A Gradient Descent-Based Backend Feedback Adaptive Motion Planning Algorithm for Autonomous Mobile Robots

Gang Li, Si-Cheng Wang, Bin Cheng, and Zhong-Pan Zhu

***Abstract*—This paper introduces an innovative motion planning algorithm for autonomous mobile robots, specifically focusing on quadrotor Unmanned Aerial Vehicles (UAVs), utilizing a gradient descent-enhanced frontend and backend architecture. A trajectory planning algorithm is proposed for the front-end part. It relies on backend optimization feedback and memorized jump points. The algorithm builds on the jump point search (JPS) algorithm and introduces an obstacle table and jump point table. A new heuristic function is proposed, which emphasizes the weight of obstacle proportion in order to avoid getting stuck in local optimal paths. In the backend trajectory optimization part, a backend space-time trajectory optimization method based on gradient descent is proposed, and an optimization objective function is designed to ensure the smoothness and safety of the UAV trajectory. The simulation results show that the algorithm proposed in this paper has significant advantages for improving real-time performance and environmental adaptability compared with the method based on ESDF and the EGO-planner. The actual flight experiments show that the proposed algorithm can avoid UAVs getting stuck in local optima during path planning. Notably, the proposed methodology also holds promise for application in path planning for other autonomous robots.**

***Index Terms*—Gradient descent, motion planning, quadrotor UAV, real-time.**

I. Introduction

AUTONOMOUS mobile robots, particularly quadrotor unmanned aerial vehicles (UAVs), have found widespread application in diverse scenarios such as detection, inspection, rescue, and exploration [1]–[4]. Attributed to their advantages of low cost, compact size, and exceptional maneuverability, these robots are utilized extensively in both military and civilian domains, holding substantial promise for future advancements. UAVs with autonomous flying technology can efficiently plan flight routes in unfamiliar surroundings, which is one of the essential basic capabilities for many UAV application directions. Faced with various dynamic scenes with a sharp increase in complexity, traditional motion planning algorithms have suffered from problems such as insufficient real-time performance, low success rates, and insufficient environmental adaptability. As a result, it is critical to investigate novel algorithms based on existing motion planning techniques.

The path planning algorithm in the field of mobile robots has a long history, such as the A* algorithm based on heuristic search [5] and the improved RRT (Rapidly Exploring Random Tree) algorithm [6]–[8]. In 2011, two Canadian professors, Harabor and Grastien [9], proposed the JPS (Jump Point Search) algorithm based on the A* algorithm model, which optimized searching for successor nodes. This algorithm is increasingly widely used in mobile robots, games, and other fields.

The UAV motion planning algorithm has been the subject of extensive research by numerous academics. It can be broadly categorized into three types: learning-based approaches, gradient-based methods, and methods based on viable region limitations. Liu *et al.* [10] proposed the RILS algorithm (Regional Inflation by Line Search). This algorithm locally takes the line segment as the initial seed of the feasible region and continuously calculates the largest ellipsoidal feasible space containing the line segment and excluding obstacles as the final feasible region convex polyhedron. Gao *et al.* [11] designed a PCCI algorithm (Parallel Convex Cluster Inflation) to calculate a convex feasible region for a three-dimensional occupancy voxel map. This algorithm employs a designated secure voxel point as the initial seed for subsequent expansion of the feasible domain, continuously expanding and absorbing new voxels visible to the existing voxel set. Savin [12] proposed a space-flipping algorithm, which takes the seed point as the center and transforms all obstacle points set by spherical pole mapping. By flipping the feasible region around the seed point and clicking the obstacle after flipping to calculate the convex hull, the region outside the convex hull corresponding to the region before flipping can be used to approximate the feasible region near the seed. Deits *et al.* [13] proposed an IRIS algorithm (Iterative Region Inflation by Semidefinite Programming). The algorithm selects the ellipsoid as the seed. It selects the inscribed ellipsoid with the maximum volume of the convex polyhedron obtained in the previous iteration as the new seed to continue a new round of expansion in subsequent

This work was supported by the National Key Research and Development Program of China (Grant No. 2023YFB4704400), National Natural Science Foundation of China (Grant Nos. 62273262, 62422314 and 62088101), Shanghai Science and Technology Commission Project (Grant Nos. 22QA1408500 and 2021SHZDZX0100), Aeronautical Science Foundation (2024M071038001), Industry-University-Research Cooperation Fund of the Eighth Research Institute of China Aerospace Science and Technology Corporation (SAST2023-019), China University Industry, University and Research Innovation Fund (Grant No. 2021ZYA03004), Special Fund for Independent Innovation of Aero Engine Corporation of China (Grant No.ZZCX-2021-007), Eastern Talent Program, Shanghai Pilot Program for Basic Research, and Fundamental Research Funds for the Central Universities.

Corresponding author: Gang Li. (e-mail:lig@tongji.edu.cn)

The authors are with the College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China, with the Shanghai Research Institute for Intelligent Autonomous Systems, Shanghai 200120, China, and also with the National Key Laboratory of Autonomous Intelligent Unmanned Systems, Shanghai 200120, China.

iterations.

Zucker *et al.* [14] proposed the covariant Hamiltonian optimization for motion planning (CHOMP) method for high-degree-of-freedom robots such as manipulators and legged robots. The CHOMP method uses ESDF and difference information for the distance field to construct the optimization objective function, which leads to deterioration of the theoretical convergence rate of the optimization method [15], and the computational efficiency of iteration in the actual process is further reduced. There are some variant algorithms based on the CHOMP method, such as the STOMP algorithm [16] (Stochastic Trajectory Optimization for Motion Planning), NFOMP algorithm [17] (Neural Field for Optimal Motion Planner), and ITOMP algorithm [18] (Incremental Trajectory Optimization). Based on the CHOMP method, these algorithms improve and enhance random optimization and deal with highly dynamic obstacles to a certain extent. Unlike previous researchers, Shulman *et al.* [19] built sequential convex programming with an L1 penalty term for the trajectory planning of rigid robots. They used convex geometric units to express all obstacles in the environment.

Tu *et al.* [20] used the Q-learning algorithm to train UAV path planning and obstacle avoidance on Microsoft's AirSim platform. In a 2021 paper, ETH Zurich [21] adopted an end-to-end approach that uses only onboard sensors and computing resources to control UAV flight. The control strategy is trained in the virtual environment. Based on predicting nonlinear optimal control problems, Tang *et al.* [22] combined neural networks and further used quadratic optimization to improve the quality of the trajectories. The tracking of the mission objectives has been completed level by level. Qureshi *et al.* [23], [24] proposed motion planning networks for the encoder-decoder structure. By combining the sampling-based planning method with online and offline supervised learning, one can dramatically exceed Informed-RRT*. Ichter *et al.* [25], [26] combined the neural network, different from the traditional method, first predicted the distribution probability of the optimal trajectory in space and performed nonuniform sampling in places with high random probability to improve efficiency. Ichter and several other scholars proposed the Critical PRMs method [27], which also introduces a learning method for prediction to obtain key nodes with high connectivity in the connected graph and adds them to the PRM graph to significantly improve the sampling efficiency of the PRM. Hua *et al.* [28] proposed a learning-based trajectory generation framework for quadrotors, which makes human-like decisions online through reinforcement learning (RL) and imitation learning (IL). Tiffany *et al.* [29] proposed a path-planning algorithm for outdoor robots, which is based on neuronal spike timing. A novel learning rule was developed and demonstrated its effectiveness on real-world environmental maps.

