An Improved A* Algorithm Based on Loop Iterative Optimization in Mobile Robot Path Planning

Gang Peng[1,2], Lu Hu[1,2(✉)], Wei Zheng[1,2], and Shan Liang Chen[1,2]

[1] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China
hulu525@foxmail.com

[2] Key Laboratory of Image Processing and Intelligent Control of Education Ministry, Wuhan, China

Abstract. In the mobile robot system, point-to-point path solving is one of the research hotspots in the field of robotics. Due to the many inflection points in the path planned by the traditional A* algorithm, the number of robot turns and the moving distance increases. Therefore, an improved A* algorithm is proposed. Based on the path of the traditional A* algorithm, a loop iterative optimization process is added. The path solved by the traditional A* algorithm is taken as the initial path of the loop iterative optimization process, from rough to fine layered iterative optimization until the total number of path nodes is minimized, and the optimal path solution is obtained. Compared with the traditional A* algorithm, the improved A* algorithm proposed in this paper effectively reduces the total number of path nodes and the number of inflection points, which can significantly improve the mobility of the robot in the actual environment. Experimental comparison results verify the feasibility and effectiveness of the proposed method.

Keywords: Loop iterative optimization · A* algorithm · Path inflection point · Path planning · Mobile robot

1 Introduction

With the development of robot technology and the continuous expansion of the application range, and the working environment of robots is becoming more and more complicated, one of the biggest technical challenges for intelligent mobile robots is efficient and smooth movement in the scene. Therefore, in the robot system, point-to-point path solving is one of the research hotspots in the field of robotics. The purpose of path solving is to find an optimal collision-free path from the starting point to the target point according to an evaluation index in an obstacle environment [1]. A lot of researches have been done on robot path solving, including sampling-based Voronoi diagram method, fast search random tree method etc. [2], node-based Dijkstra, A* algorithm, D* algorithm etc. [3, 4], based on biological heuristics Neural networks, genetic algorithms, ant colony algorithms etc. [5]. Among them, the A* algorithm is widely used in mobile robot path solving [6]. Because there are many inflection points in the path solved by the A* algorithm, there are many redundant nodes in the path. Such a path is neither optimal nor conducive to control the movement of the robot, causing the robot to move less efficiently.

Therefore, aiming at the problem of traditional A* algorithm path solving, an improved A* algorithm is proposed. Based on the traditional A* algorithm solution path, one loop iterative optimization process is added. The path solved by the traditional A* algorithm is used as the initial path of the loop iterative optimization process, and the loop iterative optimization is continuously performed until the total number of path nodes reaches the minimum, and the global optimal path is obtained.

2 Improved A* Algorithm

The A* algorithm is a classic method based on heuristic information search optimal path, which maintains a cost function and evaluates the current feasible path according to the cost function [7]. The cost function is composed of the current existing cost value and the estimated cost value. The estimated cost value, that is, the heuristic information, is the core of the whole A* algorithm [8]. The cost function has a different design depending on the application backgrounds. For the field of ground mobile robots, the 2D grid map is used to represent the robot motion space. Therefore, the A* algorithm is applied to the 2D grid map to realize the point-to-point path solution of the mobile robot. At the same time, an improved A* algorithm is proposed aiming at the deficiency of traditional A* algorithm in path solving.

2.1 A* Algorithm Principle

Using the heuristic information from the starting point of the robot to the target point, as well as selecting the appropriate cost function [9], the optimal solution is obtained by dynamically adjusting the search strategy according to the value of the cost function. Therefore, the key to the A* algorithm is to find the cost function, as in Eq. (1).

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

In Eq. (1), $g(n)$ is the cost that has been paid from the starting node *Start* to the current node n, and $h(n)$ is the cost estimation function from the current node n to the target node *Goal*. The A* algorithm starts from the starting node s and searches for the node with the smallest $f(n)$ value until the target point is searched to determine the shortest path.

