Research on the Global Path Planning of Mobile Robot Based on Improved A* Algorithm

Guoliang Tan*

School of Automotive and Transportation Engineering, Hefei University of Technology, Hefei, China

1903049204@qq.com

Abstract. Aiming at the problems of many redundant inflection points and low smoothness in the path planning field of the traditional A* algorithm, an improved A* algorithm based on a strong angle change penalty and the smooth path is proposed. Firstly, by introducing the penalty function of angle change cost to optimize the node evaluation function, redundant inflection points in the path can be effectively reduced. Finally, the fourth-order Runge-Kutta algorithm is used to smooth the path, which comprehensively considers the attraction of the original path, the smoothing force of the path movement, and the repulsive force of the obstacle, so as to improve the smoothing quality of the path. With the help of MATLAB, the simulation experiment is carried out, and the effect of path planning is compared and analyzed under the influence of different cost penalty scaling coefficient k, and the optimal k value is determined to be 2. In the experiments of diverse raster map environments, the improved A* algorithm has excellent performance in optimizing the path length and reducing the number of nodes, especially in random and complex environments, which are optimized by 3.68% and 37.50% respectively. The simulation results meet the smooth motion requirements of the actual environment, and the global path planning performance of the A* algorithm has been significantly improved.

Keywords: A* algorithm; path planning; the cost of angle change; fourth-order Runge-Kutta algorithm.

1. Introduction

At present, the key challenge of autonomous navigation technology for mobile robots lies in the global path-planning ability in complex environments, which directly determines the operational efficiency and operational stability of robot systems [1]. As a widely used global path planning method, the A* algorithm has become a basic algorithm in this field because of its clear mathematical principle, simple implementation process, and good path-finding performance in static scenes. However, when faced with increasingly complex practical application scenarios, the traditional A* algorithm has exposed some inherent defects, including insufficient dynamic environment response-ability and redundant inflection points in the generation path, which seriously restrict the practical value of the algorithm in real mobile robot systems. In this regard, Zhao Yuling et al. [2] combined the node position information of graph theory problems and grid map nodes, greatly reducing the number of grid nodes and path search time. Zhao Wei et al. [3] introduced a double-layer polyline optimization strategy, which further reduced the turning times of the A* algorithm planning path through the double-layer optimization method of node slope and path extension. Lin Tanqi et al. [4] proposed to introduce obstacle avoidance cost and steering cost, and to eliminate the redundant path of the original A* algorithm by iteratively using the A* algorithm to generate the path in reverse segments, thus optimizing the steering trajectory. Wang Yunliang et al. [5] optimized the traditional A* algorithm by combining the second-order Bessel curve, which effectively improved the smoothness of the path.

The above methods all optimize the A* algorithm to some extent. In this paper, by introducing the innovative angle change cost penalty function and the fourth-order Runge-Kutta algorithm enhancement strategy, this study is committed to building the optimal or suboptimal trajectory for the robot from the initial position to the target area, significantly improving its path planning performance under complex working conditions, thus providing algorithmic support for the high-performance operation of the mobile robot system.

2. Introduction to Basic A* Algorithm

2.1 The basic principle of the A* algorithm

A* algorithm is a global static search algorithm based on the Dijkstra and BFS algorithm [6]. The path-finding process of the algorithm is based on a heuristic search strategy, and the search direction is determined by evaluating the cost function *f (n)* of each node. In the search process, the algorithm maintains two lists: an Open List and a Closed List. The open list stores the nodes to be evaluated, and the closed list stores the evaluated nodes [7]. The algorithm selects the node with the smallest value of *f (n)* from the open list to expand until the target node is found or the open list is empty. If the target node is found, the path is reconstructed by tracing back to the parent node, and finally, the target path is planned. The schematic diagram of the basic A* algorithm pathfinding process is shown in Figure 1.

The cost function of the basic A* algorithm is as follows:

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

In the above function: *g (n)* represents the actual cost from the starting point to the node *n*, and *h (n)* is the estimated cost from the node *n* to the target point. This paper uses Euclidean distance as a heuristic function to measure the estimated cost of two points, namely:

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2}$$

In the above function: x_1 is the abscissa of the current node y_1 is the ordinate of the current node; x_2 is the abscissa of the target node, and y_2 is the ordinate of the target node.

