www.nature.com/scientificreports

scientific reports

OPEN

Gradient-based autonomous obstacle avoidance trajectory planning for B-spline UAVs

Wei Sun, Pengxiang Sun, Wei Ding, Jingang Zhao & Yadan Li

Unmanned aerial vehicles (UAVs) have become the focus of current research because of their practicability in various scenarios. However, current local path planning methods often result in trajectories with numerous sharp or inflection points, which are not ideal for smooth UAV flight. This paper introduces a UAV path planning approach based on distance gradients. The key improvements include generating collision-free paths using collision information from initial trajectories and obstacles. Then, collision-free paths are subsequently optimized using distance gradient information. Additionally, a trajectory time adjustment method is proposed to ensure the feasibility and safety of the trajectory while prioritizing smoothness. The Limited-memory BFGS algorithm is employed to efficiently solve optimal local paths, with the ability to quickly restart the trajectory optimization program. The effectiveness of the proposed method is validated in the Robot Operating System simulation environment, demonstrating its ability to meet trajectory planning requirements for UAVs in complex unknown environments with high dynamics. Moreover, it surpasses traditional UAV trajectory planning methods in terms of solution speed, trajectory length, and data volume.

Keywords B-spline, Gradient descent, L-BSGF, Trajectory planning

With the rapid development of artificial intelligence technology and machinery manufacturing, micro UAVs have expanded their capabilities and are now being utilized in various scenarios, not only in the military but also in commercial applications, including environmental detection, fire rescue, road guidance, and short-range transportation[1,2]. However, challenges such as limited load capacity and range continue to hinder the advancement of micro UAVs. Additionally, researchers are faced with the issues of limited perception ability and computational load of micro UAVs towards the external environment[3].

In response to these problems, experts and scholars have proposed various methods for solving UAV trajectories. One of the traditional path planning algorithms is the artificial potential field method, which has a simple structure and strong real-time performance. However, this method tends to result in local optimal solutions. To address this issue, many scholars have made improvements to the artificial potential field method. Fang et al. introduced a Piecewise-potential-field (PPF) based UAV formation trajectory planning method[4]. This method utilizes a suitable PPF function to avoid local optimal solutions while still satisfying the motion constraints. Li et al. proposed an artificial potential field-based particle swarm algorithm, which generates robot planning paths by adjusting the inertia weight parameter and ranking the position vector of particles[5].

With the advancement of artificial intelligence, intelligent bionics algorithms have emerged as the principal path planning algorithms[6–8]. The graph search algorithm uses multi-source sensor information to search feasible path options at high frequency and finally calculates the optimal path from the starting point to the destination through an iterative process. Scholars have used intelligent bionic algorithms to enhance traditional graph search algorithms like Dijkstra and A*, significantly boosting the efficiency of local path planning. Zhang et al. proposed the A*-ACO algorithm by fusing the A* algorithm and ant colony optimization (ACO) to improve the blindness of the initial search and the convergence speed of the ant colony algorithm, but the A*-ACO algorithm may not be able to find the optimal path when facing complex situations[9].

Model predictive control is the mainstream UAV control method. Chai et al. utilized deep neural networks trained with optimal trajectories generated by fuzzy multi-target transcription methods[10]. The trained neural network serves as a UAV command generator to ensure the real-time generation of optimal trajectories. Furthermore, they developed a centralized robust model predictive control algorithm[11]. This centralized control structure simplifies controller development and parameter tuning while guaranteeing a stable and convergent attitude-tracking process through nonlinear feedback rules and stringent constraints. Lindqvist B et al. proposed

School of Geomatics, Liaoning Technical University, Fuxin 12300, Liaoning, China. email: 516885735@qq.com

a nonlinear model predictive control algorithm to predict the future position of obstacles by parameterizing obstacle trajectory so as to conduct autonomous navigation experiments of UAVs[12]. In addition, a new nonlinear model predictive control framework is proposed[13]. Based on this framework, the UAV conforms to the dynamic characteristics of the UAV and effectively solves the problem of UAV estimation drift.

The gradient-based motion planning method, widely used in UAV trajectory optimization, involves transforming the UAV trajectory planning problem into an unconstrained nonlinear optimization problem[14,15]. Ratliffet et al. introduced the Euclidean Signed Distance Field (ESDF) into the field of robot motion planning[16], using ESDF data to optimally configure the robot's surroundings, which in turn is useful for generating collision-free paths for the robot.

The gradient motion planning method parameterizes trajectory planning[17]. However, due to the need for the UAV to maintain a high-speed motion state, the computational burden imposed by the optimal configuration of the surrounding environment is too heavy for the UAV to handle. As a result, continuous solving of the obstacle avoidance problem becomes challenging, leading to a low success rate[18,19]. To address issues such as excessive data volume in the gradient motion planning method, the B-spline generation method is introduced for trajectory initialization[20,21]. This method helps solve the challenging problem of trajectory solving, focusing on the smoothness and glossiness of the trajectory rather than considering collision problems during the initialization process. The feasibility of the trajectory is ensured by the convex packet property of the B-spline. This property allows for a quick trajectory initialization while avoiding the need for numerous calculations to establish collision-free sections using conventional methods like the potential field method. After initializing, only the gradient data of the ESDF within a narrow range around the initial trajectory are collected[22]. Meanwhile, the initial trajectory is continuously collision-tested and iteratively optimized[23,24]. If a collision is detected, the trajectory undergoes obstacle avoidance modification to generate a collision-free trajectory. This approach significantly reduces the computation of data and enhances the solution speed of trajectory optimization.

The primary objective of the algorithm proposed in this study is to streamline the process of initializing and optimizing trajectories for UAVs. The algorithm aims to enhance the smoothness of trajectories while prioritizing feasibility and safety, making them more suitable for high-speed UAV flight to prevent waste of power and time.

The key contributions of this paper include: (1) introducing a method for UAV trajectory initialization, where B-spline curve control points are chosen through polynomial fitting and connected to ensure the shortest trajectory in obstacle-free scenarios. When obstacles are present, B-spline curves are created based on control points and basis functions, with trajectory obstacle avoidance optimization achieved through the use of the convex hull property. (2) Solving the UAV obstacle avoidance problem using the repulsive force algorithm, proposing a lightweight UAV trajectory evaluation function, and utilizing the L-BSGF algorithm for fast trajectory optimization and quick program restart after planning failure. The proposed algorithm is verified in the Robot Operating System (ROS) environment.

The rest of this paper is organized as follows. Section II focuses on the initialization and obstacle avoidance optimization of the UAV trajectory. Section III simulates and validates the B-spline-based method of UAV trajectory generation in the ROS environment. Section IV draws the conclusions of the paper.

