Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2024.050113
ISSN 2790-0932 Vol. 5, Issue 1: 112-127

Improved A Algorithm Based on Fused Theta Optimization*

Tengsheng Yang[1,a] , Huiheng Suo[1,b] , Rui Rao[1,c], Jian Wu[1,d,*] , Tao Sun [2,e], Xie Ma[2,f], Bibo Yu[3,g], Yingping Bai[4,h], Xiaoqin Li[5,i], Yuhan Sun[5,j], Xiushui Ma[3,k]

[1]Nanchang Hangkong University, Nanchang, China
[2]Ningbo University of Finance & Economics, Ningbo, China
[3] Huayuan New Materials Co., Ltd, Ningbo, China
[4]NingboTech University, Ningbo, China
[5]Ningbo Polytechnic, Ningbo, China
[a]yts2610@163.com, [b]suohuiheng@163.com, [c]ruirao163@163.com, [d]flywujian@qq.com, [e]871522458@qq.com, [f]maxie@163.com, [g]583771406@qq.com, [h]3459951916@qq.com, [i]47154318@qq.com, [j]1917201715@qq.com, [k]mxsh63@aliyun.com
[*]corresponding author

Keywords: Improved A*, Bezier curves, Jump point search, Theta*, Path planning

Abstract: In this paper, an improved A* algorithm based on fused Theta* is proposed. Firstly, the traditional search strategy of the A* algorithm is changed to a jumping search strategy, which intelligently determines which nodes need to be expanded and which nodes do not need to be expanded at each step of the expansion, based on the direction that the parent node is expanding from, and whether there is any obstacle and its location information around it, The idea of Theta* algorithm is also introduced to check whether the nodes in the final path can be directly connected, if so, the intermediate nodes are skipped and the intermediate path is pruned. The improved A* algorithm is then combined with the Bessel curve optimization algorithm to eliminate redundant inflection points in the robot's path, resulting in a smoother and near-optimal path. As analyzed by simulation experiments, compared to the traditional A* algorithm, the improved A* algorithm shortens the path length by 4.34%, reduces the computation time by 76.9%, and reduces the number of search nodes by 67.4%. The experimental results show that the fusion algorithm improves the efficiency of path planning, increases the stability and path smoothness, and is easier to apply in practice.

1. Introduction

In the field of modern path planning, finding an optimal path from the starting point to the end point is a challenging task[1], especially in complex and changing environments. Traditional path planning algorithms, such as the A* algorithm[2], Dijkstra's algorithm[3], and the BFS algorithm[4],

tend to produce too many inflection points when dealing with path planning in open areas, resulting in paths that are not smooth enough, and the computational efficiency needs to be improved. In order to solve these problems, researchers have proposed a variety of improvement strategies, a path enhancement method is proposed in literature[5] to shorten the path length by determining whether the line between neighboring path nodes passes through an obstacle or not. The bi-directional A* algorithm is utilized in literature[6] and literature[7], which reduces the number of search nodes, but still suffers from many inflection points and proximity to obstacles. Literature[8] proposed a hybrid heuristic function based on the original heuristic function, which improves the computational efficiency of the algorithm, but it is easy to fall into the risk of local optimization in the case of more obstacles. Literature[9] and literature[10] extend 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 address the above challenges, this paper proposes an improved A* algorithm based on fused Theta* optimization. The algorithm not only inherits the high efficiency of the A* algorithm and the path-smoothing property of the Theta* algorithm, but also further smoothes the paths by introducing Bessel curves, and at the same time adopts the jump-point search and the improved heuristic function to accelerate the search process. This fusion strategy not only significantly improves the efficiency of path planning, but also generates smoother and more intuitive paths in complex environments, which further enhances the practicality and adaptability in application scenarios such as mobile robots.

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, calculate the cost of each neighboring node using the valuation function $f(n)$,thereby improving the computational efficiency of path planning[11] .The valuation function is calculated as follows:

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

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, $f(n) = g(n)$, 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, $f(n) = h(n)$, that is, 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 speed and accuracy. The heuristic function is usually calculated using Manhattan distance[12] or Euclidean distance[13].Euclidean distance algorithm $h_1(n)$, Manhattan distance algorithm $h_2(n)$, the expression is:

$$h_1(n) = \sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2} \tag{2}$$

$$h_2(n) = \left|X_j - X_i\right| + \left|Y_j - Y_i\right| \tag{3}$$

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

3. I mproved A* algorithm

3.1 JPS_Theta* algorithm

The traditional A* algorithm has problems in search efficiency and convergence speed [14]. In response to these problems, the traditional A* algorithm was improved, and the main innovations of the JPS_Theta* algorithm is as follows:

Jump point search: The JPS_Theta* algorithm searches for nodes in jumps, and only for nodes that match the search rules are considered, by filtering out some representative jump points. This avoids searching too many redundant nodes and reduces the space and time complexity of the search. The jump point search strategy is divided into two main aspects, one is the cropping of neighboring nodes and the other is the screening of jump points.

(1) Neighbourhood node cropping

Let the current extended node be X and node $P(x)$ be the parent of node X . $d = \{n_0, n_1, ..., n_i\}$ denotes a loop-free path starting from the start node n_0 and ending at the goal node n_i .The symbol $d \setminus X$,indicates that the path d does not contain node X . $Path()$ denotes the path distance function. For example, $Path = (\{P(x), X, n\})$ is the path distance from node $P(x)$, arriving at node n , and passing through node X . $Path = (\{P(x), ..., n\})$ is the path distance from node $P(x)$, arriving at node n , and not passing through node X .

The rules followed above to filter the search nodes are as follows: Discard inferior nodes, i.e., nodes in the gray raster region of the graph, from the parent node $P(x)$ without passing through the current node (X node) and directly arrive at the required cost is less than or equal to the node from the parent node to arrive at the current node (X node), and then to arrive at the inferior node required cost of the node. Consider natural nodes, i.e., nodes in the white raster region of the graph.

For the node to be extended, its neighboring nodes can be categorized into two types of states: without obstacles and with obstacles.

