Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

Research on Path Planning of Mobile Robot based on A* Algorithm

Cui Zhi
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin , P. R.China

Shi Xiumin
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin , P. R.China

***Abstract*—Path planning is one of the core issues in the field of intelligent robots, and it is also an important aspect in the study of artificial intelligence in robotics. The typical path planning problem refers to how to find the moving path from the starting point to the ending point for the robot in the working environment with obstacles, so that the robot can pass all obstacles safely and without collision during the movement. Firstly, the paper expounds the concept of mobile robot path planning and introduces the current research situation of mobile path planning. Then it introduces the steps of A* algorithm and A* algorithm implementation, and carries out MATLAB simulation experiment. The experiment proves that the A* algorithm can plan the global optimal path and ensure the fastness and accuracy of the path. It has certain reference significance for the future research of path planning.**

Keywords—Mobile robot; path planning; A* algorithm

INTRODUCTION

In recent years, people attach great importance to artificial intelligence technology, and the field of mobile robots has gradually become a research hotspot [1]. Intelligent mobile robots are one of the most active areas in the development of artificial intelligence. Path planning technology is one of the key technologies for mobile robots to achieve autonomous positioning and path navigation [2], and is an important research direction in the field of mobile robots. The path planning of the mobile robot mainly realizes three tasks: one is to plan the feasible path of the robot from the starting point to the target point; the other is to plan the feasible path of the robot from the starting point to the target point. The other is to make the path of the robot run around obstacles in space. The third is to optimize the robot's motion path to achieve shorter, smoother requirements [3].

A* algorithm, A* algorithm is a heuristic search algorithm in artificial intelligence [4]. It can omit a large number of unnecessary search paths and improve search efficiency. It is the most effective way to solve the shortest path by static path planning. The algorithm can make the robot seek a robot motion path trajectory from the starting point position to the target point position without any obstacles in the environment, which is a better effective path.

I. A* ALGORITHM PRINCIPLE

The A* algorithm is a very effective path optimization algorithm. It is faster than the Djkstas algorithm. It first appeared in 1968. The overall framework of the algorithm is a graph traversal search algorithm, but it is different from most graph search algorithms. A heuristic function is used to estimate how close any point on the map is to the target point. Through this heuristic, you can coordinate the selection of the best direction search. If the process fails, you can also choose another path search until you find the most optimized path. The A* algorithm is a typical heuristic search algorithm in artificial intelligence, and the inspirational valuation is expressed by the valuation function:

$$f(n) = g(n) + h(n)$$

Where $f(n)$ is the valuation function of node n; $g(n)$ is the actual cost from the initial node to the n-node in the state space; $h(n)$ is the estimated cost of the best path from the node to the target node.

The correct selection of the valuation function will directly affect the success of the A* algorithm, and the determination of the function is closely related to the actual situation. Therefore, it is very important to choose the valuation function. An inappropriate heuristic function will slow down the A* algorithm and cause it to produce a bad path. The closer the valuation value is to the actual value, the better the valuation function is achieved. For geometric path planning, the Manhattan distance between the two nodes can be taken as the valuation value, ie:

$$h(n) = \sqrt{(d_x - s_x)^2 + (d_y - s_y)^2} \qquad (2)$$

In this case, the valuation function will be more or less restricted by the valuation value. The node is close to the target point, the h value is small, and the f value is relatively small, which ensures that the shortest search is performed in the direction of the end point. If A* is required to give a "perfect" path, then the value of the heuristic function should be a low estimate (less than the actual cost) of the actual cost from one point to the end, ie, $h^*(n) < h(n)$ is used instead of $h(n)$.

The A* algorithm implementation relies on two lists: open list and close list. The open list stores all the nodes that have been generated but not accessed. These nodes are allowed access nodes from their parent nodes. They are sorted in the open list according to the valuation value from small to large.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

The first node is the lowest valuation value. Node. Only the first node in the queue is the node that will be accessed; the closed list records the nodes that have been visited. Starting from a point to test the neighboring nodes, if you can move and the current move is the starting point to the historical best method of that node, then the node is inserted into the open list according to the valuation value from small to large, and the trial ends in several directions. After taking the node with the lowest valuation value, put it into the closed list, and then try to test several adjacent nodes from the point with the lowest valuation value, and also open the list [120, 121].

II. A* ALGORITHM IMPLEMENTATION STEPS

Suppose the starting point is dark green square A and the end point is red square B. The A* algorithm is initialized as shown in Figure 1, and the middle blue is the obstacle.

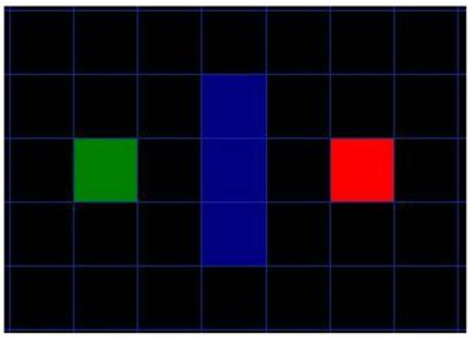

Fig.1 A* algorithm initialization

Implementation steps of the A* algorithm:

1) Start at point A and store it as a pending point in an "open list".

