International Journal for Housing Science and Its Applications
Publish August 6, 2025. Volume 46, Issue 3 Pages 7473-7482

https://doi.org/10.70517/ijhsa463641

Research on UAV Path Planning Based on Improved Dijkstra's Algorithm

Yuanyao Zhang[1], Hongji Liang[1] and Junli Li[1,*]
[1] Information Engineering and Automation School of Kunming University of Science and Technology, Kunming, Yunnan, 650500, China
Corresponding authors: (e-mail: a20222104028@stu.kust.edu.cn).

Abstract As a core technology in the field of modern automation, UAV path planning plays an important role in many fields such as military reconnaissance, disaster rescue, and environmental monitoring. Although traditional path planning algorithms such as Dijkstra's algorithm can guarantee to find the shortest path, there are problems such as large computation volume, high memory consumption, and easy to fall into local optimization in large-scale map applications. Aiming at the problems of traditional Dijkstra algorithm in UAV path planning, this paper proposes an improved path planning algorithm based on Dijkstra-PSO fusion. The method combines the precise search characteristics of Dijkstra algorithm and the fast convergence ability of particle swarm algorithm, and avoids the algorithm from falling into local optimization by dynamically changing the inertia weight strategy, adaptively adjusting the learning factor, and improving the speed updating mechanism; it constructs a composite fitness function containing the path length and the corner of the turn, and introduces the three times B-spline interpolation for the trajectory smoothing process. Simulation results show that the improved algorithm reduces the number of search nodes by 62.5% to 91.67%, the path length by 9.39% to 16.57%, the turning angle by 90.04% to 92.95%, and the computation consumption time by 85.29% to 93.27% in maps of different sizes from 15×15 to 100×100. Compared with the A* algorithm, the path nodes are reduced by 60% to 91.3% and the search time is reduced by 81.48% to 92.35%. The algorithm in this paper significantly improves the computational efficiency while guaranteeing the path quality, and provides a new solution for real-time path planning of UAVs in complex environments.

Index Terms Dijkstra's algorithm, Particle Swarm Optimization, UAV, Path Planning, Triple B-Spline Interpolation, Fitness Function

I. Introduction

A UAV is an unmanned aerial vehicle that is controlled using an on-board decision control program in conjunction with ground control commands [1]. Early UAVs were mainly used in the military field, including intelligence reconnaissance, line tracking, and strategic destruction [2]. After entering the 21st century, UAVs have developed in the direction of miniaturization, stealth, intelligence, and diversified system configurations, and their combat missions have been further expanded [3], [4]. At the same time, due to the advantages of UAVs such as low production cost, stronger survivability, and better usability, their use has become increasingly widespread, and they have been gradually expanded from the purely military field to the civil field [5]-[7]. Due to the relative complexity of the urban geographic environment in the current civil real flight environment, the timeliness and reliability of UAV path planning are more demanding [8], [9]. Therefore, optimal path planning for UAVs in geographic environments becomes a key prerequisite for UAVs to expand in the civil field.

The path planning problem, in simple terms, is to plan a collision-free optimal path from the starting point to the target point for UAVs in the environmental space according to the specific flight mission [10]. Only a planned path that fully considers the flight energy consumption, relevant environmental constraints, and the UAV's own relevant performance constraints can provide a prerequisite guarantee for safe and effective UAV flight [11], [12]. Dijkstra's algorithm, as a typical 3D raster path planning method, is more intuitive for the expression of the geographic environment, and it can also be used to divide the geographic space into a number of levels with different granularity of rasters according to different needs, and it has significant application advantages in data storage and organization [13]-[16]. However, the traditional Dijkstra's algorithm is unable to deal with dynamically changing environments as well as lack of understanding of UAV's motion characteristics to ensure real safety [17], [18]. Based on this background, it is of great significance to study the improved Dijkstra's algorithm with high efficiency and low cost and at the same time with safety to realize the autonomous decision-making flight of UAVs.

As an important part of modern aviation technology, unmanned aerial vehicles (UAVs) show a broad application prospect in many fields such as military reconnaissance, civil monitoring, logistics and distribution. As the core of autonomous UAV flight, the performance of path planning technology directly affects the efficiency and safety of mission execution. Path planning is essentially to find the optimal path from the starting point to the end point for UAVs under the given constraints, and the problem needs to consider several evaluation indexes, including rationality, completeness, optimality, real-time and adaptability. Among the traditional path planning algorithms, Dijkstra's algorithm is famous for its ability to guarantee finding the global optimal solution, but in practical applications, it has defects such as a large number of search nodes, high computational complexity, and high memory consumption, which are more prominent, especially in large-scale map environments. Particle swarm optimization algorithm, as a heuristic algorithm based on group intelligence, has the advantages of fast convergence speed and simple parameter setting, but it is easy to fall into the local optimum, which affects the quality of the final solution. In the current research, it is often difficult for a single algorithm to meet the requirements of accuracy and efficiency at the same time, so how to organically combine the advantages of different algorithms and design a hybrid algorithm that can maintain the search accuracy and significantly improve the computational efficiency has become an important direction in the current research of unmanned aircraft path planning.

