Complex & Intelligent Systems (2023) 9:7133–7153
https://doi.org/10.1007/s40747-023-01115-2

ORIGINAL ARTICLE

HPO-RRT*: a sampling-based algorithm for UAV real-time path planning in a dynamic environment

Yicong Guo[1] · Xiaoxiong Liu[1,2] · Qianlei Jia[1] · Xuhang Liu[1] · Weiguo Zhang[1,2]

Received: 24 June 2022 / Accepted: 12 May 2023 / Published online: 20 June 2023

Abstract

The real-time path planning of unmanned aerial vehicles (UAVs) in dynamic environments with moving threats is a difficult problem. To solve this problem, this paper proposes a time-based rapidly exploring random tree (time-based RRT*) algorithm, called the hierarchical rapidly exploring random tree algorithm based on potential function lazy planning and low-cost optimization (HPO-RRT*). The HPO-RRT* algorithm can guarantee path homotopy optimality and high planning efficiency. This algorithm uses a hierarchical architecture comprising a UAV perception system, path planner, and path optimizer. After the UAV perception system predicts moving threats and updates world information, the path planner obtains the heuristic path. First, the path planner uses the bias sampling method based on the artificial potential field function proposed in this paper to guide sampling to improve the efficiency and quality of sampling. Then, the tree is efficiently extended by the improved time-based lazy collision checking RRT* algorithm to obtain the heuristic path. Finally, a low-cost path optimizer quickly optimizes the heuristic path directly to optimize the path while avoiding additional calculations. Simulation results show that the proposed algorithm outperforms the three existing advanced algorithms in terms of addressing the real-time path-planning problem of UAVs in a dynamic environment.

Keywords Sampling-based path planning · Dynamic path planning · Time-based rapidly exploring random tree (RRT) · Unmanned aerial vehicles (UAVs)

Introduction

Recently, unmanned aerial vehicles (UAVs) have gradually played an increasingly critical role in civil fields (such as earthquake relief and vegetation protection) and military fields (such as urban penetration and attacking enemy targets) [1]. Path planning, as one of the most fundamental and important problems in autonomous UAV flight, has been widely studied [2]. In recent decades, many path-planning algorithms have been proposed, which can be roughly divided into nature-inspired methods, grid-based methods, and sampling-based methods. Nature-inspired algorithms include artificial potential field (APF) methods [3], the interfered fluid dynamical system (IFDS) method [4] and intelligent algorithms. Among them, the APF and IFDS have the advantages of simple calculation and strong real-time performance, but the planned path can easily fall into a local minimum. Intelligent algorithms such as a genetic algorithm [5] can obtain the global optimal solution by iteration, but the calculation is more complicated in high-dimensional spaces. Heuristic search methods include the A* algorithm [6], anytime repairing A* (ARA) algorithm [7], sparse A* algorithm [8] and D* algorithm [9]. Although such algorithms can ensure that the optimal solution can be found if it exists, their search time increases sharply with increasing problem scale and spatial dimension. Sampling-based methods such as rapidly exploring random trees (RRT) [10] and probabilistic roadmaps

✉ Xiaoxiong Liu
nwpulxx@outlook.com

Yicong Guo
guoyicong@mail.nwpu.edu.cn

Qianlei Jia
jiaql@mail.nwpu.edu.cn

Xuhang Liu
liuxuhang@mail.nwpu.edu.cn

Weiguo Zhang
zhangwg@nwpu.edu.cn

1 School of Automation, Northwestern Polytechnical University, Xi'an 710129, China

2 Shaanxi Province Key Laboratory of Flight Control and Simulation Technology, Xi'an 710129, China

(PRM) [11] usually sacrifice the optimal solution at grid search resolution for the ability to quickly find satisfactory solutions in high-dimensional complex state space and large-scale problems.

This paper focuses on solving the real-time path-planning and optimization problem of UAVs in complex dynamic environments with moving threats. Because of the influence of moving threats, planning needs to be constantly revised following the update of environmental information [12]. Therefore, the path-planning algorithm must have high planning and optimization efficiency and low storage requirements. Efficient planning and optimization calculations ensure that a UAV can generate the optimal path suitable for flight in real time, and low storage can effectively improve the scale of the UAV planning space. Because RRT and its derived algorithms [13, 14] build a tree structure through random sampling to quickly expand the search space, these algorithms still have very low computational complexity and high planning efficiency in high-dimensional environments. Therefore, these algorithms have good prospects in addressing UAV path planning in complex environments with moving threats. Note that most of the early works on RRT algorithms mainly focus on finding the feasible solutions, leaving a great possibility for improving the execution time and path optimization [15].