In summary, the advantage of existing feasible domain-constrained methods is that the trajectories obtained under hard constraints are safe. However, the hard constraint solution is time consuming, and there is a contradiction between the solution speed and conservatism of the feasible space. The effect of learning-based motion planning in specific scenarios is significantly improved compared with nonlearning traditional methods [30], but the generalization performance still needs to be improved. The optimal solution method based on gradient information can be used to reach a solution faster under soft constraints, and the overall length of the obtained trajectory is shorter. Its disadvantage is that the optimized result may still not satisfy the constraints, and it is necessary to process environmental information and introduce additional calculations. At present, the optimization algorithm based on gradient descent is superior in terms of real-time performance and versatility and can ensure safety, efficiency, and dynamic feasibility.

For this reason, based on the gradient descent algorithm of the front and rear ends, an adaptive motion planning algorithm based on the feedback of the rear end is proposed. This paper conducts research on the basis of the JPS algorithm, and the main contributions are as follows:

- A new algorithm framework has been proposed, which combines front-end path planning and back-end trajectory optimization. In the front-end path planning section, we improved the JPS algorithm by introducing obstacle tables and jump point tables and proposed a new heuristic function that incorporates weights based on obstacle density. By adjusting the value of λ in the heuristic function online, adaptive path adjustment can be achieved to avoid getting stuck in local optimal paths.
- In the backend trajectory optimization section, a gradient descent trajectory optimization algorithm based on guidance of the adjacent safety zero point is proposed. When the optimized path cannot meet the dynamic constraints, local path re-partitioning is performed, and the adjusted λ value is fed back to the front-end to increase the priority of selecting low obstacle density areas.
- We designed a UAV hardware and software framework suitable for proposing the method and deployed the algorithm on the UAV for actual flight experiments. In specific areas with dense obstacles, the proposed method outperforms several commonly used methods, including EGO-planner, in terms of path optimization time and success rate.

The rest of this work is organized as follows: In Section II, a framework for motion planning algorithms is proposed, and a global trajectory manager is designed. In Section III, in the front-end path optimization part, a memorized jump-point path planning algorithm based on backend optimization feedback is proposed. In the backend trajectory optimization part, a gradient descent trajectory optimization algorithm based on guidance of the adjacent safety zero point is proposed. In Section IV, an actual flight experiment for a UAV and a comparative experiment for four motion planning algorithm frameworks are carried out. In Section V, the conclusion is finally given.

II. Algorithm Framework

The environment faced by UAVs is often unknown and dynamic, and obstacles can appear at any time on the pre-planned flight path. Therefore, during flight, it is necessary to maintain the perception of the surrounding environment and

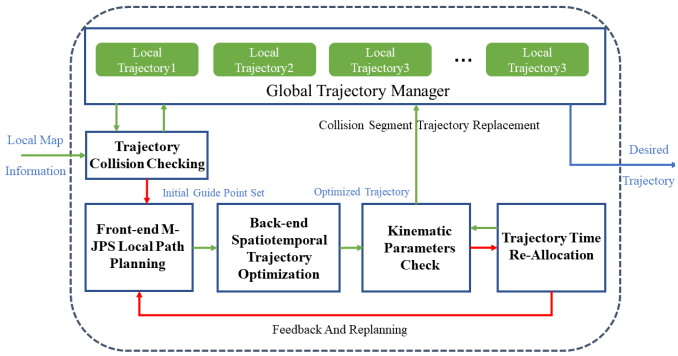

Fig. 1. Algorithm framework schematic diagram.

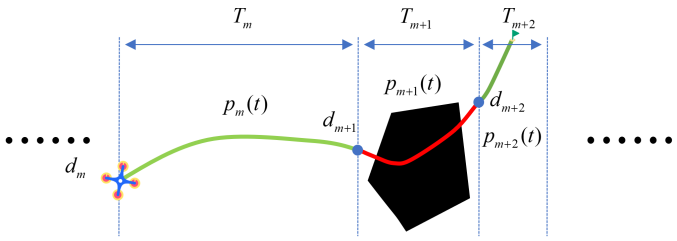

Fig. 2. A path that will collide appears on the global path of the UAV (red path).

change the flight trajectory in a timely manner in the face of upcoming obstacles that can collide with the UAV. Any change in the global trajectories will lead to many redundant calculations, which is not conducive to avoiding obstacles with high real-time performance. The most efficient way to solve this problem is to only modify the trajectory where the collision occurred. To this end, this paper proposes a backend feedback adaptive quadrotor UAV motion planning algorithm framework based on gradient descent, as shown in Fig. 1.

The framework comprises a global trajectory manager, a front-end path planning module, a back-end trajectory optimization module, and a trajectory time re-allocation module [31]. The global trajectory manager will detect collisions on the trajectory segment at the current time t_{cur}. If there is no collision, the RPC mechanism of the ROS system is used to publish to the trajectory tracking module to control the UAV to track the trajectory. If a collision is detected, it is necessary to obtain the collision continuous trajectory set Φ_{collide} by first using the memorized jump points search algorithm (hereinafter referred to as M-JPS) proposed in this paper to perform fast path planning according to the starting point and end point of the trajectory to generate an initial safe point set. A local vector distance field is calculated using the control points of the point set and the trajectory Φ_{collide}, and then the objective function is designed to optimize the trajectory using a gradient descent algorithm. After optimization, the kinematic parameters, such as velocity v_i and acceleration a_i, are checked to ensure that the upper limit of the UAV's motion is not exceeded. When the motion data obtained by optimizing the trajectory and matching it with the trajectory segment exceeds the UAV dynamics limit, a time reallocation program needs to be performed.

Regarding the global trajectory manager, in the face of immediate collision with obstacles, in this paper, only the collision trajectory is modified, as shown in Fig. 2.