(a) Searching neighboring nodes without obstacles

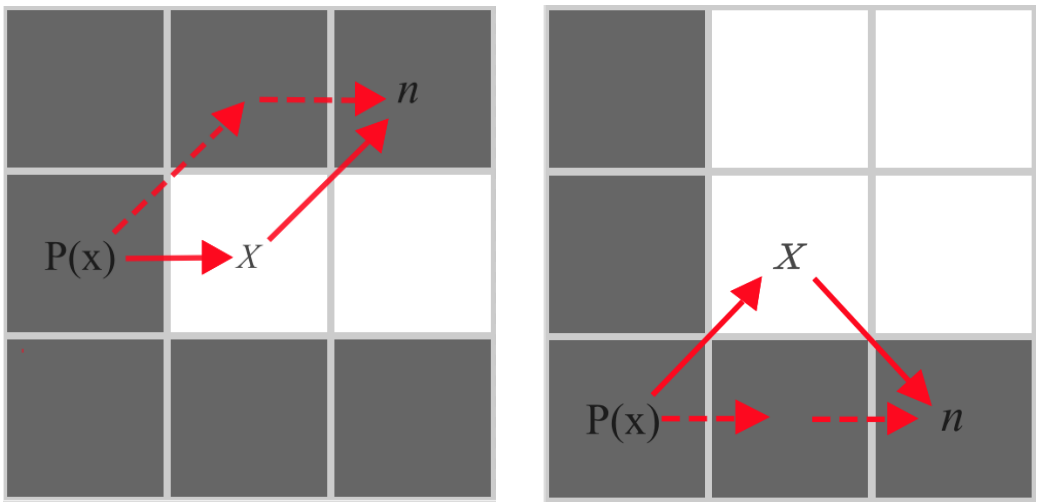
If there is no obstacle in the neighboring nodes of the current to-be-extended node X , as shown in Fig. 1, the gray neighboring nodes around it can be discarded. Since the cost of reaching all gray nodes without going through node X from its parent node $P(x)$ is less than or equal to the cost of going through node X , there is no need to extend it. For example, the shortest path for a node to reach that node can be reached directly by its parent node $P(x)$ without having to go through node X . The cost of the path is the same for the solid and dashed lines as in Fig.1, the gray nodes in the graph are called the inferior nodes of node X, and the white nodes are called the natural neighbor nodes of node X .

For the extension in the linear direction, the node tailoring can be expressed by equation (4):

$$Path(< P(x), ..., n > \setminus X) \le Path(< P(x), X, n >) \tag{4}$$

For the extension in the diagonal direction, which is slightly different from the extension in the linear direction, i.e., the cost of the path without passing through node X needs to be smaller than the cost of passing through node X , the node tailoring can be expressed by equation (5):

$$Path(< P(x),...,n > \backslash X) < Path(< P(x), X, n >) \quad (5)$$

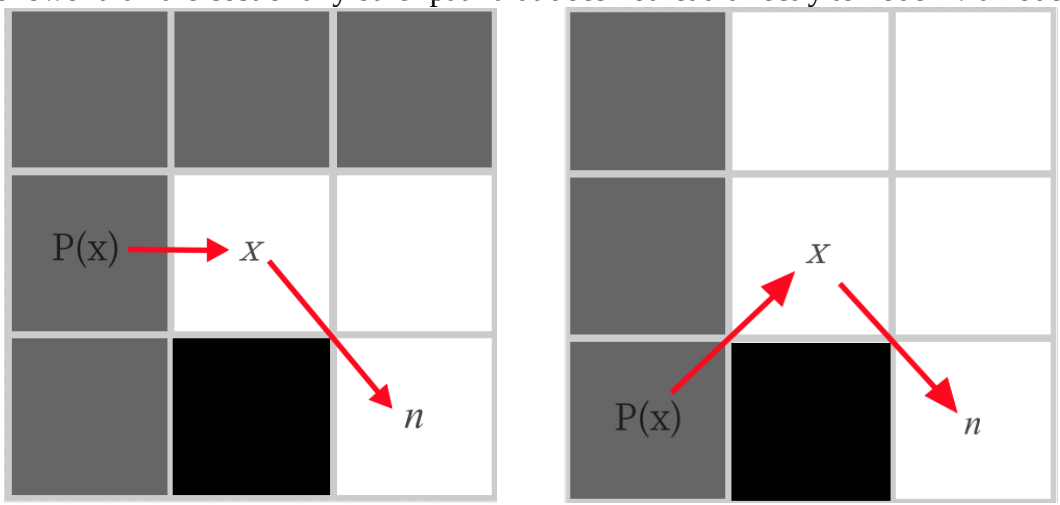


(1) linear direction *(2) diagonal direction*

Figure 1 No obstacle nodes around the node

(b) Neighboring nodes with obstacles

If there is an obstacle in the neighboring node of the current node X to be extended, as shown in Fig. 2, where the black grid indicates the obstacle, for the gray grid, the same does not need to be extended according to the above description. And for node n, node n is considered to be a mandatory neighbour of node X if the path cost from the parent node $P(x)$ to node n via the current node X is lower than the cost of any other path that does not lead directly to node n via node X.



(1) linear direction (2) diagonal direction

Figure 2 Obstacle nodes around the node

Node cropping with obstacles around neighboring nodes can be expressed by equation (6):

$$Path(< P(x), X, n >) < Path(< P(x),...,n > \backslash X) \quad (6)$$

(2) Screening of jump points

As shown in Fig. 3, the algorithm adds only the jump points that meet the search planning to the priority queue during the extended search for paths, and a node is said to be a jump point if it

satisfies the following conditions. The steps are as follows:

Step 1: A node is a jump point if it is either a start or an end point. In Fig. 3, node S is both the start point and the jump point, and node G is both the end point and the jump point.

Step 2: A node is a jump point if there are forced neighbor nodes around the node. According to the above description of neighbor node cropping, there exists a forced neighbor node B at node A in the graph, then node A is a jump point.

Step 3: If the current node is extended diagonally from its parent node and the node is able to reach the jump point during the search along the horizontal or vertical direction, the node is also a jump point. For example, node C in the graph is extended from its parent node A along the diagonal direction, and node C is able to reach the jump point D in the process of searching along the horizontal direction (there exists a forced neighbor node F at node D, so D is a jump point), so node C is also a jump point.