To deal with the path-planning problem in dynamic environments with moving threats, many improved RRT algorithms have been proposed, which are based on two different ideas. One idea is that algorithms will update the environment information and replan the path immediately after detecting changes in the environment, which requires the algorithms to have faster response speed, higher planning and optimization efficiency. Another idea is that algorithms actively avoid threats by predicting the trajectory of dynamic threats and planning collision-free paths. The algorithms with this idea can reduce the update frequency so that more time can be used to optimize the planning path to deal with the dynamic environment. Recently, many scholars have studied these two categories of RRT algorithms for dynamic environment path planning. The DRRT [16] algorithm uses tree pruning to update the plan. It takes the target position as the root of the tree to extend the tree, which simplifies the extension process. Bryant proposed an online RRT* algorithm [17] that ensures that new targets are always added to the spanning tree through online replanning. RRT^X [18] uses a fast rewiring operation to repair damaged branches to rebuild the tree structure after detecting changes in the environment. These three algorithms mainly consider the information of moving threats from the current time to the next step of planning, meaning that path replanning is needed whenever the environment changes. Although the above algorithms optimize the replanning process to improve computing efficiency and ensure real-time performance, they still need a high replanning update frequency to address the movement of threats. In contrast to the above three replanning algorithms, the time constraint is considered by an algorithm based on the partial path-planning method [19]. This kind of algorithm adds the nodes without collision with the known threat trajectory to the tree and parallels the path planning by executing the UAV. Influenced by the partial path-planning method, a time-based RRT algorithm named Risk-RRT is proposed in [20]. Risk-RRT predicts the movement of a threat through a Gaussian process. Then, RRT is used to plan part of the path and control the robot to advance a certain distance along the planned path. Note that because Risk-RRT uses the traditional RRT as the underlying planner, the algorithm can quickly obtain feasible paths even though it cannot obtain optimized paths. Therefore, to improve the quality of the planned path, Zhang proposed the Risk-RRT* algorithm [21]. Risk-RRT* improves the underlying planner-based RRT to RRT*. The rewiring process of RRT* can optimize the tree structure to improve the planning path optimality. However, the rewiring process of RRT* will change the tree structure. Since Risk-RRT* is a time-based RRT* algorithm, the change in tree structure means that each affected node in the tree needs to update the information, including the parent and child node, timestamp and depth of the node. This process often brings many additional calculations, which reduces the percentage of time devoted to the path optimization process within the limited planning time and then affects the quality of the planned path.

This analysis shows that the above algorithms still have space and a possibility to be improved.

(1) The flight environment of UAVs is usually a large three-dimensional (3D) space with many moving threats, and the moving frequency of threats is also high. Adopting replanning algorithms requires maintaining a high planning update frequency, which is difficult to achieve.
(2) In a large 3D space, if the random sampling method is used to search the whole planning space, many useless sampling nodes may be generated. This result will make the planning inefficient and increase the planning time.
(3) The longer a UAV performs a mission in a battlefield environment, the greater is the probability of being detected and attacked by the enemy. This circumstance necessitates improving the efficiency and quality of the planned path so that the UAV can quickly complete its task on the battlefield and reduce this probability. Therefore, combined with the above motivations, this paper will focus on further improving the partial planning algorithm based on time RRT. The algorithm can improve the efficiency of feasible path planning and quickly optimize the path based on avoiding additional computational consumption to ensure rapid planning

efficiency and high path quality in the online dynamic path planning of UAVs with moving threats.

Therefore, in this paper, we propose the HPO-RRT* algorithm to solve the real-time path-planning problem of UAVs in a 3D dynamic environment. Figure 1 shows the basic framework of the algorithm, which adopts a hierarchical architecture. The HPO-RRT* algorithm can quickly plan a path suitable for UAV flight and ensure the optimization of the planned path. First, the HPO-RRT* algorithm uses the UAV perception system to update the world information and predict the moving threats before each plan. Then, an improved time-based RRT* algorithm is used in the path planner to obtain the heuristic path: the sampling bias method based on the APF function is used to guide the sampling towards the target position and avoid threats, the process of selecting the optimal parent node for the new node in RRT* is introduced, and lazy collision checking is adopted to reduce the running time of the algorithm while satisfying the avoidance of dynamic threats. Thus, a partial heuristic path can be quickly obtained. Finally, a low-cost path optimizer is used to quickly optimize the path generated by the path planner to ensure the homotopy optimality of the final planned path.