In this paper, a new hybrid path planning method is constructed by integrating the advantages of Dijkstra's algorithm and particle swarm algorithm. Firstly, the shortcomings of the traditional algorithm are analyzed, and then the dynamic inertia weight strategy, adaptive learning factor adjustment mechanism and improved speed update strategy are introduced into the particle swarm algorithm in order to enhance the convergence performance of the algorithm and avoid the phenomenon of premature maturity. Then the Dijkstra-PSO fusion strategy is designed to determine the critical path points by using the Dijkstra algorithm for local fine search, and then the global optimization is carried out by the improved particle swarm algorithm. At the same time, the composite fitness function containing path length and corner is constructed so that the algorithm can comprehensively consider multiple performance indicators. Finally, three times B-spline interpolation is introduced to smooth the path to ensure that the UAV can perform the flight mission safely and stably. Through this systematic research approach, this paper seeks to provide an efficient and practical solution to the UAV path planning problem.

II. Based on Dijkstra-PSO three-dimensional path planning model

II. A. Traditional algorithms and UAV path planning

II. A. 1) Route planning

Path planning is the technique of solving an optimal path connecting the starting point and the termination point in the target area with reference to one or more index parameters (minimum energy consumption, shortest distance of the traveled path, least resources consumed by the algorithm, etc.), which is essentially the optimal solution of an equation or a locally optimal solution under some constraints [19]. The performance of path planning algorithms is related to the real-time performance of task completion and good or bad results. At present, path planning is widely used in many fields, for example, in high-tech fields such as robot path planning, unmanned aerial vehicle (UAV) reconnaissance paths and cruise missile surprise defense, and in daily life fields such as urban road planning, map navigation systems and logistics capacity scheduling. Path planning technology takes a real-world problem, molds and topologizes it into a planning problem with a grid of points and lines in order to solve it using algorithms. In order to measure the performance of path planning in different application scenarios, the following evaluation metrics are proposed:

(1) Rationality: the planning paths found by the algorithm should comply with both complete and incomplete constraints.

(2) Completeness: the algorithm can find all the objectively existing planning paths.

(3) Optimality: the planning path found by the algorithm is optimal in one or more index parameters, including time, distance, cost loss and other indexes.

(4) Real-time: the complexity of the algorithm is in line with the actual arithmetic, including time complexity, space complexity, etc., in order to achieve real-time planning path.

(5) Adaptability: the algorithm can cope with the ability of dynamic changes in the environment.

II. A. 2) UAV path planning modeling

When the UAV is carrying out path planning, it is usually necessary to put the UAV in the most real three-dimensional environment space to be analyzed and model construction, when the three-dimensional spatial model construction is completed, the amount of data contained in the model as well as the accuracy will have a great improvement, and the success rate and safety of path planning with the three-dimensional environment model can also be greatly guaranteed. At the same time UAV in the implementation of some specific tasks, does not require the UAV in three-dimensional space on the height of the change, this case can be simplified into a two-dimensional spatial model of

the three-dimensional model, usually using a top view of the two-dimensional space modeling, which will be in the space of the obstacles and other environmental information is projected to the two-dimensional plane for processing, to reduce the processing of one-dimensional information, so as to also reduce the problem of the complexity. Therefore, in this subsection, the two-dimensional and three-dimensional environmental space modeling and analysis are carried out respectively.

(1) Two-dimensional path planning modeling

There are many common methods to build 2D environment maps, this chapter mainly introduces the raster map method, the raster map method is a relatively simple and reliable method for modeling the 2D environment of the UAV, which evenly divides the UAV's workspace into a number of spaces with the same size, and the size of the rasters will have a certain impact on the path planning, and when the rasters are too small, more storage space is needed to store the relevant information of each raster, which will increase the running time of the path planning algorithm. When the grid is too small, more storage space is needed to store the information related to each grid, which increases the running time of the UAV path planning algorithm. When the grid is too large, the information described by each grid is not accurate enough, which will result in the planning of too long a trajectory, or even may not meet the requirements of the trajectory, so a suitable grid size is particularly important. If the space of the UAV environment has a length of l and a width of w, and the shape of each grid is square with a side length of d, the total number of grids of the whole simulation environment is $(l/d)*(w/d)$. The expression of the raster method is shown in equation (1):

$$map_k = \begin{cases} 1, \\ 0, \end{cases} \begin{matrix} i\ is\ an\ \text{int}eger \end{matrix} \tag{1}$$

(2) Three-dimensional path planning modeling

The UAV may collide with the towering mountains or buildings on the ground when flying in the 3D spatial environment, in order to ensure that the UAV causes collision in the complex terrain environment formed by the mountains and towering buildings, the threat of the terrain to the UAV needs to be taken into account, and the towering mountains and buildings will be approximated as an environment composed of cylinders and spheres in the flight environment, and the whole mountain environment can be regarded as the superposition of different cylinders and spheres. The whole mountain environment can be regarded as a superposition of different cylinders and spheres.