In Fig. 2, d_m is the intersection point between the trajectory numbered m and the previous trajectory, and $p_m(t)$ is the m-th trajectory. The red local trajectory needs to be replanned, and the original collision trajectory needs to be replaced. Since it is impossible for the UAV to complete sudden changes in speed, position, and acceleration, the new trajectory $p'_{m+1}(t)$ that replaces the $p_{m+1}(t)$ segment trajectory in the figure can be expressed as

$$p'_{m+1}(t) = c^{\mathrm{T}}\beta^{\mathrm{T}}(t) \tag{1}$$

where $\beta(t) = \left[0, t, \ldots t^{n-1}, t^n\right]$, t is the time of the current time relative to the beginning of the collision segment, and c is the trajectory polynomial coefficient matrix. For the new trajectory segment to satisfy position and higher-order derivative continuity at both the beginning and the end of the maneuver, the coefficient matrix c should satisfy the following constraint:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2 & 0 & \cdots & 0 \\ 1 & T_{m+1} & {T_{m+1}}^2 & {T_{m+1}}^3 & \cdots & {T_{m+1}}^n \\ 0 & 1 & 2T_{m+1} & 3{T_{m+1}}^2 & \cdots & n{T_{m+1}}^{n-1} \\ 0 & 0 & 2 & 6T_{m+1} & \cdots & n(n-1){T_{m+1}}^{n-2} \end{bmatrix} c = \begin{bmatrix} d_{m+1} \\ \dot{p}_m(T_m) \\ \ddot{p}_m(T_m) \\ d_{m+2} \\ \dot{p}_{m+2}(0) \\ \ddot{p}_{m+2}(0) \end{bmatrix}. \tag{2}$$

From the above inference, it can be seen that if the replanned path for the UAV is made as short as possible, it is necessary to subdivide the UAV trajectory with small granularity and select a suitable UAV trajectory model to ensure easier calculation and planning and meet the position of the head and tail of the trajectory and the dynamic constraints.

The curve model chosen in this paper is a cubic uniform B-spline curve. The general expression for the B-spline curve is expressed as follows:

$$\Phi(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i \tag{3}$$

where $\Phi(t)$ represents the trajectory of the UAV, P_i represents the control point, $N_{i,k}(t)$ represents the basis function, n represents the order of the curve, and k represents the degree of expression.

The primary function expression for the uniform B-spline curve is given by

$$N_{i,k} = \frac{1}{k!} \sum_{r=0}^{k-i} (-1)^r C_{k+1}^r (t + k - i - r)^k, i = 0, 1, ..., k. \tag{4}$$

Substituting $k = 3$ into (4) yields the basis function of the

cubic uniform B-spline curve as follows:

$$\begin{cases} N_{0,3}(t) = \dfrac{1}{6}(-t^3 + 3t^2 - 3t + 1) \\ N_{1,3}(t) = \dfrac{1}{6}(3t^3 - 6t^2 + 4) \\ N_{2,3}(t) = \dfrac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \\ N_{3,3}(t) = \dfrac{1}{6}t^3 \end{cases} \tag{5}$$

Combining (3) and (5), the initial point position constraint can be obtained as follows:

$$p_0 = \Phi_0(0) = \frac{1}{6}(P_0 + 4P_1 + P_2). \tag{6}$$

Since the primary function of the uniform B-spline curve is periodic, the initial point constraint of the second segment of the B-spline curve $\Phi_1(0)$ is also the end point constraint of the first segment of the curve $\Phi_0(1)$, so the nth curve can be obtained and the initial point constraint is given by

$$\Phi_n(t) = \frac{1}{6}(P_n + 4P_{n+1} + P_{n+2}). \tag{7}$$

Deriving $\Phi_n(t)$, combined with (7) and the relationship between the new control point and original control point, the constraints of the velocity and acceleration at the beginning and end of the curve can be obtained as follows:

$$\begin{cases} \dot{\Phi}_n(t) = \dfrac{1}{2t_s}(P_{n+2} - P_n) \\ \ddot{\Phi}_n(t) = \dfrac{1}{t_s^2}(P_n - 2P_{n+1} + P_{n+2}) \end{cases} \tag{8}$$

where t_s is the planning time for each section of the B-spline curve.

III. Front-end Path Planning and Backend Trajectory Optimization

A. Front-end Path Planning

This paper proposes an M-JPS algorithm based on backend optimization feedback, as shown in Algorithm 1. 3D Jump Point Search (JPS), as shown in Algorithm 2, is a path search algorithm based on the traditional 2D JPS algorithm and extended to adapt to the 3D environment. The core idea is to search for *jump points* [9] in the three-dimensional grid, which are key points that can influence path selection. The algorithm reduces the number of nodes that need to be evaluated by identifying these skip points, thereby improving search efficiency. On the basis of the JPS algorithm, memorized jump points and obstacle tables are added. The algorithm flow is shown in Fig. 3. The jump points table can be used to solve the issue of JPS by only planning once and not reusing environmental processing history information. It can be further used to speed up the algorithm's work process to solve the path. To improve the speed and success rate for subsequent trajectory optimization, the heuristic function is modified by combining the new obstacle table. The heuristic function plays a guiding role in planning. Adjusting the value of λ in the heuristic function can adaptively reduce the collision risk for UAVs caused by the concentration of obstacles, optimizing their flight path planning.

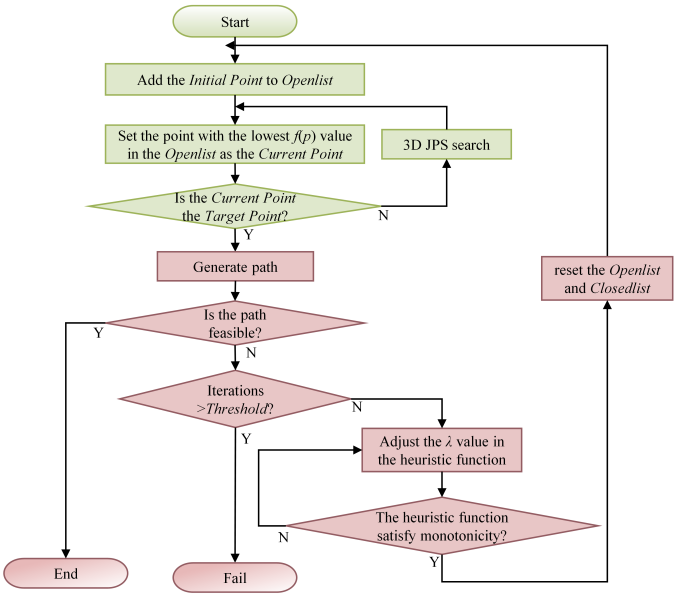

Fig. 3. Flow chart of front-end path search algorithm based on backend feedback optimization. $f(p)$ is the cost function, λ represents the weight of the proportion of obstacles around point p.

Algorithm 1 M-JPS

Input: *Initial Point*, *Target Point*
Output: optimal path

```
create Openlist and Closedlist
for i = 1 to Threshold do
    calculate f(p), g(p), and h(p) at the Initial Point
    Initial Point → Openlist
    Current Point = argmin f(p), p ∈Openlist
    Current Point → Closedlist
    if Current Point == Target Point
      then generate path
      if the path is feasible
        then break
      else adjust the λ value
    else 3D JPS search
end for
```

Remark: In the proposed M-JPS algorithm, the expression of the heuristic function $h(p)$ is reflected in the following text, and λ in the function is adaptively adjusted. Firstly, λ has an initial value, and during each iteration, the monotonicity of the heuristic function is evaluated. If monotonicity is not satisfied, the value of λ is adjusted until monotonicity is satisfied. Secondly, when the optimized path fails to meet the dynamic constraints, the backend will perform local re-partitioning and adjust the λ value as feedback to the frontend.

