Scholar Publishing Group
International Journal of Multimedia Computing
https://doi.org/10.38007/IJMC.2024.050101
ISSN 2789-7168 Vol. 5, Issue 1: 1-13

Improved A Algorithm Based on Bessel Curve Optimization*

Huiheng Suo[1,a,*], Tengsheng Yang[1,b], Qiang Hu[1,c], Jian Wu[1,d*], Xie Ma[2,e], Qingwei Jia[3,f], Zizhen Chen[4,g], Xiushui Ma[5,h]

[1]*Nanchang Hangkong University, Nanchang, China*

[2]*Ningbo University of Finance & Economics, Ningbo, China*

[3]*WILDSC (Ningbo) Intelligent Technology CO.,LTD, Ningbo, China*

[4]*Ningbo Polytechnic, Ningbo, China*

[5]*NingboTech University, Ningbo, China*

[a]*suohuiheng@163.com*, [b]*yts2610@163.com*, [c]*1289762339@qq.com*, [d]*flywujian@qq.com*, [e]*maxie@163.com*, [f]*qingwei.jia@wildsc.com.cn*, [g]*e0400326@nbpt.edu.cn*, [h]*mxsh63@aliyun.com*

[*]*corresponding author*

Keywords: Improved A*, Bezier Curve, Path Planning, ROS Robots

Abstract: Aiming at the traditional A* algorithm in solving the robot path planning, there are the problems of longer path trajectory and computation time, more nodes for searching, and the path is not smooth enough. In this paper, an improved A* algorithm based on Bessel curve optimization is proposed. First, the traditional unidirectional search strategy of A* algorithm is changed to a bidirectional search strategy, which dynamically defines the target nodes of forward and reverse search; at the same time, an improved heuristic function is introduced to improve the search efficiency of A* algorithm by reducing the complexity of the planning space; and then the improved A* algorithm is combined with the Bessel curve optimization algorithm to eliminate the redundant inflection points in the robot's path, to make the path smoother and closer to the optimum. The experimental results show that the improved A* algorithm improves the efficiency of path planning, increases the stability and path smoothness, and is easier to apply in practice.

1. Introduction

Path planning, as one of the key technologies for robots, is a prerequisite for accomplishing navigation and other complex tasks [1]. Traditional path planning algorithms, such as the A* algorithm [2], Dijkstra's algorithm [3], and BFS algorithm [4], show good search performance in many application scenarios, but there are problems such as too large search space and easy to fall

into the local optimization when dealing with complex environments. To address the limitations of the traditional A* algorithm, a path enhancement method is proposed in the paper [5] to shorten the path length by determining whether the line between neighboring path nodes passes through an obstacle or not. The A* algorithm for jump-point search was utilized in paper [6] and paper [7], which reduces the number of search nodes, but still suffers from many inflection points and proximity to obstacles. Paper [8] proposes a hybrid heuristic function that improves the computational efficiency of the algorithm but is prone to the risk of falling into local optimality when there are more obstacles. Paper [9] extends the traditional A* algorithm to an infinite number of search directions based on the 8 search directions, which greatly reduces the number of bending points, but ultimately the time taken is too long.

To solve these problems, this paper proposes an improved A* algorithm based on fused Bessel curve optimization. The method uses dynamic weighting, improved heuristic functions, and a two-way search strategy to speed up the search. Extract global path critical points as guidance points, smoothing of guide points using third-order Bessel curves, which improves the efficiency of path planning for mobile robots in complex environments.

2. Traditional A* Algorithm

The essence of the A* algorithm is the heuristic search algorithm, based on the classical Dijkstra's algorithm a heuristic function is introduced, calculating the cost of each neighboring node using the valuation function $f(n)$, thereby improving the computational efficiency of path planning [10]. The valuation function is calculated as follows:

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

Where, $g(n)$ denotes the actual cost consumed from the starting point to the current node. $h(n)$ is the heuristic function used to estimate the predicted cost from the current node to the target node. In extreme cases, when $h(n) = 0$ and $f(n) = g(n)$, Considering only the actual cost, path planning is prioritized to ensure that the shortest path is found, at this point the algorithm degenerates into Dijkstra's algorithm, this path planning approach leads to too many search nodes and reduces the efficiency of the search; When $h(n) = g(n)$, At this time, the optimal path can be found very quickly, however, in practice it is difficult to calculate the distance to the target point, so it is difficult to implement; When $g(n) = 0$ and $f(n) = h(n)$, only the estimated cost needs to be considered, at this time, the algorithm may degenerate into the BFS algorithm, this method has fewer search nodes and can search quickly, but it cannot guarantee to find the optimal path.