Trajectory initialization

The UAV trajectory should be collision-free, with the shortest path length, and conform to the smooth curve of UAV dynamics. The B-spline is capable of generating a curve that precisely satisfies these conditions in the given environment. The expression of the B-spline is shown in Eq. (1).

$$P(u) = \sum_{i=0}^{n} C_i B_{i,k}(u) \qquad u \in [u_{k-1}, u_{n=1}] \tag{1}$$

where C_i represents the control point, $B_{i,k}$ represents the B-spline basis function of order K, and u represents the node vector. To simplify B-spline generation and improve the efficiency of path generation, the optimal 4th-order basis functions are used to generate B-spline curves. The curve segmentation generates each B-spline trajectory by four control points and four B-spline basis functions calculated by Eq. (2)

$$\begin{cases} f_1(s) = \frac{1}{6}(1-s)^3 \\ f_2(s) = \frac{1}{6}(3s^3 - 6s^2 + 4) \\ f_3(s) = \frac{1}{6}(-3s^3 + 3s^2 + 3s + 1) \\ f_4(s) = \frac{1}{6}s^3 \end{cases} \tag{2}$$

where s from 0 to 1 represents the normalized distance, and each segment of the trajectory is generated as shown in Eq. (3).

$$P_i(s) = f_1(s)C_1 + f_2(s)C_2 + f_3(s)C_3 + f_4(s)C_4 \tag{3}$$

where C_1, C_2, C_3, C_4 is the four control points of the trajectory and P_i belongs to R^3. The complete trajectory of the UAV is stitched by each segment of the trajectory, and the next segment is generated by the control points C_2, C_3, C_4, C_5. When there are N control points, N-3 segments of UAV trajectories are generated. The velocity $V(s)$ and acceleration $A(s)$ of the position curve can be deduced from 3.

$$V(s) = \frac{\partial P(s)}{\partial s} = \begin{bmatrix} V_x(s) \\ V_y(s) \\ V_z(s) \end{bmatrix} \tag{4}$$

$$A(s) = \frac{\partial^2 P(s)}{\partial^2 s} = \begin{bmatrix} a_x(s) \\ a_y(s) \\ a_z(s) \end{bmatrix} \tag{5}$$

Since the control points of the trajectory are immobile, differentiating the trajectory is differentiating the spline function. Then the velocity spline functions V_1, V_2, V_3, V_4, and the acceleration spline functions a_1, a_2, a_3, a_4 can be obtained.

$$\begin{cases} v_1(s) = -(1-s)^2/2 \\ v_2(s) = \left(3s^2/3 - 2s\right) \\ v_3(s) = -3s^2/3 + s + 1/2 \\ v_4(s) = s^2/2 \end{cases} \tag{6}$$

$$\begin{cases} a_1(s) = 1 - s \\ a_2(s) = 3s - 2 \\ a_3(s) = 1 - 3s \\ a_4(s) = s \end{cases} \tag{7}$$

Therefore, both the velocity and acceleration of the B-spline trajectory P_i are functions of the parameter S, as shown in Eqs. (8) and (9).

$$V(s) = v_1(s)C_1 + v_2(s)C_2 + v_3(s)C_3 + v_4(s)C_4 \tag{8}$$

$$A(s) = a_1(s)C_1 + a_2(s)C_2 + a_3(s)C_3 + a_4(s)C_4 \tag{9}$$

This shows that the end point of the former curve and the start point of the latter curve are equal in position, velocity, and acceleration, which ensures that the B-spline trajectory is smooth and continuous. It is also assumed that additional control points are added to verify that the trajectory passes through the start point and end point. We expand the starting point C_1 as c_0, c_1, and c_2, where c_1 is the origin C_1, c_0 and c_2 as in Eq. (10).

$$\begin{cases} c_0 = c_1 - v_1 L \\ c_2 = c_1 - v_1 L \end{cases} \tag{10}$$

where v_1 represents the velocity vector when the UAV passes through the point, L is a suitable constant, and for the experiment we take half the distance required for the UAV to gain maximum velocity. Solving Eq. (9) by substituting it into Eqs. (7) and (8) verifies that the trajectory passes through the starting point C_1 and has velocity v_1 acceleration a_1 of zero. Similarly, the target point C_n is extended to c_{n-1}, c_n,c_{n+1} and c_{n+1}, where c_n is the origin C_n, c_{n-1} and c_{n+1} are denoted by Eq. (11).

$$\begin{cases} c_{n-1} = c_n - v_n L \\ c_{n+1} = c_n + v_n L \end{cases} \tag{11}$$

The solution of the equation also verifies that the trajectory passes through the control point C_N, the speed of the UAV is v_n, and the acceleration a_n is zero. Therefore as long as we adjust the control points C_1, C_2, C_3, C_4, …, C_N, the initialization of the UAV B-spline trajectory can be completed after solving.

In order to quickly initialize the UAV B-spline trajectory, the polynomial fitting method is used for the selection of control points. After obtaining the starting point C_1 and the ending point C_N of the trajectory, the maximum singular value d_z between the two points in space is calculated and compared with the set distance d_c of the distribution of the control points, to determine whether to add a waypoint. If d_z is larger, it is necessary to set waypoints. The expression for the number m of setting waypoints is:

$$\begin{cases} m = [d_z/d_c] + 1 \ \ (d_z > d_c) \\ \quad m = 2 \qquad\quad d_z \le d_c \end{cases} \tag{12}$$

The coordinates of the path point C_N are calculated as:

$$\begin{cases} x_n = x_0(1 - n/m) + \frac{1}{m}x_N \\ y_n = y_0(1 - n/m) + \frac{1}{m}y_N \quad (0 < n \le m) \\ z_n = z_0(1 - n/m) + \frac{1}{m}z_N \end{cases} \tag{13}$$

After obtaining each path point it is known that m-1 segments of UAV trajectories are generated, and the polynomial coefficients of each UAV trajectory in the x-axis, y-axis, and z-axis directions, F_x, F_y, F_z are solved as shown in Eq. (14).

$$\begin{cases} F_x = B_x/A \\ F_y = B_y/A \\ F_z = B_z/A \end{cases} \tag{14}$$

where, B_x, B_y, B_z are the parameter vectors in the x-axis, y-axis, and z-axis directions at the beginning and end of each UAV trajectory, and their expressions are:

$$\begin{cases} B_x = [x_0, v_{x0}, a_{x0}, x_n, v_{xn}, a_{xn}] \\ B_y = [y_0, v_{y0}, a_{y0}, y_n, v_{yn}, a_{yn}] \\ B_z = [z_0, v_{z0}, a_{z0}, z_n, v_{zn}, a_{zn}] \end{cases} \tag{15}$$

where $x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0}, a_{x0}, a_{y0}, a_{z0}$ are the coordinate, velocity, and acceleration of each segment of the trajectory on the three axes of the starting point. $x_n, y_n, z_n, v_{xn}, v_{yn}, v_{zn}, a_{xn}, a_{yn}, a_{zn}$ are the coordinate, velocity, and acceleration of the corresponding segment of the trajectory on the three axes of the ending point. The parameter matrix in Eq. (14) is expressed as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ t^5 & t^4 & t^3 & t^2 & t & 1 \\ 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \\ 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix} \tag{16}$$

where t represents the time required to pass through the corresponding trajectory at the UAV's maximum speed, and the time is set to be twice the time required to fly at the maximum speed, considering that the UAV may not be able to maintain the maximum speed during the start and end segments. t_n can be formulated as follows.