II. A. 3) Dijkstra planning algorithm

Dijkstra's algorithm is a very typical graph search algorithm [20], initially to solve the shortest path problem from one node to another node, the principle is to start from a starting point, and gradually extend the expansion of the child nodes outward with its center, until the end point is included in the many child nodes expanded to, then the search stops, it is a typical greedy search strategy, Dijkstra's algorithm The greedy search side strategy is shown in Fig.1. S represents the starting point, G represents the goal point, the purple area is the obstacle, the pink area is all the nodes extended by this algorithm, and the green path is the final searched path.

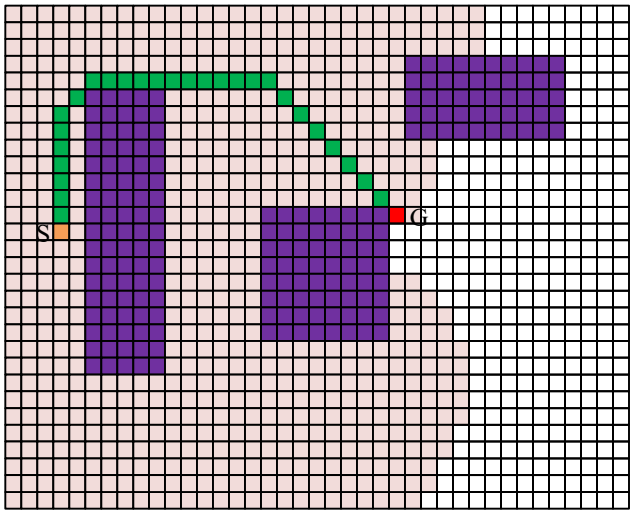

Figure 1: Dijkstra's algorithm greedy search side strategy

In each sub-node expansion process, the next sub-node selected is the node with the closest distance, which ensures that the path of each search is the shortest. In order to ensure that the final search path is the shortest, the distance between the nodes is recalculated each time. The process of sub-node expansion and selection of Dijkstra's algorithm is shown in Fig. 2.

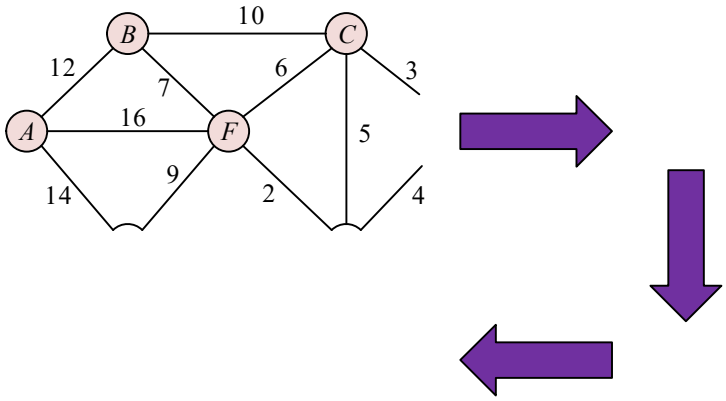

Figure 2: Dijkstra algorithm sub-node expansion and selection process diagram

Two sets S and U need to be introduced, respectively S contains the nodes that have been visited and will not continue to be visited, while U contains the nodes that have not yet been visited; calculate the distance of each node in the set U to the starting point, filter the node with the smallest distance from U to be added to S and remove the node from U; and update the shortest distance in U as the node expands, traversing all the nodes. Where D represents the starting point and the first expansion node and A represents the goal point at the end of the traversal. Nodes C and E are children of node D, and C is shorter path from the starting point and serves as the second expansion parent node, so there are $S=\{D(0),C(3)\}$ and $U=\{A(\infty),B(13),E(4),F(9),G(\infty)\}$; Similarly E is the third round expansion parent, so we have $S=\{D(0),C(3),E(4)\}$ and $U=\{A(\infty),B(13),F(6),G(12)\}$; F is the fourth round expansion parent node, $S=\{D(0),C(3),E(4),F(6),G(12),B(13),A(22)\}$, and $U=\varnothing$. Thus all nodes are traversed and the shortest path is filtered as $D\rightarrow E\rightarrow F\rightarrow A$.

The disadvantage of Dijkstra's algorithm is also based on its greedy strategy search mode, almost all the nodes in the map will be traversed, the amount of computation and memory space occupied is also large, when applied to the mobile robot in a large map to search for paths, the algorithm will require a very large amount of computation and memory space, the computational efficiency will be reduced dramatically, if the memory space is consumed before the failure to find a target location If the target location is not found before the memory space is consumed, the algorithm may cause a crash situation.