The choice of $h(n)$ directly affects the algorithm's speed and accuracy. The heuristic function is usually calculated using Manhattan distance [11] or Euclidean distance [12]. Euclidean distance algorithm $h_1(n)$, Manhattan distance algorithm $h_2(n)$, the expression is:

$$\begin{cases} h_1(n) = \sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2} \\ h_2(n) = \left|X_j - X_i\right| + \left|Y_j - Y_i\right| \end{cases} \tag{2}$$

Where, (X_i, Y_i) represents the starting point position coordinates, and (X_j, Y_j) represents the target point position coordinates.

3. Improved A* Algorithm

3.1. Bidirectional A*

The traditional A* algorithm has problems in search efficiency and convergence speed [13]. In response to these problems, the traditional A* algorithm is improved, the main innovations of the improved A* algorithm are as follows:

(1)Bidirectional search: In the improved A* algorithm, a bidirectional search strategy is adopted, utilizing the information of the starting point and the end point, and continuously updating the search paths of the starting point and the end point alternately during the search process until some intermediate point is jointly searched. This avoids searching too many redundant nodes and reduces the space and time complexity of search.

(2)Improved heuristic function: The $h(n)$ of the traditional algorithm cannot meet the actual needs. It is crucial to choose the appropriate $h(n)$ .This paper combines the advantages of Chebyshev distance and proposes an improved heuristic function, then adds the corresponding weighting factors. The forward search valuation function $f_F(n_F)$ as follows:

$$f_F(n_F) = g_F(n_F) + w(n) * h_F(n_F) \tag{3}$$

And the backward search valuation function $f_B(n_B)$ of the weighted algorithm are

$$f_B(n_B) = g_B(n_B) + w(n) * h_B(n_B) \tag{4}$$

In equation(3) and equation(4),

$$w = \begin{cases} 2 & f_F(n_F) = f_B(n_B) > 12 \\ 0.8 & f_F(n_F) = f_B(n_B) \le 12 \end{cases} \tag{5}$$

$$h_F(n_F) = h_B(n_B) = \begin{cases} dist_2 & dist_2 > dist_1 \\ dist_1 & ohters \end{cases} \tag{6}$$

In equation(6),

$$dist_1 = \left| xn_F - xn_B \right| \tag{7}$$

$$dist_2 = \left| yn_F - yn_B \right| \tag{8}$$

where, w is the weighting factor, n_F represents the current node in the OpenList table of forward search, n_B represents the current node in the OpenList table of backward search, $g_F(n_F)$ represents the estimated cost from the starting point to n_F , $g_B(n_B)$ represents the estimated cost from target point n_B . $h_F(n_F)$ and $h_B(n_B)$ are heuristic functions, representing the minimum path cost from n_F to n_B and the minimum path cost from n_B to n_F respectively.

The specific process of improving the A* algorithm for path search is as follows:

Step 1: Create two Openlist tables: one for searching in the starting direction and one for searching in the ending direction. At the same time, create two Closelist tables and two Parentlist tables, which are used to save the visited nodes and the parent nodes of the current node respectively.

Step 2: Add the starting point and end point to the starting point Openlist table and the end point

Openlist table respectively, and set the estimated cost values of the starting point and end point.

Step 3: Select the node with the smallest estimated cost value from the starting point Openlist table and the ending point Openlist table respectively, which are called the current starting point node and the current ending node respectively.

Step 4: Check whether the current starting point node and the current ending point node meet, that is, whether the same node exists in the Closelist table of the starting point and ending point. If they meet, the shortest path has been found.

Step 5: If the estimated cost value of the current starting point node is less than the estimated cost value of the current ending node, then the expansion in the starting point direction is performed. Move the current starting point node from the starting point Openlist table to the starting point Closelist table, add its adjacent unvisited nodes to the starting point Openlist table, and update their estimated cost values and parent nodes.

Step 6: If the estimated cost value of the current end node is less than or equal to the estimated cost value of the current start node, the end direction is expanded. Move the current end point node from the end point Openlist table to the end point Closelist table, adding its adjacent unvisited nodes to the end point Openlist table, and update their estimated cost values and parent nodes.

Step 7:Repeat steps 3 to 6, until the shortest path is found or the starting point Openlist table and the destination Openlist table are empty (that is, the path cannot be found).

Step 8: If the shortest path is found, backtrack the path from the Closelist table of the start and end points respectively. The parent node of each node can be obtained through the Parentlist table, backtrack from the end point until you reach the starting point to get the shortest path.