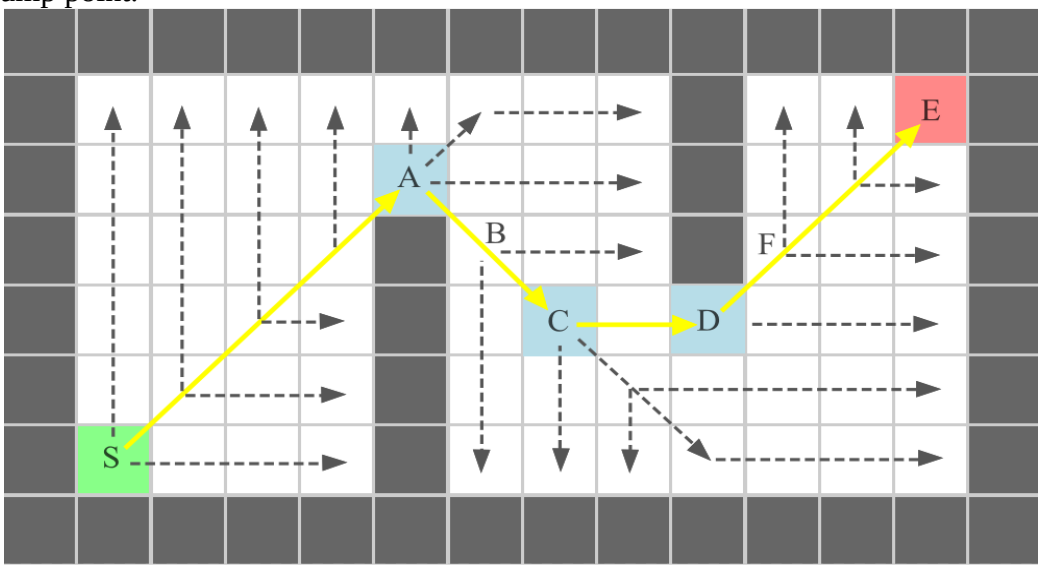


Figure 3 Jump Point Screening Process

Improvement of the heuristic function: Traditional algorithm's do not meet the actual needs, and it is vital to choose the right one, In this paper, which propose an improved heuristic function by combining the advantages of the Chebyshev distance [15], and then add the corresponding weighting factors, The search valuation function $f(n)$ of the weighted algorithm is

$$f(n) = g(n) + w * h(n) \tag{7}$$

In equation (7),

$$w = \begin{cases} 2.0 & h(n) > 12 \\ 0.8 & h(n) \le 12 \end{cases} \tag{8}$$

$$h(n) = \begin{cases} dist_2 & dist_2 > dist_1 \\ dist_1 & others \end{cases} \tag{9}$$

In equation (9),

$$dist_1 = \left| X_i - X_j \right| \tag{10}$$

$$dist_2 = \left| Y_i - Y_j \right| \tag{11}$$

Where, w is the weighting factor and n denotes the current node in the Open List table $g(n)$ denotes the path generation value from the starting point to n . $h(n)$ is the heuristic function and (X_i, Y_i) and (X_j, Y_j) are the coordinates of the nodes respectively.

Theta* algorithm: The core of the Theta* algorithm lies in the idea of checking, for the nodes in the path, whether there exists a direct access to the latter node between the three points without passing through an intermediate node and without passing through an obstacle at the same time. If such a path exists and the total cost of reaching the current node via this path is lower, then the intermediate node is skipped. To operate on the path from Fig. 3, the cases can be categorized into two types of cases: passing through obstacles and not passing through obstacles. This is shown in Fig. 4.

(1) Passing through an obstacle

For the process of starting point S passing through intermediate point A and then to point C, if the path connecting point S to point C is directly connected, this path will pass through the gray obstacle, it is regarded as the case of passing through the obstacle, and this case is not feasible, and the original path is retained.

(2) No passing of obstacles

For the path from point A through the intermediate point C and then to point D, if the path connecting point A to point D between this path does not pass through the obstacle, it is regarded as the case of not passing through the obstacle, and at the same time the distance of the path from point A directly to point D is shorter than that of the previous arrival through the intermediate point, so it will skip the intermediate point C, and pruning operation will be carried out on the path of AC,CD.

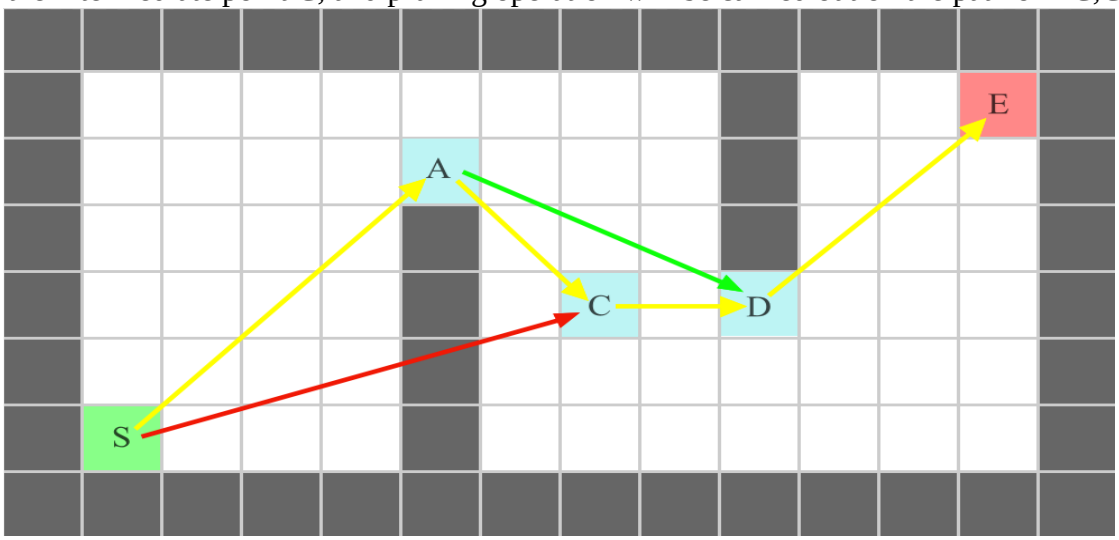


Figure 4 route pruning process

The flow of the JPS Theta* algorithm is shown in Figure 5.