II. A. 4) Particle Swarm Algorithms

The PSO algorithm was inspired by the migration of birds [21]. Let the population of the particle swarm be n, the dimension of the particle be D, and the number of iterations be t, then the position vector $X_i(t)=(x_{i1}(t),x_{i2}(t),x_{i3}(t),\cdots,x_{iD}(t))$ is the position of the particle i in the second iteration of t, and the velocity vector $V_i(t)=(v_{i1}(t),v_{i2}(t),v_{i3}(t),\cdots,v_{iD}(t))$ is the velocity of the particle i in the second iteration of t, the vector $p_i=(p_{i1},p_{i2},p_{i3},\cdots,p_{iD})$ is the position of the optimal particle that records the previous iterations of the particle i by the evaluation function, and the vector $p_g=(p_{g1},p_{g2},p_{g3},\cdots,p_{gD})$ is the position of the globally optimal particle. The velocity update of a particle consists of three components, namely, the particle's original velocity, the velocity at which the particle learns from the global optimal $(p_{gd}^k-x_{id}^k)$, and the velocity at which the particle learns from the previous optimal particle $(p_{id}^k-x_{id}^k)$. The velocity update formula for particles is as follows Eq. (2), and the position update formula for the particle swarm algorithm is as follows Eq. (3):

$$v_{id}^{k+1}=\omega v_{id}^k+c_1 rand()(p_{id}^k-x_{id}^k)+c_2 rand()(p_{gd}^k-x_{id}^k) \tag{2}$$

$$x_{id}^{k+1}=x_{id}^k+v_{id}^{k+1} \tag{3}$$

In Eq. (2), v_{id}^k, v_{id}^{k+1} are the velocity components of the k th and $k+1$ th iterations. Eq. (3) x_{id}^{k+1}, x_{id}^k are the position components of the k th, and $k+1$ th iterations; ω is the inertial weight of the velocity. c_1, c_2 are the

weights for learning to the i th iteration optimal position and learning to the global optimal position, respectively. $rand()$ is a random number between $(0,1)$.

The inertia weight ω is an important parameter in the algorithm, representing the weight of the particle's original velocity in the iteration. For most continuous optimization problems, ω decreasing in a linear way is better than decreasing in a convex function, so this paper adopts the strategy of linear decreasing, as in equation (4):

$$\omega_{i+1} = d_{amp} * \omega_i \tag{4}$$

where ω_i is the inertia weights for the velocity of the previous iteration, ω_{i+1} is the inertia weights for the velocity of the current iteration, and d_{amp} is the inertia weight decay coefficient.

II. B.Improved particle swarm algorithm

II. B. 1) Dynamically changing inertia weighting strategies

The inertia weight factor w is a constant in the standard particle swarm algorithm, which is used to balance the search ability of particles globally and locally. Inertia weights in particle swarm optimization algorithms analyze the relationship between inertia weights and particle search, the larger the value of inertia weights the better for global search, and the smaller the value of inertia weights the better for local search. Linearly decreasing inertia weights are proposed to balance the local and global search ability. The specific formula is:

$$w(k) = w_{\max} - \left(w_{\max} - w_{\min}\right)\left(\frac{k}{k_{\max}}\right) \tag{5}$$

where $w_{\max} = 0.9$, $w_{\min} = 0.4$, and $k_{\max}$ maximum number of iterations.

Therefore, this paper proposes an adaptive inertia weight updating method. The method is set larger in the early stage of particle iteration, w, to quickly complete the search of the global map. In the later stage, the value of w is smaller, and the speed of particle search decreases accordingly, making the particles search more finely in the local map. The formula for updating the inertia weights w is:

$$w_k = w_{\max} - (w_{\max} - w_{\min}) \times \log_{k_{\max}}^{k} \tag{6}$$

II. B. 2) Adaptive change learning factor strategy

Since the learning factor can accelerate the speed of particle search, this paper uses an adaptive adjustment algorithm to continuously correct the learning factor. The particle pre-traversal is mainly able to fly through the whole search space quickly, so set $c_1 > c_2$. The later traversal is mainly to keep the particle approaching the global optimum, and will be $c_1 < c_2$. Therefore the equations for the variation of c_1 and c_2 are shown below:

$$c_1 = (c_{1i} - c_{1f})\frac{k_{\max} - k}{k_{\max}} + c_{1f} \tag{7}$$

$$c_2 = (c_{2i} - c_{2f})\frac{k_{\max} - k}{k_{\max}} + c_{2f} \tag{8}$$

where c_{1i} and c_{1f} are the initial and termination values of c_1. c_{2i} and c_{2f} represent the initial and termination values of c_2, respectively. $c_{1i} = 2.5$, $c_{1f} = 0.5$, $c_{2i} = 0.5$, and $c_{2f} = 2.5$.

II. B. 3) Speed update strategy

In the standard particle swarm algorithm, each particle moves to the local optimal position and the global optimal position at a certain speed with the number of iterations. The particle adjusts its speed based on the flight experience of its own individual and other individual particles in the population to change the flight state of the particle, and the movement of the particle is shown in Fig. 3.

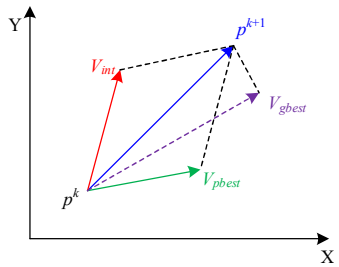

Figure 3: Particle motion

where p^k denotes the position of the particle in the k th generation and $V_{g_{best}}$ denotes the velocity of the globally optimal particle.