$$\begin{cases} t_n = 2(C_{n+1} - C_n)/v_{\max}\ (n = 0) \\ t_n = (C_{n+1} - C_n)/v_{\max}\ (0 < n < m) \\ t_n = 2(C_{n+1} - C_n)/v_{\max}\ (n = m) \end{cases} \tag{17}$$

After obtaining the coordinates of the waypoints and the polynomial coefficients of the m-1 group of coordinates, the polynomial fitting is used to generate the coordinates of the control points, and the solution formula is shown in Eq. (18).

$$\begin{cases} x_n = \dfrac{\mathbf{K}_x}{\mathbf{C}} \\ y_n = \dfrac{\mathbf{K}_y}{\mathbf{C}} \\ z_n = \dfrac{\mathbf{K}_z}{\mathbf{C}} \end{cases} \tag{18}$$

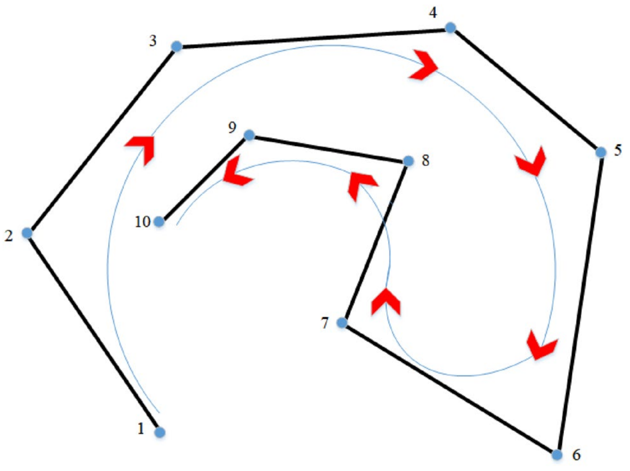

Figure 1. Structure of clamped B-spline.

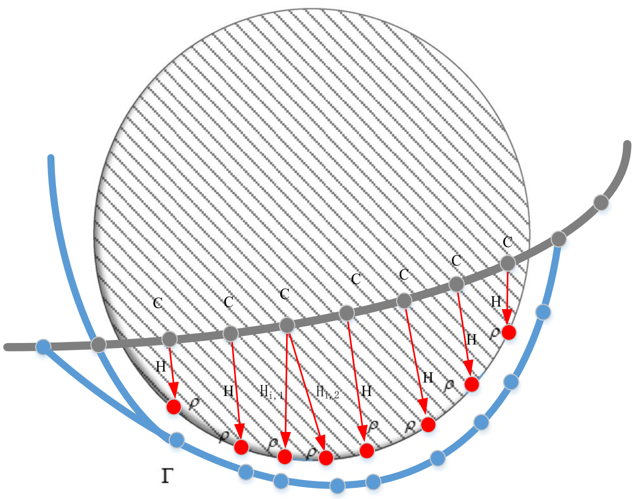

Figure 2. Trajectory obstacle avoidance control point variation diagram.

where **C** is the coefficient matrix associated with the number of waypoints and $\mathbf{K}_x, \mathbf{K}_y, \mathbf{K}_z$ are the augmentation vector of the coefficient vector F_x, F_y, F_z of the polynomial. After obtaining the control points according to Eq. (3), the path initialization is completed.

The trajectory initialization process has only the starting point and the endpoint of the trajectory, and to facilitate the trajectory solving, the initialization process of the trajectory is calculated using the parameter S, but the trajectory optimization process is more convenient to use the time parameter t to solve the trajectory. The conversion formula for the parameters S and t of the UAV trajectory is:

$$P(s) = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} = P(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \tag{19}$$

$$\frac{dP(t)}{dt} = \frac{\partial P(s)}{\partial s}\bar{s} \Rightarrow \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} = \begin{bmatrix} v_x(s) \\ v_y(s) \\ v_z(s) \end{bmatrix}\bar{s}(t) \tag{20}$$

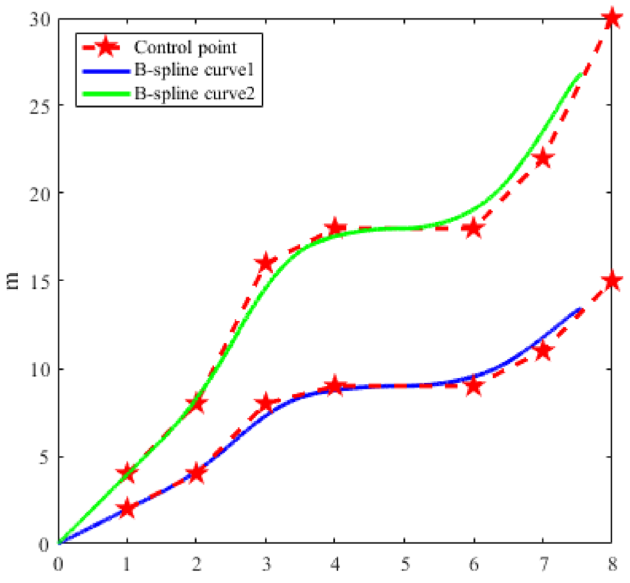

Figure 3. Change of B-spline curve by control points selection.

Optimization of trajectory

Clamped B-splines

According to de Boor-Cor's recursive formula (21), it can be shown that each higher-order B-spline is a convex linear combination of two lower first-order B-splines. Each segment of the B-spline curve must have enough basis functions to match the control points if the interval is legal. Therefore, in the proposed algorithm, $n+4$ node vectors $(u_0, \cdots, u_{n+3})$ are required if there are n control points. Four nodes $(u_i, u_{i+1}, u_{i+2}, u_{i+3})$ are needed if one wants to determine a 4th-order B-spline P_i. Neighboring B-splines have 2 control points that are the same to ensure that the splicing between every two segments is good.

$$\begin{cases} B_{i,1}(u) = \begin{cases} 1 \; u_i < u_{i+1} \\ 0 \text{ otherwise} \end{cases} \\ B_{i,k} = \dfrac{u-u_i}{u_{i+k-1}-u_i} B_{i,k-1}(u) + \dfrac{u_{i+k}-u}{u_{i+k}-u_{i+1}} B_{i+1,k-1}(u) \end{cases} \tag{21}$$

The UAV trajectory initialization process, by assuming the increase of control points to ensure that the UAV passes through the trajectory start point, the solution passes through is the beginning and end of the two points of the velocity of v_1 and v_n, respectively. However, the speed of the UAV in the beginning two points can only be zero. Therefore, in the trajectory solution process outlined in this paper, the first four and the last four node vectors remain consistent. This B-spline is called clamped B-spline and its structure is shown in Fig. 1.