In order to verify the effectiveness of the JPS Theta* algorithm, simulation experiments were conducted to compare it with the conventional A* algorithm in a static environment. The system used for the simulation environment is Windows 11 and the simulation platform is matlab Fig. 6, Fig. 7 and Fig. 8 represent the simulation results of conventional Astar algorithm, JPS algorithm and JPS Theta* algorithm respectively.

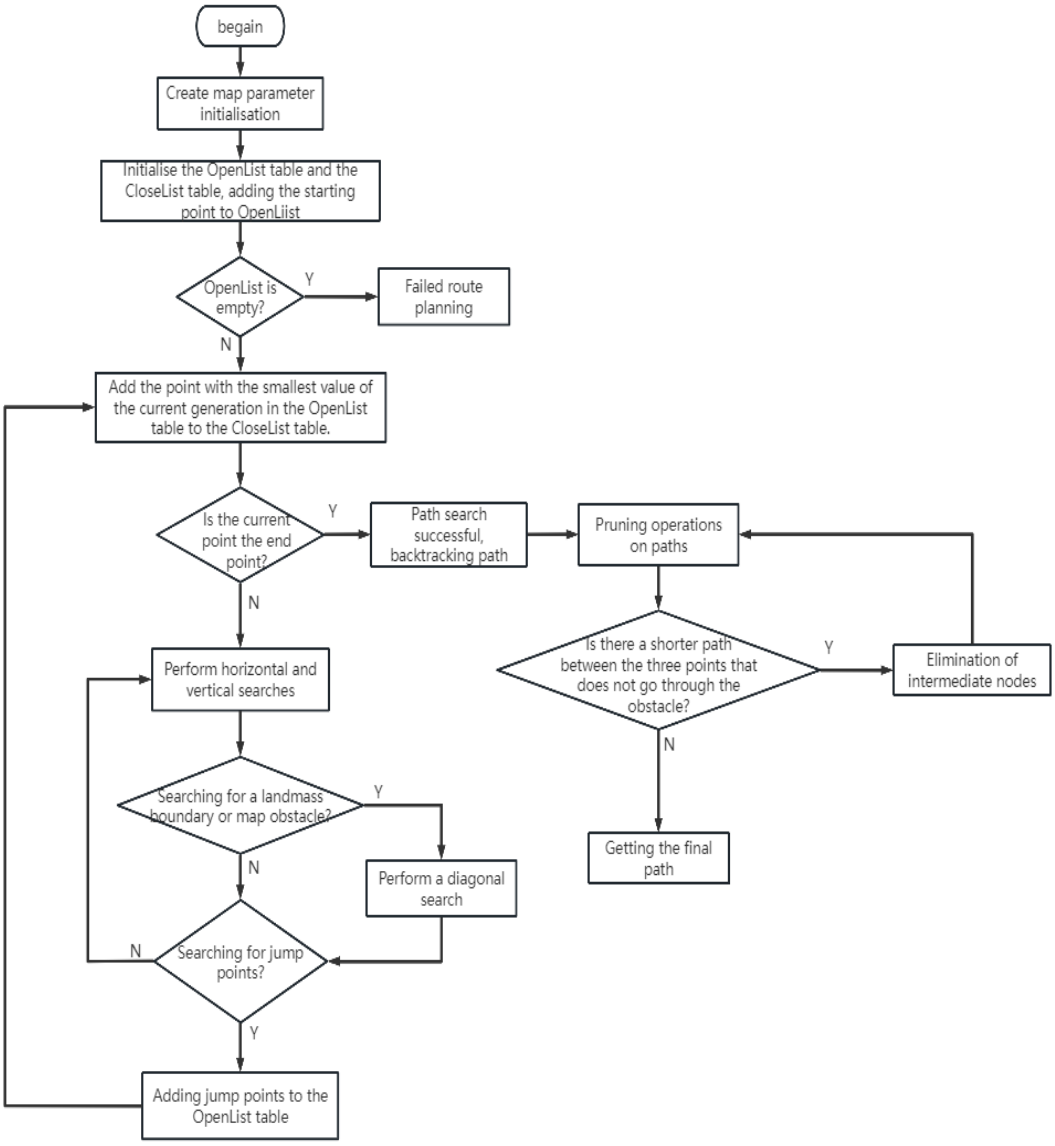


Figure 5 JPS Theta algorithm flow chart*

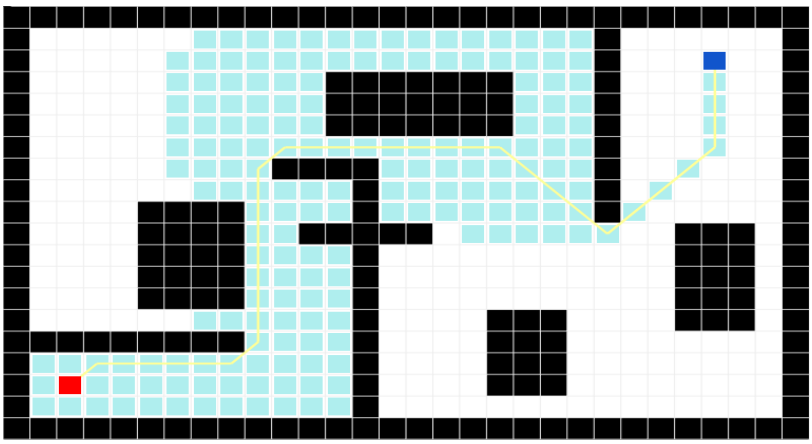


Figure 6 Traditional A algorithm*

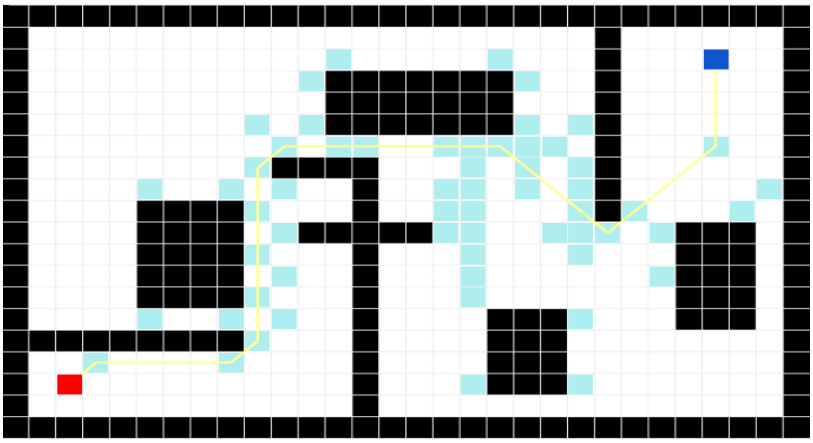


Figure 7 JPS algorithm

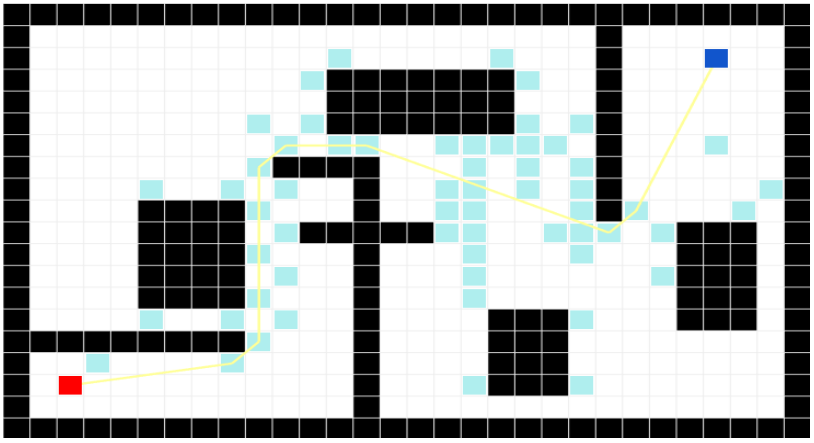


Figure 8 JPS_Theta algorithm*

The red squares are the starting point, the blue squares are the end point, the planned paths are represented by yellow line segments, and the light blue squares are the nodes that have been visited during the path search. A comparison of the algorithm performance metrics is shown in Table 1.

Table 1 Simulation experiment data statistics

Algorithms	Number of turns	Track length/cm	Number of search nodes	Calculation time/s
Traditional A* algorithm	8	40.556	181	2.56
JPS algorithm	8	40.554	59	0.58
JPS_Theta* algorithm	7	38.79	59	0.59

According to the data in Table 1, it can be seen that the JPS_Theta* algorithm shortens the length of the trajectory from 40.55cm to 38.79cm in a simple raster environment map of 20×30 compared with the traditional A* algorithm, The number of search nodes is reduced from 181 to 59,the computation time is reduced from 2.56s to 0.59s. Therefore, combining the four indicators of the algorithm, JPS Theta* search algorithm has more advantages.