The algorithm suffers from falling into local minima at the late stage of particle iteration. Therefore, a population dynamic update method is proposed to avoid the whole population falling into local optimum. When the particle swarm algorithm evolves to the late stage, if the global optimal adaptation value in the particle swarm is not updated for many times in a row, the particles of the population may fall into the local optimum. At this time, it is necessary to change the speed of the globally optimal particles:

$$v_j^{m+1} = g_{best} - p_j^m + \alpha \cdot v_j^m + \beta \cdot r \tag{9}$$

where r is a random number in $[0,1]$. $\alpha = 0.2$ and $\beta = 0.3$. p_j^m denotes the position of the j th particle when the global optimum is obtained.

II. C.Dijkstra-PSO based path planning strategy

II. C. 1) Construction of the fitness function

To plan the flight path of the UAV well, the choice of the fitness function affects the algorithm to get the optimal value. In the standard algorithm the fitness function is only related to the length of the flight path. However, according to the real environment of UAV flight, when there are a lot of turns during the flight, it affects the flight of the UAV. Therefore, this thesis adds the corner of the flight to the fitness function:

$$C = \sum_{p=2}^{n-1} \frac{\phi_{(p-1,p)}\phi_{(p+1,p)} + \varphi_{(p-1,p)}\varphi_{(p+1,p)} + \gamma_{(p-1,p)}\gamma_{(p+1,p)}}{\sqrt{\phi_{(p-1,p)}^2 + \varphi_{(p-1,p)}^2 + \gamma_{(p-1,p)}^2}\sqrt{\phi_{(p+1,p)}^2 + \varphi_{(p+1,p)}^2 + \gamma_{(p+1,p)}^2}} \tag{10}$$

where $\phi_{(p-1,p)} = x_{p-1} - x_p$, $\varphi_{(p-1,p)} = y_{p-1} - y_p$, and $\gamma_{(p-1,p)} = z_{p-1} - z_p$.

In this paper, the path and corner functions together form the fitness function. As in equation (11):

$$F = m_1 \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} + m_2 \frac{L}{C} \tag{11}$$

where L represents a linear function and the function is proportional to the number of corners.

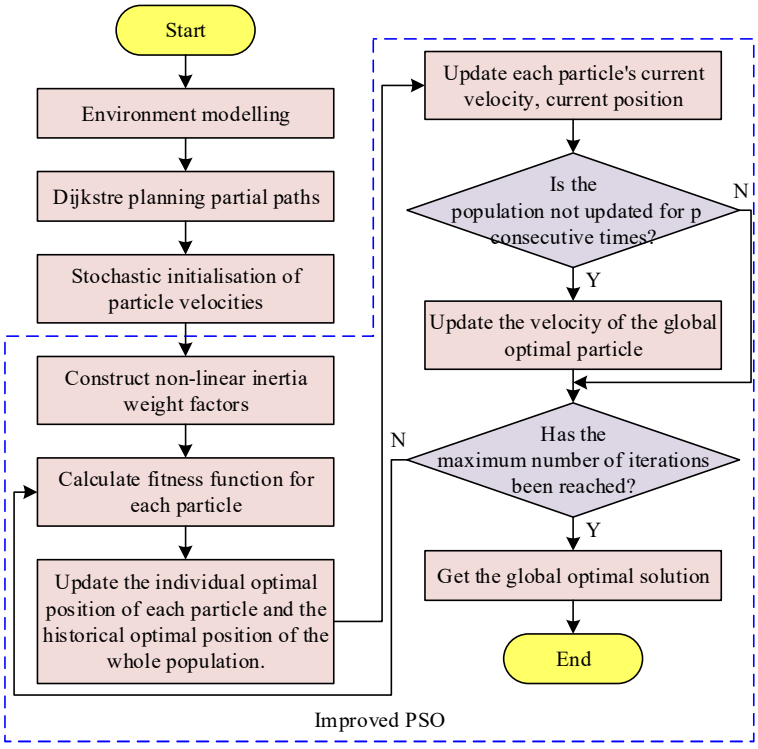

Figure 4: Dijkstra-PSO algorithm process

II. C. 2) Dijkstra-PSO fusion strategy

In Dijkstra's algorithm each time the node with the closest distance from the starting point is found and that node is used as a new center to expand a point with the closest distance from the starting point and update the current distance value to finally find the shortest distance from the starting point to all other points. In this algorithm, the shortest path can be found by one-time traversal, but when the number of nodes in the environment is large, then the amount of computation increases, which does not meet the time requirements of UAV flight. Particle swarm algorithm can search the global optimum in a shorter time. So in this paper, we will utilize the characteristics of Dijkstra algorithm's searching fineness combined with PSO algorithm with fast convergence to complete the path planning for the global environment.

The improved particle swarm algorithm incorporating Dijsktra mainly uses Dijkstra to realize the search of a small range of paths to determine the shortest path of the part, followed by determining a particle position according to each group of path points, and finally using the improved particle swarm algorithm to iterate globally until the global optimal value is found. The specific process is shown in Fig. 4.