The A* algorithm solution path is divided into the following steps:

(1) Create two tables: Open list and Close list, Open list records the nodes to be detected, and Close list records the nodes that have been detected;
(2) Add the starting node to the Open list;
(3) Repeat the following steps:
    (a) Find the node with the smallest $f(n)$ value in the Open list, use it as the current node, and transfer it to the Close list table;

- (b) Consider the eight neighbor nodes of the current node:
  - (b.1) If the neighbor node is unreachable or already in the Close list, ignore the neighbor node, otherwise continue with the following steps:
  - (b.2) If it is not in the Open list table, add it, calculate the $f(n)$ of the neighbor node, and then use the current node as its parent node;
  - (b.3) If it already exists in the Open list table, it is necessary to judge whether the path from the current node to the neighbor node is better, the method is to use the cost $g(n)$ to judge, if the $g(n)$ from the current node to the neighbor node is smaller than the neighbor node The original $g(n)$, then change the original parent node of the neighbor node to the current node, and recalculate its $f(n)$, and reorder the Open list table;
- (c) When the target node has been added to the Open list, it means that a path has been successfully searched; or if the target node is not found when the Open list is empty, it means that the path cannot be found.

(4) Starting from the target node, based on the direction of its parent node, back to the state of the starting node, an optimal path is generated.

In the above algorithm, the proportion of $h(n)$ in the whole cost function $f(n)$ determines the efficiency of the path solution [10]. If the $h(n)$ ratio is too small, that is, the heuristic information of the current node n to the target node g is too small, the search rate of the algorithm becomes slow, increases the solution time, but can get a better path. Conversely, the A* algorithm can have a faster search rate, and you can find a passable path if you search for fewer nodes, but it may not be the optimal path.

Therefore, considering the rate and quality of the solution, this paper uses the D-Euclidean distance to design the heuristic function $h(n)$, as shown in Eq. (2). First, a better path is solved in advance in a short time, and then optimized based on the solution path to obtain the optimal path.

$$h(n) = D * \sqrt{(n.x - Goal.x)^2 + (n.y - Goal.y)^2} \tag{2}$$

In Eq. (2), n is the current node, *Goal* is the target node, and D is the cost to move one step from the current node along the Euclidean distance, Eq. (2) represents the pre-estimated cost that needs to be paid to move from the current node n along the Euclidean distance to the target point *Goal*.

2.2 Loop Iterative Optimization Process

In the grid map, no matter how the heuristic function $h(n)$ is designed when using the traditional A* algorithm, there are always many inflection points in the path of the solution, resulting in many redundant nodes in the obtained path, as shown by the black solid line in Fig. 1. The path is not optimal and is not conducive to controlling the motion of the robot. Therefore, the path to be solved needs to be optimized to reduce the number of redundant nodes in the path and improve the efficiency of robot movement.

The black square in Fig. 1 represents the obstacle, the white square represents the passable area, the solid black line represents the unoptimized path, and n1 to n5 are the

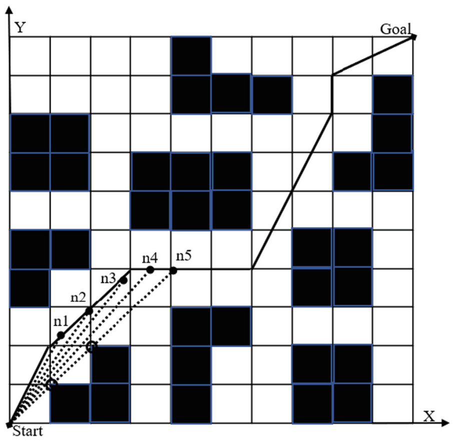


Fig. 1. Path optimization principle

nodes on the unoptimized path. Connect to subsequent nodes from the starting point of the path, such as In Fig. 1, the starting node *Start* is connected to the dotted line between n1 and n5 respectively. If there is an obstacle on the dotted line between the starting node *Start* and the current node, then the dotted line between the starting node *Start* and the previous node will be as the optimal path. Looking at Fig. 1, it can be seen that the point on the dotted line between *Start* and n5 contact with the obstacle area. As shown by the circle in Fig. 1, the line between *Start* and n5 is impassable and cannot be optimized. Therefore, the dotted line between *Start* and n4 is selected as the current optimal path.

From the above analysis, the idea of improving the A* algorithm is as follows, on the basis of the path solved by the traditional A* algorithm, a path loop iterative optimization process is added until the total number of path nodes from the starting node to the target node is minimized, and the optimal path solution is found.