3.2. Bezier curve optimization

As can be seen from Figure 8, the path generated by the improved JPS Theta* algorithm has many inflection point problems. Therefore, a cubic Bezier curve optimization algorithm [16] is introduced to remove the concave and convex points in the path to make it smooth and continuous. By formula

$$B(t)=\sum_{i=0}^{n} p_i b_{i,n}(t) \tag{12}$$

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

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

Where,

$$b_{i,n}(t)=\binom{n}{i} t^i (1-t)^{n-i} \tag{14}$$

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

Derivation of t on the basis of Eq. (12) yields:

$$\begin{aligned} B'(t) &= \sum_{i=0}^{n} n(p_{i+1}-p_i) b_{i,n-1}(t) \\ &= \sum_{i=0}^{n-1} n(p_{i+1}-p_i) \binom{n-1}{i} t^i (1-t)^{n-1-i} \end{aligned} \tag{15}$$

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

$$B_i(t)=\sum_{i=0}^{3} p_i b_{i,n}(t) \tag{16}$$

From formula (12, 14, 16), the basis function expression of cubic Bezier curve can be obtained as:

$$\begin{cases} B_{1,3}(t)=(1-t)P_0+tP_1 \\ B_{2,3}(t)=(1-t)^2 P_0+2t(1-t)P_1+t^2 P_2 \\ B_{3,3}(t)=(1-t)^3 P_0+3t(1-t)^2 P_1+3t^2(1-t)P_2+P_3 t^3 \end{cases} \tag{17}$$

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 the fusion of JPS Theta* algorithm and Bessel curve optimization is shown in Fig. 9.The yellow line is the path generated by the JPS Theta* algorithm, and the green curve is the path generated by fusing the Bessel curve optimization. It can be seen that the paths generated by incorporating the Bessel curve optimization algorithm are shorter and smoother.