II. C. 3) Three times B-spline interpolation

The above is mainly to improve the global path algorithm based on known maps to prevent from falling into local optimization and premature convergence. However, the trajectory generated in the UAV flight is not necessarily optimal and the flight trajectory is not smooth enough, the speed of the sudden change and other problems, so the introduction of three times B spline interpolation [22], more conducive to the UAV air flight.

The expression of cubic B spline is:

$$M(t)=\sum_{i=0}^{3}A_{i,3}(t)B_i \tag{12}$$

B_i is the control point of the curve and $A_{i,3}(t)$ is the cubic B spline basis function:

$$A_{i,3}(t)=\frac{1}{3!}\sum_{m=0}^{3-i}(-1)^m C_4^m(t+3-m-j)^3 \tag{13}$$

Bring Eq. (13) to Eq. (12) to get the control points of the curve and remove the duplicate control points to reduce the computational complexity of the algorithm. Finally the remaining control points are linked to get the B-spline curve.

III. Simulation experiment analysis

III. A. Validation of validity

Aiming at the obvious difference between the improved Dijkstra algorithm and the traditional Dijkstra algorithm in the path planning results, this paper designs four sizes of maps, 15×15, 25×25, 50×50 and 100×100, and carries out the two algorithms path planning tests in the same size of raster maps respectively, and through the comparison of the relevant data, it is fully proved that the critical path point adopted in this paper Screening and high-order Bessel curve optimization methods used in this paper. The comparison details of the relevant parameters of Dijkstra's algorithm in different raster maps before and after improvement are shown in Table 1.

As can be seen from Table 1, in the four sizes of maps, when the starting point and the target point are under the same conditions, the search nodes are reduced by 62.5%, 75%, 86.67%, 91.67%, the path lengths are reduced by 9.39%, 10.58%, 16.57%, and 12.83%, and the angle of turn is reduced by 92.95%, 90.11%, and 90.04%, respectively, 90.47%, and the computational consumption time is reduced by 85.29%, 88.57%, 91.46%, and 93.27%, respectively. From Table 1, it can be seen that the improved Dijkstra algorithm has obvious optimization in searching nodes, path length, turning angle, computation consuming time, etc., and the effect is more obvious in large maps, which fully proves the effectiveness of the improved Dijkstra-PSO algorithm in this paper.

Table 1: Comparison of Dijkstra-PSO algorithm and traditional Dijkstra algorithm

Map size	Comparison algorithm	Path node	Path length	Turning angle(°)	Consuming time (ms)
15×15	Dijkstra	16	20.13	342.56	68
	Dijkstra-PSO	6	18.24	24.16	10
25×25	Dijkstra	32	45.36	410.62	105
	Dijkstra-PSO	8	40.56	40.63	12
50×50	Dijkstra	60	87.69	452.68	164
	Dijkstra-PSO	8	73.16	45.08	14
100×100	Dijkstra	120	161.25	658.48	223
	Dijkstra-PSO	10	140.56	62.74	15

In this paper, the simulation comparison between the improved Dijkstra's algorithm and the classical path planning algorithm (A* algorithm) is accomplished in the environment of raster maps of different sizes, 15×15, 25×25, 50×50, and 100×100, respectively. In the same size map, set the same location of the starting point and the goal point, respectively, using two algorithms for path planning. The path search time, search nodes, path length and turning angle of the two algorithms are compared, and the comparison results are shown in Table 2.

As can be seen from Table 2, the improved Dijkstra-PSO algorithm reduces the path nodes by 60%, 73.33%, 87.1%, 91.3%, the path length by 7.49%, 3.12%, 7.16%, 5.4%, the angle of turn by 88.69%, 80.08%, 88.58%, 87.7%, and the path searching time is reduced by 81.48%, 87.23%, 90.21%, and 92.35%, respectively. The experimental results show that the improved Dijkstra algorithm has a more obvious advantage over the commonly used A* algorithm, which effectively improves the relevant performance of the algorithm, proving the necessity of the improved Dijkstra algorithm and the effectiveness of the improved Dijkstra-PSO algorithm.

Table 2 Comparison of Dijkstra-PSO algorithm and A* algorithm

Map size	Comparison algorithm	Path node	Path length	Turning angle(°)	Path search time (ms)
15×15	Dijkstra-PSO	6	18.52	24.26	10
	A*	15	20.02	214.53	54
25×25	Dijkstra-PSO	8	40.35	40.47	12
	A*	30	41.65	203.14	94
50×50	Dijkstra-PSO	8	73.24	44.85	14
	A*	62	78.89	392.64	143
100×100	Dijkstra-PSO	10	140.74	62.55	15
	A*	115	148.77	508.48	196

III. B. Optimization of Path Planning Algorithm Parameters

In the path planning results described in the previous section, only the effectiveness of the algorithm is considered. Therefore, when setting up the simulation environment, the parameters α and K (α denotes the ratio of triggering the search restriction time, and K is the initial weight coefficient) are both set to initial values and do not change during the computation process. In order to verify the effect of parameters α and K on the performance of the Dijkstra-PSO algorithm, orthogonal experiments are used to test the effect of path planning under different parameter values. Among them, the value range of parameter α is defined as [0,1] and the value range of parameter K is defined as [1,2].