The graph features 10 control points and 14 nodes with node vectors [0, 0, 0, 0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1, 1, 1, 1]$^{\mathrm{T}}$. Figure 1 perfectly illustrates the two main properties of the clamped B-spline. The curve passes through the head and tail control points and has a strong convex hull[25,26]. Strong convex hull property means that if the node vectors u are located in $[u_i, u_{i+1}]$, then the curve $P(u)$ is located within the convex hull defined by control points $C_i,\ C_{i-1},\ C_{i-2},\ C_{i-3}\,(i \geq 3)$, which indirectly verifies that a control point only affects local trajectories. The B-spline function is localized, and this property can solve the problem of local changes in the trajectory affecting the overall trajectory planning when the UAV encounters an obstacle. The collision part of the trajectory is changed by adjusting the relevant control points, while the convex hull of the B-spline ensures that the trajectory meets the restricted range.

Obstacle avoidance optimization

When a trajectory collides with an obstacle, the control point of the collision is selected, and the corresponding point of the control point is generated on the surface of the obstacle. Each control point C_i may have more than one corresponding point $\rho_{i,j}$ on the obstacle surface, but each corresponding point $\rho_{i,j}$ belongs to only one control point. According to the corresponding repulsive direction vector $H_{i,j}$ generated by the positions and $\rho_{i,j}$, the relationship between the three is shown in Fig. 2. i coincide with the index of the original control point, and j is the index of the corresponding point and the opposite direction vector. The distance between the obstacle surface point and the original control point is calculated as shown in Eq. (22).

$$D_{i,j} = \left(C_i - \rho_{i,j}\right) \cdot H_{i,j} \tag{22}$$

Since the trajectory is continuously collision detected and optimized, to avoid the resulting duplicates of $H_{i,j}$ and $\rho_{i,j}$, we consider obstacles encountered by the control points C_i as newly discovered obstacles only by restricting $D_{i,j} > 0$. This restriction helps us to find obstacles that affect the new trajectory, reducing the amount of computation that can be done to update the new trajectory faster.

Environment as an important reference factor in UAV obstacle avoidance algorithms, a large amount of environmental data must be stored as a prerequisite for obstacle avoidance, and at the same time, appropriate trajectory algorithms are established to avoid obstacles. This means that there is a large amount of data to be computationally processed, which is undoubtedly an insurmountable problem for the arithmetic power of micro UAVs. The article algorithm only needs to store the environmental data in the narrow space near the B-spline trajectory, while the repulsive force algorithm is simple and efficient, which greatly reduces the amount of computation[27]. Moreover, because it does not require a lot of environmental information, it avoids the problem of UAVs falling into local minima and does not require collision-free trajectories to be generated in advance.

Since the trajectory is generated based on the B-spline curve, it is affected by the selection of basis function and control point, the B-spline curve structure is shown in Fig. 3. The distance between control points has a great influence on trajectory generation. If the distance between the trajectory control points is too large, the trajectory deformation caused by each iteration of the trajectory will be large, and unnecessary UAV performance waste will occur when facing small obstacles. If the distance between control points is too small, more iterative calculations are needed and even failed to planning when facing large obstacles.

Gradient-based trajectory optimization

According to Eq. (12) it can be seen that in the B-spline parameterization process, we use a uniform B-spline so that the time interval Δt in the control points is the same and very small, so the UAV velocity v_i, acceleration a_i, and jerk j_i between the two control points can be expressed as:

$$\begin{cases} v_i = \dfrac{C_{i+1} - C_i}{\Delta t} \\ a_i = \dfrac{v_{i+1} - v_i}{\Delta t} \\ j_i = \dfrac{a_{i+1} - a_i}{\Delta t} \end{cases} \tag{23}$$

The trajectory planning task for UAVs is carried out in a differentially flat space with set control points C_i, $C_i \in R^3$, so the trajectory optimization process transforms into:

$$J = \lambda_s J_s + \lambda_c J_c + \lambda_f J_f \tag{24}$$

where J_s is the smoothness penalty, J_c represents the collision penalty, J_f represents the feasibility penalty, λ_s, λ_c and λ_f are the weights of the three penalties.

Smoothness penalty: the smoothness penalty is formulated as the curvature of the trajectory, the smoothness of the trajectory is judged by the geometric information of the human–computer trajectory, if the higher curvature of the trajectory represents more difficulty for the UAV to track the trajectory. Therefore, the functional expression of the smoothing penalty is:

$$J_s = \sum_{i=0}^{n} \sqrt{|a_i|^2 |v_i|^2 - (v_i \cdot a_i)^2 / |v_i|^3} \tag{25}$$

Minimizing each control point acceleration a_i and jerk j_i, makes the whole trajectory smooth.

Collision penalties: the purpose of establishing collision penalties is to keep the UAV away from obstacles and keep the two at a safe distance s_f. A segmentation function is established to determine the rate of change of the collision penalty based on the size of $D_{i,j}$. Therefore, the expression of the collision penalty function for the corresponding point $\rho_{i,j}$ of each control point is:

$$\begin{gathered} j_c(i,j) = \begin{cases} 0 \left(w_{i,j} \le 0 \right) \\ w_{i,j}^3 \left(0 < w_{i,j} \le s_f \right) \\ 3s_f w_{i,j}^2 - 3s_f^2 c_{i,j} + 3s_f^3 \left(w_{i,j} > s_f \right) \end{cases} \\ w_{i,j} = s_f - D_{i,j} \end{gathered} \tag{26}$$

where $j_c(i,j)$ is the cost of generating the repulsive direction vector $H_{i,j}$ at the corresponding point $\rho_{i,j}$ on the control point C_i. The cost of each control point is calculated independently, and when a control point generates multiple repulsive direction vectors, it represents an increase in the cost of obstacle avoidance. Thus, the collision penalty incurred by a control point is calculated as:

$$J_c(i) = \sum_{j=1}^{e} j_c(i,j) \tag{27}$$

where e is the number of control points generating repulsive direction vectors $\mathbf{H}$. The cost of a UAV trajectory collision is the accumulation of the collision penalty $J_c(i)$ for all control points. Therefore, the total cost of the collision penalty is:

$$J_c(i) = \sum_{j=1}^{e} j_c(i,j) \tag{28}$$

To obtain the total cost of collision J_c, we obtain the gradient by deriving it with the expression:

$$\frac{\partial J_c}{\partial C_i} = \sum_{i=1}^{n} \sum_{j=1}^{e} H_{i,j} \begin{cases} 0 \left(w_{i,j} \le 0 \right) \\ -3w_{i,j}^2 \left(0 < w_{i,j} \le s_f \right) \\ -6s_f w_{i,j} + 3s_f^2 \left(w_{i,j} > s_f \right) \end{cases} \tag{29}$$