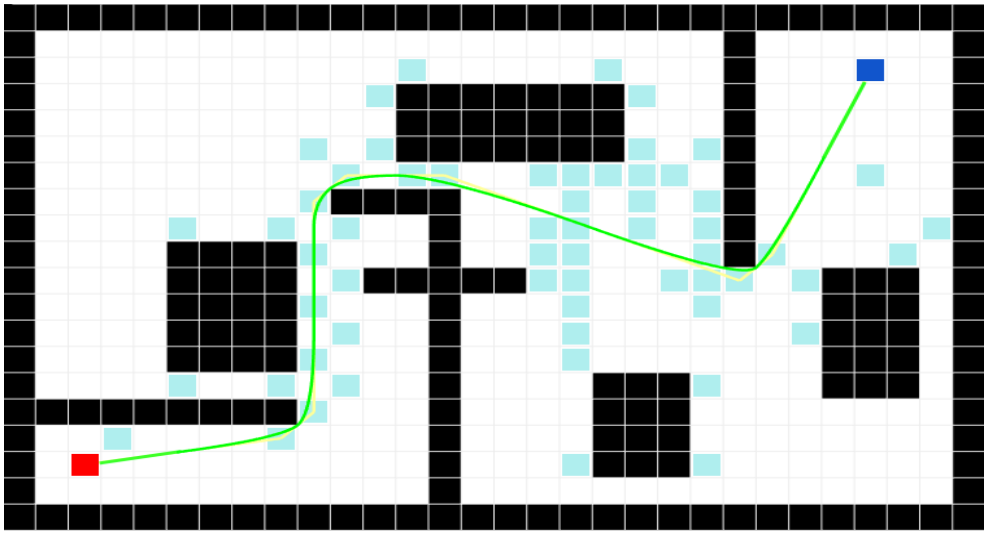


Figure 9 Optimized Fusion of JPS_Theta and Bessel Curve*

4. Mobile Robot Simulation Experiment

The previous section describes the significant performance improvement of the JPS Theta* algorithm over the traditional A* algorithm in the matlab simulation environment. In order to further verify the significant advantages of the JPS Theta* algorithm, the JPS Theta* algorithm will be deployed and experimentally analysed in the ROS simulation environment in this paper.

4.1 Experimental platforms

Figure 10 shows the ROS simulation experiment environment with Ubuntu system 20.04 and ROS version noetic. Build the simulation environment in Gazebo and display it graphically. The simulation cart is equipped with LiDAR and the corresponding environment map is constructed by Cartographer laser SLAM map building algorithm [17] as shown in Figure 11.

