

Path Planning for Mobile Robots Based on Improved A-star Algorithm

Yuanyuan Zhang

College of Electronic Information and Automation, Tianjin University of Science and Technology,
300222, China

Miao Zhang*

College of Electronic Information and Automation, Tianjin University of Science and Technology,
300222, China

E-mail: z004389@126.com, *miaozhang@tust.edu.cn

www.tust.edu.cn

Abstract

In this paper, an improved A-star (A*) algorithm is presented, which decreases the search time and path cost by introducing a bidirectional search strategy, enhancing the evaluation function. Through the addition of inflection points deletion method, the path smoothness is augmented, and the running speed and reliability of A* are enhanced. Eventually, the superiority of the improved A* algorithm is verified through comparative experiments.

Keywords: Path planning, A-star, Robot

1. Introduction

Path planning is an important research topic in the field of robotics [1]. With the continuous progress of technology, there is a growing demand for the application of path planning algorithms in complex environments, especially in the fields of autonomous navigation and unmanned driving [2]. Among them, A* algorithm [3] is widely used in path planning tasks due to its high efficiency.

The traditional A* algorithm searches for the shortest path in a graph through a heuristic search strategy, and although it performs well in many scenarios, it still has some shortcomings when dealing with complex environments or dynamic obstacles. For example, the traditional A* algorithm may generate more transitions and lengthy paths during the path planning process, leading to an increase in computation time and resource consumption [4]. In order to solve these problems, researchers have proposed a variety of improved algorithms to increase the efficiency and path quality of path planning. These improvements mainly focus on optimizing the heuristic function, reducing node expansion and improving path smoothness [5]. With these optimizations, the improved algorithms show better performance in several aspects such as computation time, path length and resource consumption.

The rest of this article is organized as follows. The second section introduces the principle of A* algorithm and describes the improvement scheme. The third part conducts comparative experiments between the improved A* algorithm and the A* algorithm, and demonstrates the

superiority of the improved A* algorithm. The fourth part summarizes the main content of this paper, and introduces future work.

2. Principles of A* algorithm and improvement methods.

Although the A* algorithm is an effective path planning algorithm, in some complex or irregular environments, the A* algorithm can face problems such as long planning time and high path cost. Therefore, the search time of the A* algorithm is reduced by introducing a bidirectional search strategy, improving the evaluation function, and removing redundant nodes in the search process. After that, the path cost is reduced and the smoothness of the path is increased by introducing a corner optimization algorithm.

2.1. A* algorithm

The A* algorithm is a widely used graph search algorithm, mainly for path planning and graph traversal. It combines the advantages of heuristic search and shortest path search to find the optimal path from the starting point to the end point among multiple nodes. The A* evaluation function is defined as $f(n) = g(n) + h(n)$, where $f(n)$ is the cost estimate from the initial state to the goal state via state n , $g(n)$ is the actual cost from the initial state to state n , and $h(n)$ is the estimated cost of the optimal path from state n to the goal state. The A* algorithm needs to maintain two state tables called openList and closeList.

openList consists of nodes to be examined and closeList consists of nodes that have already been examined.

Initially, the algorithm adds the starting point to the openList table and sets the closeList table to empty. Then start the loop: select a node n from the openList table with the smallest value of f . If n is the target node, the algorithm ends and returns the path. Otherwise, move n to closeList. for each neighbor node of n : if the neighbor is in closeList, skip. If the neighbor is not in openList, add it and compute g , h and f values. If the neighbor is already in the openList table, check if the new g value is smaller, and if so, update its g , h and f values. Repeat the above steps until the target node is found or the openList table is empty (indicating no solution). The A* algorithm flowchart is shown in Fig.1.

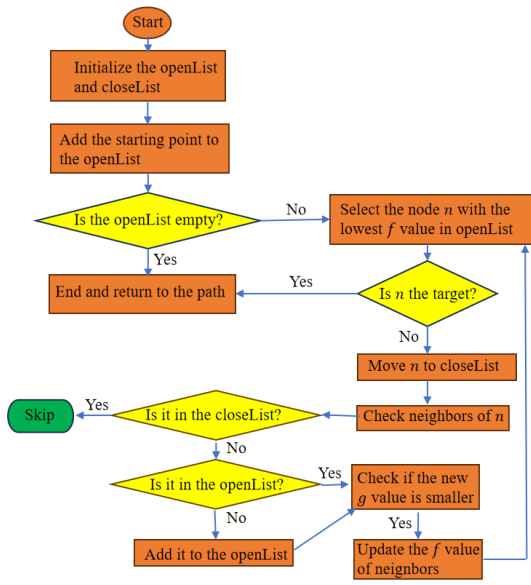


Fig.1. A* algorithm

2.2. Bidirectional A*

The bidirectional search strategy of the A* algorithm is an optimization method that speeds up the path finding process by searching from both the start and goal points. As shown in Fig.2, a forward search (blue line) from the starting point and another reverse search (green line) from the goal point, when the forward search and reverse search meet at a certain point, the complete path can be obtained by backtracking.