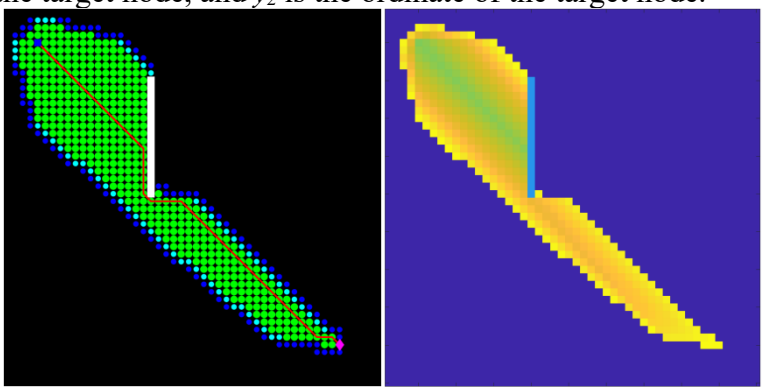


Figure 1 The Schematic Diagram of the Path-finding Process of Basic A* Algorithm

2.2 Disadvantages of A* algorithm

1) Unable to handle dynamic environment. The basic A* algorithm is based on the static map for path planning and assumes that the map's state is unchanged during the search process. However, when new obstacles in the environment or the target position change, it is difficult for the basic A* algorithm to adjust the path in time, resulting in the planned path no longer being the optimal solution.

2) Weak angle change requirements. The basic A* algorithm usually only considers the movement from the current node to eight fixed directions, that is, up, down, left, right, left, right, left, and right, and cannot consider the movement in any other directions. At the same time, the basic A* algorithm sets the horizontal or vertical movement cost to 1 and the diagonal movement cost to $\sqrt{2}$. This simplified cost calculation method will lead it to give priority to the jagged combination path of multiple discrete directions to approach the target direction, so as to avoid the higher cost of diagonal movement, which will lead to more unnecessary inflection points in the path.

3) The planned path is not smooth enough, and the smoothness is poor in complex maps. When the basic A* algorithm is running, only the shortest path is considered, and the generated path is composed of grid points, which leads to more inflection points and frequent turns. This will make the robot or autopilot navigation in the real scene unstable, increase mechanical wear and energy consumption, and may even affect the safety performance.

3. Global Path Planning Based on Improved A* Algorithm

3.1 The introduction of angle change cost penalty function

The traditional A* algorithm only considers two kinds of moving costs in an 8-connected grid. Although this simplified model is computationally efficient, it does not pay attention to direction consistency, and it will frequently change the moving direction, resulting in a jagged path that does not meet the actual movement requirements.

For the above shortcomings, the angle cost penalty function *angleCost* (*n)* is introduced, and the node evaluation function *f* (*n)* is optimized from the bottom logic of path planning. Additional penalties are imposed on the movement with large direction changes, thus encouraging the algorithm to choose the path with the same direction and effectively solving the redundant inflection point problem in the traditional A* path. The cost function of the improved A* algorithm is as follows:

$$F(n) = g(n) + h(n) + k \times angleCost(n) \tag{3}$$

The specific steps to improve the strategy are as follows:

Step 1: Detecting the modulus length of the direction vector of the preamble movement. When it is 0, it means that the current node is the starting point of the path, and there is no preamble movement direction, so there is no direction change. Therefore, the cost of angle change is directly set to 0, and the subsequent calculation is skipped. This treatment ensures that the starting point is not punished by unreasonable direction.

Step 2: Calculating the dot product and modulus length of the previous moving direction vector (that is, the moving direction vector between the parent node and its parent node) and the current moving direction vector (that is, the moving direction vector between the current node and its parent node). Among them, the dot product results reflect the "consistency" of the two vectors in the direction: when the dot product value is 1, the two vectors are in the same direction and there is no direction change; when the value is -1, the two vectors are opposite and the direction changes the most; when the value is 0, the two vectors are vertical and the direction changes significantly. The vector modulus length is used for subsequent normalization to ensure that the angle calculation is not affected by the vector size.

Step 3: According to the vector dot product formula, the cosine value of the included angle between the vectors in the above two directions is derived, and the calculation formula is as follows:

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|a| \times |b|} \tag{4}$$

Where: $\vec{a} \cdot \vec{b}$ is the dot product of the above two direction vectors, and $|a| \times |b|$ is the module length of the above two direction vectors.