To verify the effectiveness of the improved A* algorithm, a simulation experiment was conducted to compare with the traditional A* algorithm in a static environment. The system used in the simulation environment is Windows 11, and the simulation platform is PyCharm Community Edition 2023.2.4. Figure 1 and Figure 2 show the simulation effects of the traditional A* algorithm and the improved A* algorithm respectively.

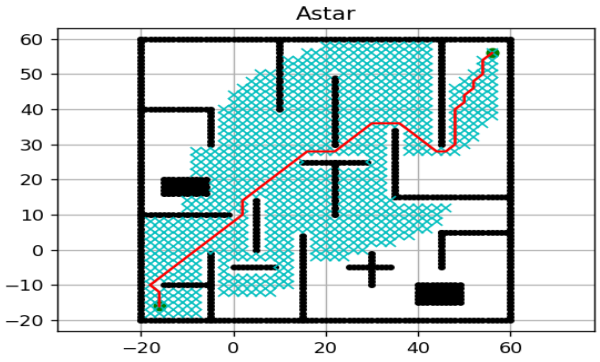

Figure 1. Traditional A algorithm*

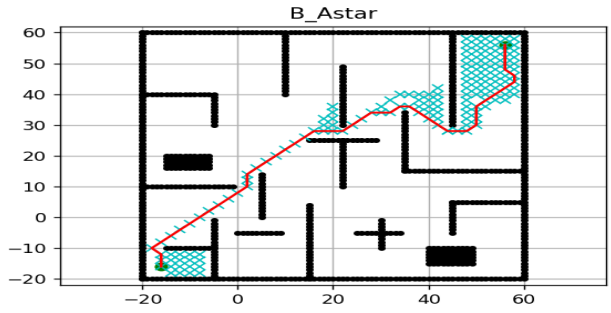

Figure 2. Improved A algorithm*

The white area in the figure represents the free area without obstacles, black areas represent obstacles, the starting point is set at the green circle with coordinates (-5,-5), the end point is set at the green circle with coordinates (55,55), the blue × represents the node searched during the path planning process, and the red line represents the generated path. The comparison of algorithm performance indicators is shown in Table 1.

Table 1. Simulation experiment data statistics

Algorithms	Number of turns	Trajectory length	Number of search nodes	Calculation time
Traditional A*	18	65.41cm	947	5.8s
Improved A*	16	62.17cm	176	1.6s

According to the data in Table 1, it can be seen that in a simple 80×80 raster environment map, compared with the traditional A* algorithm, the trajectory length of the improved A* algorithm is shortened from 65.41cm to 62.17cm; the number of search nodes was reduced from 947 to 176; the calculation time was reduced from 5.8s to 1.6s. Therefore, by combining the three indicators of the algorithm, an improved A* algorithm search algorithm has more advantages.

3.2. Bezier Curve Optimization

As can be seen from Figure 2, the path generated by the improved A* algorithm has many inflection point problems. Therefore, a cubic Bezier curve optimization algorithm is introduced to remove the concave and convex points in the path to make it smooth and continuous. By formula

$$B(t)=\sum_{i=0}^{n} p_i b_{i,n}(t) \tag{9}$$

The mathematical expression of the Bezier curve when n=3 can be obtained as:

$$B(t)=\sum_{i=0}^{n} p_i b_{i,3}(t) \tag{10}$$

Where,

$$b_{i,n}(t)=\binom{n}{i} t^i (1-t)^{n-i} \tag{11}$$

Where, $b_{i,n}(t)$ is the Bernstein polynomial, t∈[0,1],i=0,1,2,.....,k-1

Derivation of t based on Eq. (9) yields:

$$B^{'}(t)=\sum_{i=0}^{n} n(p_{i+1}-p_i)b_{i,n-1}(t)$$
$$=\sum_{i=0}^{n-1} n(p_{i+1}-p_i)\binom{n-1}{i}t^i(1-t)^{n-1-i} \quad (12)$$

When the vector interpolation of each node of the cubic Bezier curve is a constant, it denotes a cubic uniform Bezier curve [14]. The expression of the i-th cubic uniform Bezier curve is:

$$B_i(t)=\sum_{i=0}^{3} p_i b_{i,n}(t) \quad (13)$$

From formula (9, 11, 13), the basis function expression of cubic Bezier curve can be obtained as:

$$\begin{cases} B_{1,3}(t)=(1-t)P_0+tP_1 \\ B_{2,3}(t)=(1-t)^2P_0+2t(1-t)P_1+t^2P_2 \\ B_{3,3}(t)=(1-t)^3P_0+3t(1-t)^2P_1+3t^2(1-t)P_2+P_3t^3 \end{cases} \quad (14)$$

where, t is the normalized variable, P_0 P_1 P_2 and P_3 are the four control points of the curve.