Our work mainly includes the following innovations:

(1) A hierarchical planning framework suitable for real-time fast path planning in a UAV dynamic environment is designed (Fig. 1), which comprises a UAV perception system, a path planner based on the improved time-based RRT* and a low-cost path optimizer, to improve the planning efficiency and ensure path optimization.
(2) In the bottom planner, a bias sampling method based on the APF function is used to guide the generation of sampling nodes, which can improve the sampling efficiency and reduce the planning time.
(3) In the path planner, the process of neighbourhood optimal parent node selection and lazy collision checking is applied to the tree expansion, which improves the planning efficiency and obtains a path optimized as much as possible.
(4) A low-cost path optimizer is designed to quickly optimize the path generated by the path planner without changing the tree structure.
(5) The probability completeness and homotopy optimality of the HPO-RRT* algorithm are discussed and proven.

This paper is organized as follows. The following section describes the related work. "Problem formulation" describes the problems related to path planning in a dynamic environment. "HPO-RRT* algorithm" introduces the HPO-RRT* algorithm in detail in three aspects: the overall framework of HPO-RRT* ("Overall framework"), tree structure optimization and heuristic path planning ("Path planner"), and the low-cost heuristic path optimization method ("Low-cost path optimizer"). The analysis and proof of the probability completeness and homotopy optimality of HPO-RRT* are introduced in "Analysis". In "The results of simulation experiments", the simulation experiments are reported. In "Conclusions and future work", we draw conclusions and discuss future work.

Related works

In this section, we first introduce the sampling strategy and related works to improve the RRT convergence speed. Then, Risk-RRT is taken as a typical example to describe the relevant basic knowledge of time-based RRT.

Sampling strategy

As a single-query path-planning method based on sampling, the RRT algorithm has been widely studied in the past three decades because of its high computational efficiency and strong scalability. However, RRT has the problem of slow convergence. Therefore, how to quickly converge has become a research focus.

The results show that changing the sampling strategy can effectively increase the convergence speed of the algorithm. Most RRT-like algorithms use an unbiased and uniform sampling strategy to explore the configuration space, which is pointless and greatly increases computational consumption, resulting in slower convergence speed. Therefore, an importance sampling strategy is proposed, which can extract samples from the preset distribution or configuration space. This method effectively increases the convergence speed [22]. Using prior information to guide local sampling is a strategy based on importance sampling. This strategy uses the planned path as a priori information to guide sampling around it, such as informed RRT* [23] and its batch sample iteration improved algorithm BIT* [24]. In addition, the ellipsoid heuristic sampling strategy is also adopted in [25, 26] to increase the convergence speed. Compared with the prior information-guided sampling strategy, direct bias sampling avoids the uniform sampling process when exploring feasible paths and usually brings a higher convergence speed. This strategy biases random samples to favorable areas through specific methods, such as Theta*-RRT*, to explore the configuration space. Using the APF method as a bias sampling strategy is a good choice. The attractive potential field of the APF is used to guide random sampling towards the target position bias in [27, 28]. Pharpatara [29] introduced a rotating potential field to the traditional APF to guide offset sampling with direction information. This paper also adopts the offset sampling method based on the APF function to

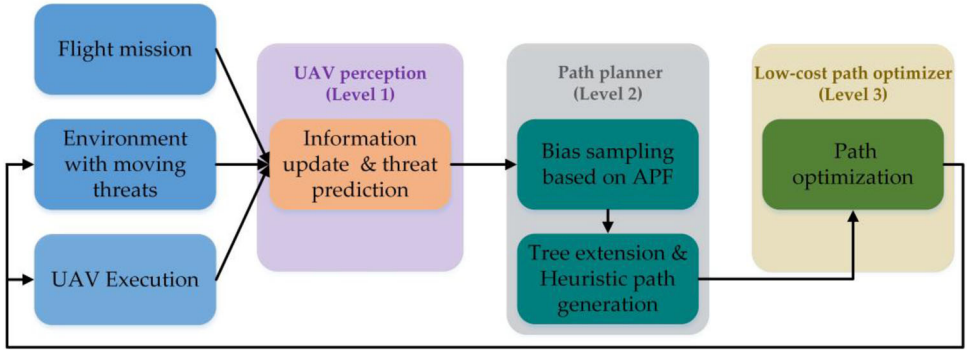


Fig. 1 Basic framework of HPO-RRT*

increase the convergence speed. In contrast to [28], we combine attractive and repulsive potential fields to guide random sampling to further improve the sampling efficiency.