It should be noted that in order to obtain the global optimal path solution, the idea of hierarchical iterative optimization is adopted, that is, the path optimized by each layer is used as the initial path of the next layer of iterative optimization, and the hierarchical iteration from coarse to fine. Until the total number of path nodes reaches the minimum or the number of loop iterations is reached, the optimal path solution is obtained. The detailed process is shown in Fig. 2.

Define a local start node Local_start, local target node Local_goal, last local target node Last_local_goal, and assign the global start node *Start* to Local_start, Local_goal, Last_local_goal.

Firstly, the path solved by the traditional A* algorithm is taken as the initial path of the optimization process, and then each path node is considered from the Local_goal, and it is judged whether there is an obstacle on the line of Local_goal to Local_start, and the optimal path is found by the method until the global The target node *Goal* is Local_goal, backtracks to the global starting node *Start*, generates an optimal path, completes an iterative optimization, calculates the total number of path nodes obtained by the current optimization, and determines whether the current iteration number

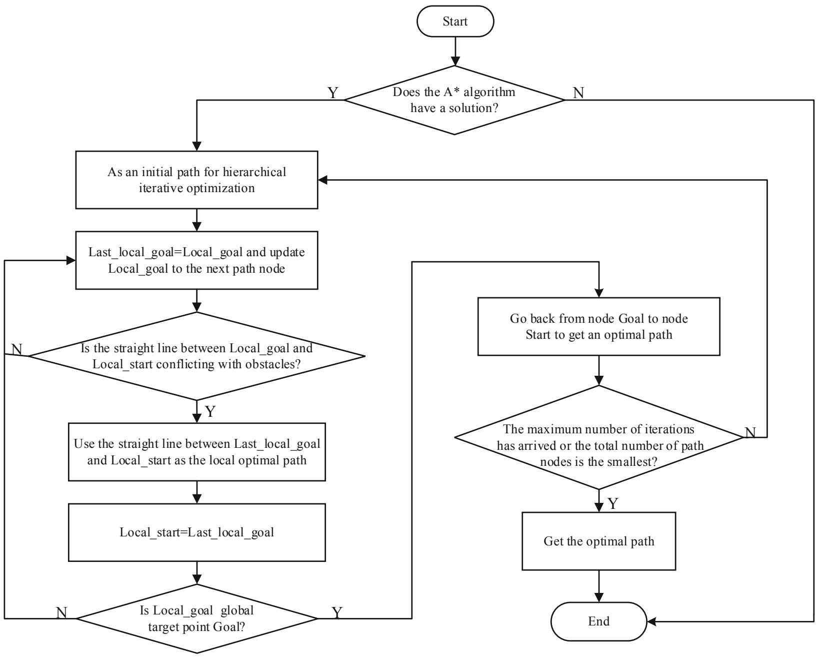

Start
Does the A* algorithm have a solution?
Y
N
As an initial path for hierarchical iterative optimization
Last_local_goal=Local_goal and update Local_goal to the next path node
Is the straight line between Local_goal and Local_start conflicting with obstacles?
N
Y
Use the straight line between Last_local_goal and Local_start as the local optimal path
Local_start=Last_local_goal
Is Local_goal global target point Goal?
N
Y
Go back from node Goal to node Start to get an optimal path
The maximum number of iterations has arrived or the total number of path nodes is the smallest?
N
Y
Get the optimal path
End

Fig. 2. Flow chart of the loop iterative optimization process

reaches the set maximum number of iterations, or current Whether the number of path nodes reaches the minimum, otherwise the path obtained this time is used as the initial path for the next iteration optimization, and the layered iteration is continued until the optimal path is obtained.

In the hierarchical iterative optimization process, it is the most time-consuming step in the whole optimization process to determine whether there are obstacles on the line between the two nodes, which determines the rate of path optimization. Considering the real-time motion of the robot, the optimal path needs to be solved quickly. Therefore, in order to accelerate the path optimization process, the Bresenham [11] line algorithm is used to quickly determine whether there are obstacles on the line of the two-pixel node. The algorithm is as follows.