Step 4: Calculating the angle change cost through the angle change cost penalty function *angleCost (n)*, and the specific formula is as follows:

$$angleCost(n) = \begin{cases} 2e^{-\cos\theta} - 1 & 0 < \cos\theta \le 1 \\ 1 - \cos\theta & -1 \le \cos\theta \le 0 \end{cases} \tag{5}$$

This formula reflects the magnitude of the change angle of the direction vector. The greater the angle difference, the greater the value of *angleCost (n)*, and the higher the additional cost when moving. Meanwhile, the cost penalty scaling coefficient *k* plays a role in adjusting the penalty weight. Through this mechanism, the algorithm will tend to choose the path branch with little direction change in the search process and reduce the sharp inflection point in the path.

3.2 Path smoothing optimization strategy based on fourth-order Runge-Kutta algorithm

In the aspect of path smoothing, the Bezier curve and spline curve generate smooth curves by fitting path points, but these methods need to adjust complex parameters of path points and are difficult to deal with obstacles. The path smoothing method based on the physical model compares

the path optimization to the forced motion in the physical system and adjusts the path by taking the resultant force calculated by the force field model as the definition of the first-order motion derivative. However, the traditional physical model often adopts the relatively simple Euler method in numerical solution, which has problems such as insufficient accuracy and slow convergence. Therefore, this paper adopts the classic fourth-order Runge-Kutta algorithm for numerical solution, and its mathematical principle formula is as follows:

$$\begin{cases} y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_i,\ y_i) \\ k_2 = f\left(x_i + \frac{h}{2},\ y_i + \frac{hk_1}{2}\right) \\ k_3 = f\left(x_i + \frac{h}{2},\ y_i + \frac{hk_2}{2}\right) \\ k_4 = f(x_i + h, y_i + hk_3) \end{cases} \tag{6}$$

In the interval [x_i , x_{i+1}], four different points can be taken to construct a fourth-order Runge-Kutta scheme, and its numerical integration diagram is shown in Figure 2 [8]. This scheme is often used to solve ordinary differential equations and has the following advantages: ① the accuracy is high, reaching the fourth-order accuracy O (h^5); (2) only the first derivative needs to be known, and there is no need to explicitly define or calculate other higher derivatives; ③ y_{i+1} can be calculated only by giving y_i (in this paper, it is assumed that $y_i = 1$ is used to solve the road strength planning problem), so it can start itself [9].

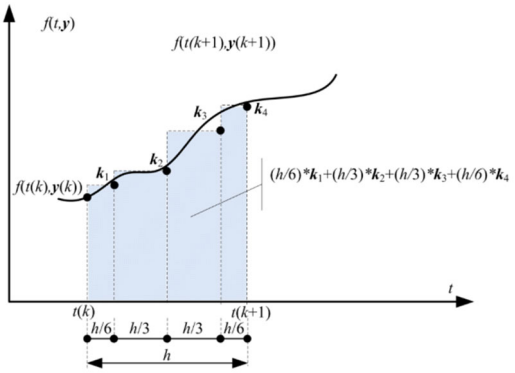


Figure 2 Schematic Diagram of Numerical Integration of Fourth-order Runge-Kutta Algorithm

The optimization algorithm first uses the improved A * algorithm to obtain the initial path and ends if the path is empty. Otherwise, the fourth-order Runge-Kutta algorithm is applied to the initial path for path smooth optimization. As for the definition of the first derivative, in classical mechanics, the motion of the object follows Newton's second law, and the model is simplified in the path smoothing optimization problem. Assuming that the moving object is a unit mass, the position update of the path point is regarded as a dynamic system, and the resultant force is directly regarded as the first derivative of the position with respect to time, and the differential equation of the position with respect to time is constructed. The resultant force is divided into the following three parts:

1) Original route attraction:

$$\mathrm{F_{attract}} = \alpha \times \left(\mathrm{P_{original}} - \mathrm{P_{current}}\right) \tag{7}$$

Where: $P_{original}$ is the location of the original path point; $P_{current}$ is the current path point position; α is the attraction scaling factor, which is used to control the closeness between the smooth path and the original path (α = 0.1 in the simulation experiment in this paper). The force is proportional to the distance from the current position to the corresponding point of the original path; The direction of the force points to the corresponding point of the original path; The function of force is to make the smoothed path close to the original path to keep the original key features, and at the same time prevent the path from deviating from the feasible area due to excessive smoothing.