Time-based RRT

The difference between time-based RRT and classical RRT mainly lies in the representation of nodes in the tree. The tree node of classical RRT only records the location information of the node, while node x of time-based RRT can be defined as a structure containing six kinds of information, i.e. $x : \left(x_{coor}, x_{parent}, x_{sons}, n, t, P_{collisioncheck}(t)\right)$.

(1) x_{coor} represents the position information of x and is represented in $O - xyz$. (2) x_{parent} represents the parent node of x. Note that x_{parent} is also represented by a structure like x. (3)x_{sons} represents any child node of x, similar to x_{parent}. (4) n indicates the depth of x, that is, it needs to pass through n nodes from the root node to the x. (5) t represents the timestamp of x, which can be calculated by depth, i.e. $t = t_0 + n\Delta t$, where t_0 is the timestamp of the root node and Δt is the time increment required for the UAV to pass through the adjacent nodes. (6) $P_{collisioncheck}(t)$ is the collision risk probability between the node and the obstacles in the environment at time t. In the Risk-RRT algorithm, $P_{collisioncheck}(t)$ is defined as [20]:

$$\begin{cases} P_{collisioncheck}(t) = P_{static} + (1 - P_{static}) P_{moving}(t) \\ P_{moving}(t) = 1 - \prod_{k}^{K} \left(1 - P_{moving}(p_k(t))\right) \end{cases}, \tag{1}$$

where P_{static} represents the collision risk probability of the static threat, $P_{moving}(t)$ represents the collision risk probability obtained for the moving threat at time t, and $P_{moving}(p_k(t))$ represents the collision risk probability between the k-th moving obstacle and the node at time t.

Time-based RRT is also an incremental algorithm similar to RRT, i.e. the tree structure continues to expand with increasing sampling nodes. In contrast to RRT, in the tree structure of time-based RRT, tree nodes with high collision risk or timestamps smaller than the current root node will be deleted.

Problem formulation

In this section, we define the path-planning problems in a dynamic environment with moving threats that will be solved in this paper and the notation used to describe them.

The planning environment is abstracted as a configuration space $X \subseteq \mathbb{R}^d$. Let $X_{obs} \subseteq X$ be the threat space composed of static threat X_{sta} and moving threat X_{mov}, i.e. $X_{sta}, X_{mov} \in X_{obs}$. The obstacle-free space is defined as $X_{free} = X \backslash X_{obs}$. In a dynamic environment, the configuration space is time varying, so the above two spaces are represented as $X_{obs}(t)$ and $X_{free}(t)$. x_{init}, X_{goal} are the initial position and the target area, respectively. In this paper, $X_{goal} = \left\{x \in X_{free} | \rho\left(x, x_{goal}\right) \leq r_{goal}\right\}$ is defined in the dynamic environment to detect whether the goal is reached, where $\rho(x_1, x_2)$ represents the Euclidean distance between x_1 and x_2. A ball with centre $x \in X$ and radius $r \in \mathbb{R}_{>0}$ is denoted as $\mathfrak{B}_{x,r}$. The APF function is $U : \mathbb{R}^d \to \mathbb{R}$, where the attractive and repulsive potential fields are U_{att} and U_{rep}, respectively. The potential field forces generated by these fields are $\mathbf{F}_{att}$ and $\mathbf{F}_{rep}$.

The following paragraphs describe some problems in path planning in a dynamic environment.

In contrast to the static environment, obstacles in a dynamic environment change their positions over time. Therefore, for the dynamic path-planning algorithm, time measurement must be introduced into the nodes so that the collision detector can use the time component to check the feasibility of the planned path. The dynamic environment proposed in this paper is described in Problem 1.

Problem 1 (*Dynamic environment*). If threats change location over time, the environment is called dynamic, i.e.

$$\Delta X_{obs} = \bigcup_{i=0}^{I} f_i(x, t_i). \tag{2}$$

The feasibility of the path is described in Problem 2.