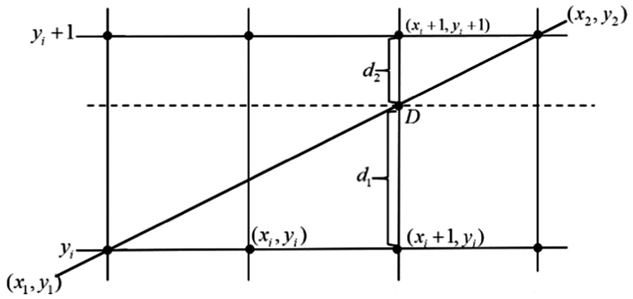


Fig. 3. Bresenham line algorithm principle

As shown in Fig. 3, the intersection of the horizontal line and the vertical line is regarded as a pixel node, and if there is an obstacle on the line between the two pixel nodes (x_1, y_1) and (x_2, y_2), it is necessary to calculate the coordinates of each pixel node between the two pixel nodes. The calculation is stopped until a pixel node whose coordinates are obstacles is encountered.

Suppose the line equation is $y = k * x + b$, and its slope k is between 0 and 1, if $x_2 > x_1$, then $\Delta x = (x_2 - x_1)$, $\Delta y = (y_2 - y_1)$. The principle of the algorithm is to find the pixel node closest to the point D on the straight line. The judgment of the distance is determined by comparing the sizes of d1 and d2. It is assumed that the i-th step has determined that the coordinates of the i-th pixel node is (x_i, y_i), then the i+1th pixel coordinate:

$$\begin{cases} d_1 \geq d_2 & (x_i + 1, y_i + 1) \\ d_1 < d_2 & (x_i + 1, y_i) \end{cases} \tag{3}$$

As can be seen from Fig. 3, $d_1 = (y - y_i)$, $d_2 = (y_i + 1 - y)$ in the formula (3). Multiply both sides of the equal sign of d_1 and d_2 by Δx, and bring the line equation $y = k * x + b$ into d_1 and d_2, as shown in Eq. (4):

$$\begin{cases} \Delta x \cdot d_1 = \Delta x \cdot (y - y_i) = \Delta y \cdot (x_i + 1) + \Delta x \cdot b - \Delta x \cdot y_i \\ \Delta x \cdot d_2 = \Delta x \cdot (y_i + 1 - y) = \Delta y \cdot (y_i + 1) - \Delta x \cdot (x_i + 1) - \Delta x \cdot b \end{cases} \tag{4}$$

Among them, the size of d_1 and d_2 can be compared by difference, so it can be derived from Eq. (4).

$$\Delta x \cdot (d_1 - d_2) = 2\Delta y \cdot x_i - 2\Delta x \cdot y_i + c \tag{5}$$

In Eq. (5), c is a constant. Since $\Delta x = (x_2 - x_1)$ and $x_2 > x_1$, therefore Δx is greater than 0, then the sign of $d_1 - d_2$ does not change. Let $p_i = \Delta x * (d_1 - d_2)$, then the final point-finding equation is:

$$\begin{cases} p_i \geq 0, & (x_i + 1, y_i + 1), & p_{i+1} = p_i + 2(\Delta y - \Delta x) \\ p_i < 0, & (x_i + 1, y_i), & p_{i+1} = p_i + 2\Delta y \end{cases} \tag{6}$$

The pixel coordinates of the next node are selected by the positive and negative of p_i, and it is judged whether the pixel node is an obstacle. If yes, the judgment is stopped. Otherwise, the value of p_{i+1} is calculated by Eq. (6), and then the coordinates of the next pixel are selected by the positive and negative of p_{i+1}, and the relationship between p_{i+1} and p_{i+2} is calculated, thereby continuously recursively calculating. Finally, it is determined whether the line between the two pixels (x_1, y_1) and (x_2, y_2) conflicts with the obstacle.

The algorithm can effectively reduce the amount of computation in the iterative optimization process and greatly accelerate the solution process of path loop iterative optimization.

3 Experimental Analysis

In order to verify the application effect of the improved A* algorithm in the actual complex environment. The original A* algorithm and the improved A* algorithm is integrated into the ROS system environment, and a comparative experiment was carried out on a mobile robot equipped with a single-line lidar. Figure 4 shows the mobile robot system used in this experiment. Figure 5 shows an experimental environment with a length of 10 m and a width of 7 m. Set the maximum linear speed of the robot to 0.5 m/s and the maximum angular velocity to 1 rad/s.