The simulation after integrating the improved A* algorithm and Bezier curve optimization is shown in Figure 3.The red line is the path generated by the improved A* algorithm, the blue curve is the path generated by merging Bezier curve optimization. It can be seen that the path generated after integrating the Bezier curve optimization algorithm is shorter and smoother.

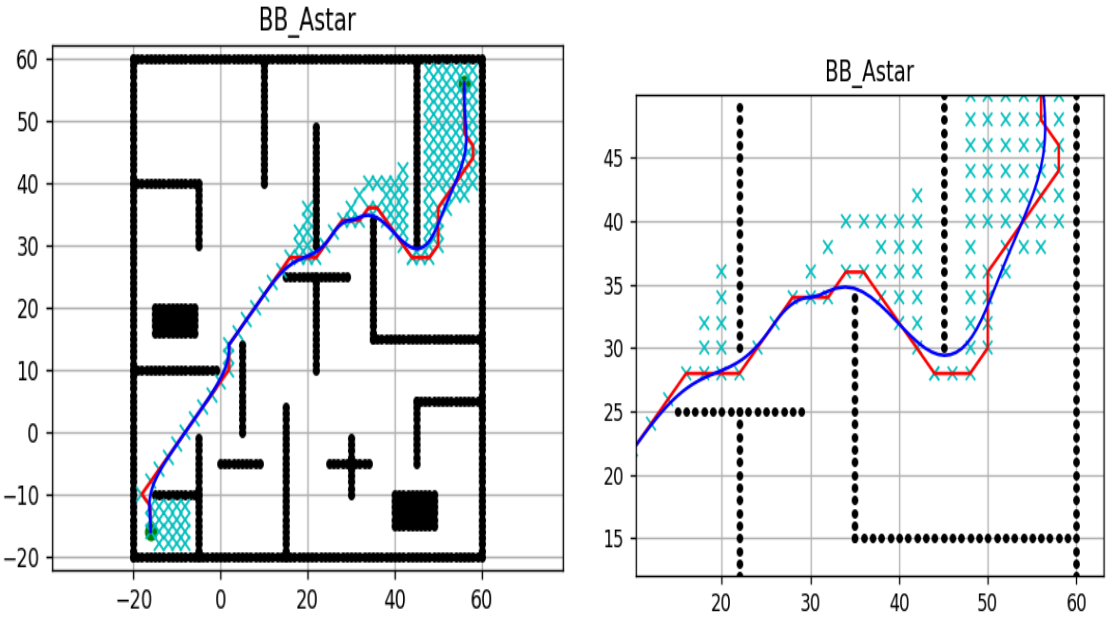

Figure 3. Bidirectional A and Bessel Curve Fusion*

3.3. Fusion Algorithm Path Planning

This paper integrates the above two methods and extracts the global key points of the improved A* algorithm as the guidance points of the Bezier curve optimization algorithm. The path generated by the fusion algorithm can reduce unnecessary turns and oscillations when the robot performs navigation tasks while ensuring global optimality. The algorithm flow is shown in Figure 4.