2) Smooth force of path movement:

$$F_{smooth} = \beta \times \left(P_{prev} + P_{next} - 2P_{current}\right) \quad (8)$$

Where: P_{prev} and P_{next} are the positions of adjacent points respectively; β is the scaling factor of smoothing force, which controls the smoothness of the global path (β = 0.3 in the simulation experiment in this paper). Based on the spring particle model, this force makes the path point move to the midpoint of its adjacent points, reducing the "sawtooth" phenomenon of the path and reducing the total length of the path.

3) Repulsive force of obstacle:

$$F_{repel} = \gamma \times \sum \left[\left(\frac{1}{d} - \frac{1}{r}\right) \times \frac{\vec{d}}{d^2}\right] \quad (9)$$

Where: d is the distance from the current route point to the obstacle, r is the detection radius, $\vec{d}$ is the unit vector of the obstacle direction, and γ is the repulsive force scaling factor (γ = 0.6 in the simulation experiment in this paper). This force automatically avoids the obstacle area by using the repulsive potential field generated by the distance from the obstacle. At this time, the force is inversely proportional to the distance, which is effective within the detection radius (when the distance approaches 0, the force tends to infinity to prevent collision; When the distance is greater than R, the force is 0), which ensures that the smoothed path will not pass through obstacles and keep the safety of the planned path.

After calculating the resultant force of the above three forces, the fourth-order Runge-Kutta algorithm is used to solve the motion equation of the path point by numerical integration. Before solving the fitting path, it is necessary to set the maximum number of iterations and the convergence threshold (100 and 0.01 respectively in the simulation experiment in this paper) and update the position of the path point through continuous iteration until the path change is less than the threshold or reaches the maximum number of iterations. Finally, the smoothed path is checked to ensure that the points on the path are not in the obstacle area. If there are points on the obstacle, they are corrected by linear interpolation.

4. Simulation Verification Experiment and Data Result Analysis of the Improved A* Algorithm

MATLAB R2023a is used to carry out simulation and verification experiments on global path planning based on an improved A* algorithm, and the data results are analyzed. The map environment is a grid map of a random obstacle environment with a size of 20 × 20, with black squares indicating obstacles, green circles indicating the starting point position, purple intersections indicating the target point position, and solid lines in the map indicating the generated global planning path.

4.1 Selection of optimal K value

In the improved A* algorithm, the effect of angle change cost highly depends on the choice of cost penalty scaling coefficient k. When the cost penalty scaling coefficient k takes different values, the simulation results are shown in Figure 3, in which the solid green line on the map represents the generated global planning path, and the proportion of obstacles in the map environment is 30%.

