and reasonable safety distance setting not only determines whether the algorithm can plan the correct path, but also can greatly improves the path optimization. Therefore, the path planning safe distance must be considered in practical applications.

4. Conclusions

The shortest path cannot be as the judgment that the path is optimal for warehouse autonomous mobile fire-fighting robots in indoor structure space. It should also be based on the safety distance between the path and the obstacle to make the robot move at high speed, and the executive time is the shortest. Therefore, this paper improves the problems of the conventional A∗ algorithm in the path planning of autonomous mobile robots.

In the design, Floyd algorithm and A∗ algorithm are fused by the inflection point priority strategy. Experiments show that the method can reduce the path optimization time and significantly reduces the total number of the path of the inflection points and the cumulative turning angle and thus shortens the path length and increases the smoothness of the path. Finally, the problem of safe path of all kinds of robots in different space is solved by expanded obstacles; the time of the path planning is reduced greatly; the path is optimized in many aspects; the planning efficiency and the algorithm practicability are improved.

Although the method of the path planning has achieved good experimental results, there is still a long planning time in the process of high-resolution map path planning. In the next research, we will consider how to add new search heuristic functions and constraints, so as to improve the search efficiency of high-resolution grid map in complex environment and ensure the algorithm has better applicability.

In the future, the fire-fighting ability of a single robot will not meet the demand for a large indoor space or a flammable environment. Therefore, we will also consider the cooperation and game of multirobots in this environment [35] and study the wireless mobile networking and anticollision theory of multirobots [36].

Data Availability

The MATLAB path planning data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Fundamental Research Funds for the Central Universities (3142018047).

References

[1] E. Frazzoli, A. Munther, and M. A. Dahleh, "Real-time motion planning for agile autonomous vehicles," in *Proceedings of the 2001 American Control Conference*, vol. 25, no. 1, pp. 43–49, IEEE, Arlington, VA, USA, June 2001.

[2] X. Xu, Y. Li, T. Huang et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.

[3] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[4] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.

[5] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[6] H. Wu, "Performance modeling of delayed offloading in mobile wireless environments with failures," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2334–2337, 2018.

[7] J.. Zhao, X.. Guan, and X.. Li, "Power allocation based on genetic simulated annealing algorithm in cognitive radio networks," *Chinese Journal of Electronics*, vol. 22, no. 1, pp. 177–180, 2013.

[8] X. Xu, S. Fu, L. Qi et al., "An IoT-Oriented data placement method with privacy preservation in cloud environment," *Journal of Network and Computer Applications*, vol. 124, pp. 148–157, 2018.

[9] H. Wu, Z. Han, K. Wolter, Y. Zhao, and H. Ko, "Deep learning driven wireless communications and mobile computing," *Wireless Communications and Mobile Computing*, vol. 20192 pages, 2019.

[10] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[11] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings of the 1994 IEEE*, Munich, Germany, 1994.

[13] Q. Bu, Z. Wang, and X. Tong, "An improved genetic algorithm for searching for pollution sources," *Water Science and Engineering*, vol. 6, no. 4, pp. 392–401, 2013.

[14] J. Ye, "Tracking control of a nonholonomic mobile robot using compound cosine function neural networks," *Intelligent Service Robotics*, vol. 6, no. 4, pp. 191–198, 2013.

[15] M. M. Mohamad, M. W. Dunnigan, and N. K. Taylor, "Ant colony robot motion planning," in *Proceedings of the EUROCON 2005-The International Conference on "Computer as a Tool"*, November 2006.

[16] F. Zhang, J. Liu, and Q. Li, "A new way of network analysis based on dijkstra," *Remote Sensing Information*, vol. 2, pp. 38–41, 2004.

[17] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[18] F. Duchoň, A. Babinec, M. Kajan et al., "Path planning with modified a star algorithm for a mobile robot," *Procardia Engineering*, vol. 96, no. 1, pp. 159–169, 2014.

[19] S. Uttendorf, B. Eilert, and L. Overmeyer, "Combining a fuzzy inference system with an A∗ algorithm for the automated generation of roadmaps for automated guided vehicles," *At-Automatisierungstechnik*, vol. 65, no. 3, pp. 189–197, 2017.

[20] M. Wodziński and A. Krzyżanowska, "Sequential classification of palm gestures based on A∗ algorithm and MLP neural network for quadrocopter control," *Metrology and Measurement Systems*, vol. 24, no. 2, pp. 265–276, 2017.

[21] S. G. Cui, H. Wang, and L. Yang, "A simulation study of a-star algorithm for robot path planning," *16th Int Conf on Materials. Beijing*, vol. 282, pp. 33–38, 2013.

[22] A. R. Soltani, H. Tawfik, J.Y. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms," *Advanced Engineering Informatics*, vol. 16, no. 4, pp. 291–303, 2002.

[23] F. Duchoň, D. Huňady, M. Dekan, and A. Babinec, "Optimal navigation for mobile robot in known environment," *Applied Mechanics and Materials*, vol. 282, no. 1, pp. 33–88, 2013.

[24] Y. Qin, H. Wang, and C. Du, "Mobile robot path planning based on double-layer A∗ algorithms," *Manufacturing Automation*, vol. 24, pp. 21–25, 2014.

[25] M. Hawa, "Light-assisted A∗ path planning," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 2, pp. 888–898, 2013.

[26] D. František, B. Andrej, and K. Martin, "Path planning with modified A star algorithm for a mobilerobot," *Procedia Engineering*, vol. 96, no. 96, pp. 59–69, 2014.

[27] M. Gao, Y. Zhang, and L. Zhu, "Bidirectional time- efficient A∗ algorithm for robot path planning," *Application Research of Computers*, vol. 36, no. 4, pp. 1–6, 2019.

[28] X. Zhao, Z. Wang, and C. Huang, "Mobile robot path planning based on an improved A∗ algorithm," *Robot*, vol. 40, no. 6, pp. 903–910, 2018.

[29] R. Chen, C. Wen, and L. Peng, "Improve A∗ algorithm and apply to indoor path planning for mobile robots," *Journal of Computer Applications*, vol. 39, no. 4, pp. 1006–1011, 2019.

[30] Y. Ren, F. U. Lixia, and Y. Zhang, "Smoothing A∗ algorithm extended search neighborhood for robot path planning," *Electronic Science and Technology*, vol. 31, no. 5, pp. 33–43, 2018.

[31] W. Wang and Z. Feng, "The shortest path planning for mobile robots using improved A∗ algorithm," *Journal of Computer Applications*, vol. 38, no. 5, pp. 1523–1526, 2018.

[32] W. Lu, J. Lei, and Y. Shao, "Path planning for mobile robot based on an improved A∗ algorithm," *Journal of Ordnance Equipment Engineering*, vol. 40, no. 4, pp. 197–201, 2019.

[33] C. Cheng, X. Hao, J. Li et al., "Global dynamic path planning based on fusion of improved A∗ algorithm and dynamic window approach," *Journal of Xi'an Jiaotong University*, vol. 51, no. 11, pp. 137–143, 2017.

[34] Y. Zhang, L.-L. Lin, and H.-C. Lin, "Development of path planning approach using improved A-star algorithm in AGV system," *Journal of Internet Technology*, vol. 20, no. 3, pp. 915–924, 2019.

[35] J. Zhao, Y. Tao, G. Yi et al., "Power control algorithm of cognitive radio based on non-cooperative game theory," *China Communications*, vol. 10, no. 11, pp. 143–154, 2013.

[36] X. Xu, X. Liu, Z. Xu et al., "Trust-oriented IoT service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2019.