Feasibility penalty: we need to constrain the UAV trajectory in x, y, and z directions to ensure that the speed, acceleration, and jerk of the UAV in each dimension satisfy the UAV performance constraints. Due to the convex envelopment of the B-spline, we can restrict the UAV trajectory by restricting the derivatives of the control points.

$$J_f = \sum_{i=1}^{n} w_v F(v_i) + \sum_{i=1}^{n-1} w_a F(a_i) + \sum_{i=1}^{n-2} w_j F(j_i) \tag{30}$$

where w_v, w_a and w_j are the weights of the feasibility penalty in terms of velocity, acceleration and jerk. F is a quadratically continuous differentiable function of the higher-order derivatives of the control points.

$$F(c) = \sum_{r=x,y,z} f(c_r) \tag{31}$$

$$f(c_r) = \begin{cases} a_1c_r^2 + b_1c_r + c_1\,(c_r \le -c_j) \\ (-\lambda c_m - c_r)^3\,(-c_j < c_r < -\lambda c_m) \\ 0\,(-\lambda c_m \le c_r \le \lambda c_m) \\ (c_r - \lambda c_m)^3\,(\lambda c_m < c_r \le c_j) \\ a_2c_r^2 + b_2c_r + c_2\,(\,c_r \ge c_j) \end{cases} \tag{32}$$

where $c_r \in C \in \{v_i, a_i, j_i\}$, a_1, b_1, c_1, a_2, b_2 and c_2 are able to satisfy the second-order continuity of the function F. c_m is the limit of the derivatives, c_j is the node of the second and third intervals. λ is the elasticity coefficient and $0 < \lambda < 1$ to make the final result satisfy the constraints.

Time adjustment

The time T_i of an individual trajectory segment P_i is mainly determined by the number and position of control points. In the initialization phase of the trajectory, we assign times to individual segments. When the UAV changes its path due to an obstacle, the speed V_i, acceleration A_i and jerk J_i of the UAV exceed the UAV's own limits, needs to adjust the time, the expression for the time conversion ratio is:

$$u = \max\left\{|v_m/V_i|, \sqrt{|a_m/A_i|}, \sqrt[3]{|j_m/J_i|}\right\} \tag{33}$$

where v_m, a_m and j_m respectively represent maximum speed, acceleration, and jerk on the x, y, and z-axis. The time for the UAV to complete the reassignment of the path fragments is as follows:

$$T_{i'} = uT_i \tag{34}$$

Numerical solution

This paper proposes the use of gradient for trajectory optimization, which can be seen as solving extreme value problems of multivariate objective functions[28]. The optimal approach for solving such unconstrained optimization problems is to use the Newton method. By utilizing the solved data during the objective function generation process, the requirement for the fast restart of trajectory computation can be met, and repeated computation can be avoided, thereby improving the solving speed. However, solving the UAV trajectory requires extremely high real-time performance. In the solving process of Newton's method, the inverse of the Hessian matrix needs to be computed, which exceeds the computational power of the UAV. Therefore, the proposed quasi-Newton algorithm is used for solving[29–31].

There are various quasi-Newton algorithms, and in this paper, the L-BFGS algorithm is utilized[32–34]. This algorithm ensures a high success rate in obstacle avoidance and also exhibits good performance in terms of solving accuracy and restart loss. The solution process is as follows, $x \in R^3$, $f(x)$ and the update formula for x is:

$$x_{k+1} = x_k - \lambda H_k^{-1}\nabla f_k \tag{35}$$

where λ is the learning rate and the iterative formula for H_K is:

$$H_{k+1} = V_k^T H_k V_K + \rho_k s_k s_k^T \tag{36}$$

where $\rho_k = (y_k^T s_k)^{-1}$, $v_k = I - \rho_k y_k s_k^T$, $s_k = x_{k+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$. H_K also does not need to be solved explicitly, and H_K is approximated by B_k. Thus, the proposed Newtonian condition is:

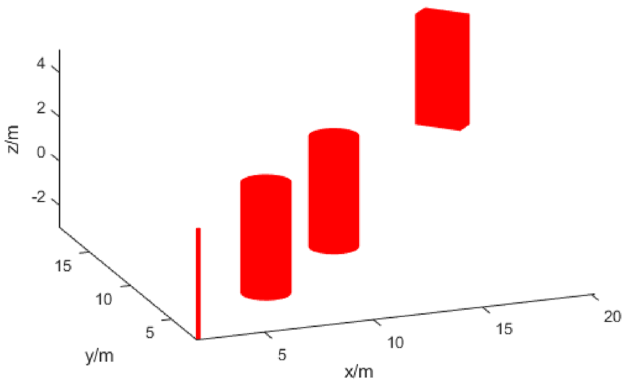

Figure 4. Simple experimental environment.

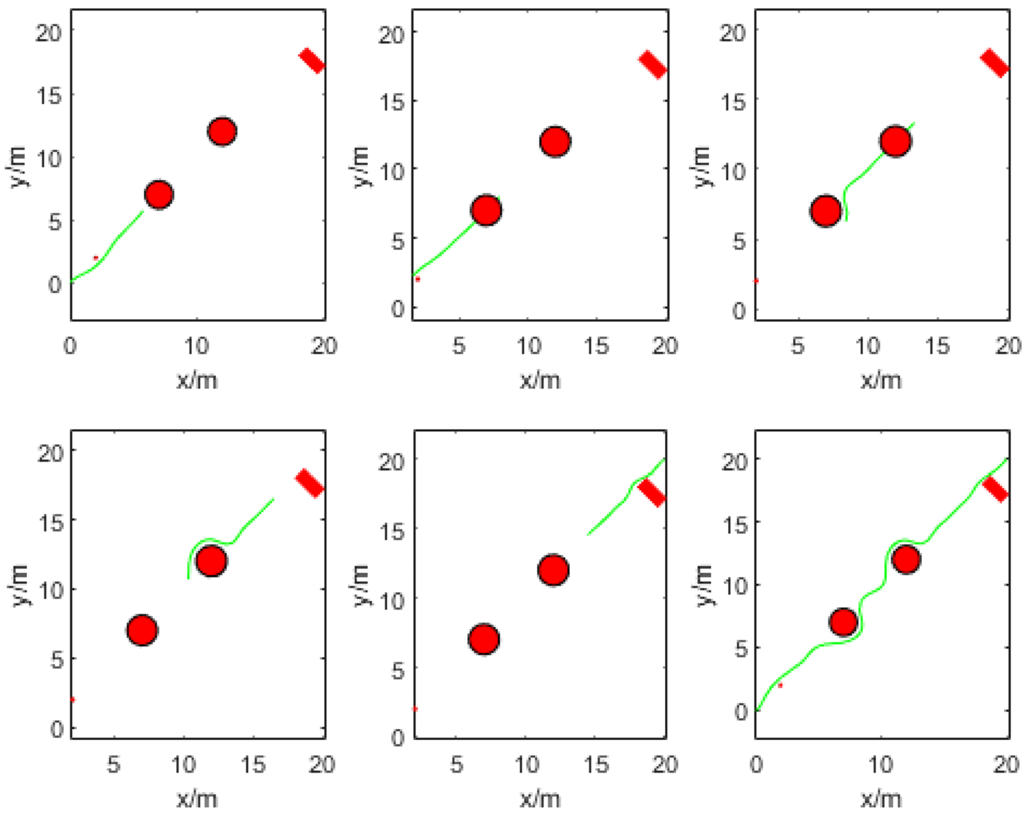

Figure 5. Path generation process in a simple obstacle environment.