It may or may not be, this is a list of squares to be checked. List. Although there is only one element in the list, it will be added later. "Before you open it and treat it as a special point in the "open". The list is like a square that the path of the week may pass through.

2) Find all accessible or passable squares around the starting point, skip obstacles or other unreachable squares, and add them to the list. Save point A as the "parent square" for all of these squares.

3) Remove point A from the open list and add it to a "close list" that holds all the squares that do not need to be checked again.

As shown in Figure 2, the dark green square is the starting point, and its outer frame is bright blue, indicating that the square is added to the close list. The black squares adjacent to it need to be checked, and their frames are bright green. Each black square has a gray pointer to their parent node.

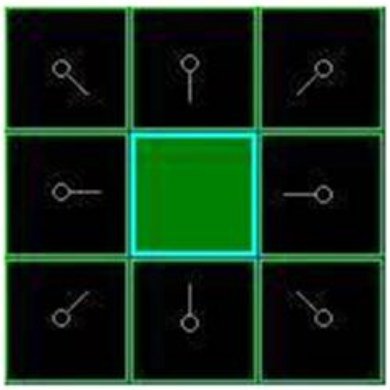

Fig.2 Start selection chart

The next step is to open the adjacent squares in the list and roughly repeat the previous process. The one with the lowest f value is selected. Which square in the selection path is determined by equation (1):

The path selected by A* is generated by iterating through the open list and selecting the square with the lowest f value. g represents the cost of movement along the path from the starting point to the current point, so that the horizontal and vertical movements make the horizontal and vertical movements cost 10, and since the diagonal distance is $\sqrt{2}$ times the horizontal or vertical distance, in order to simplify the calculation and improve the search speed, The cost of moving diagonally is 14. Calculate the g value along a specific path to a square. The method of evaluation is to take the g value of its parent node, and then increase it by 14 or 10 according to whether it is diagonal or right angle to the parent node. H is calculated by the Manhattan method, which calculates the sum of the horizontal and vertical squares from the current cell to the destination cell, and then multiplies the result by 10, where the obstacles in the path are ignored, and the values of f are g and h. with. The result of the first step is shown in Figure 4-3. Each square is marked with the values of F, G, and H, where the upper left corner is F, the lower left corner is G, and the lower right corner is H.

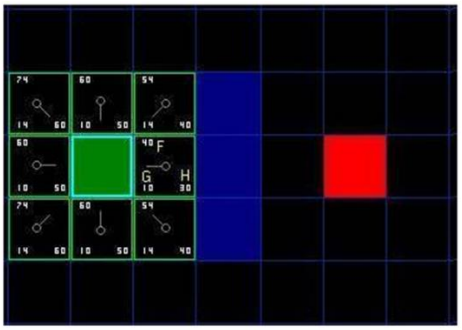

Fig.3 Numerical example

In order to continue the search, select the square with the lowest f value from the open list, and then treat the selected square as follows:

1) Remove it from the open list and add it to the close list.

2) Check all adjacent grids, skip the grids that are already in the closed list or the grids that are not passable, add it to the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

open list, and if they are not already inside, use the selected square as the new square. Parent node.

3) If a neighboring cell is already in the open list, check if the current path is better.

In the first 9 squares, after the start point is switched to the close list, 8 left in the open list. Among the 8 squares, the lowest value of f is the grid immediately to the right of the starting grid, so select this grid as the next grid to be processed, and select the best as shown in Figure 4-3.

Take this grid out of the open list, put it in the close list, and then check the adjacent grid around it. The three grids on the right are walls, so skip it, the grid on the left is in the close list, skip it, the other 4 The grid is already in the open table, so it is determined by checking the value of g, if the path is better through this grid. After checking the four grids, it was found that none of the paths can be improved by using the current grid, so no changes are made.

Continue to search the open list, now there are only 7 grids, still choose the lowest value of f, but now there is a minimum value of the grid value, select the last one to open the grid in the list can increase the search speed, in fact, which one to choose All the same, if the first choice is wrong, you will definitely choose another one. Here you select the grid at the bottom right of the starting grid, and select f as shown in Figure 4.