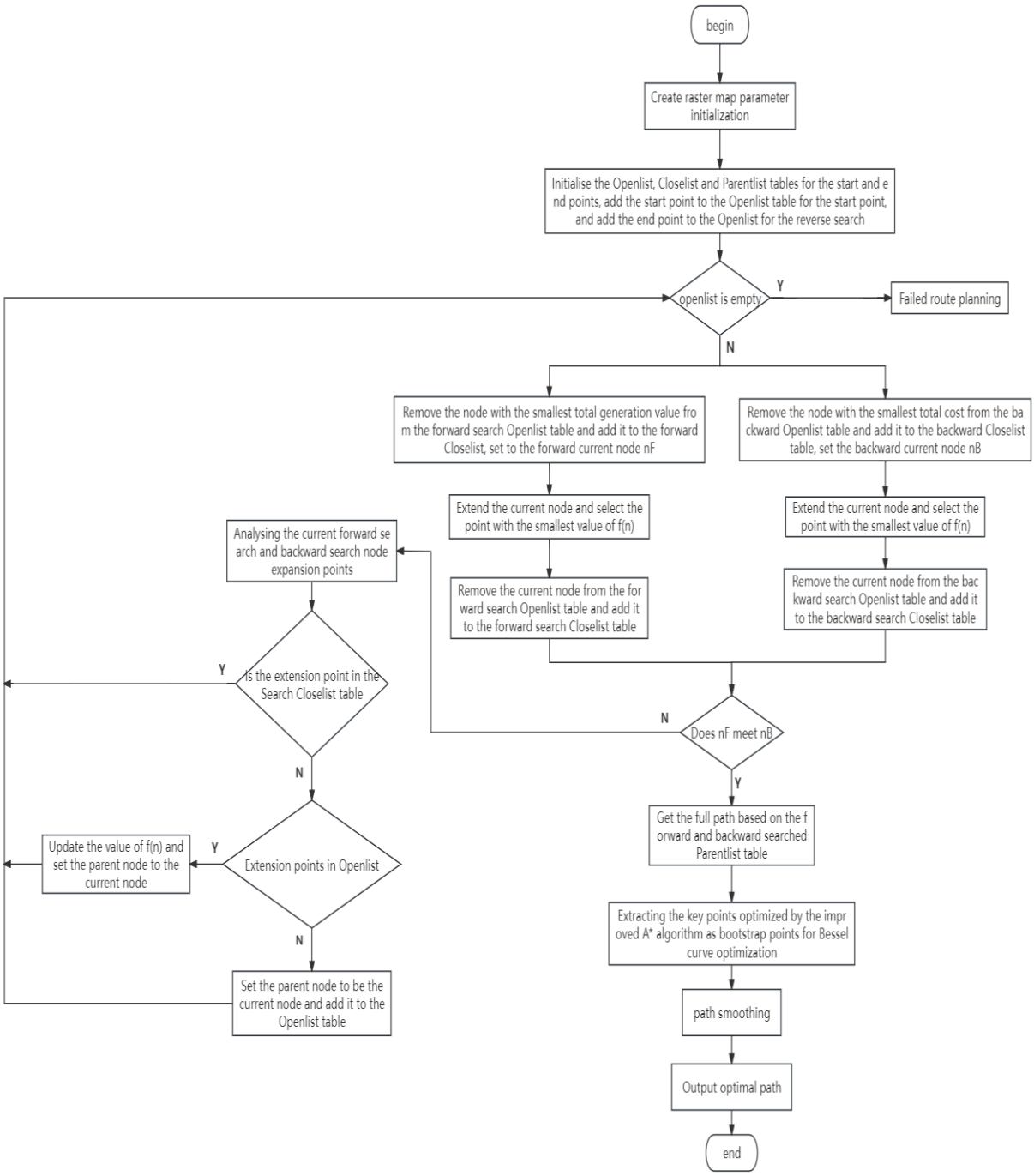

Figure 4. Fusion algorithm flow chart

4. Experimental Verification

The previous section describes the PyCharm simulation environment under the, compared with the traditional A* algorithm, the improved A* algorithm has significant performance improvements. In order to further verify the significant advantages of the improved A* algorithm, In this chapter, the improved A* algorithm will be deployed and experimentally analyzed in ROS simulation environment and real vehicle environment, respectively.

4.1. Simulation Experiments and Analysis

4.1.1 Experimental Platforms

Figure 5 shows the ROS simulation experiment environment with Ubuntu system 20.04 and ROS version noetic. Building the simulation environment in Gazebo and displaying it graphically. The simulated cart is equipped with LiDAR and the corresponding raster map is constructed by LiDAR, as shown in Figure 6 to Figure 9.