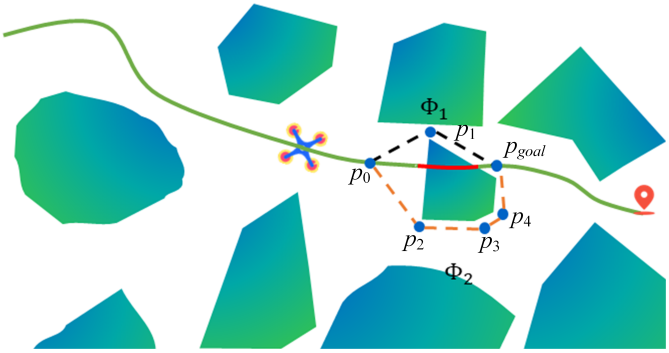

Fig. 4. Path planning optimal path (black) and suboptimal trajectory (orange).

Algorithm 2 3D JPS search

```
for direction in [up, down, left, right, front, back, diagonals] do
    neighbor = the point where the Current Point moves one step along the direction
    if neighbor is not walkable or neighbor in Closedlist
        continue
    calculate g(neighbor)
    calculate h(neighbor)
    if neighbor not in Openlist
        then f(neighbor) = g(neighbor) + h(neighbor)
        Openlist.add(neighbor, f(neighbor))
        neighbor.parent = Current Point
    else if g(p) < g(neighbor)
        Openlist.decrease_priority(neighbor, g + h)
        neighbor.parent = Current Point
    prune_middle_jump_points(Closedlist,Openlist)
end for
```

In general, the heuristic function of A* is fully competent in situations where subsequent backend optimization is usually not considered. However, when considering the situation shown in Fig. 4, the following problems arise. As shown in Fig. 4, there are two paths to choose from: $\Phi_1 = \{p_0, p_1, p_{\text{goal}}\}$ and $\Phi_2 = \{p_0, p_2, p_3, p_4, p_{\text{goal}}\}$. Obviously, the heuristic function of the trajectory Φ_1 will be lower, and the final planning result will mostly coincide with the trajectory Φ_1. However, since a large part of the trajectory Φ_1 has a relatively narrow safe area and considering the size of the UAV, the safe distance required by the UAV contradicts the narrow safety planning environment. This can lead to the need for a longer time to optimize the backend and may not even lead to one obtaining the final desired trajectory.

Therefore, this paper proposes an explicit description of the narrowness of the safe area around each voxel as

$$f_{\text{dens}}(p) = \frac{\iiint_R Occ(x)\mathrm{d}x}{V_R} \tag{9}$$

where R is a three-dimensional area including voxel p, which is the green area shown in Fig. 5. The black areas shown in Fig. 5 represent obstacles. $Occ(x)$ is the proportion of the nonobstacle coverage area in the three-dimensional area R.

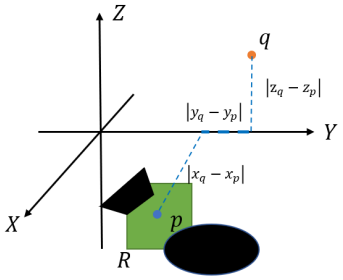

Fig. 5. Description of the cost parameters in the process of the UAV from position p to destination q.

The heuristic function can be expressed as

$$h(p,q) = \frac{(D(p,q) + D_{\text{cheb}}(p,q)) \times (1 + \lambda f_{\text{dens}}(p))}{\lambda_{H\text{eur}}} \tag{10}$$

where $D(p,q)$ and $D_{\text{cheb}}(p,q)$ represent the Euclidean distance and Chebyshev distance from the current point p to the end point q, respectively. λ represents the weight of the influence of the proportion of obstacles around point p on the selected path. The weight can be comprehensively selected according to the size, wheelbase, and expected safe distance from obstacles of the UAV. The λ in the heuristic function is adjusted online, and based on the backend optimization, the parameter λ is adjusted in a timely manner. $\lambda_{H\text{eur}}$ is the proportion of the heuristic function value. The Chebyshev distance in Fig. 5 is given by

$$D_{\text{cheb}}(p,q) = \max\{|x_q - x_p|, |y_q - y_p|, |z_q - z_p|\}. \tag{11}$$

The $(1+\lambda f_{\text{dens}}(p))$ term in the heuristic function determines whether the original mathematical proof of the algorithm still holds. In the flowchart of Fig. 3, the heuristic function after adjusting λ must satisfy monotonicity before the subsequent process can be carried out. Otherwise, the value of λ will be recalculated.

B. Backend Trajectory Optimization and Motion Planning Algorithm Framework

Figure 6 shows the spatial layout of the global path in two-dimensional space when it passes through obstacles. Optimization aims to design a suitable optimization objective function and use the gradient value to guide the path of the collision segment away from the obstacle. To avoid redundant voxel information processing, the calculation process is only started when a collision segment exists in the global path. First, the obstacle's control point is designated as Q_i, the edge guide point closest to Q_i on the obstacle surface is called P_i, and the vector V_i is $P_i - Q_i$. The vector distance d_i of the guide point P_i on the edge of the control point Q_i is defined as

$$d_i = (P_i - Q_i)V_i. \tag{12}$$

From (12), it is easy to see that d_i is negative when Q_i is inside the obstacle and positive when Q_i is outside the obstacle. When the control point coincides with Pi, it is zero bound, so P_i is also called the adjacent safety zero boundary point of the control point Q_i. Therefore, as long as the

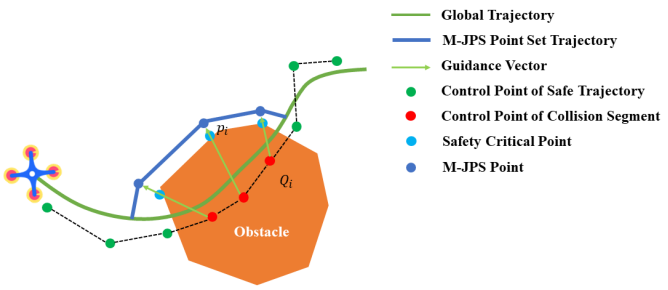

Fig. 6. The UAV uses the M-JPS algorithm to quickly generate a safe point set path and the trajectory control points within the obstacle to form a guidance vector.

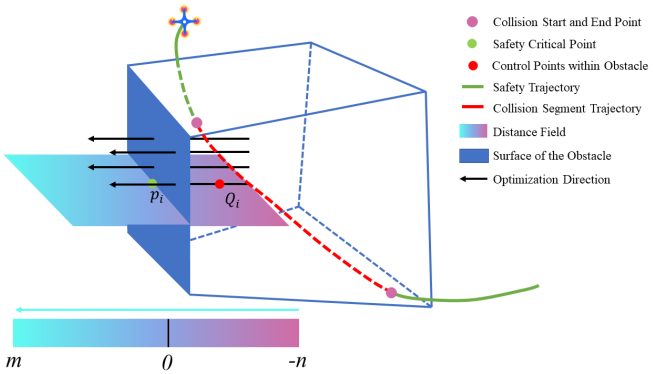

Fig. 7. The collision segment utilizes the direction optimization of the safety zero point away from the obstacle.

gradient direction is directed toward the nonnegative direction of d_i when designing the optimization objective function, the control point inside the obstacle can be optimized to the safe area to complete obstacle avoidance.