4.2 Global path planning experiment

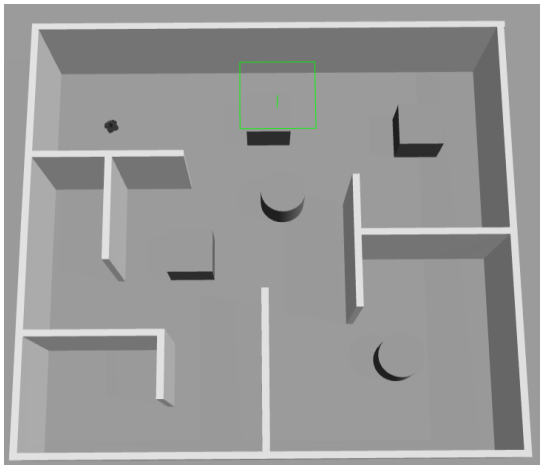


Figure 10 Gazebo simulation environment

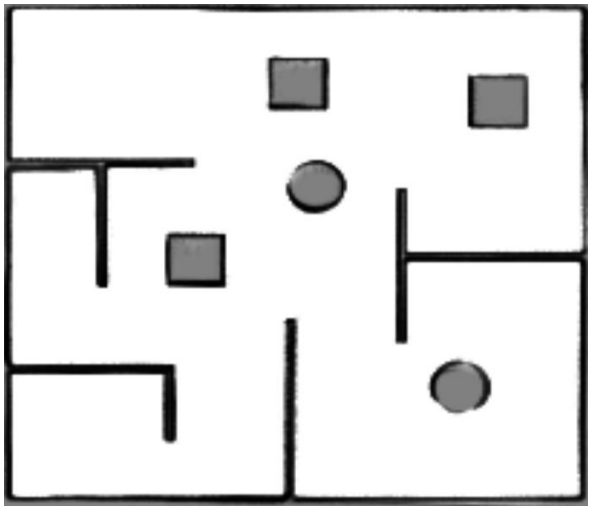


Figure 11 Environmental maps built by Cartographer SLAM

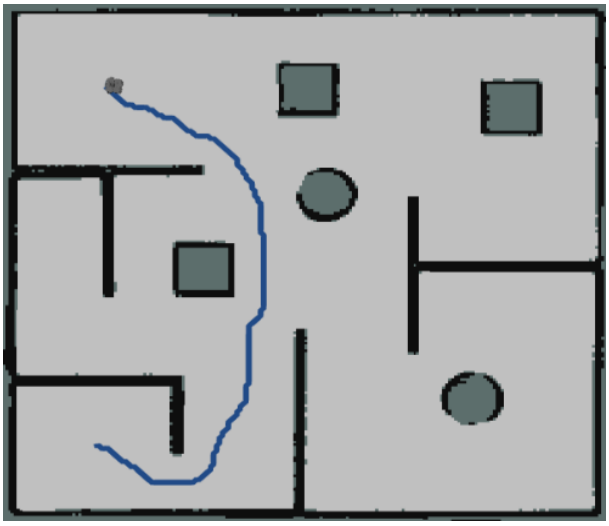


Figure 12 Traditional A algorithm*

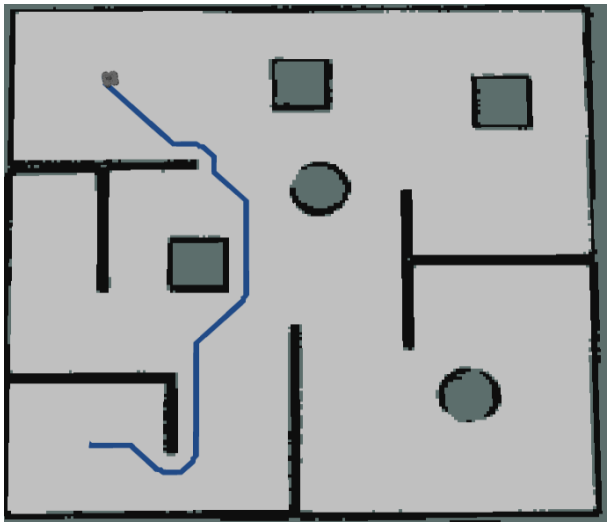


Figure 13 JPS algorithm

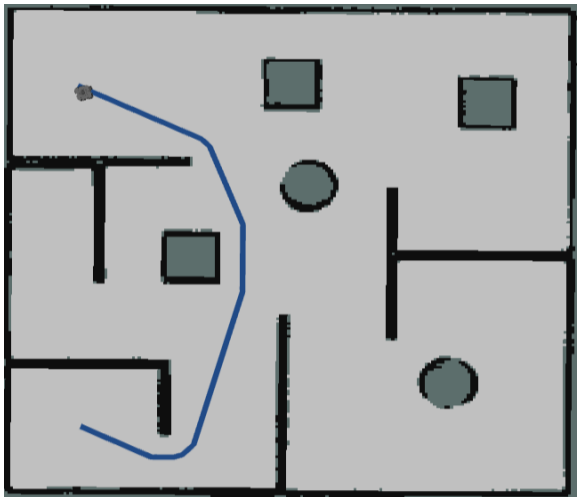


Figure 14 JPS Theta algorithm*

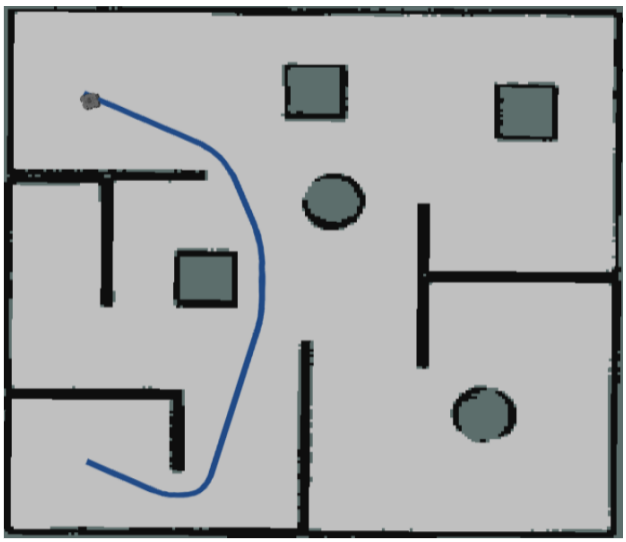


Figure 15 Fusion Bessel curve optimisation algorithm

Table 2 shows the algorithmic performance index of the two algorithms in path planning, it can be seen that the JPS Theta* algorithm is better than the traditional A star algorithm in terms of both the path length and the search time, and the planned paths have smaller twists and turns and better smoothness.

Table 2 Comparison of search results

Algorithms	Trajectory length/m	Plan time /ms
Traditional A*	9.75	4.56
JPS	8.65	2.29
JPS_Theta*	7.43	2.33

4.3 Experiments with local path planning algorithms incorporating TEB

The TEB algorithm is an efficient local path planning method for mobile robots in dynamic

environments. The optimisation process of the TEB algorithm is based on a graph optimisation framework, which is usually implemented using a nonlinear optimisation approach. In the optimisation process of the TEB algorithm, the main concern is to minimise a cost function which consists of several components including path length, obstacle avoidance, time optimisation and satisfying dynamic constraints. The solution is then solved by G2O, which in turn obtains the robot's velocity, angular velocity information and drives the robot.

The path length cost of the robot in the TEB algorithm can be represented by (18):

$$C_{len} = \sum_{i=1}^{N-1} \| p_{i+1} - p_i \| \tag{18}$$

where p_{i+1} and p_i denote the positions of two consecutive bit positions on the path, N is the total number of bit positions on the path, and $\|.\|$ denotes the Euclidean distance.

The obstacle avoidance cost can be expressed by (19)

$$C_{obs} = \sum_{i=1}^{N-1} \sum_{j=1}^{M} \max(0, d_{safe} - \| p_i - o_j \|) \tag{19}$$

where o_j denotes the position of the jth obstacle, M is the total number of obstacles, d_{safe} is the safe distance, and $\max(0, x)$ denotes that a cost is incurred only when the distance between the robot and the obstacle is less than the safe distance.

The time optimisation cost can be expressed by (20):

$$C_{time} = \sum_{i=1}^{N-1} \Delta t_i \tag{20}$$

Where, Δt_i denotes the time interval required to reach two consecutive bit positions on the path.

The dynamic constraint cost can be expressed by (21):

$$C_{dynamic} = \sum_{i=1}^{N-1} (\| v_i - v_{\max} \| + \| a_i + a_{\max} \|) \tag{21}$$

Where, v_i and a_i denote the velocity and acceleration at the ith position, respectively, and $v_{\max}$ and $a_{\max}$ are the maximum allowable values of velocity and acceleration, respectively.

The optimisation objective is to minimise the sum of the above cost functions can be expressed by (22):

$$f(T) = \omega_{len} C_{len} + \omega_{obs} C_{obs} + \omega_{time} C_{time} + \omega_{dynamic} C_{dynamic} \tag{22}$$

Where, ω_{len}, ω_{obs}, ω_{time}, $\omega_{dynamic}$ are the weight coefficients of the respective cost terms.

The optimal sequence of local path points can be obtained by the optimal solution of the G2O solution, which can be expressed by Eq. (23):

$$T^* = \arg_T f(T) \tag{23}$$

The robot navigation process is shown in Figure 16 and is based on the ROS visualization tool Rviz. The gray area is the unknown area not scanned by LiDAR, the black place is the obstacle scanned by the laser, the starting point of the robot is at the center point of the map, the green arrow mark indicates the final goal point location of the robot, and the red line segments with arrows are the paths generated by the fusion TEB algorithm. At moment t_1, the target point information in the

unknown environment is sent down to the robot through Rviz. Moments t_2-t_5 are for the robot to navigate while building the map, and the fusion algorithm continues to plan new local paths as obstacles are continuously scanned. The robot can plan a trajectory that satisfies the relevant constraints of the robot during the travelling process, and is able to plan a feasible trajectory to avoid obstacles in a timely manner after detecting obstacles, and the path is smooth, so as to reach the target point safely and efficiently in the end.