4.1.2 Simulation Experiment

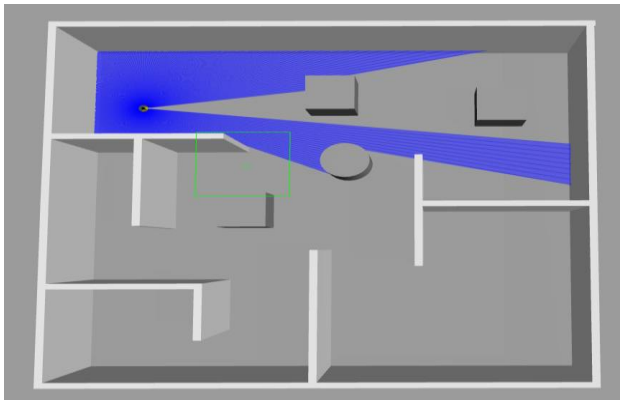

Figure 5. Gazebo simulation environment

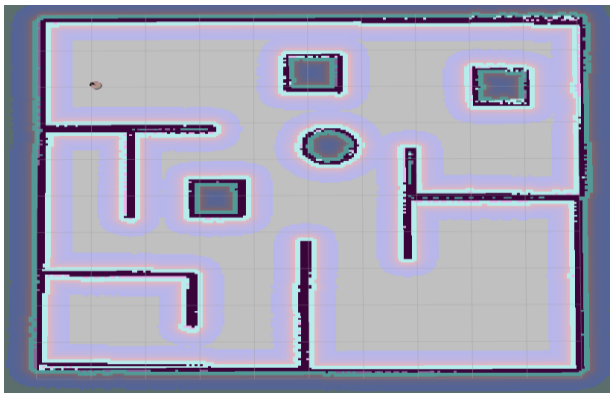

Figure 6. SLAM map constructed by LiDAR

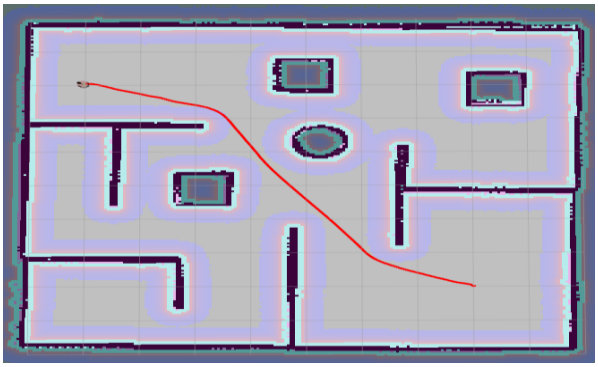

Figure 7. Traditional A algorithm*

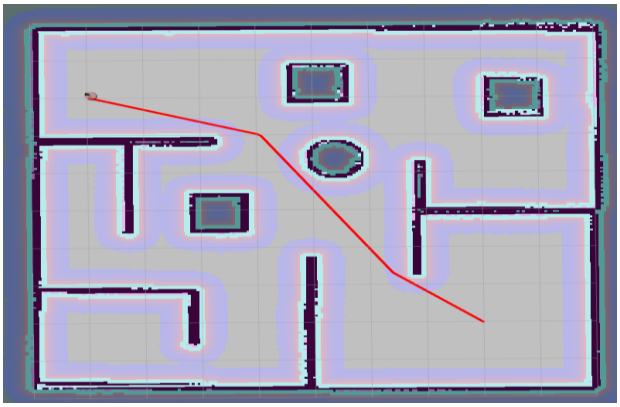

Figure 8. Improvement of the A algorithm*

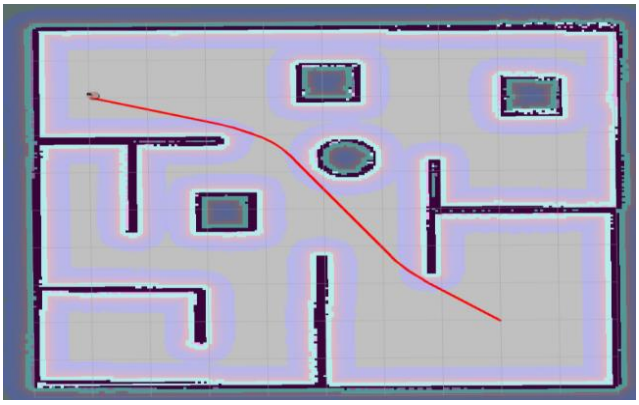

Figure 9. Fused Bessel curve optimization algorithm

Table 2 shows the algorithmic performance metrics of the two A* algorithms for path planning in the simulation environment, and the comparison shows that the improved A* algorithm is more advantageous in terms of both path length and search time.

Table 2. Comparison of search results

Algorithms	Trajectory length	Time
Traditional A*	12.47cm	4.72ms
Bidirectional A*	11.86cm	2.11ms

4.2. Real Vehicle Experiment and Analysis

4.2.1 Experimental Platforms

The real vehicle experiments were conducted on the ROS mobile robot platform shown in Figure 10. The robotic platform is equipped with LIDAR and is connected to the development host via WIFI. Ubuntu 20.04 was used as the development host system to control the mobile robot using the ROS operating system, where the ROS version is Noetic. On the robot side, a Raspberry Pi is used as a controller to realize motion control and acquisition of sensory data for the robot.

