Path Planning Research Based on An Improved A* Algorithm for Mobile Robot

Wang Xin*, Lu Wanlin, Feng Chao, Hao Likai
Troop No.66133rd of PLA, Beijing 100144, China
*Corresponding author’s e-mail: 786164744@qq.com

Abstract. With the development of technology, the path planning technology of mobile robots has received extensive attention. The principle and process of A* algorithm are outlined. Because there are many break points in the A* algorithm, the path is not smooth and there are too many problems in searching for grid points. So we need improve the A* algorithm. Firstly, in order to reduce the number of search grid points, a threshold value N is added to the openlist. If the number of search times is greater than N and the first node inserted is not expanded, the node is set to the highest priority. Secondly, we use the path planning based on the combination of Floyd algorithm and A* algorithm to remove redundant nodes and reduce the sharpness of the breakpoints. Finally, the geometric optimization process results in a smooth path. Simulations verify that the improved A* algorithm reduces the path length and the path is smoother, more suitable for robot navigation.

1. Introduction
Path planning technology is a hot topic in the research of mobile robots, which lays a foundation for the control of autonomous driving of robots[1]. The definition of path planning can be expressed as: after the starting point and target point are set, search for an optimal route that can avoid obstacles from the starting point to the target point according to the planning algorithm. According to the environmental conditions of the robot, the motion path is divided into two categories: One is the global path planning based on the known map information, and the information in such path planning environment includes all the known obstacles and drivable areas; The other is local path planning that can obtain real-time information based on the camera, inertial navigation, odometer, etc., and the local path planning is in the environment where the obstacle information is unknown[2]. Common global path planning algorithms are[3]: A* algorithm, Dijkstra algorithm, particle swarm optimization, genetic algorithm, etc. The A* algorithm is a heuristic algorithm first proposed by Nilsson. Continuously searching for target point to obtain the obstacle avoidance path of the robot is the core of A* algorithm. By evaluating the search points in the state space, the optimal node is obtained, and then the search continues according to the location node until the target point is found. This algorithm has the characteristics of relatively simple programming method, less parameter setting and high efficiency of searching path[4].

2. A* algorithm
A* algorithm is an effective search method to obtain the shortest path, and it is also a typical heuristic algorithm[5]. A* algorithm is generally used for static global planning, and its core expression is:

\[ f(n) = g(n) + h(n) \]  

(1)
Where: \(f(n)\) represents the path length from the starting node through the current node to the target node; \(g(n)\) represents the path length from the starting node to the current node \(n\), and \(g(n)\) is the shortest distance from the starting node to the current node \(n\); \(h(n)\) is the estimated cost function distance from the state node \(n\) to the target state. Since \(g(n)\) is the actual distance, in order to search for the optimal solution in the static map, the selection of cost type \(h(n)\) is very important, and the value of \(f(n)\) depends on the size of \(h(n)\). The green grid represents starting point position, the red grid represents target point position, and the eight points around the blue grid represent the candidate points of the child nodes that can be set as the current state space. Let one side length of the grid be 10, and formula (1) be abbreviated as \(F=G+H\), where \(G\) represents the Manhattan distance from the starting point to the current state, and \(H\) is the estimated cost function value from the current state node to the target node using Euclidean distance. The \(F\) value of eight squares around starting point are shown in figure. The figure shows that No.2 square has the smallest \(F\) value. Use this as the next point to deal with, and then calculate \(G\) and \(H\) value of squares around No.2 square, black square is an obstacle, ignore the impassability when calculating the values of \(G\) and \(H\), determine the optional square of No.2 is No.1 and No.3. Since No.1 and No.3 are the 8 adjacent nodes of the parent node (the starting point), and there is no optional new adjacent node around No.3 node, the node to be processed from the starting point is No.1 instead of No.2 and No.3. According to this idea, node 4 is a child of node 1.