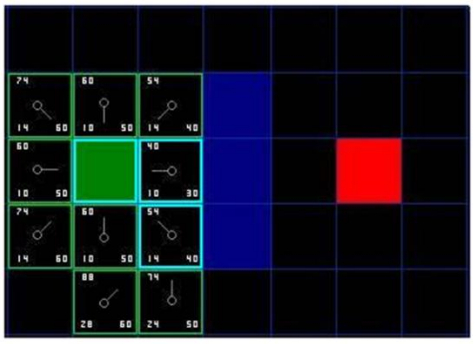

Fig.4 Choose f minimum

Now check the adjacent grid again, the right side and the upper right are the walls, skip it, the top and top left are in the close list, also skipped, the grid below the wall is also skipped here, can you cross the corner directly to the grid, depending on How to set up a node. So there are three grids left. The two grids below the current grid are not in the open list, so add them and make the current grids their parent nodes. The last grid is on the left side of the current grid. Check this one. Whether the path g value is lower. Repeat the above operation until the target cell is added to the open list and find the target path as shown in Figure 5.

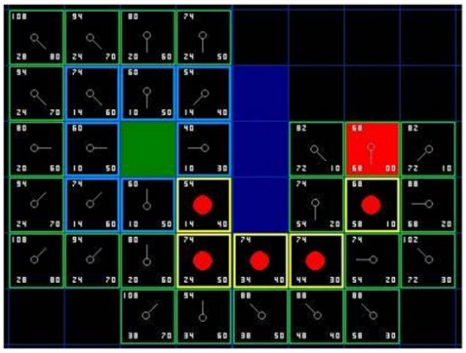

Fig.5 route plan

Note that the parent node of the grid below the starting grid is different from the previous one. Before it has a g value of 28 and points to the upper right grid, its g value is now 20, pointing to the grid above it. Moving from point B to the parent node in the direction of the arrow will eventually lead back to the starting grid, which is the path required.

III. A* ALGORITHM SIMULATION EXPERIMENT

The A* algorithm is simulated in MATLAB to verify its feasibility. Design a 30x30 rectangular space, set the coordinates of the starting point to (3, 3), and the coordinates of the target point to (29, 23). The simulation results are shown in Figure 4.4. The black dots in the figure indicate obstacles, the blue dots represent the starting points, the red dots represent the target points, and the green dots are the trajectory points generated according to the A* algorithm. The generated trajectory points are extracted and optimized by fitting to obtain a smooth trajectory as shown in Figure 6.

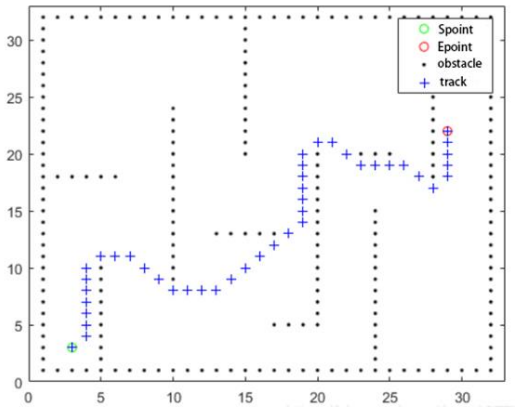

Fig.6 A* algorithm simulation results

IV. CONCLUSION

Mobile robot path planning technology has always been one of the research hotspots in the field of artificial intelligence. The ultimate goal is to plan an optimal path for mobile robots to avoid obstacles from the starting point to the end point. This paper briefly introduces the definition of mobile robot path planning and the tasks that need to be completed. Then it introduces the steps of A* algorithm and A* algorithm implementation. Finally, the MATLAB simulation experiment of A* algorithm is carried out to verify the effectiveness of the algorithm. It has certain reference significance for the future research on path planning.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

ACKNOWLEDGMENTS

This paper is supported by the key technologies R&D program of Tianjin (18YFZCSF00600)

REFERENCES

[1] Faster R-CNN:towards real-time object detection w ith region proposal netw orks. Ren S,He K,Girshick R.et al. IEEE Trans Pattern Anal M ach Intell(NIPS) . 2017.

[2] Yang Juncheng, Li Shuxia, Cai Zengyu. Research and Development of Path Planning Algorithms[J]. Control Engineering, 2017, 24(7): 1473-1480.

[3] ZHANG Yidong, ZHENG Rui, YAN Yuxi. Current Status and Prospects of Mobile Robot Path Planning Technology[J]. Journal of System Simulation, 2005, 17(2): 439-443.

[4] Xin Xin, Liang Huawei, Du Mingbo, et al. An improved A algorithm for searching infinite neighborhoods[J]. ROBOT, 2014, 36(5): 627-633.

[5] MILAD N, ESMAEEL K, SAMIRA D. Multi-objective multi-robot path planning in continuous environment using an en-Hanced genetic algorithm [J]. Expert Systems with Applications, 2019 (115): 106-120.

[6] Al-Sultan K S, Aliyu M D S. A new potential field-based algorithm For path planning[J]. Journal of Intelligent and Robotic Systems:Theory and Applications, 1996, 17(3): 265-282.

[7] Dong Zhuo-ning, Chen Zong-ji, Zhou Rui, et al. A hybrid approach Of virtual force and A* search algorithm for UAV path re-planning [P]. Industrial Electronics and Applications (ICIEA) , 2011 6th IEEE Conference on, 2011: 1140-1145.