According to the above definition of the adjacent safety zero point, the intersection of the obstacle, the straight line formed by the current control point Q_i, and the guidance point generated by the nearest M-JPS algorithm gives the adjacent safety zero point of the control point Q_i. In the actual flight space of the UAV, the environment is three-dimensional, and a schematic diagram showing the use of the direction of the zero-boundary safe point to optimize away from obstacles is shown in Fig. 7.

In this paper, three penalty items are used in the design of the optimization objective function, namely, the smoothness penalty J_s, the collision segment penalty J_c and the ease of use penalty J_f, and the control points of the B-spline curve in the obstacle are optimized, assuming that it is necessary to optimize the trajectory segment to be controlled by N_c control points, respectively $\{Q_1, Q_2, Q_3, ..., Q_{Nc}\}$. The final optimization objective function is given by [31]

$$\min J(Q_i) = \lambda_1 J_s + \lambda_2 J_c + \lambda_3 J_f \tag{13}$$

where λ_1, λ_2 and λ_3 represent the proportions of J_s, J_c, and J_f in the objective function, respectively.

The smoothness penalty is used mainly to prevent the high-order derivative value of the curve from being too high and to suppress the acceleration jitter existing in the flight path of the UAV, so it is designed to suppress the jerk speed value. Benefiting from the convex hull property of the B-spline curve, minimizing the third-order derivative of the control point will be enough to reduce the third-order derivative value on the entire B-spline curve, so the primary expression for the smoothness penalty J_s is given by

$$J_s = \sum_{i=1}^{N_c-2} ||jerk_i||^2 \tag{14}$$

where N_c is the number of control points, $jerk_i$ represents the acceleration change rate at the i-th point.

For the collision segment penalty J_c, a safe distance threshold d_{safe} is introduced, which indicates the minimum distance between the UAV and obstacles during the flight of the current task. The safe distance needs to be selected according to the actual scene and the size of the UAV. If the safety distance is too small, one can encounter problems such as being too close to obstacles during autonomous flight or even losing kinetic energy after passing by obstacles and destroying the power structure of the UAV. If the safety distance is too large, the planning results cannot be obtained in some obstacle-intensive areas. J_c is expressed as follows:

$$J_c = \sum_{i=1}^{N_c} j_i \tag{15}$$

where j_i represents the collision segment penalty for each control point Q_i, and its specific expression is given as follows:

$$j_i = \begin{cases} 0, d_i > d_{\text{safe}} \\ (d_{\text{safe}} - d_i)^3, 0 \le d_i < d_{\text{safe}} \\ d_{\text{safe}}^3 + d_{\text{safe}}(d_{\text{safe}} - d_i)^2 + d_{\text{safe}}^2(d_{\text{safe}} - d_i), d_i < 0 \end{cases}. \tag{16}$$

Using the characteristics that the gradient value d_i inside the obstacle is less than 0 and the gradient value d_i outside the obstacle is greater than 0 when designing the penalty of the collision section, the speed of optimization convergence is accelerated when the control point is still inside the obstacle. The final result converges to a result greater than or equal to the safe distance threshold d_{safe}.

For the ease of use penalty J_f, it is necessary to strictly limit the value of the high-order derivative of the trajectory in each dimension at any time to be less than the maximum value, that is,

$$|\Phi_d^k(t)| \le \Phi_{d,\max}^k \tag{17}$$

where $d \in \{x, y, z\}$, k represents the order, and $\Phi_{d,max}^k$ represents the maximum value of a certain motion parameter under the k-order of the d dimension. The main motion parameters for the ease-of-use restriction are velocity v_i and acceleration a_i, so the ease-of-use penalty J_f is designed as follows:

$$J_f = \varpi_v \sum_{i=1}^{N_c} F(v_i) + \varpi_a \sum_{i=1}^{N_c-1} F(a_i) \tag{18}$$

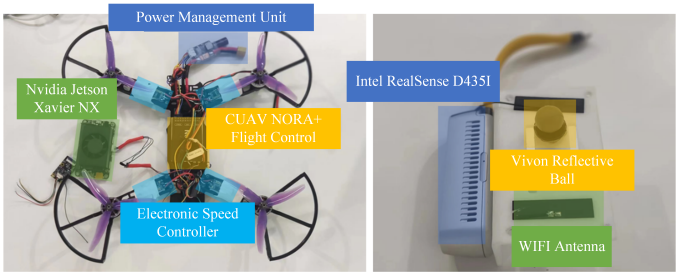

Fig. 8. Partial structure diagram of UAV.

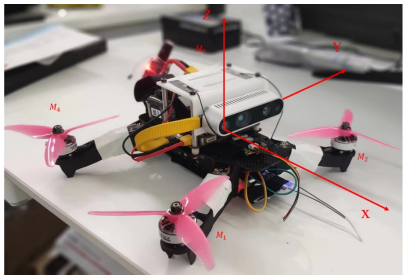

Fig. 9. General structure diagram of UAV.

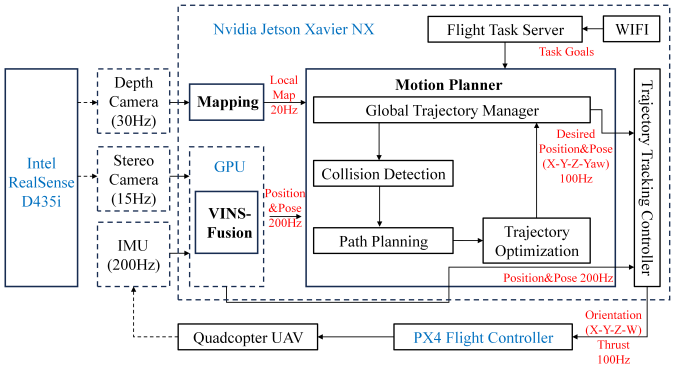

Fig. 10. Software Framework.