Problem 2 (*Feasible path planning*). In the configuration space X containing $X_{obs}(t)$ and $X_{free}(t)$, given x_{init} and $X_{goal}\left(x_{goal}, t\right)$, a collision-free path $\sigma : [0, T] \rightarrow X_{free}$ can be planned, where $\sigma(0) = x_{init}$, $\sigma(T) \in X_{goal}\left(x_{goal}, T\right)$.

Let $\sum$ be the set of all feasible paths in obstacle-free space. The cost function $c(\sigma) \in \mathbb{R}_{>0}$ represents the generation value of each feasible path σ. Then, the optimal path-planning problem is described by Problem 3.

Problem 3 (*Optimal path planning*). Assuming that there is a solution set $\sum$ for Problem 2, the path in which the cost function $c(\sigma)$ is the smallest is the solution σ^* of the optimal path planning, i.e.

$$\sigma^* = \arg\min_{\sigma \in \sum} c(\sigma). \tag{3}$$

In UAV path planning, constraints are placed on the maximum steering angle $\phi_{\max}$ and the maximum climbing/diving angle $\gamma_{\max}$ [30]. Assume that the coordinates of any path node are (x_i, y_i, z_i). The constraints on $\phi_{\max}$ and $\gamma_{\max}$ are calculated as follows:

$$\begin{cases} \phi_i = \arccos\left(\dfrac{\mathbf{a}_i^T \mathbf{a}_{i+1}}{\|\mathbf{a}_i\|_2 \|\mathbf{a}_{i+1}\|_2}\right) \le \phi_{\max} \\ |\gamma_i| = \left|\arctan\left(\dfrac{z_i - z_{i+1}}{\|\mathbf{a}_i\|_2}\right)\right| \le \gamma_{\max} \end{cases} \tag{4}$$

where the vector $\mathbf{a}_i$ is $\left[x_i - x_{i-1}, y_i - y_{i-1}\right]^T$.

Therefore, in this paper, the cost function between two points x_1 and x_2 in the configuration space is defined as follows:

$$f_c(x_1, x_2) = \begin{cases} \rho(x_1, x_2) & \phi \le \phi_{\max}, |\gamma| \le \gamma_{\max} \\ +\infty & \text{otherwise} \end{cases}, \tag{5}$$

where ϕ and γ are the steering angle and climbing/diving angle of the UAV flying along the planned path, respectively, which can be calculated through the geometric relationship between the two adjacent path segments. If these angles exceed their limits, it is considered that the planned path cannot meet the needs of the UAV, so the cost function is set to infinity. Thus, the cost function of the planned path can be expressed as:

$$c(\sigma) = \sum_{t=1}^{T-1} f_c(\sigma(t), \sigma(t+1)). \tag{6}$$

HPO-RRT* algorithm

In this section, we introduce the HPO-RRT* algorithm proposed in this paper in detail.

Overall framework

To address the path-planning problem of UAVs in dynamic environments with moving threats, the algorithm must be able to solve dynamic problems quickly. However, only obtaining the feasible path is not an optimal choice: the algorithm should also be able to quickly obtain the optimal solution. Note that many time-based RRT* algorithms need to prune the tree structure when optimizing the path, which inevitably results in more wasted branches and additional calculations of time-based nodes in the correction tree.

Therefore, the HPO-RRT* algorithm is proposed in this paper to satisfy the efficiency of real-time path planning and the superiority of paths in the dynamic environment of UAVs. The algorithm adopts a hierarchical framework for path planning and optimization. First, the UAV perception system is used for threat prediction and world information updates. Then, an improved time-based RRT* algorithm is used as the path planner. Finally, a low-cost path optimizer is used to optimize the path obtained by the planner. The pseudocode of the algorithm is shown in Algorithm 1, including one main cycle and five main processes.

(1) Initialization (lines 1–3). This process initiates the HPO-RRT* algorithm. σ_{opt} is treated as an empty set. Similarly, the root node of tree T is set to x_{init}, and the current positions x_{cur} of the UAV and time are initialized.

(2) Termination condition (line 4). After detecting that x_{cur} has reached the target area X_{goal}, the program is immediately terminated. Otherwise, continue running the program until x_{cur} arrives at X_{goal}.