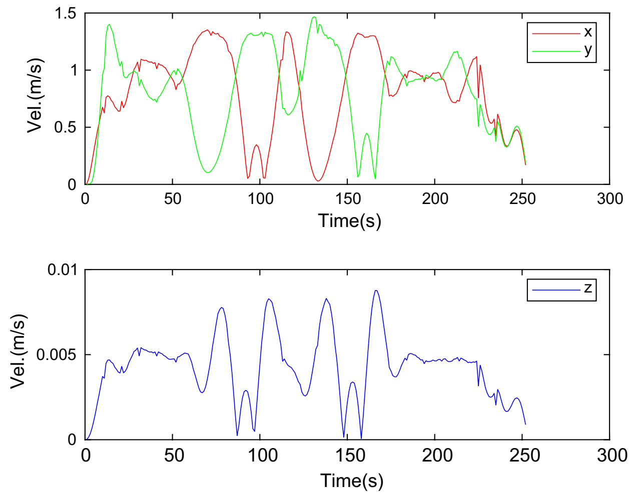

Figure 6. UAV three-axis speed in simple obstacle environment.

$$B_{k+1}s_k = y_k \tag{37}$$

$$B_{k+1} = B_k + P_k + Q_k \tag{38}$$

Make P_k and Q_k satisfy:

$$\begin{cases} P_k s_k = y_k \\ Q_k s_k = -B_k s_k \end{cases} \tag{39}$$

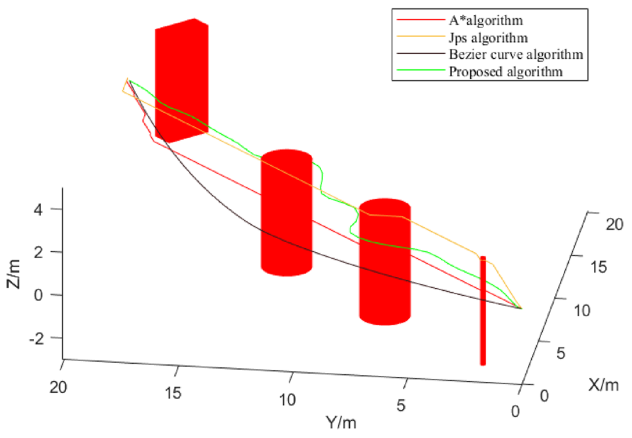

Figure 7. Comparison of trajectories in Experiment 1.

Algorithms	Lengths (m)	Time (s)	Nodes used
A*	34.705	20.15	1540
JPS	33.529	19.29	3126
Bezier curve	31.572	20.52	1728
Proposed	30.956	20.32	236

Table 1. Path comparison of the three algorithms.

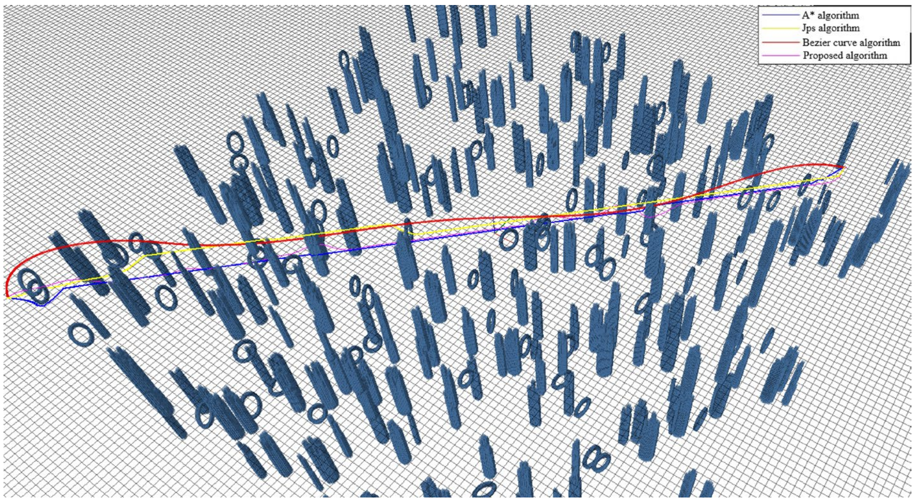

Figure 8. Comparison of trajectories in Experiment 2.

Algorithms	Lengths (m)	Time (s)	Nodes used
A*	73.24	42.32	3751
JPS	72.85	41.25	23,466
Bezier curve	73.59	42.68	3962
Proposed	72.30	40.11	952

Table 2. Comparison of paths in complex environments.

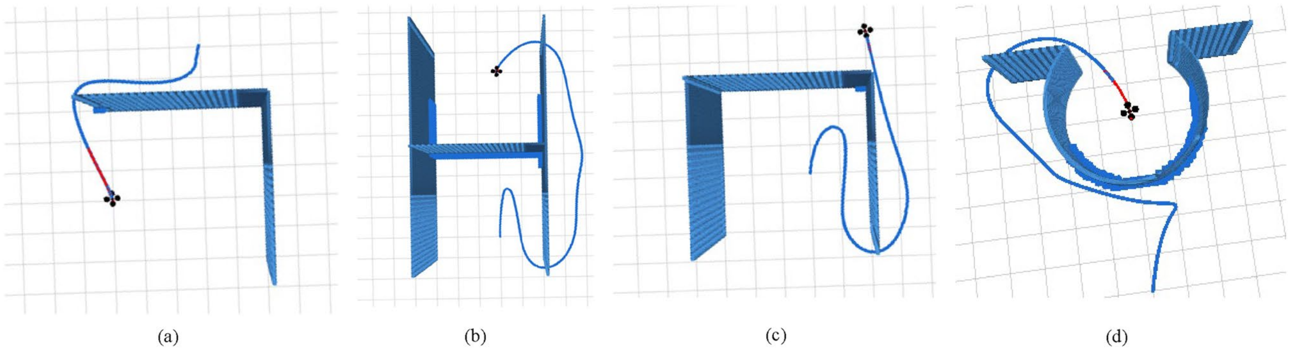

Figure 9. Obstacle avoidance trajectory of UAV in special obstacle environments: (**a**) L-shape obstacle; (**b**) H-shape obstacle; (**c**) U-shape obstacle; (**d**) Omega shape obstacle.