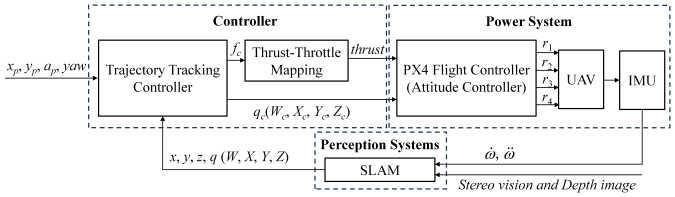

Fig. 11. UAV trajectory tracking control structure diagram.

where ϖ_v and ϖ_a are weights for each term, and $F(\cdot)$ is a twice continuously differentiable metric function of higher-order derivatives of control points.

$$F(p) = \sum_{i=x,y,z} f(p_i) \tag{19}$$

$$f(p_i) = \begin{cases} k_1 p_i^2 + k_2 p_i + k_3, p_i \le -p_j \\ (-\lambda p_m - p_i)^3, -p_j < p_i < -\lambda p_m \\ 0, -\lambda p_m \le p_i \le \lambda p_m \\ (p_i - \lambda x_m)^3, \lambda p_m < p_i < p_j \\ k_4 p_i^2 + k_5 p_i + k_6, p_j \le p_i \end{cases} \tag{20}$$

where $P \in \{v_i, a_i\}$, $k_1, k_2, k_3, k_4, k_5, k_6$ are used to ensure the continuity of the function $f(p_i)$ within the second order, p_m is the limit of the current kinematic parameters on the current direction axis, p_j is the dividing point between the two intervals of the quadratic and cubic curves, and λ is the elastic coefficient used to set the proportion of the safety margin.

IV. Experiments

A. Hardware

In this paper, an Nvidia Jetson Xavier NX is used as the onboard computer in the UAV, and a radiator is added. The flight controller uses CUAV NORA+, and an Intel Realsense D435I is used as the visual perception unit. Part of the structure diagram is shown in Fig. 8. The overall object of the final UAV is shown in Fig. 9.

B. Software Framework

The software framework for the autonomous flight system of the quadrotor UAV designed in this paper includes the following parts: an autonomous positioning module based on binocular vision inertial navigation fusion, a front-end path planning module based on memorized jump points based on backend optimization feedback, a backend space-time trajectory optimization module, a construction module for voxel map fusion, a global trajectory manager, and a trajectory tracking controller. The SLAM method adopted refers to VINS-Fusion [32]. The overall software framework is shown in Fig. 10.

C. Trajectory Tracking Controller

The overall trajectory tracking controller structure is shown in Fig. 11. The trajectory tracking controller currently uses PID control. After the hardware pattern recognition parameters are determined, one can also expect to use Model Predictive Control (MPC) or Linear Quadratic Regulator (LQR) controllers to improve the performance of trajectory tracking.

The effect of trajectory tracking is shown in Fig. 12.

The data results shown in Fig. 12 show that the trajectory of the UAV basically tracks the expected trajectory released by the onboard computing unit with a small error.

D. Flight State Finite State Machine

The UAV can enter an uncontrollable state during the flight for some reason. For example, when the positioning data are abnormal and a positioning mutation occurs, the position control loop can output a large degree of control instantaneously, causing the UAV to collide at a high speed. Therefore, this paper introduces a flight state finite state machine in actual operation. This enables the UAV to perform self-checks for each state in advance when switching between states to ensure safe switching. This simultaneously avoids abnormal UAV operation caused by unreachable state switching. The specific

This article has been accepted for publication in IEEE Transactions on Cognitive and Developmental Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2024.3519319

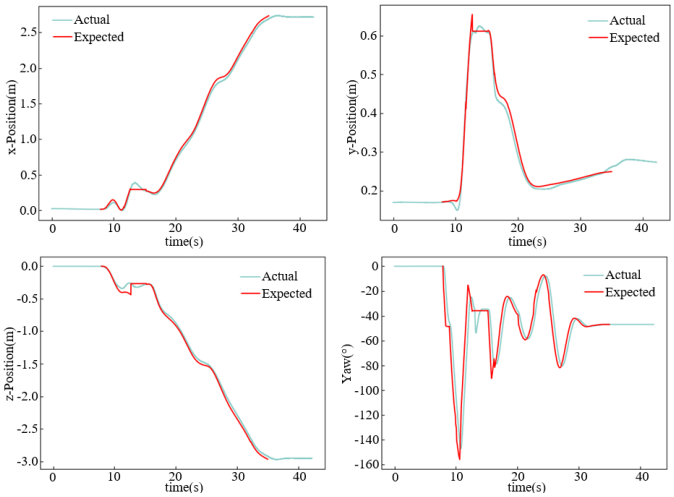

Fig. 12. Comparison chart of UAV actual flight trajectory and planned expected trajectory.

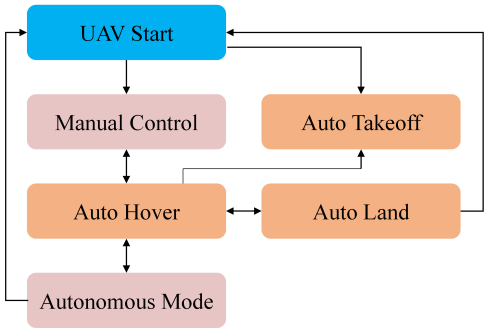

Fig. 13. Flight State Finite State Machine.

relationship diagram for jumping between states is shown in Fig. 13.

When the UAV is powered on, the UAV enters the UAV Start state. After inspecting the basic modules in the software framework, it enters the Manual Control mode. At this time, the UAV performs flight actions according to the remote control input. The following three automatic modes, namely, Auto TakeOFF, Auto Hover, and Auto Land, still receive commands from the remote control but can activate the trajectory tracking controller for control, which can lead to a self-stabilizing and automatic process. The final autonomous mode model fully starts the motion planning module, which is calculated, input by the program and used to control the movement of the UAV through the trajectory tracking controller.

E. Actual Flight Experiments

In this section, a scenario with actual obstacles as shown in Fig. 14 is constructed to verify the flight effectiveness of the actual aircraft, some obstacles and circles were arranged in the scene. The video of UAV flight was uploaded on https://youtu.be/S6so_7-XnU8. Figure 15 shows the performance of a fragment of the autonomous flight system designed in this paper for a flight mission. The corresponding real-time mapping information is also included below. Due to the limited sensing range of the D435I, only the lower part of the obstacle

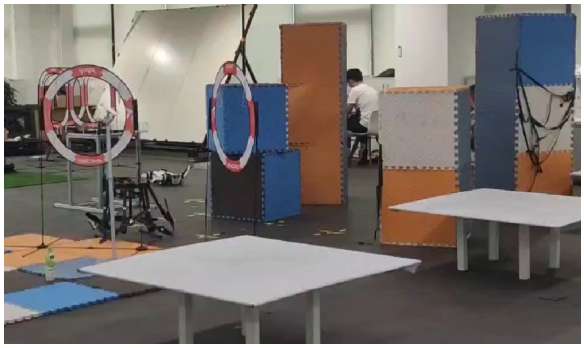

Fig. 14. Actual scenario.