(3) UAV perception system (lines 8–15). During any execution, the UAV will fly a certain distance according to the planned path, and then the planner will update the final state x_{cur} of the drone in this execution to the root of the new tree T^*_{new}. Next, nodes with timestamps less than x_{cur} in the tree and their descendants need to be deleted. Then, taking zero as the initial depth of x_{cur}, the trajectory of the threat in the future time $N_l \Delta t$ is predicted through the navigation system, where N_l is the maximum depth. Because of the movement of threats and UAVs, all information of maps, trees, UAVs and moving threats must be updated . This process is basi-

cally the same as that of Risk-RRT [20] except that the perception system is expanded from 2 to 3D. This paper mainly focuses on improving path planning and optimization, so the description of the perception process refers to [20].

(4) Path planner (lines 16–20). The existing time-based RRT algorithm usually uses RRT as the bottom planner [15, 31]. Obviously, the planned path is not optimal, and a feasible path may not be obtainable within the time interval. In this paper, we use an improved time-based RRT* as the path planner. First, the bias sampling method based on the APF function is used to guide sampling node $x_{apfrand}$ towards the target region and away from the obstacle. Then, the time-based RRT* process of selecting the optimal parent node in the neighbourhood of the new node is introduced to optimize the tree structure, and the rewiring process in time-based RRT* is removed to avoid damage to the tree structure. Meanwhile, the tree structure is extended in combination with the lazy collision checking process. Finally, the heuristic path is obtained through the tree structure.

(5) Path optimizer (line 21). Because all the nodes in the tree have time parameters and the time is irreversible, using the rewiring process in the tree structure to optimize the path will lead to the disorder of node timestamps in the tree, and repairing the tree structure will consume considerable extra time. Therefore, this paper designs a low-cost path optimizer that directly optimizes the heuristic path without changing the tree structure to obtain the optimal path σ_{opt}.

Algorithm 1: HPO-RRT*

1 σ_{opt}, T.E $\leftarrow \varnothing$, T.V $\leftarrow \{x_{init}\}$
2 $x_{cur} \leftarrow x_{init}$
3 $t \leftarrow$ **UpdateClock()**
4 **while** $x_{cur} \notin X_{goal}$ **do**
5 **if** $\sigma_{opt} = \varnothing$
6 break
7 **else**
8 $x_{cur} \leftarrow$ *move along* σ_{opt} *for one step*
9 **endif**
10 $T^{*}_{new} \leftarrow$ **DeleteUnreachability**(T, x_{cur}, t)
11 $t \leftarrow$ **UpdateClock()**
12 **Observe & PredictMovingObs**$(t, N_l, \Delta t)$
13 **if** *new information is dectected*
14 $\Delta \mathbf{X}_{obs} \leftarrow$ **UpdateObs**$(t, \Delta t)$
15 **endif**
16 **while UpdateClock()** $< t + \Delta t$ **do**
17 $x_{apfrand} \leftarrow$ **SampleAPF**$(x_{goal}, \mathbf{X}_{obs})$
18 $T_{new} \leftarrow$ **Extend**$(T^{*}_{new}, x_{apfrand}, \mathbf{X}_{obs}, N_c, N_l)$
19 **endwhile**
20 $\sigma_{heu} \leftarrow$ **HeuristicPathFinding**(T_{new}, x_{goal})
21 $\sigma_{opt} \leftarrow$ **Low- costOptimizer**(σ_{heu})
22 $t \leftarrow$ **UpdateClock()**
23 **endwhile**

Path planner

Bias sampling based on the APF function

In the traditional RRT algorithm, the sampling nodes are randomly selected in the collision-free configuration space, and then the extended nodes generate a search tree to the target area. Although this method can ensure the completeness of probability, in the three-dimensional space of UAV flight, the search efficiency is low because of the large random sampling space. In addition, HPO-RRT* is a planning algorithm based on time constraints. The path-planning time of the UAV will be limited to a fixed time interval, so the time of path planning is relatively short. Random sampling is inefficient and may be unable to plan a feasible path. Therefore, to plan the path efficiently, this paper proposes a bias sampling method based on APF functions, which makes the sampling nodes expand towards the target region and away from the threat to plan the path efficiently in a limited time.