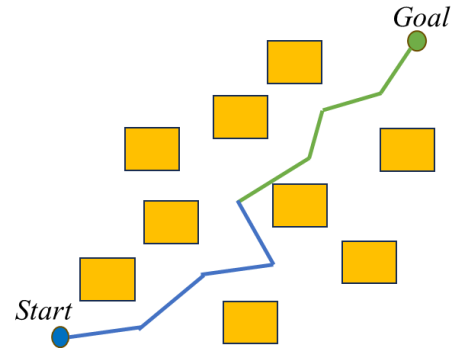


Fig.2. Bidirectional A* algorithm

2.3. Improvement of the evaluation function

In order to make the A* algorithm have better search efficiency in complex environment, the evaluation function of A* algorithm is improved. The evaluation function of the traditional A* algorithm is shown in Eq. (1).

$$f(n) = g(n) + h(n) \quad (1)$$

Where, $f(n)$ is the composite evaluation value, $g(n)$ is the path cost function from the current point to the start point, $h(n)$ is the heuristic function of the path cost from the current node to the goal point.

Since the heuristic function is a Euclidean distance, the value of the heuristic function is never greater than the actual distance from the current point to the target point. When the current point is far away from the target point, the estimated value of the heuristic function is much smaller than the actual value, the algorithm searches for more nodes, low computational efficiency, at this time the weight of the estimated value should be increased to improve the computational efficiency; when the current point is gradually close to the target point, the estimated value is gradually approaching the actual value, in order to prevent the estimated value of the search of the optimal path is not too large, the estimated value of the weight should be reduced. Therefore, the improved evaluation function is shown in Eq. (2).

$$f(n) = g(n) + \left(\frac{R+r}{R}\right)h(n) \quad (2)$$

Where, R is the distance from the start point to the target point, and r is the distance between the current point and the target point.

2.4. Inflection points deletion method

As shown in Fig.2, the paths planned by the A* algorithm always have a lot of inflection points, which are very unfavorable for the movement of the robot. Therefore, after the A* algorithm generates the path, there is a parent-child relationship between the inflection points, and the last inflection point of the goal point is the parent inflection point. As shown in Fig.3, the goal point is the initial inflection point and the goal point is connected to its grandfather inflection point. If there is no obstacle between

it and the grandfather inflection point, then connect the goal point and the grandfather inflection point, delete the parent inflection point of goal point, the goal point's grandfather node becomes its parent node, and continue to connect the following grandfather node until there is a collision with the obstacle. After that, the parent inflection point before hitting the obstacle is the initial inflection point, continue to find its grandfather node, and so on until the initial inflection point is connected. By removing redundant inflection points, the total cost of the path is reduced and the smoothness of the path is increased.

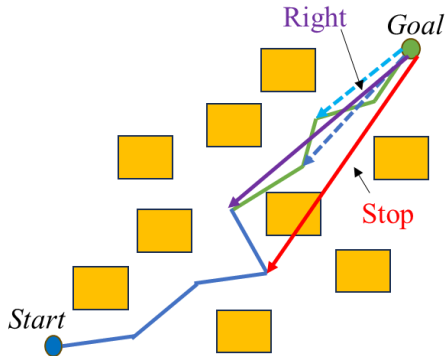


Fig.3. Inflection point deletion method

3. Simulation

Based on the three improvement schemes in section 2, the improved A* algorithm is designed. In this section, two different sets of environments are set up in MATLAB to compare the performance of the A* algorithm and the improved A* algorithm. The map is in 30*30 size with black squares as obstacles. The first set of environments is a simple environment with a start point at [22.5, 15.5] and a goal point at [26.5, 28.5], and the Euclidean distance from the start point to the end point is 13.60. The second set of environment is relatively complex environment, the starting point is [28.5, 19.5], the target point is [13.5, 16.5], the Euclidean distance from the starting point to the end point is 15.30.