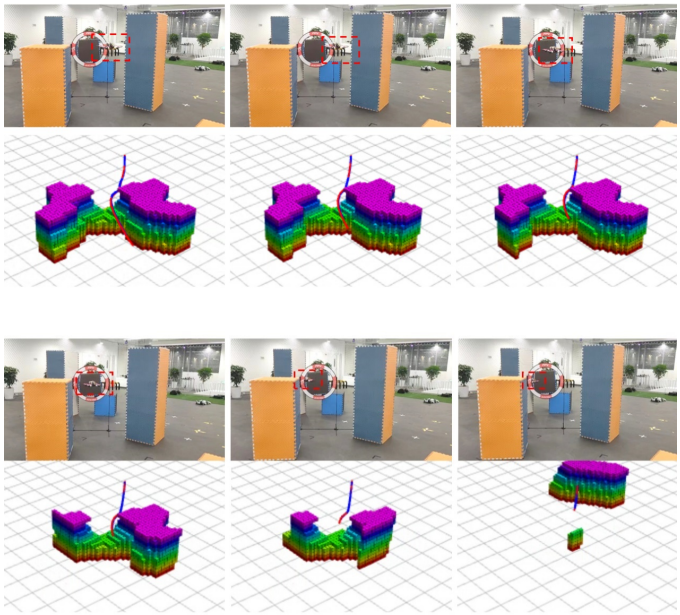

Fig. 15. The process of UAV motion planning leaping over obstacles and the actual mapping process.

information was constructed while passing through the FPV obstacle gate, which is generally in line with expectations.

Judging from the effect of actual mapping, it is observed that that the UAV can relatively accurately estimate the surrounding obstacles, which aligns with the initial experimental design expectations.

Figure 16 shows the trajectory optimization and trajectory replacement during UAV flight, where the blue part is the global trajectory, and the red part is the locally modified trajectory of the collision segment. According to the initial trajectory obtained by the minimum snap-based multitarget point global initial polynomial trajectory generation algorithm, the UAV perceives the subsequent collision segment through the perception and collision detection algorithm. At this time, the UAV first quickly generates a local safety reference point and then generates a local vector distance field according to the generated local reference point. Furthermore, the modified red trajectory of the new collision segment is calculated according to the spatiotemporal trajectory optimization algorithm of the

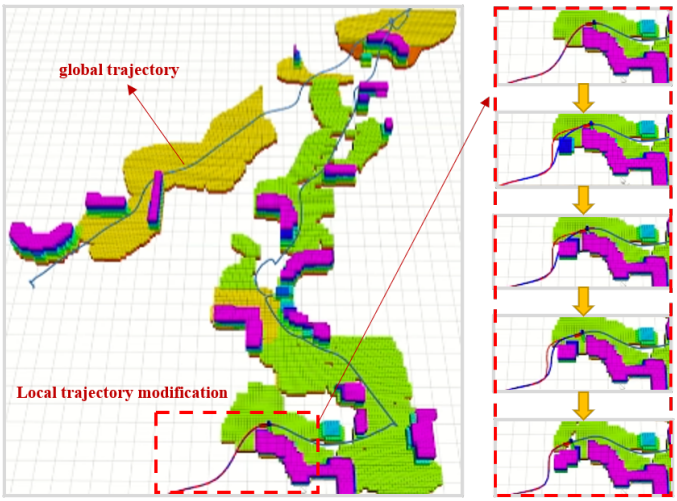

Fig. 16. Trajectory optimization, trajectory dynamics detection, trajectory reassignment and trajectory replacement process during UAV flight.

TABLE I
COMPARISON OF PLANNING RESULTS AND TIME-CONSUMING OF DIFFERENT MOTION PLANNING ALGORITHMS

Motion Planning Algorithm	Path Length (m)	Average Speed (m/s)	Max Speed (m/s)	Optimize Time (ms)	Mapping Time (ms)
Proposed	58.487	2.109	2.276	**0.386**	**0**
EGO [31]	56.410	2.115	2.251	0.426	**0**
ENI[1]	**43.727**	**2.086**	2.237	0.496	5.61
EI[2]	52.203	2.103	**2.354**	0.531	5.70

[1] ENI represents the collision-free initialization method based on ESDF.
[2] EI represents the collision-based initialization method based on ESDF.

UAV rear end based on gradient information. Then, to ensure the trajectory's dynamic availability, the dynamic parameter detection for the global adjacent trajectory is performed on the basis of the optimized trajectory. After the detection finds that the optimized trajectory exceeds the dynamic limit of the UAV, the system will execute a trajectory time redistribution algorithm based on the dynamics. After satisfying the dynamic constraints of the UAV, the trajectory replacement is completed in the global trajectory manager in the correct figure, and the condition for multiorder dynamic continuity of the entire trajectory is still satisfied. Thus far, the UAV has been observed to complete the process of autonomously planning and adjusting its motion state while satisfying safety, smoothness, and ease of use with local flying in a very efficient trajectory.

F. Simulation Comparison Experiments

In the same simulation scenario, this paper tests and compares the planning trajectory length, speed, and time consumption during the entire task process for some existing motion planning algorithm frameworks based on gradient information in single planning. The comparison results are shown in Table I.

Based on the data presented in Table I, it is evident that the velocity of our proposed method aligns closely with that of the two ESDF dependent methods. However, a notable distinction is that our method exhibits a mapping time of zero and achieves the shortest optimization time. Although the path length generated is slightly longer than that of the ENI and EI methods, this is strategically intended to circumvent regions with high obstacle density. Compared to the EGO method, our approach demonstrates a significant advantage in terms of optimization time, while maintaining consistency in other measured parameters. The shorter optimization time, compared to the other three methods, underscores the enhanced computational efficiency of our method. This optimization not only accelerates the execution speed of the algorithm but also bolsters its adaptability and robustness in complex environments.

TABLE II
COMPARISON OF THE SUCCESS RATE OF DIFFERENT MOTION PLANNING ALGORITHMS

Motion Planning Algorithm	Success Rate
Proposed	**0.9201**
EGO [31]	0.8802
ENI	0.6807
EI	0.8911

Table II shows a comparison of the task success rates obtained under multiple random scene tests.

The comparison results show that the average speeds of the algorithm proposed in this paper and the compared algorithms are basically the same. In the same scenario, the success rate of the algorithm planning in this paper can reach 92.01%, which is higher than that obtained using the other three algorithms, which proves the improvement of the success rate brought about by the adaptive ability of the motion planning algorithm framework used in this paper. The single planning time for the motion planning algorithm is 0.386 ms, which is only 90.61% of the shortest time among the other three algorithms. The total planning time and success rate are significantly improved compared with other algorithms. Although the length of the planned path is increased compared with other algorithms, essentially, the obstacle density and efficiency are balanced because the obstacle density to find a path is considered in the front-end path planning. This is very important for the fast, safe, and smooth avoidance of an obstacle by a UAV.

V. CONCLUSION AND DISCUSSION

This paper proposes a backend feedback adaptive quadrotor UAV motion planning algorithm based on gradient descent, which enhances the integration and feedback between front-end path planning and backend trajectory optimization. The framework of the motion planning algorithm consists of three components: a global trajectory manager, front-end path planning, and backend trajectory optimization. To address the limitations of traditional frameworks, an M-JPS algorithm based on backend optimization feedback is introduced, and a new heuristic function is proposed to improve environmental adaptability by incorporating obstacle and jump point tables.