The iterative formula for the algorithmic matrix B_{k+1} can be obtained after obtaining the appropriate P_k and Q_k:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \tag{40}$$

The L-BFGS algorithm requires H_k^0 to be a positive definite matrix in order to ensure gradient descent. To achieve convergence, a monotone line search is performed under strong Wolfe conditions, with the Hessian matrix H_k^0 being used.

$$H_k^0 = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} I \tag{41}$$

Experiment results and analysis

Utilizing the ROS platform[35], we conducted UAV autonomous flight experiments across various simulated environments and compared them with other algorithms to assess the proposed algorithm's effectiveness. During the experiment, the UAV faced limitations in acquiring global obstacle information. When approaching an obstacle, the UAV was able to gather local obstacle information through a simulated camera. This information may not even be all the information about an obstacle.

Initially, the simulation experiment was conducted in a setting with simple obstacles. In experiments, the UAV flew from an initial point to a target point at a constant speed and stable attitude, while traversing an area with four obstacles. Four obstacles were distributed within the simulation map of [25.0 m, 25.0 m, 5.0 m], consisting of three cylinders with varying thicknesses and a wall. Experiments can intuitively reflect the trajectory generation process of UAVs. When the trajectory generated in the previous iteration fails to avoid obstacles, the proposed algorithm can quickly generate a new path. The experimental environment is depicted in Fig. 4.

The UAV trajectory is generated in segments, without considering collision during the initialization process. The B-spline order is 3, resulting in approximately 25 control points per segment of the trajectory. Each segment has a horizontal length of around 7 m, with an initial distance interval of about 0.3 m between neighboring points. The safety distance between the UAV and obstacles is set at 0.8 m. These parameters are considered optimal as they effectively balance the degrees of freedom and complexity of the UAV trajectory. Experiment 1 involves generating the UAV running trajectory using the B-spline algorithm in a simple obstacle environment. The total length of the trajectory is 30.956 m. The process of UAV running obstacle avoidance is illustrated in Fig. 5, and the UAV's three-axis speed is shown in Fig. 6.

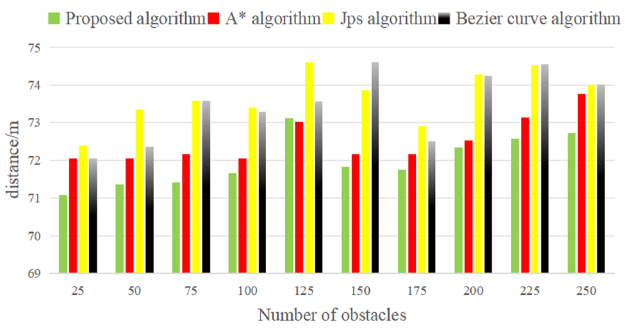

Figure 10. Comparison of UAV trajectory lengths.

In this paper, the proposed path planning algorithm is compared with two traditional graph search algorithms, A* and Jump Point Search (JPS), as well as the Bezier curve fitting algorithm. Three algorithms require sensing most of the obstacles in the environment in order to discover the optimized path. The article algorithm focuses more on path optimization compared to the other three algorithms, particularly in a simple environment, resulting in smaller data processing. The paths generated by the three algorithms in the identical experimental environment are compared, and the results are shown in Fig. 7.

Based on the findings presented in Fig. 7 and Table 1, it can be observed that the A* and JPS algorithms demonstrate better performance in terms of time efficiency in simple environments. This is mainly because the obstacle environment is relatively simple, and the data processing amount is too small. When faced with more complex environments, the advantages of algorithms will be shown. Meanwhile, these algorithms are not as effective as the algorithm proposed in the paper in terms of the trajectory length and the number of nodes used. On the other hand, the B-spline trajectory planning method algorithm and Bezier curve algorithm generate smoother trajectories that are well-suited for UAV dynamics and can be easily implemented in real-life scenarios.

To further validate the effectiveness of the algorithm proposed in this paper, Experiment 2 randomly generated 300 cylindrical obstacles and Circular obstacles within a square range of 50 m on each side. The position, radius, and height of each cylindrical obstacle were randomly generated, and there was no overlap between the obstacles. Three path planning methods operate independently in the same environment mentioned above. The UAV initiates its flight from the coordinates [− 25, − 25, 5] and navigates towards the final destination point[5,25]. The resulting trajectories generated by the UAV using the three trajectory calculation methods are illustrated in Fig. 8.

The results presented in Fig. 8 and Table 2 indicate that the proposed algorithm effectively tackles real-time obstacle avoidance in randomized and complex environments. While the A*, JPS, and Bezier curve algorithms also demonstrate success in solving the obstacle avoidance problem during the experiment, they struggle to handle the significant amount of data generated by more complex environments, leading to suboptimal solutions. This highlights the superior stability of the algorithms proposed in this paper. The trajectory shape in Fig. 8 demonstrates that the proposed algorithm in the paper primarily achieves obstacle avoidance through in-plane roundabouts. This approach ensures the stability of the UAV while increasing the solving speed and generating a smoother UAV trajectory.

In addition, we conducted the autonomous flight obstacle avoidance simulation experiment of UAVs in L, H, U, and omega under four special obstacle environments, and the generated UAV trajectories are shown in Fig. 9. In these cases, the algorithm still provides a safe and smooth UAV trajectory that allows the UAV to traverse these obstacles.

When the UAV passes L-shaped obstacles, a smooth trajectory is generated, as shown in Fig. 9a. The UAV initially fits the control points by polynomial and connects the control points to generate the trajectory. When no obstacle is found, the trajectory flies along the line between the starting point and the destination place, as shown in Fig. 9b,c, which may cause the UAV to waste kinetic energy. Of course, this is limited by the UAV's ability to sense obstacles, and similar situations can be avoided by carrying better sensors. Secondly, when the algorithm pursued smoothness, the UAV circled a part of the distance, as shown in Fig. 9c,d. However, the UAV is easier to complete the smoother trajectories than the sharp ones without requiring global obstacle information, and it has higher adaptability to the unknown environment.

Under the same conditions of obstacle generation methods, boundary constraints, and route constraints, the four algorithms operate independently in environments with ten different numbers of obstacles. The comparison of trajectory lengths generated by the four algorithms is presented in Fig. 10. In 10 obstacle density environments, the average trajectory lengths for the four algorithms are 71.98 m, 72.50 m, 73.69 m, and 73.47 m respectively. The comparison in Fig. 10 reveals that the proposed algorithm exhibits a shorter trajectory and runtime.

Conclusion

This paper focuses on developing a B-spline curve-based UAV path planner to achieve a smoother and more suitable flight path. The proposed algorithm demonstrates comparable performance in planning distance and time to other advanced path planning algorithms while producing a trajectory that is better suited for UAV flight. In addition, the algorithm only requires obstacle information in the narrow space around the trajectory, leading to reduced data processing and enabling the UAV to navigate through unknown obstacle environments swiftly and safely. Simulation experiments confirmed the effectiveness and reliability of the algorithm.