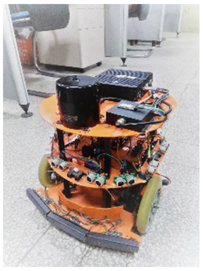


Fig. 4. Mobile robot system

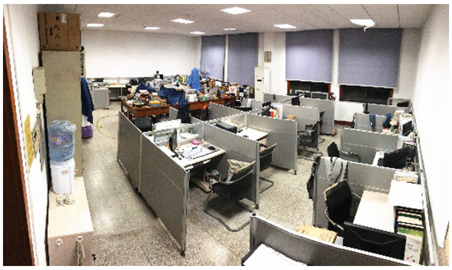


Fig. 5. Experimental environment

3.1 Improved A* Algorithm Experiment

In order to verify the path optimization performance of the improved A* algorithm, this paper pre-uses the laser mapping algorithm to build a grid map with a resolution of 0.025 m in a laboratory scene with a length of 10 m and a width of 7 m, as shown in Fig. 6. And the occupied grid map is binarized and corroded. In the figure, white is a passable area, black is an obstacle area, S is a starting node, and G is a target node.

Under the premise of the same starting node S and target node G, the traditional A* algorithm and the improved A* algorithm is used to solve the path separately. Figure 7 shows the path solution of the traditional A* algorithm, there are many inflection points in the path in the Fig. 7. As shown in the blue circle in Fig. 7, there are many redundant nodes in the path. Such a path will reduce the motion efficiency of the robot. Therefore, the path can be further optimized. Under the premise of ensuring effective obstacle avoidance, the path is used as the initial value of hierarchical iterative optimization, and the path is optimized. Figure 8a is the result of iterative optimization once. Compared with the path before optimization, the number of inflection points is significantly reduced, but the total number of path nodes does not reach the minimum at this time. Therefore, the path is used as the initial value of hierarchical iterative optimization, and the second optimization is continued, as shown in Fig. 8b, at this time, the path is optimal. The number of inflection points is the smallest and the total number of path nodes is minimized.

The experimental results show that, based on the initial value, the improved A* algorithm only needs to be iterated several times to make the path optimal. Table 1 compares the performance of the algorithm. The final optimization result is compared with the path of the traditional A* algorithm. The path length is reduced by 13.9%, and the cumulative number of inflection points is reduced by 62.5%. Using the optimized path to control robot movement can effectively reduce the number of turns and improve the robot's moving efficiency.

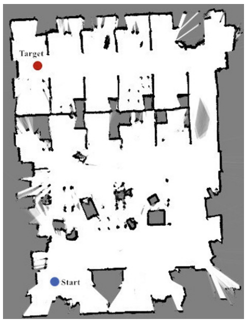


Fig. 6. Occupy the grid map

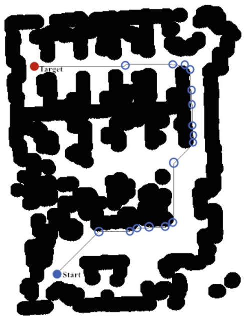
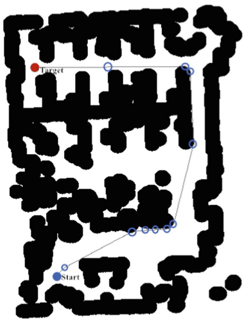
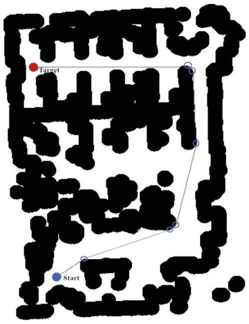


Fig. 7. A* algorithm solution results



(a) Iterative optimization 1 time



(b) Iterative optimization 2 times

Fig. 8. Improved A* algorithm solution results

Table 1. Comparison of algorithm performance

Algorithm	Traditional A* algorithm	Improved A* algorithm	
		Iterative optimization 1 time	Iterative optimization 2 times
Number of inflection points	16	10	6
Path length	17.302 m	15.515 m	14.895 m