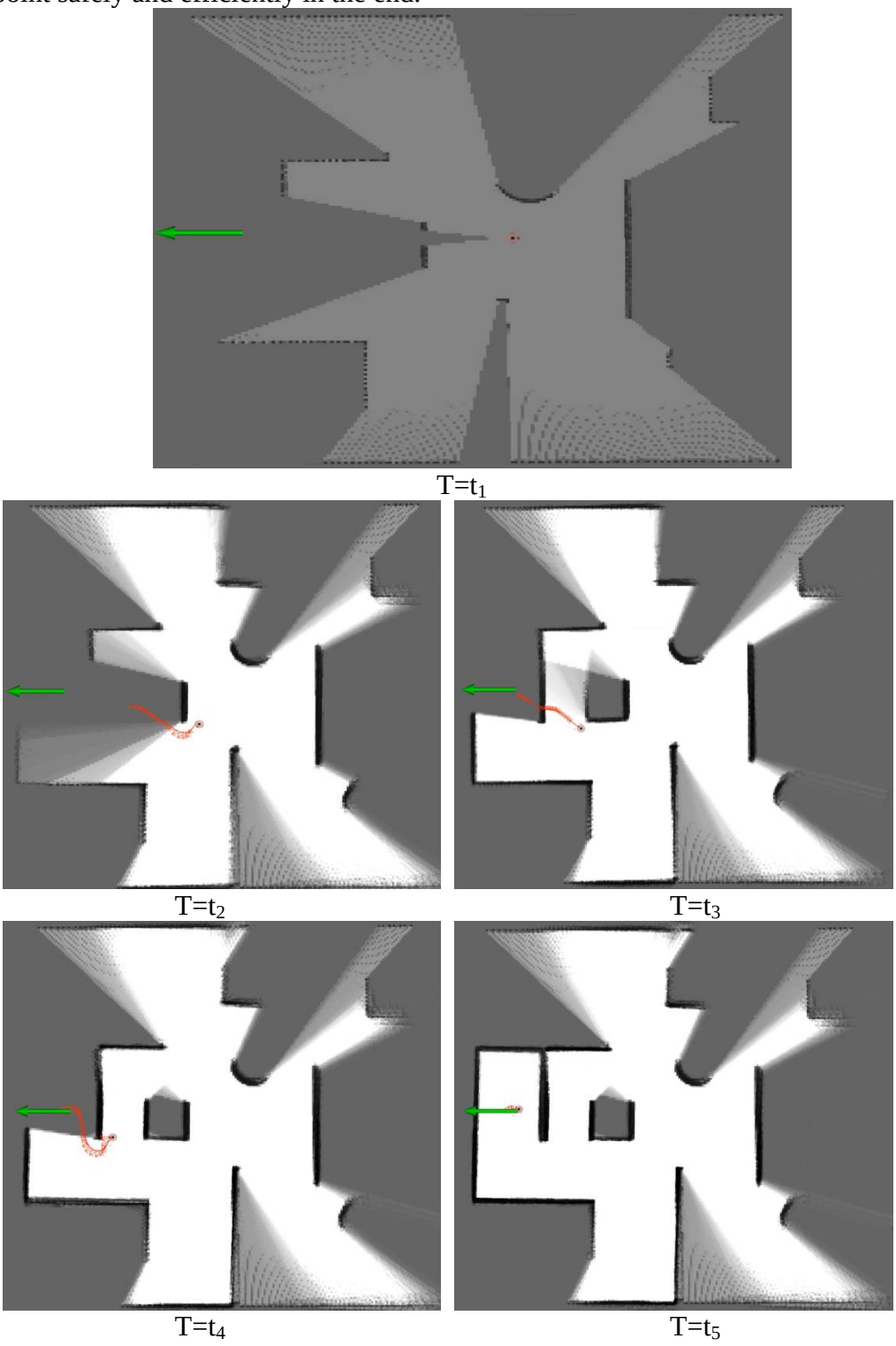


Figure 16. Fusion of TEB algorithms for local path planning

The experimental results proved that coping with the influence of unknown obstacles in the environment on the robot's navigation is solved by using the fusion TEB algorithm, which can meet the practical needs of navigation and obstacle avoidance, and the effect is obvious.

5. Conclusions

The JPS_Theta* algorithm carries out several optimizations on the traditional A* algorithm. Firstly, on the basis of the traditional A*, it proposes a jump-point search strategy and the idea of Theta*, which picks out the nodes that only conform to the search rules, replaces the traditional A* algorithm's evaluation of each neighboring node, and thus reduces the amount of computation, and at the same time, it performs the pruning operation of the paths, which reduces the lengths of the paths, and improves the efficiency of path planning. Aiming at the problem that there are many inflection points in the paths, the generated paths are smoothed and optimized by using 3rd order Bessel curves to remove the bumps in the paths, which reduces the number of iterations for node selection and the inflection points of the paths. Finally, the improved algorithm is fused and experimented in a simulation scenario to verify the feasibility and effectiveness of the proposed algorithm.

Acknowledgements

This paper is supported by Projects of major scientific and technological research of Ningbo City (2020Z065, 2021Z059,2022Z090(2022Z050), 2023Z050(the second batch)), 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).

References

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

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

[3] Yongquan Wang, Robot path planning based on A algorithm and Bessel curve [D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2018.*

[4]Sujit, P. B., et al. "Path planning and optimization: a review." Journal of Mechanical Science and Technology 25.7 (2011): 1553-1563.

[5] R. Cao, Research on robot path planning based on A algorithm and Bessel curve[J]. 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[J]. Robotics Technology and Application, 2017(1): 37-40.

[8] Tao Wang, Peiyi Xu, Robot path planning based on A algorithm[J]. Computer and Modernization, 2015(10): 84-87.*

[9] Zhen Liu, Robot path planning in unknown environment based on fuzzy logic[J]. Robotics Technology and Application, 2015(2):22-25.

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

[11] Schiller, Ulf, et al. "Smooth paths for mobile robots using Bézier curves." Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003). IEEE, 2003.

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

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

[14] Qi Li. Robot path planning in unknown environment based on A algorithm[J]. Robotics and Applications, 2018(1): 32-35.*

[15]Nash, Alex, and Simon Chapuis. "Dynamic Path Smoothing for Mobile Robots using A Path Planning. "Robotics and Autonomous Systems 123 (2020). 103340.*

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

[17] Hess, Wolfgang, Damon Kohler, Holger Rapp, and Daniel Andor. "Real-time loop closure in 2D LIDAR SLAM." in 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1271-1278. ieee, 2016.