The APF constructs the attractive potential field of the target position and the repulsive potential field of the threat. The resultant force $\mathbf{F}_{apf}$ generated by these two potential fields guides the UAV to move towards the target point. In this paper, the following functions are used to define the attractive U_{att} and the repulsive U_{rep} potential fields, as well as their corresponding force vectors $\mathbf{F}_{att}$ and $\mathbf{F}_{rep}$, respectively. U_{att} and $\mathbf{F}_{att}$ are:

$$U_{att}(x) = \frac{1}{2} k_{att} \rho^2(x, x_g), \tag{7}$$

$$\mathbf{F}_{att}(x) = -\nabla(U_{att}) = -k_{att}\rho(x, x_g)\frac{\partial \rho(x, x_g)}{\partial x}, \tag{8}$$

U_{rep} and $\mathbf{F}_{rep}$ are

$$U_{rep}(x) = \begin{cases} \frac{1}{2} k_{rep}\left(\frac{1}{\rho(x, x_{obs})} - \frac{1}{\rho_o}\right)^2 & \rho(x, x_{obs}) \le \rho_o \\ 0 & \rho(x, x_{obs}) > \rho_o \end{cases}, \tag{9}$$

$$\mathbf{F}_{rep}(x) = \begin{cases} k_{rep}\left(\frac{1}{\rho(x, x_{obs})} - \frac{1}{\rho_o}\right)\frac{1}{\rho(x, x_{obs})^2}\frac{\partial \rho(x, x_{obs})}{\partial x} & \rho(x, x_{obs}) \le \rho_o \\ 0 & \rho(x, x_{obs}) > \rho_o \end{cases}, \tag{10}$$

where ρ_o is the maximum impact distance of the threat, and x_{obs} is the location of the threat.

According to Eqs. (7) and (9), the calculation of random sampling through APF bias is described as follows:

$$\mathbf{F}_{att\_bias} = -k_{att}\rho(x_{rand}, x_{goal}), \tag{11}$$

$$\mathbf{F}_{rep\_bias} = k_{rep}\left(\frac{1}{\rho(x_{rand}, x_{obs})} - \frac{1}{\rho_o}\right)\frac{1}{\rho(x_{rand}, x_{obs})^2}, \tag{12}$$

$$\mathbf{F}_{apf\_bias} = \mathbf{F}_{att\_bias} + \mathbf{F}_{rep\_bias}. \tag{13}$$

Then, the bias sampling node can be described as:

$$x_{apfrand} = x_{rand} + k_{bias} \frac{\mathbf{F}_{apf\_bias}}{\left|\mathbf{F}_{apf\_bias}\right|}, \tag{14}$$

where $k_{bias} \in \mathbb{R}_{>0}$ is the bias step.

Note that path planning using the APF function has disadvantages, such as inaccessible targets and easily falling into local minima. However, in this paper, the APF function is only used to bias the random sampling, not to find the path. At the same time, each sampling is an independent random event. Therefore, the shortcomings of the APF function can be effectively avoided.

In this paper, we further improve the approach described in [28]. Qureshi and Ayaz [28] calculate the influence of the attractive potential field on x_{rand} and determine whether to reserve bias sampling nodes by determining whether the distance between x_{rand} and x_{obs} is within the range of ρ_o. Through the strategy of [28], the random sampling nodes outside the influence range of the repulsive potential field can be biased towards the goal, thereby improving the sampling efficiency. However, if the nodes are in the influence range of the repulsive potential field, they will be discarded. At this time, new nodes will be randomly sampled again to bias. Note that if the random sampling node within the influence range of the repulsive potential field can also be biased, repeated random sampling and collision check of this node will be avoided, thus further improving the sampling efficiency. Therefore, this paper introduces biased sampling of the repulsive potential field to retain all random sampling and expand the expansion space of the tree as much as possible. The pseudocode of the APF bias sampling is shown in Algorithm 2.

Algorithm 2: **SampleAPF**$\left(x_{goal}, \mathbf{X}_{obs}\right)$

$x_{rand} \leftarrow$ **sample_ random** $\left(\mathbf{X}_{free}\right)$
$\mathbf{F}_{att\_bias} = -k_{att}\rho(x_{rand},\ x_{goal})$
if $\rho(x_{rand},\ x_{obs}) \le \rho_o$
  $\mathbf{F}_{rep\_bias} = k_{rep}(\frac{1}{\rho(x_{rand}, x_{obs})} - \frac{1}{\rho_o})\frac{1}{\rho(x_{rand}, x_{obs})^2}$
else
  $\mathbf{F}_{rep} = 0$
endif
$\mathbf{F}_{apf\_bias} = \mathbf{F}_{att\_bias} + \mathbf{F}_{rep\_bias}$
$x_{apfrand} = x_{rand} + k_{bias}\frac{\mathbf{F}_{apf\_bias}}{\left|\mathbf{F}_{apf\_bias}\right|}$
return $x_{apfrand}$