3.2 Comparison of Original A* Algorithm and Improved A* Algorithm

In order to verify the improvement effect of the loop iterative optimization process on the original A* algorithm, a comparative experiment was conducted in a laboratory environment with dense obstacles and a narrow space. As shown in Fig. 5, a Cartographer mapping algorithm was used in advance to establish a high-precision global occupied grid map in this environment. the current position of the robot taken as the global starting point, the global target point is randomly selected in the blank area occupying the grid map, as shown by the blue and red dots in Figs. 9 and 10. On the map, the point-to-point path solving experiment is performed using the original A* algorithm and the improved A* algorithm, respectively, and the robot is controlled to move along the pre-planned path at the same linear velocity and angular velocity. Figure 9 shows the path solved by the original A* algorithm, among which the blue circle is the path inflection point. Figure 10 is the path after the improved A* algorithm is optimized twice in the loop iteration.

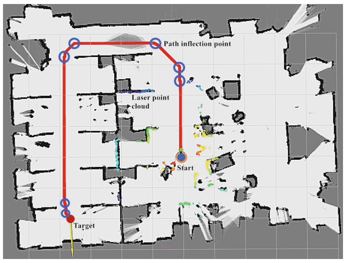


Fig. 9. Original A* algorithm results (Color figure online)

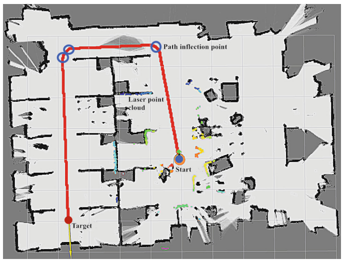


Fig. 10. Improved A* algorithm results (Color figure online)

In order to effectively evaluate the performance of the improved A* algorithm, multiple sets of different global starting points and target points were used to perform multiple sets of comparison experiments using the original A* algorithm and the improved A* algorithm. Among them, the total length of the path solved by the two algorithms, the total time of the robot moving to the target point, the number of turns along the path movement are used as performance evaluation indicators, and the robot is controlled to complete the motion experiment at the same linear velocity and angular velocity, the experimental results are shown in Table 2.

It can be seen from the analysis in Table 2 that, compared with the original A* algorithm, no matter where the global starting point and the target point are set, the total length of the solved path is the shortest, and the number of path turns is the least, the time to move to the global target point is minimal. Therefore, using the improved A* algorithm to solve the path can effectively reduce the number of turns, shorten the movement time, and improve the robot movement efficiency.

Table 2. Multi-group path solving comparison experiment

Number of experiments		1	2	3	4	5
Start(X, Y)m		1.17, 0.34	1.43, 0.06	0.08, −2.97	2.72, 3.82	−0.28, −2.86
Target(X, Y)m		0.05, −2.88	0.08, −2.79	2.72, 3.82	1.50, 0.19	−0.48, 3.78
Original A* algorithm	Path length	9.22 m	8.99 m	12.25 m	8.23 m	15.78 m
	Number of turns	8 Times	9 Times	7 Times	6 Times	8 Times
	Time of movement	61.1 s	62.3 s	69.25 s	53.32 s	80.21 s
Improved A* algorithm	Path length	8.42 m	8.33 m	11.42 m	7.85 m	14.52 m
	Number of turns	4 Times	3 Times	3 Times	3 Times	3 Times
	Time of movement	49.39 s	47.26 s	59.03 s	45.41 s	66.21 s
	Iterative times	3 Times	2 Times	2 Times	2 Times	3 Times
Path length reduction (%)		8.67%	7.34%	6.77%	4.61%	7.98%
Reduced number of turns (%)		50%	66.6%	57.1%	50%	62.5%

3.3 Experimental Comparison of Obstacle Avoidance Path Planning Methods

In order to verify the actual effect of the improved A* algorithm in obstacle avoidance path planning, in the same experimental environment, the same obstacle environment, the same linear velocity and angular velocity, the improved A* algorithm and the artificial potential field method were compared in multiple groups.

Obstacle Avoidance Path Planning Experiment with Improved A* Algorithm
In the experimental environment shown in Fig. 5, the improved A* algorithm is used to perform the obstacle avoidance path planning experiment for the special obstacle of the human body. The experimental verification process is shown in Fig. 11.