Additionally, a gradient descent trajectory optimization algorithm guided by the adjacent safety zero point is proposed, addressing the issue of unguided redundant mapping found in traditional gradient-based algorithms. Experimental results demonstrate that this algorithm framework achieves a higher success rate and shorter optimization time in rapid UAV reprogramming, outperforming three other algorithms.

The method proposed in this article is mainly suitable for environments with dense obstacles. It is not simply pursuing the shortest path but rather integrating the volume of UAVs to plan a suitable path. The adaptability of the method proposed in this article is primarily demonstrated by its ability to autonomously adjust planned paths in environments with obstacles, effectively responding to dynamically changing conditions. By incorporating dynamic continuity constraints into the global trajectory manager, local trajectories are seamlessly integrated, ensuring dynamic continuity and enhancing the efficiency and safety of UAV navigation. Furthermore, the λ value quantifies obstacle density, guiding UAVs to select paths with lower obstacle concentrations, further improving the algorithm's adaptability. This study enhances the autonomous navigation capabilities of UAVs in complex environments and offers an effective solution for achieving efficient and safe motion planning in various scenarios. The proposed method can be applied in future research path planning for various mobile robots, not just quadcopter UAVs.

REFERENCES

[1] J. H. Yu, H. W. Gao, J. Sun, D. L. Zhou, and Z. J. Ju, "Spatial Cognition-Driven Deep Learning for Car Detection in Unmanned Aerial Vehicle Imagery," *IEEE Trans. Cogn. Develop. Syst.*, vol. 14, no. 4, pp. 1574–1583, Dec. 2022.

[2] S. H. Alsamhi, A. V. Shvetsov, S. Kumar, S. V. Shvetsova, M. A. Alhartomi, A. Hawbani, N. S. Rajput, S. Srivastava, A. Saif, and V. O. Nyangaresi, "UAV Computing-Assisted Search and Rescue Mission Framework for Disaster and Harsh Environment Mitigation," *Drones*, vol. 6, no. 7, p. 154, Jul. 2022.

[3] N. A. Letizia, B. Salamat, and A. M. Tonello, "A Novel Recursive Smooth Trajectory Generation Method for Unmanned Vehicles," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1792–1805, Oct. 2021.

[4] B. Y. Zhou, H. Xu, and S. J. Shen, "RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1816–1835, Jun. 2023.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Sys. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[6] Y. C. Guo, X. X. Liu, X. H. Liu, Y. Yang, and W. G. Zhang, "FC-RRT*: An Improved Path Planning Algorithm for UAV in 3D Complex Environment," *ISPRS Int. J. Geoinf.*, vol. 11, no. 2, p. 112, Feb. 2022.

[7] J. K. Wang, T. G. Li, B. P. Li, and M. Q. H. Meng, "GMR-RRT*: Sampling-Based Path Planning Using Gaussian Mixture Regression," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 690-700, Sep. 2022.

[8] Z. H. Chen, J. B. Yu, Z. Y. Zhao, X. Y. Wang, and Y. Chen, "A Path-Planning Method Considering Environmental Disturbance Based on VPF-RRT*," *Drones*, vol. 7, no. 2, p. 145, Feb. 2023.

[9] D. D. Harabor, and A. Grastien, "Online Graph Pruning for Pathfinding on Grid Maps," in *Proc. 25th AAAI Conf. Artif. Intell.*, San Francisco, California, USA, Aug. 2011, pp. 1114–1119.

[10] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.

[11] F. Gao, L. Q. Wang, B. Y. Zhou, X. Zhou, J. Pan, and S. J. Shen, "Teach-Repeat-Replan: A Complete and Robust System for Aggressive Flight in Complex Environments," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1526–1545, Oct. 2020.

[12] S. Savin, "An Algorithm for Generating Convex Obstacle-free Regions Based on Stereographic Projection," in *Proc. Int. Siberian Conf. Control Commun. (SIBCON)*, Astana, Kazakhstan, Jun. 2017, pp. 1–6.

[13] R. Deits and R. Tedrake, "Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming," in *Proc. 11th Workshop Algorithmic Found. Robot. (WAFR)*, Istanbul, Turkey, Aug. 2014, pp. 109–124.

[14] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9–10, pp. 1164–1193, Aug. 2013.

[15] Y. Nesterov, *Lectures on Convex Optimization*. Berlin, Germany: Springer, 2018.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011.

[17] M. Kurenkov, A. Potapov, A. Savinykh, E. Yudin, E. Kruzhkov, P. Karpyshev, and D. Tsetserukou, "NFOMP: Neural Field for Optimal Motion Planner of Differential Drive Robots With Nonholonomic Constraints," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10991–10998, Oct. 2022.

[18] C. Park, J. Pan, and D. Manocha. "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proc. 22nd Int. Conf. Autom. Plan. Sched. (ICAPS)*. Atibaia, Sao Paulo, Brazil, Jun. 2012.

[19] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.

[20] G. -T. Tu and J. -G. Juang, "Path Planning and Obstacle Avoidance Based on Reinforcement Learning for UAV Application," in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Ho Chi Minh City, Vietnam, Aug. 2021, pp. 352–355.

[21] A. Loquercio, E. Kaufmann, R. Ranftl, M. Muller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Sci. Robot.*, vol. 6, no. 59, p. eabg5810, Oct. 2021.

[22] G. Tang, W. D. Sun, and K. Hauser, "Learning Trajectories for Real-Time Optimal Control of Quadrotors," in *Proc. 25th IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 3620–3625.

[23] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion Planning Networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Montreal, Canada, May 2019, pp. 2118–2124.

[24] A. H. Qureshi, Y. L. Miao, A. Simeonov, and M. C. Yip, "Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 48–66, Feb. 2021.

[25] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, Australia, May 2018, pp. 7087–7094.

[26] B. Ichter, and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2407–2414, Jul. 2019.

[27] B. Ichter, E. Schmerling, T. W. E. Lee, and A. Faust, "Learned Critical Probabilistic Roadmaps for Robotic Motion Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Electr network, May 2020, pp. 9535–9541.

[28] H. A. Hua, and Y. C. Fang, "A Novel Learning-Based Trajectory Generation Strategy for a Quadrotor," *IEEE Trans. Neural Netw. Learn. Syst.*, Nov. 2022.

[29] T. Hwu, A. Y. Wang, N. Oros, and J. L. Krichmar, "Adaptive Robot Path Planning Using a Spiking Neuron Algorithm With Axonal Delays," *IEEE Trans. Cogn. Develop. Syst.*, vol. 10, no. 2, pp. 126–137, Jun. 2018.

[30] S. Rezwan, and W. Choi, "Artificial Intelligence Approaches for UAV Navigation: Recent Advances and Future Challenges," *IEEE Access*, vol. 10, pp. 26320–26339, 2022.

[31] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.

[32] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," 2019, *arXiv:1901.03642*.