![Figure 1. A* algorithm searches for nodes](image)

In the constructed map, let the coordinate of starting node be \(S(S_x, S_y)\), the current node \(C(C_x, C_y)\), the target node \(T(T_x, T_y)\). Simple evaluation function \(f(n)\) value in Euclidean distance can be expressed as:

\[
f(n) = \left( (C_x - S_x)^2 + (C_y - S_y)^2 \right)^{1/2} + \left( (T_x - C_x)^2 + (T_y - C_y)^2 \right)^{1/2}
\]

\[\text{(2)}\]

3. A* algorithm flow
The algorithm flow of A* will be described in form of pseudocode. First, set the parameters. start: The starting point of robot path planning. goal: The target point of robot path planning. \(g(n)\): The actual movement distance of \(n\) node to start point. \(h(n)\): The theoretical travel distance of \(n\) node to the target point. openlist: a list of undetected nodes while searching for nodes; closelist: nodes that have been detected; comaFrom: save the child and parent nodes, the resulting path is saved in this list. The pseudocode:

Add the starting point to openlist

\[
\begin{align*}
\text{do} & \ \\
& \quad \text{Search for the node in the openlist with the smallest } f(n) \text{ value, it represents the current node} \ \\
& \quad \text{Add to closelist} \ \\
& \quad \text{For each node in All candidate nodes adjacent to the current node} \ \\
& \quad \quad \text{if(Node is the obstacle point or node is in closelist)} \ \\
& \quad \quad \quad \text{Continue to search for the next adjacent candidate node.} \ \\
& \quad \quad \text{if( This node does not exist in the openlist)} \ \\
& \quad \quad \quad \text{Add this candidate node to the openlist, Update parent node, calculate its } f(n) \text{, } g(n) \text{, } h(n) \text{ value} \ \\
& \quad \quad \quad \text{if(Candidate nodes are already in the openlist)} \ \\
& \quad \quad \quad \quad \text{if(using } g(n) \text{ value as criteria, detect the new path, Determine if there is a lower } g(n) \text{)}
\end{align*}
\]

2
{ Replace its parent node with the current node, calculate the \( f(n)h(n) \) value of current node }

} save the parent node with comaFrom

}while(target node is not in openlist)

if openlist is empty, there is no path to search.

At last connect all parent node from start point, this path is result of A* algorithm.

The path planning program structure of traditional A* algorithm is shown in figure 2.

![Figure 2. Traditional A* algorithm path planning block diagram.](image)

4. A* algorithm improvement

Although the traditional A* algorithm has become a widely used path planning algorithm in the navigation of mobile robots, the path planned by this algorithm has many problems such as multiple inflection points and turning times, which is not conducive to the running of the robot in the actual environment[6]. Therefore, based on the study of A* algorithm, An A* algorithm improvement is proposed which is suitable for robot driving.

In order to solve the problem that A* algorithm cannot find the shortest path in the actual path planning, Anthony Stentz proposed the D* method[7] based on interpolation method to make the path
smooth. But the interpolation method increased the calculation amount and made the real-time performance worse. Intelligent algorithms such as genetic algorithm and neural network algorithm are also used for path planning, but for static map, intelligent algorithm takes more time. In order to optimize the planning path of A* algorithm and reduce the length of the path, three aspects of A* algorithm are improved:

4.1. Set the openlist access time threshold
In the traditional A* algorithm, when the current node expands the child nodes, all candidate nodes will be extended every time. This method may have some invalid searches in complex environments, which cost some time. On the basis of traditional A* algorithm, improvement is proposed to avoid getting into the local minimum and solve the problem of not being able to plan the global path. This improvement can be described as that every time the first inserted node in the openlist table is detected to determine whether it has been extended after passing a certain time threshold of N, and if not, this node will be regarded as the highest priority extension.

4.2. Floyd algorithm optimization
Floyd algorithm refers to the idea of dynamic programming to search for the shortest path, which is used to solve the optimal distance problem between two points [8]. The combination of Floyd algorithm and A* algorithm is used to reduce the path length, which meets the actual application requirements. The principle of Floyd algorithm is shown in figure 3.

\[
L(A,D) \text{is the distance between A and D, in figure 3, there is a obstacle between A and D. Let } L(A,D) = +\infty, R(A, D) = A \rightarrow D. \\
\text{Point B is the planned node between point A and D:} \\
\text{If} \\
L(A,B) + L(B,D) < L(A,D) \\
\text{Then} \\
L(A,D) = L(A,B) + L(B,D) \\
R(A,D) = A \rightarrow B \rightarrow D
\]

\[
\text{Insert point C between B and D:} \\
\text{If} \\
L(A,C) + L(C,D) < L(A,B) + L(B,D) \\
\text{Then} \\
L(A,D) = L(A,C) + L(C,D) \\
R(A,D) = A \rightarrow C \rightarrow D
\]
Remove point C. The optimized path from A to D is represented as an optimized arc path \( R(A,D) \). By adopting Floyd algorithm to optimize the path planned by A* algorithm, redundant points can be removed, which is more suitable for mobile robot driving.