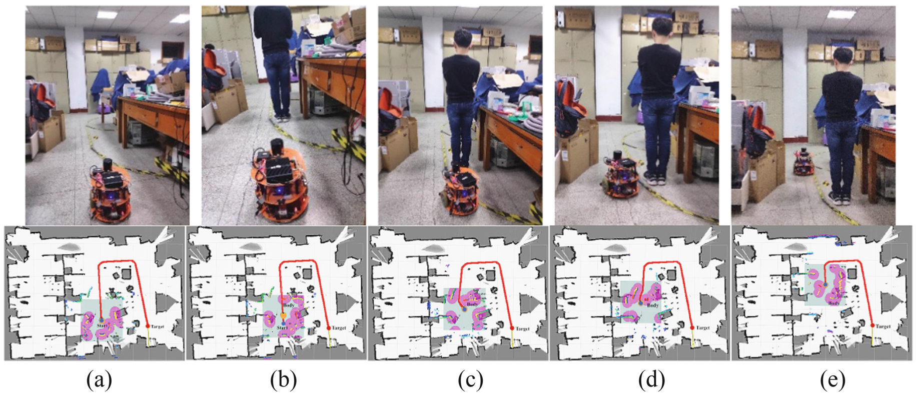


Fig. 11. Robot avoids the human body

The obstacle avoidance process in Fig. 11 shows that, after the robot detects the obstacle, in order to bypass the obstacle, an improved A* algorithm is used to plan an optimal path from the current position to the global target point. The robot is controlled to move along the path to avoid obstacles. At the same time, in the obstacle avoidance process, the obstacle avoidance method can adjust the obstacle avoidance path in real time according to the safety distance between the robot and the obstacle. Therefore, the improved A* algorithm is used for obstacle avoidance path planning, which having the advantages of short moving distance and stable motion, which can effectively shorten the motion time of the robot and improve the motion efficiency.

Multiple Sets of Comparison Experiments

In order to effectively compare the obstacle avoidance path planning performance of the improved A* algorithm, five different global starting points and target points, five different obstacle environments are shown in Fig. 12. A comparative experiment was conducted on two obstacle avoidance path planning methods. Among them, the minimum safe distance between the robot and the obstacle, the time to complete the obstacle avoidance, the moving distance of the robot to bypass the obstacle is the performance evaluation index, and the control robot completes the motion experiment under the same expected linear velocity and angular velocity. The results are shown in Table 3.

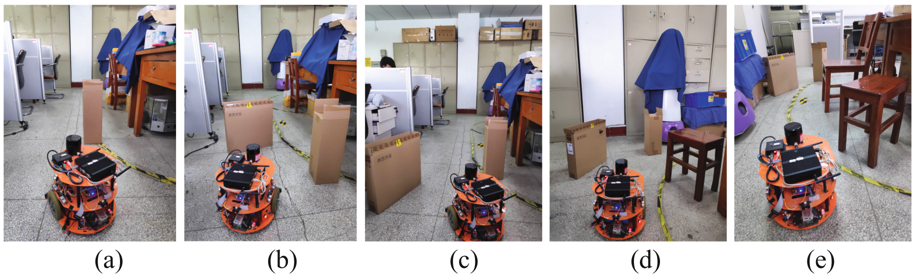


Fig. 12. Five obstacle environments

Table 3. Multiple sets of contrast experiments for two obstacle avoidance path planning methods

Number of experiments		1	2	3	4	5	Average
Artificial potential field algorithm	Minimum safe distance	28 cm	30 cm	26 cm	27 cm	20 cm	26.2 cm
	Distance to bypass obstacles	2.95 m	3.39 m	3.78 m	3.99 m	4.41 m	3.704 m
	Complete an obstacle avoidance time	25.32 s	28.13 s	32.5 s	34.1 s	42.3 s	32.47 s
Improved A* algorithm	Minimum safe distance	20 cm	14 cm	10 cm	13 cm	8 cm	13 cm
	Distance to bypass obstacles	1.62 m	1.56 m	2.56 m	2.32 m	2.96 m	2.204 m
	Complete an obstacle avoidance time	9.93 s	8.76 s	15.2 s	13.4 s	17.2 s	12.898 s
Avoid obstacle distance reduction %		45.08%	53.98%	32.27%	41.85%	32.87%	4.49%
Reduced timed for obstacle reduction %		60.7%	68.85%	53.23%	60.70%	59.33%	60.27%