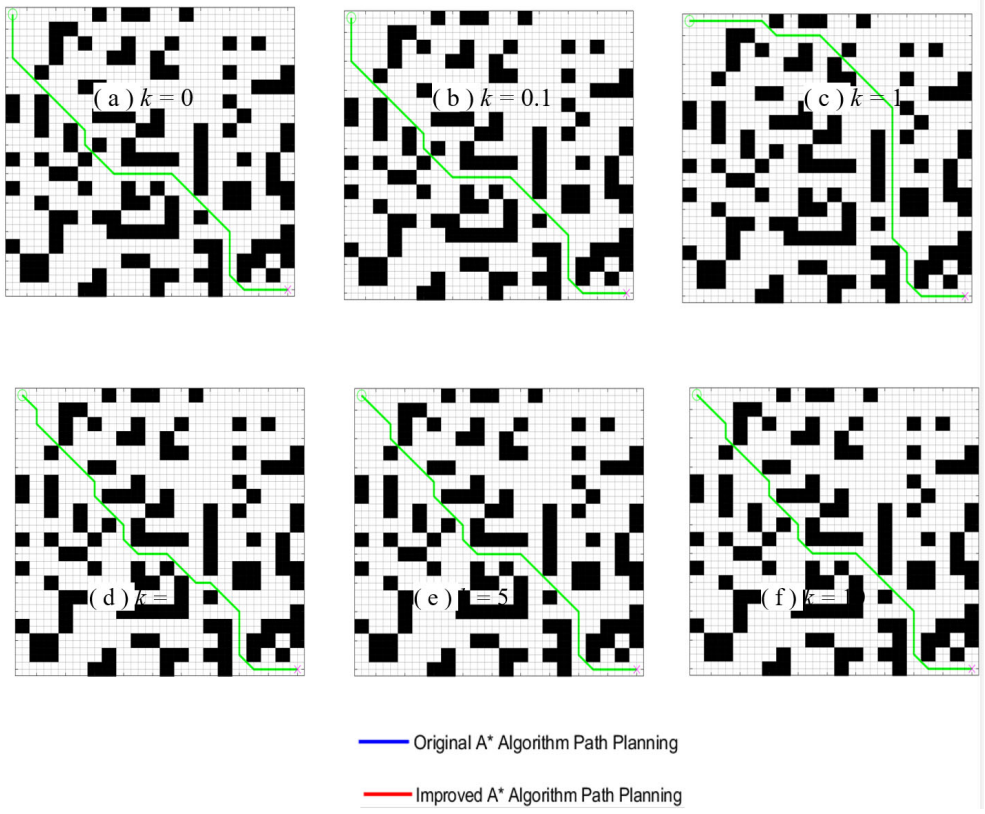


Figure 3 Simulation Results of Path Planning Based on Changing K Value

Table 1 Comparative Data Results of Different K Values

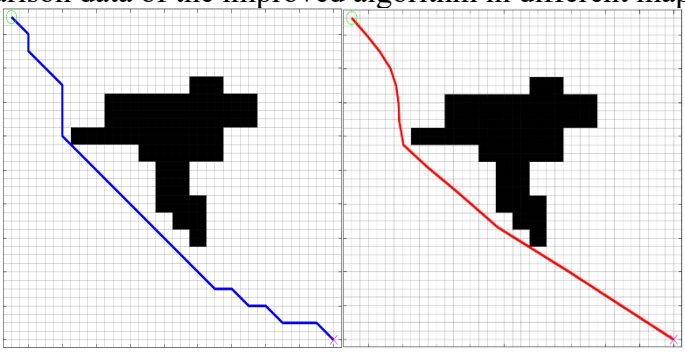
k	Path length	The number of path nodes	The number of path inflection points	Average search time/ms
0	30.38	24	13	28.169
0.1	30.38	24	11	29.641
one	30.38	24	11	43.452
2	30.97	25	8	44.801
five	30.97	25	8	54.925
10	33.31	29	8	55.870

When k takes different values, the path length, the number of path nodes, the number of path inflection points, and the average search time planned by the improved A* algorithm are shown in Table 1 respectively. The simulation data show that when the weight k is too small, the penalty for direction change will be insufficient and the path smoothing effect will not be obvious; When the weight k is too large, it will make the improved strategy excessively pursue the consistency of direction and choose a longer detour path, which will lead to the path deviating greatly from the optimal path and increase the path length, the number of nodes and the search running time. Therefore, considering the smoothness of the path, the shortest path, and the search time, $k = 2$ is chosen as the optimal cost penalty scaling factor for further simulation experiments.

4.2 Simulation experiment analysis of global path planning

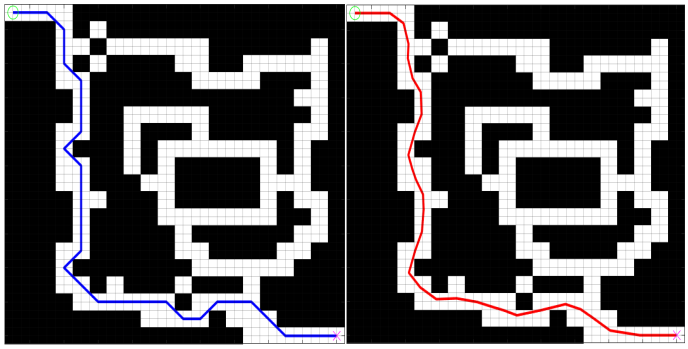
In order to better meet the actual movement requirements and deeply analyze the performance of the improved algorithm, grid maps including single obstacle environment, narrow passage environment, and random obstacle environment (obstacles account for 20%) are set up in the simulation environment and experimental analysis is carried out. The generated planning simulation results are shown in Figures 4, 5, and 6, in which the blue solid line on the map represents the

unimproved planning path and the red solid line represents the improved planning path. Table 2 gives the experimental comparison data of the improved algorithm in different map environments.