Data availability

The data that support the findings of this study are available on request from the corresponding author, Pengxiang Sun, upon reasonable request.

Received: 21 October 2023; Accepted: 20 June 2024
Published online: 24 June 2024

References

1. Xinyu, L. *et al.* Agricultural UAV trajectory planning by incorporating multi-mechanism improved grey wolf optimization algorithm. *Expert Syst. Appl.* **233**, 1 (2023).
2. Cheng, Z. *et al.* Automated UAV image-to-BIM registration for building façade inspection using improved generalised Hough transform. *Autom. Constr.* **153**, 104957 (2023).
3. Pestana, J. *et al.* Overview obstacle maps for obstacle-aware navigation of autonomous drones. *J. Field Robot.* **36**(4), 734–762 (2019).
4. Yuxuan, F. *et al.* Piecewise-potential-field-based path planning method for fixed-wing UAV formation. *Sci. Rep.* **13**(1), 2234–2234 (2023).

5. Zheng Li, Yu. *et al.* Particle swarm algorithm path-planning method for mobile robots based on artificial potential fields. *Sensors* **23**(13), 6082 (2021).
6. Lei, Z. H. L. Intelligent control of swarm robotics employing biomimetic deep learning. *Machines* **9**(10), 236–236 (2021).
7. Kaustubh, C. *et al. Image invariant robot navigation based on self organising neural place codes* 88–106 (Springer, 2005).
8. Denis, S. *et al. Spatial representation and navigation in a bio-inspired robot* 245–264 (Springer, 2005).
9. Chengwei, Z., & Qi, F. Research on UAV path planning combined with ant colony and A*, pp. 1228–1236 (Springer Nature Singapore, 2023).
10. Chai, R. *et al.* Real-time reentry trajectory planning of hypersonic vehicles: A two-step strategy incorporating fuzzy multiobjective transcription and deep neural network[J]. *IEEE Trans. Ind. Electron.* **67**(8), 6904–6915 (2019).
11. Chai, R. *et al.* Attitude tracking control for reentry vehicles using centralised robust model predictive control [J]. *Automatica* **145**, 110561 (2022).
12. Lindqvist, B. *et al.* Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles[J]. *IEEE Robot. Autom. Lett.* **5**(4), 6001–6008 (2020).
13. Mansouri, S. S. *et al.* A unified nmpc scheme for mavs navigation with 3d collision avoidance under position uncertainty[J]. *IEEE Robot. Autom. Lett.* **5**(4), 5740–5747 (2020).
14. Quan, L. *et al.* Survey of UAV motion planning. *IET Cyber-Syst. Robot.* **2**(1), 14–21 (2020).
15. Li, D., Ren, X., Gu, S., *et al.* Attitude calculation of quadrotor UAV based on gradient descent fusion algorithm, pp. 351–360 (Springer Nature Singapore, 2022).
16. Ratliff, N., Zucker, M., Bagnell, J. A., & Srinivasa, S. Chomp: Gradient optimization techniques for efficient motion planning. In *Proc IEEE Int. Conf. Robot. Autom*, pp. 489–494 (2009).
17. Zhongyang, M., Huaguang, Z., Juan, Z. *et al.* A novel actor–critic–identifier architecture for nonlinear multi-agent systems with gradient descent method. *Automatica* **155** (2023).
18. Kalakrishnan, Z. M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. Stomp: Stochastic trajectory optimization for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom*, pp. 4569–4574 (2011).
19. Sucan, I. A., Kalakrishnan, M., & Chitta, S. Combining planning techniques for manipulation using realtime perception. In *IEEE International Conference on Robotics and Automation* (2010).
20. Aijuan, L. *et al.* Map construction and path planning method for a mobile robot based on multi-sensor information fusion. *Appl. Sci.* **12**(6), 2913–2913 (2022).
21. Wenchao, D. *et al.* An efficient b-spline based kinodynamic replanning framework for quadrotors. *Trans. Robot.* **35**(6), 1287–1306 (2019).
22. Xin, Z. *et al.* EGO-planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robot. Autom. Lett.* **6**(2), 478–485 (2021).
23. Zhenping, W. *et al.* A vision-based approach for autonomous motion in cluttered environments. *Appl. Sci.* **12**(9), 4420–4420 (2022).
24. Theodorou, E., Buchli, J., & Schaal, S. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *IEEE International Conference on Robotics and Automation* (2010).
25. Yuan, Li. *et al.* Smooth trajectory planning for a cable driven parallel waist rehabilitation robot based on rehabilitation evaluation factors. *Chinese J. Mech. Eng.* **36**(1), 1–13 (2023).
26. Shamaila, S. & Muhammadl, S. Mohamed Abullah. A quadratic trigonometric B-Spline as an alternate to cubic B-spline. *Alex. Eng. J.* **61**(12), 11433–11443 (2022).
27. Yifei, H. *et al.* A path smoothing scheme for micro-line tools using cubic B-spline fitting with dominant points. *J. Phys. Conf. Ser.* **2355**(1), 012018 (2022).
28. Orliński, M. & Jankowski, N. Fast t-SNE algorithm with forest of balanced LSH trees and hybrid computation of repulsive forces. *Knowl. Based Syst.* **206**, 106318 (2021).
29. Liu Yucong, Yu. & Shixing, L. T. Hessian regularization of deep neural networks: A novel approach based on stochastic estimators of Hessian trace. *Neurocomputing* **536**, 13–20 (2023).
30. Abhishek Kumar, K. & Parhi Dayal, R. Dynamic walking of multi-humanoid robots using BFGS Quasi-Newton method aided artificial potential field approach for uneven terrain. *Soft Comput.* **27**(9), 5893–5910 (2022).
31. Kadir, K. A comparison of Quasi-Newton methods considering line search conditions in unconstrained minimization. *J. Inf. Optim. Sci.* **43**(8), 2031–2053 (2022).
32. Nguyen, D. T. *et al.* Training the RBF neural network-based adaptive sliding mode control by BFGS algorithm for omni-directional mobile robot. *Int. J. Mech. Eng. Robot. Res.* **7**(4), 367–373 (2018).
33. Gao, F. *et al.* Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Trans. Robot.* **36**(5), 1526–1545 (2020).
34. Keung, L. K. *et al.* A modified q-BFGS algorithm for unconstrained optimization. *Mathematics* **11**(6), 1420 (2023).
35. Angelos, A., Lagoudakis, M. G. & Panagiotis, P. A ROS multi-tier UAV localization module based on GNSS, inertial and visual-depth data. *Drones* **6**(6), 135 (2022).

Acknowledgements
The authors gratefully acknowledge the support of the Discipline Innovation Team of Liaoning University of Engineering and Technology (LNTU20TD-06), Liaoning Province Applied Basic Research Program (2022JH2-101300231).

Author contributions
Wei Sun, Pengxiang Sun, and Wei Ding wrote the main manuscript text, Jingang Zhao, and Yadan Li prepared Figs. 5, 6, 7 and 8.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to P.S.