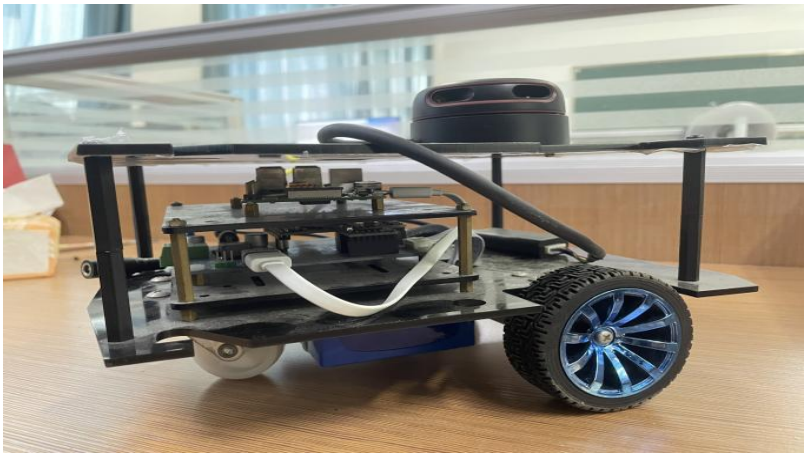

Figure 10. Experimental Robot

4.2.2 Experimental Environment and Map Construction

In this paper, a corridor outside the laboratory was chosen as the experimental environment for robot navigation testing, as shown in Figure 11. The map of the experimental environment was constructed using the Gmapping SLAM algorithm using an on-board LiDAR sensor [15].

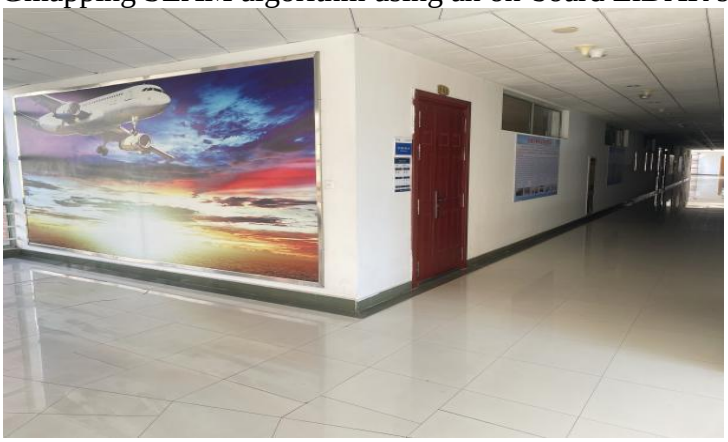

Figure 11. Experimental Environment

Figure 12 shows the environment map constructed by the robot in the experimental site. It can be seen that this map has a clear structural outline and high accuracy, which can meet the environmental needs of subsequent research and provide a reliable global map for subsequent comparisons of the two path planning algorithms.

Figure 12. Map of the experimental environment

4.2.3 Autonomous Navigation Implementation

On the maps constructed in the previous section, navigation experiments are conducted on the traditional A* algorithm, the improved A* algorithm, and the fusion algorithm, respectively. Open the Rviz visualization interface, complete the corresponding display configuration, perform the initial position estimation and specify the target point.