4.3. Path smoothing

There are many sharp inflection points in the path planned by A* algorithm, which do not conform to the smooth movement of the mobile robot. In order to make the mobile robot move smoothly on the planned path, the planned path needs to be smoothed[9]. Use smooth curves to replace sharp points. The principle is shown in figure 4.

\[
\begin{align*}
A(x_1, y_1), & B(x_2, y_2), & C(x_3, y_3) \text{ is known, do the bisector of Angle } \angle ABC \text{ through point B}, \\
\text{The lines } AB \text{ and } BC \text{ are denoted as } & \vec{AB} \text{ and } \vec{BC} \text{ respectively, P1 and P2 are the intersections of arcs and lines, The center of the arc is } r, Q \text{ is the Angle between the two lines.} \\
\text{The linear equation of } AB: & y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x - \frac{x_1 y_2 - y_1 x_2}{x_2 - x_1} \\
\text{The linear equation of } BC: & y = \frac{y_3 - y_2}{x_3 - x_2} \cdot x - \frac{x_2 y_3 - y_2 x_3}{x_3 - x_2} \\
\text{The figure shows:} & \delta = \arccos \left( \frac{\vec{AB} \cdot \vec{BC}}{|AB| \cdot |BC|} \right) \\
\text{Let arc radius:} & |p_1 r| = |p_2 r| = R \\
\text{By equation:} & |p_1 B| = |p_2 B| = R \cdot \cot \left( \frac{\delta}{2} \right) \\
\text{Let} & \epsilon_1 = \frac{|p_1 B|}{|AB|}, \epsilon_2 = \frac{|p_2 B|}{|BC|} \\
\text{The coordinates of } p_1 \text{ and } p_2 \text{ can be obtained from the coordinates of A and B:} & p_1 \cdot x = \epsilon_1 \cdot x_1 + (1 - \epsilon_1) \cdot x_2 \\
& p_1 \cdot y = \epsilon_1 \cdot y_1 + (1 - \epsilon_1) \cdot y_2 \\
& p_2 \cdot x = (1 - \epsilon_2) \cdot x_2 + \epsilon_2 \cdot x_3 \\
& p_2 \cdot y = (1 - \epsilon_2) \cdot y_2 + \epsilon_2 \cdot y_3
\end{align*}
\]
Make perpendicular lines of line AB and line BC respectively through $p_1$ and $p_2$, and their intersection point is. Then the arc equation is:

$$\left( x - ry \right)^2 + \left( y - ry \right)^2 = R^2$$  \hspace{1cm} (16)

So the equation for the trajectory $A \rightarrow P1 \rightarrow P2 \rightarrow C$ is:

$$y_i = \begin{cases} 
\frac{y_2 - y_1}{x_2 - x_1}x - \frac{x_1y_2 - y_1x_2}{x_2 - x_1}, & x \leq p_1 \\
\sqrt{R^2 - \left( y - ry \right)^2} + ry, & p_1 \leq x \leq p_2 \\
\frac{y_3 - y_2}{x_3 - x_2}x - \frac{x_2y_3 - y_2x_3}{x_3 - x_2}, & x \geq p_2
\end{cases}$$  \hspace{1cm} (17)

Through the above methods, Floyd algorithm can be adopted to remove the redundant points at the inflection point and the folding point to shorten the robot's driving path, and then the smoothing algorithm can be used to process the path, so as to reduce the degree of bending at the inflection point, which is conducive to the robot's steering and driving and obstacle avoidance operation[10]. A* program block diagram after algorithm improvement is shown in figure 5.

![Figure 5.Block diagram of improved A* algorithm path planning](image-url)
5. Simulation experiment and analysis

To verify the effect of the improved A* algorithm, MATLAB2016b was used for simulation programming. A* algorithm is designed using the GUI in MATLAB graphical interface, as shown in figure 6, inside the guide set two buttons, one named A* algorithm for path planning, another name for generating obstacles, they will be automatically generated .m file. Use GUI editable text to set starting point coordinates and target point coordinates, import related code in the .m file of A* algorithm for path planning and obstacle button generated. modify the parameters of the corresponding Function, to generate the figure 7 GUI interface, click the generate obstacles button, there will be a 32 x 32 map with some obstacles around. A grid unit on a map has a length of 1

Figure 6. GUI design of simulation interface.