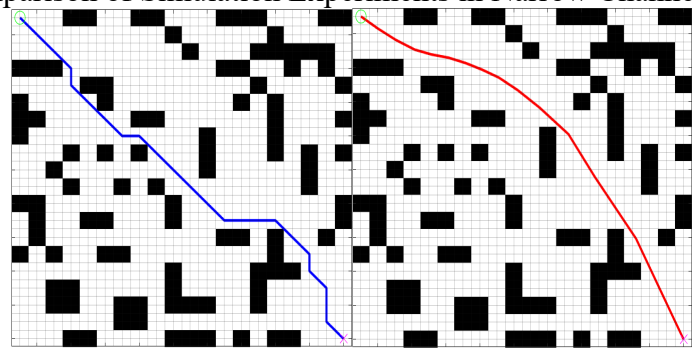
(a) Improved front path (b) Improved rear path

Figure 4 Comparison of Single Obstacle Environment Simulation Experiments



(a) Improved front path (b) Improved rear path

Figure 5 Comparison of Simulation Experiments in Narrow Channel Environment



(a) Improved front path (b) Improved rear path

Figure 6 Comparison of Simulation Experiments in Random Complex Environment

Table 2 Data Comparison Between Basic A * Algorithm and Improved A * Algorithm

Map environment	Path length		Optimization rate	The number of path nodes		Optimization rate
	Before improvement	After improvement		Before improvement	After improvement	
Single obstacle environment	29.21	28.08	3.87 %	24	13	45.83 %
Narrow channel environment	37.56	35.57	5.30 %	34	32	5.88 %

Random complex environment	29.21	28.14	3.68 %	24	15	37.50 %

The results of three groups of simulation experiments show that under the same grid map, the optimization rate of the improved A * algorithm in a narrow channel environment is higher than that in a random complex environment, but the optimization rate of the number of path nodes is significantly lower than that in other environmental maps. At the same time, the improved algorithm optimizes the path length and the number of inflection points by 3.68% and 37.50%, respectively in the random and complex environment, and the smoothness of the path trajectory is improved, which is more in line with the movement requirements of the actual environment and has obvious advantages that the traditional A* algorithm cannot match.

5. Conclusion

Aiming at the problems of many redundant inflection points and low smoothness existing in the traditional A* algorithm in the global path planning of mobile robots, this paper puts forward an improved strategy by introducing the penalty function of angle change cost and the fourth-order Runge-Kutta algorithm. By optimizing the node evaluation function, the path inflection point is reduced, and the smoothness and safety of the path are improved based on the resultant force calculation of the physical model and high-precision numerical solution. Simulation results show that when the cost penalty scaling coefficient k = 2, the improved algorithm shows better performance in different raster map environments, with the optimization rate of path length reaching 3.68% ~ 5.30%, the maximum optimization rate of node number reaching 45.83%, and the smoothness significantly improved. In addition, the limitations of this study are that the coverage of dynamic environment is insufficient and the computational efficiency of large-scale scenes needs to be optimized. In the future, it is planned to combine real-time sensing data to enhance dynamic adaptability and explore the integration with other algorithms to improve the planning efficiency in complex scenes.

References

[1] Hu Jingbo, Chen Dingfang, Wu Junfeng, et al. Research on autonomous obstacle avoidance of mobile robots based on improved fuzzy algorithm [J]. Automation and Instrument, 2018, 33 (6): 48-52.

[2] Zhao Yuling, Shao Tengwu, Soviet Union. An improved A-star grid path planning algorithm based on graph theory nodes [J]. Journal of Langfang Normal University, 2023, 23 (4): 34-38.

[3] Zhaowei, Wu Ziying. Bi-level optimization A* algorithm and dynamic path planning with dynamic window method [J]. Computer Engineering and Application, 2021, 57 (22): 295-303.

[4] Lin Tanqi, Li Cunrong, Liu Shuaiwen. Path planning of inland lake unmanned ship based on improved A* algorithm [J]. Journal of Dalian Maritime University, 2024, 50 (3): 87-96.

[5] Wang Yunliang, Zhang Sai, Wu Yanjuan. Research on Path Planning Algorithm Based on Improved A* Smoothness [J]. Computer Applications and Software, 2025, 42 (1): 258-263.

[6] Huang Chonghua. Research on path planning and dynamic obstacle avoidance decisions based on improved A* algorithm [D]. Chongqing: Chongqing University of Technology, 2024: 1-66.

[7] Sun Shuguang, Sun Tao. Research on UAV Path Planning Based on Fusion A* Algorithm [J]. Electronic Measurement Technology, 2022,45 (9): 82-91.

[8] Zhihu. Matlab program for solving ordinary differential equations by Runge-Kutta method [EB/OL]. (2021-11-14) [2025-05-09].https: //zhuanlan.zhihu.com/p/427409568? s_r=0.

[9] Li Lin, Jin Xianji. Numerical calculation method (MATLAB language version) [M]. Guangzhou: Sun Yat-sen University Press, 2006: 97-10