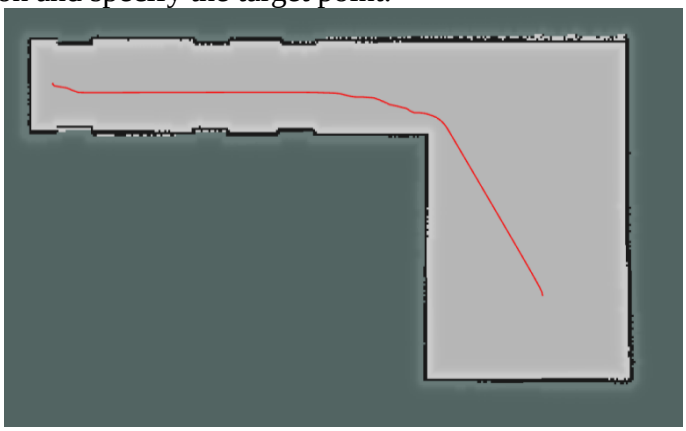

Figure 13. Traditional A Algorithm*

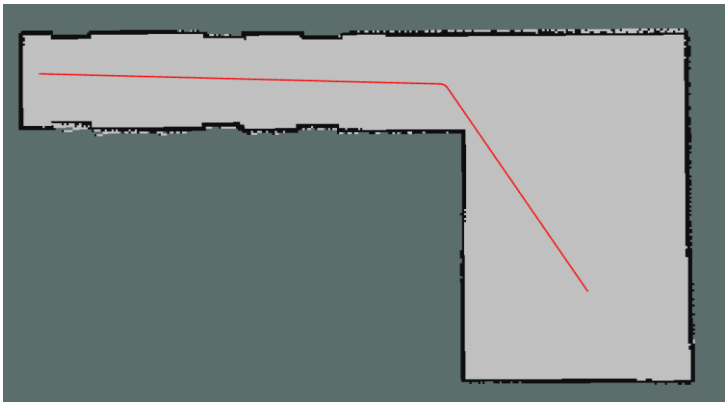

Figure 14. Improved A algorithm*

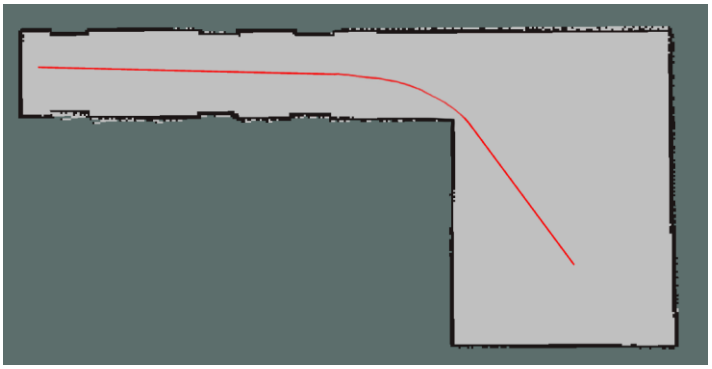

Figure 15. Fusion Bezier Curve Optimization Algorithm

Table 3 shows the algorithmic performance metrics of the two A* algorithms for experimental path planning in real vehicles. It can be seen that the improved A* algorithm is better than the traditional A* algorithm in terms of path length.

Table 3 Comparison of search results

Algorithms	Trajectory length	Time
Traditional A*	10.32m	5.26ms
Bidirectional A*	10.15m	2.47ms

Experimental results prove that compared with the traditional A* algorithm, the improved A* algorithm is more efficient in terms of overall planning efficiency, and the planned paths have smaller transitions and better smoothness.

5. Conclusions

The Improved A* algorithm performs several optimizations based on the traditional A* algorithm. First, based on the traditional A*, a bidirectional search strategy is proposed, which uses the information of the starting point and the end point to approach the target from two directions at the same time to avoid searching for too many redundant nodes during the path planning process; Secondly, an improved heuristic function is introduced to reduce the path length and planning time. In order to solve the problem that the path has many inflection points, the Bezier curve optimization algorithm is introduced to smooth the path, remove the concave and convex points in the path, and reduce the number of iterations of node selection and path inflection points. Finally, the improved algorithm was fused and experiments were conducted in actual scenarios to verify the feasibility and effectiveness of the proposed algorithm.

Funding

This paper is supported by Projects of major scientific and technological research of Ningbo City (2020Z065, 2021Z059,2022Z090(2022z050), 2023Z050(the second batch)), Major instrument special projects of the ministry of science and technology of China(2018YFF01013200), Projects of major scientific and technological research of Beilun District, Ningbo City(2021BLG002, 2022G009), Projects of engineering research center of Ningbo City (Yinzhou District Development and Reform Bureau [2022] 23), Projects of scientific and technological research of colleges student's of China(202313022036).

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Liu Y, Huang RY, Xiong QH. Robot path planning based on improved A algorithm. Automation and Instrumentation, 2015(4): 1-4.*

[2] Jian Zhang, Liguo Liu, Wei Chen. Research on robot path planning based on Bessel curve. Mechanical Design and Manufacturing, 2017(7): 61-64.

[3] Y.Q. Wang, Robot path planning based on improved A algorithm with Bessel curves. Nanjing: Nanjing University of Aeronautics and Astronautics, 2018.*

[4] Xiao Yaming, Li Hongmei, Huang Jun, et al. A robot path planning method based on Bessel curves. Computer Applications and Software, 2015, 32(10): 188-191.

[5] Rong Cao, Research on robot path planning based on improved A algorithm with Bessel curve. Automation and Instrumentation, 2016(1): 9-12.*

[6] K. Karaman and E. Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. International Journal of Robotics Research, 30(7):846-894, 2011.

[7] Qi Li, Jian Zhang, Research on robot path planning algorithm based on Bessel curve. Robotics Technology and Application, 2017(1): 37-40.

[8]D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1):23-33, 1997.

[9]state-of-the-art robot path planning techniques for unknown environments. In 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 28-33).

[10]Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31(5), 1147-1163.

[11]Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2017). On-manifold preintegration for real-time visual–inertial odometry. IEEE Transactions on Robotics, 33(1), 1-21.

[12] H. Zhang, SLAM-based mobile robot localisation and map construction. Beijing: Tsinghua University Press, 2011.

[13] Pan Jianwei, LiDAR technology and its application in UAV navigation and control. Computer Science and Applications, 2014, 4(2): 163-171.

[14] M. Gao, Robot path planning in unknown environment based on ant colony algorithm. Automation and Control, 2011(3): 23-26.

[15]Guo Jian, Zhang Hongxia, Li Li. Research on path planning algorithm in unknown environment based on UAV and mobile robot collaboration_Journal of Automation. 2015(3):550-558.