Figure 7. A* algorithm verification interface design.

The simulation environment map was constructed, and obstacles were randomly set in the map. At the same time, the starting point was randomly set at (3,3) and the target point was randomly set at (29,22) in the GUI editing text. The simulation function is to find an optimal path from the starting point (3,3) to (29,22). In experiment 1, the traditional A* algorithm was used for path planning, and the simulation results were shown in figure 8 (a). Experiment 2 only sets the openlist access time threshold for path planning, and the path simulation results are shown in figure 8 (b). In experiment 3, Floyd and smoothing algorithm are added on the basis of experiment 2. See figure 8 (c) for path simulation results.

Figure 8(a). Traditional a * algorithm path planning.

Figure 8(b).Algorithm path planning A* after only adding threshold values.

Figure 8(c).Improved A* algorithm path planning.

Figure 8 comparison of simulation path optimization effect before and after.

The analysis and comparison of experimental results show that the improved algorithm shortens the total length of the path, reduces the number of inflection points and fold points, improves the smoothness of the path, and is more suitable for the motion of mobile robots. The statistical experimental data are shown in table 1: the improved planning time is slightly increased, which is due to the addition of Floyd algorithm and smoothing algorithm. However, the planning time difference is not too large, which is within the acceptable range. This is due to the improvement of the algorithm, adding a threshold value N in the openlist to improve the efficiency of finding child nodes.
### Table 1 comparison of simulation experiment data

| Types of algorithms      | Planning time (s) | Shortest path length |
|--------------------------|-------------------|----------------------|
| Traditional A* algorithm | 24.6952           | 150.7342             |
| Improved A* algorithm    | 25.2152           | 145.3538             |

### 6. Conclusion

On the basis of A* algorithm research in static environment, a method is designed to improve the real-time performance of the path planning and reduce the search node number. In this method, openlist in the A* algorithm is proposed to join judgment threshold N, if the search node is more than N, judge whether the node of the first to join extension, not extend it as the highest extension node. Then, Floyd algorithm is used to optimize the traditional A* algorithm for the problems of multiple inflection points and high degree of path bending in the search path, and then the path smoothing process is carried out: the arc curve is used to replace the broken point to make it suitable for robot movement. The MATLAB simulation results of the improved A* algorithm and the traditional A* algorithm show that there is little difference in the planning time between them, but the improved path is smoother and shorter, which is more conducive to the running of the robot.

### References

[1] Wang, T.R., (2017)The development of robotics. Robot, 39(4): 385-386.(in Chinese).
[2] Zhu, D.Q., Yan, M.Z.,(2010)Overview of path planning technology for mobile robots. Control and Decision,25(7):961-967.(in Chinese).
[3] Zhang,G.L.,Hu, X.M., Chai, J.F.,(2011)Summary of Path Planning Algorithm and its Application . Modern Machinery, (5): 85-90.(in Chinese).
[4] Liu, S.,(2016)Research on route planning algorithm of autonomous driving resource exploration vehicle. Changchun: Jilin university.(in Chinese).
[5] Song, B.B., Jin, H.Q., Li, Q.C. (2016)Application of Improved A* Algorithm in Penetrating Path Planing Journal of Ordnance Equipment Engineering, (7):85-89.(in Chinese).
[6] Wang, Z.Z.,(2018)Automatic Robot Path Planning Under Complicit Dynamic Environment. Modular Machine Tool & Automatic Manufacturing Technique, (1).(in Chinese).
[7] Shi, J, Liu, C, Xi, H. Improved D* path planning algorithm based on CA model. Journal of Electronic Measurement & Instrumentation, 2016.
[8] Zhao, L.F., Liang, J., (2014)Improved Floyd Algorithm for Shortest Path Problem. Computer Technology and Development, (8) : 31-34.
[9] Wang, H.W., Ma, Y., Xie, Y.,(2010)Mobile Robot Optimal Path Planning Based on Smoothing A* Algorithm. Journal of TongjiUniversity(Natural Science), 38(11): 1647-1650.(in Chinese).
[10] Deng, Q., Xue, Q., Chen, L., (2018)Application of a Hybrid Path Planning Method on Armored Vehicles CGF. Journal of Ordnance Equipment Engineering,39(7):120-122,150.(in Chinese).