It can be seen from Table 3 that in the five experiments, the improved A* algorithm compared with the artificial potential field method. In order to avoid obstacles, the distance of movement is reduced by an average of 40.49%, and the time to complete an obstacle avoidance is reduced by an average of 60.27%. Therefore, the obstacle avoidance method of this paper can effectively shorten the obstacle avoidance time and reduce the moving distance. However, in Table 3, the minimum safe distance performance of the obstacle avoidance method of this paper is not as good as the artificial potential field method. The reason is that the closer the artificial potential field method is to the obstacle, the greater the repulsive force generated, which makes the safe distance of the robot from the obstacle larger. The obstacle avoidance method in this paper is based on the principle of solving the optimal path to avoid obstacles, so that the safety distance between the robot and the obstacle is small. At the same time, due to the error of the localization accuracy of the robot, the estimated distance between the robot and the obstacle is deviated from the actual distance.

4 Conclusion

Aiming at the problem of the original A* algorithm having many inflection points in the field of robot path planning which leads to many redundant nodes in the path, this paper proposes an improved A* algorithm. That is, adding a loop iterative optimization process based on the original A* algorithm, using the path solution solved by the original A* algorithm as the initial value of the loop iterative optimization process, and minimizing the total number of path nodes through continuous loop iteration, so as to obtain the optimal path. The experimental results show that the improved A* algorithm only needs a simple iteration several times to optimize the path, which makes the algorithm feasible in engineering applications. At the same time, the improved A* algorithm is applied to the obstacle-intensive and complex experimental environment for multi-group comparison experiments. The results show that the improved A* algorithm proposed in this paper can effectively reduce the number of path inflection points and shorten the path length. The moving efficiency of the mobile robot in the actual environment is improved. The experimental comparison results fully verify the feasibility and effectiveness of the improved A* algorithm proposed in this paper.

When the mobile robot turns around the obstacle, due to the localization error and the noise of the laser sensor, there is a certain error between the estimated distance between the robot and the obstacle and the actual distance. Therefore, the optimal obstacle avoidance path solved by the improved A* algorithm is close to the obstacle, which causes the risk of the robot to increase along the obstacle avoidance path. This problem can be solved by increasing the range of obstacles in the grid map, or by improving the accuracy of the lidar and localization.

References

1. Eele, A.J., Richards, A.: Path-planning with avoidance using nonlinear branch-and-bound optimization. J. Guid. Control Dyn. **32**(2), 384–394 (2015)
2. Kothari, M., Postlethwaite, I.: A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. J. Intell. Rob. Syst. **71**(2), 231–253 (2013)
3. Zhang, B., Cao, Q.X., Wang, W.S.: Path planning of mobile robot using 3D grid map. J. Xi'an Jiaotong Univ. **47**(10), 57–61 (2013)
4. Montiel, O., Sepúlveda, R., Orozco-Rosas, U.: Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field. J. Intell. Robot. Syst. **79**(2), 237–257 (2015)
5. Zhu, D.Q., Sun, B., Li, L.I.: Algorithm for AUV's 3-D path planning and safe obstacle avoidance based on biological inspired model. Control Decis. **30**(5), 798–806 (2015)
6. Wei, W., Dong, P., Zhang, F.: The shortest path planning for mobile robots using improved A* algorithm. J. Comput. Appl. (2018)
7. Fu, B., Chen, L., Zhou, Y., et al.: An improved A* algorithm for the industrial robot path planning with high success rate and short length. Robot. Auton. Syst. **106**, 26–37 (2018)
8. Jing, X., Yang, X.: Application and improvement of heuristic function in A ~ * algorithm. In: The 37th China Control Conference (2018)
9. Wang, Y., Liu, Z., Zuo, Z., et al.: Local path planning of autonomous vehicles based on A ~ * algorithm with equal-step sampling. In: The 37th China Control Conference (2018)
10. Wei, W., Dong, P., Zhang, F.: The shortest path planning for mobile robots using improved A ~ * algorithm. J. Comput. Appl. (2018)
11. Li, X., Shao, X.: Fast line drawing algorithm by circular subtraction based on Bresenham. In: Proceedings of SPIE - The International Society for Optical Engineering, vol. 8349, p. 20 (2012)