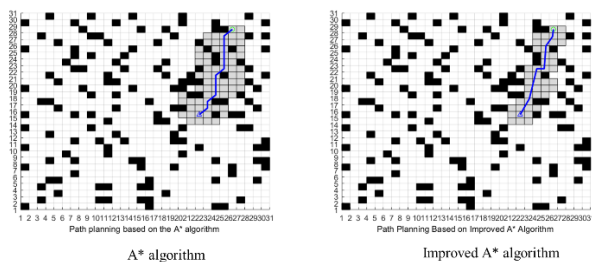


Fig.4. Simulation in simple environment

Table 1: Simulation date in simple environment

	Planning time	Sum of transition degrees	The path cost	Iterate over the sum of the nodes
A*	0.0018	270.00	14.65	69
Improved A*	0.0015	229.17	13.40	45

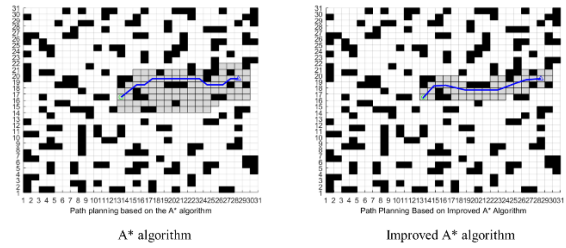


Fig.5. Simulation in complex environment

Table 2: Simulation date in complex environment

	Planning time	Sum of transition degrees	The path cost	Iterate over the sum of the nodes
A*	0.0020	315.00	17.56	89
Improved A*	0.0016	309.14	16.40	46

As shown in Fig.4 and Fig.5, improved A* algorithm has lower path cost and better smoothing compared to A* algorithm. As can be seen from

Table 1, the improved A* reduces the planning time by 16.67%, the sum of corner degrees by 15.12%, the path cost by 8.5%, and the sum of iterative nodes by 34.78% compared to the A* algorithm in the simple environment. As can be seen from Table 2, the improved A* reduces the planning time by 20%, the sum of corner degrees by 1.86%, the path cost by 6.6%, and the sum of iterative nodes by 48.31% compared to the A* algorithm in the complex environment. These experimental results prove that the improved A* algorithm is more efficient compared to the A* algorithm.

4. Conclusion

This paper introduces the principle of the A* algorithm and improves the A* algorithm. The search efficiency of the A* algorithm is improved and the search time is reduced by introducing a bidirectional search strategy and improving the evaluation function. After the path generation, the path cost is reduced by the corner optimization function and the path smoothness is improved. Finally, the improved A* algorithm is compared with the A* algorithm in MATLAB, and the experiment proves that the improved A* algorithm requires less search time and generates a lower path cost and has better smoothness. This improved A* algorithm does not consider targeted path smoothing for vehicles. In the future, vehicle kinematics will be considered to impose kinematic constraints on the A* algorithm.

References

1. Y. Wang, H. Li, Y. Zhao, et al, A Fast Coordinated Motion Planning Method for Dual-Arm Robot Based on Parallel Constrained DDP, *IEEE/ASME Transactions on Mechatronics*, 2024, 29 (3): pp.2350–2361.
2. V. N. Hartmann, A. Orthey, D. Driess, et al, Long-Horizon Multi-Robot Rearrangement Planning for Construction Assembly, *IEEE Transactions on Robotics*, 2023, 39(1): pp.239–252.
3. P. E. Hart, N. J. Nilsson and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, 1968, 4(2): pp.100–107.
4. Bing Fu, Lin Chen, Yuntao Zhou, et al, An improved A* algorithm for the industrial robot path planning with high success rate and short length, *Robotics and Autonomous Systems*, 2018, 106, pp.26–37.
5. Rui Song, Yuanchang Liu, Richard Bucknall, Smoothed A* algorithm for practical unmanned surface vehicle path planning, *Applied Ocean Research*, 2019, 83, pp. 9–20.

Authors Introduction

Ms. Yuanyuan Zhang



She is pursuing a bachelor's degree in engineering from Tianjin University of Science and Technology.

Ms. Miao Zhang



She is a postgraduate tutor of Tianjin University of Science and Technology. In 2019, she received a doctorate from the University of Windsor, Ontario, Canada. The research direction is intelligent algorithms design filters, the control system design of industrial robots and control theory.