It is found that, firstly, when the parameter α is a quantitative quantity, the larger the parameter K is the longer the path is planned. This path length grows in a stepwise manner with the variation of parameter K until it stabilizes at 48 units of length. On the other hand, as the parameter K increases, the amount of variation in the search time per extended node is insignificant and remains stable at about 18ms. Moreover, when the parameter K exceeds a certain level, the search time and path length of each extended node no longer change significantly, which indicates that the algorithm tends to the upper performance limit. In addition, if the parameter K is a constant value, as the parameter α keeps increasing, the path length calculated each time does not change significantly.

Overall, when the parameter K takes a small and constant value, choosing the parameter α between 0 and 0.5 ensures that the search time remains constant, i.e., it is maintained at around 80ms. And when the value of α exceeds 0.5, the node search time starts to increase sharply until it stabilizes until it reaches about 120ms. On the other hand, when the parameter α reaches 1, the jump in node search time is most obvious, with the maximum search time reaching 205ms.

After clarifying the value intervals of the parameters, in order to verify the effectiveness of the optimization algorithm, path planning simulation experiments are carried out using the types of complex environment graphs (with 15%, 35% and 55% obstacles) and the optimized parameters α and K. The results are shown in Tables 3 to 5. The comparison results show that the optimized Dijkstra-PSO algorithm is basically the same as the original algorithm in solving the obtained path length. Moreover, both algorithms before and after optimization complete the search in a limited time and ensure convergence to the optimal path. In the environment maps with 15%, 35% and 55% obstacles, the parameter optimization leads to a reduction of 32%, 11.38% and 31.88% in the average number of extended nodes and 14.64%, 17.21% and 17.15% in the average search time, respectively.

Table 3: Parameter optimization results in 15% obstacle map

Search step	Path length		Extended node quantity		Consuming time/ms	
	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra
10	38.28	37.94	31	35	8.83	25.79
20	37.11	37.09	32	35	6.84	19.35
30	37.23	37.79	34	63	13.71	12.57
40	37.47	38.35	33	34	38.99	42.35
50	37.42	36.72	27	28	66.45	86.37
60	37.02	38.26	26	52	8.93	23.12
70	37.01	37.56	27	67	31.31	50.26
80	37.47	37.14	38	65	36.24	22.37
90	38.14	37.91	35	60	35.31	24.48
100	36.95	38.07	40	36	36.76	25.32

Table 4: Parameter optimization results in 35% obstacle map

Search step	Path length		Extended node quantity		Consuming time/ms	
	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra
10	40.27	42.58	32	35	20.32	25.64
20	43.01	43.11	41	35	15.23	21.37
30	40.02	40.16	33	46	27.61	29.42
40	39.82	40.26	41	44	20.13	24.87
50	41.56	41.97	40	46	17.33	20.46
60	42.76	41.43	38	40	14.48	24.81
70	58.71	56.56	32	43	29.16	30.57
80	52.68	54.95	41	49	21.69	23.88
90	54.28	52.04	32	41	18.66	22.24
100	46.71	47.74	36	34	22.28	26.67

Table 5: Parameter optimization results in 55% obstacle map

Search step	Path length		Extended node quantity		Consuming time/ms	
	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra	Dijkstra-PSO	Dijkstra
10	51.62	53.97	86	86	28.09	32.17
20	71.36	64.11	72	86	22.26	51.14
30	59.86	61.36	70	128	32.51	41.19
40	45.22	44.54	82	79	38.69	25.97
50	50.32	49.97	91	93	35.15	24.38
60	62.71	52.06	33	81	35.18	21.73
70	62.07	51.23	75	125	35.71	51.65
80	52.87	67.69	31	63	26.78	26.23
90	51.58	50.13	67	111	23.36	46.56
100	49.77	62.94	34	89	21.37	39.99

IV. Conclusion

The hybrid Dijkstra-PSO algorithm exhibits significant performance advantages in the field of UAV path planning. Simulation experiments show that the algorithm achieves substantial performance improvements in four different map sizes, most notably in the 100 × 100 map environment, where the number of search nodes is reduced from 120 to 10, the computation consumption time is reduced from 223 ms to 15 ms, the angle of turn is reduced from 658.48 degrees to 62.74 degrees, and the path length is shortened from 161.25 units to 140.56 units. These data clearly demonstrate the superior performance of the algorithm in large-scale environments.

The introduction of dynamic inertia weighting strategy and adaptive learning factor effectively solves the problem of early convergence of the traditional particle swarm algorithm, which enables the algorithm to quickly explore the global search space in the early stage, and accurately locate the local optimal region in the later stage. The design of the composite fitness function takes into account the path length and flight safety, and makes the planned path more suitable for the actual flight characteristics of the UAV by introducing the corner factor. The application of three times B-spline interpolation further improves the smoothness of the path and ensures the stability and safety of UAV flight.

Comparative analysis with the classical A* algorithm further verifies the effectiveness of this paper's algorithm, which significantly reduces the computational complexity and time consumption while maintaining similar path quality. The parameter optimization experiments reveal the relationship between the performance of the algorithm and the key parameters, providing a theoretical basis for parameter adjustment in practical applications. The algorithm provides a new technical way to solve the real-time path planning problem of UAVs in complex environments, which has important theoretical value and application prospects.

Funding

This work was supported by Yunnan Horizontal Science and Technology Service Program KKF0202303309.

References

[1] Laghari, A. A., Jumani, A. K., Laghari, R. A., & Nawaz, H. (2023). Unmanned aerial vehicles: A review. Cognitive Robotics, 3, 8-22.
[2] Chaturvedi, S. K., Sekhar, R., Banerjee, S., & Kamal, H. (2019). Comparative review study of military and civilian unmanned aerial vehicles (UAVs). INCAS bulletin, 11(3), 183-198.
[3] Gargalakos, M. (2024). The role of unmanned aerial vehicles in military communications: application scenarios, current trends, and beyond. The Journal of Defense Modeling and Simulation, 21(3), 313-321.
[4] Kozera, C. A. (2018). Military use of unmanned aerial vehicles–a historical study. Safety & Defense, 4, 17-21.
[5] Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., ... & Guizani, M. (2019). Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. Ieee Access, 7, 48572-48634.
[6] Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks, 72(4), 411-458.
[7] Mohamed, N., Al-Jaroodi, J., Jawhar, I., Idries, A., & Mohammed, F. (2020). Unmanned aerial vehicles applications in future smart cities. Technological forecasting and social change, 153, 119293.
[8] Xu, C., Liao, X., Tan, J., Ye, H., & Lu, H. (2020). Recent research progress of unmanned aerial vehicle regulation policies and technologies in urban low altitude. Ieee Access, 8, 74175-74194.
[9] Mohsan, S. A. H., Othman, N. Q. H., Li, Y., Alsharif, M. H., & Khan, M. A. (2023). Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. Intelligent service robotics, 16(1), 109-137.
[10] Zhao, Y., Zheng, Z., & Liu, Y. (2018). Survey on computational-intelligence-based UAV path planning. Knowledge-Based Systems, 158, 54-64.
[11] Aggarwal, S., & Kumar, N. (2020). Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. Computer communications, 149, 270-299.
[12] Ait Saadi, A., Soukane, A., Meraihi, Y., Benmessaoud Gabis, A., Mirjalili, S., & Ramdane-Cherif, A. (2022). UAV path planning using optimization approaches: A survey. Archives of Computational Methods in Engineering, 29(6), 4233-4284.
[13] Luo, M., Hou, X., & Yang, J. (2020). Surface optimal path planning using an extended Dijkstra algorithm. IEEE access, 8, 147827-147838.
[14] Liu, L. S., Lin, J. F., Yao, J. X., He, D. W., Zheng, J. S., Huang, J., & Shi, P. (2021). Path planning for smart car based on Dijkstra algorithm and dynamic window approach. Wireless Communications and Mobile Computing, 2021(1), 8881684.
[15] Li, X. (2021, November). Path planning of intelligent mobile robot based on Dijkstra algorithm. In Journal of Physics: Conference Series (Vol. 2083, No. 4, p. 042034). IOP Publishing.
[16] Zhu, D. D., & Sun, J. Q. (2021). A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute. IEEE Access, 9, 19761-19775.
[17] Wang, J., Li, Y., Li, R., Chen, H., & Chu, K. (2022). Trajectory planning for UAV navigation in dynamic environments with matrix alignment Dijkstra. Soft Computing, 26(22), 12599-12610.
[18] Dhulkefl, E., Durdu, A., & Terzioğlu, H. (2020). Dijkstra algorithm using UAV path planning. Konya Journal of Engineering Sciences, 8, 92-105.
[19] Yuanzheng Zuo,Hongtao Wang,Ning Dai,Kanghui Zhou,Hongyu Chen & Jun Liu. (2025). Path planning for curved layer fused deposition modeling of continuous fiber reinforced structures with iso-void depth. Composites Science and Technology,266,111179-111179.
[20] Huanyu Liu,Jiahao Luo,Lihan Zhang,Hao Yu,Xiangnan Liu & Shuang Wang. (2025). Research on Traversal Path Planning and Collaborative Scheduling for Corn Harvesting and Transportation in Hilly Areas Based on Dijkstra's Algorithm and Improved Harris Hawk Optimization. Agriculture,15(3),233-233.
[21] Xinyue Li,Yu Zhang & Wang Hu. (2025). Ensemble learning training strategy based on multi-objective particle swarm optimization and chasing method. Expert Systems With Applications,281,127777-127777.
[22] He Shitao,Shen Liyong,Wu Qin & Yuan Chunming. (2024). A Certified Cubic B-Spline Interpolation Method with Tangential Direction Constraints. Journal of Systems Science and Complexity,37(3),1271-1294.