First, the **sample_random** function is used for random sampling in collision-free space X_{free}. Second, the attractive force $\mathbf{F}_{att}$ of x_{goal} to x_{rand} and the repulsive force $\mathbf{F}_{rep}$ of x_{obs} to x_{rand} are calculated. Third, the resultant force $\mathbf{F}_{apf}$ is calculated and then used to bias x_{rand} to $x_{apfrand}$. The final bias sample node $x_{apfrand}$ is obtained.

The expansion of the time-based tree

Similar to RRT*, time-based Risk-RRT* can also optimize the tree structure through the rewiring process to obtain asymptotically optimal partial planned paths when addressing path-planning problems with moving threats. However, because of the irreversibility of time, the rewiring process of Risk-RRT* will destroy the time-based tree structure. The time spent repairing the tree structure increases exponentially with the size of the tree, reducing the planning efficiency. In addition, in the sampling-based path-planning algorithm, collision checking is very time-consuming. Because the planning time using HPO-RRT* is a fixed short time interval, the planning efficiency is particularly important. Therefore, in this paper, the neighbouring parent selection process of Risk-RRT* is retained in the process of expanding the tree to optimize the tree structure, while the rewiring process is eliminated to ensure the efficiency of the path planner. Although this approach will sacrifice the optimality of the path, the path can be quickly optimized by a low-cost path optimizer that is more efficient than the rewiring process. At the same time, to expedite collision checking, a method of lazy collision checking is proposed in this paper, which improves the computational efficiency by ensuring that the planned path is collision free.

In the HPO-RRT* algorithm, the expansion of nodes is closely related to time. Each forward expansion of a node in the tree indicates that the execution time of the UAV increases by the time step Δt. Therefore, any node $x_i \in T.V$ in the tree needs to record its timestamp, i.e. $x_i : \left(x_i^*,\ t + N_i \Delta t\right)$, where x_i^* is the position coordinate of x_i, t is the start time of this extension process, and N_i is the number of time steps required to extend from the root node x_{root} of the current spanning tree to x_i.

Note that the **Extend** process is improved by using time-based RRT* as the basic planner. It only includes the process of selecting the optimal parent node in the neighbourhood of the new node x_{new} and omits the rewiring process. This approach is used because only after selecting the optimal parent node of x_{new}, x_{new} will be added to the time-based tree structure to obtain the timestamp. However, the time-based tree needs to update the timestamp of the node and all its descendants when rewiring the node and conducting collision checking again, which greatly increases the calculation time. A simple but typical example is shown in Fig. 2. In Fig. 2a, after adding x_{new} to the time-based tree, the rewiring process will be executed in the neighbourhood of x_{new}. Obviously, the cost from x_0 through x_{new} to x_8 is less than the cost from x_0 to x_8 in the original tree. If rewiring is performed, i.e. the new parent node of x_8 is x_{new} (called x_2' in (b)), the time-based

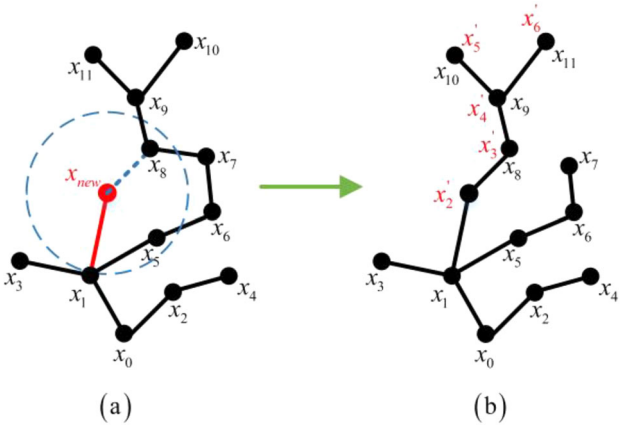


Fig. 2 Rewiring process of the time-based tree

tree will be updated to the structure shown in Fig. 2b. At this time, the timestamps of x_8, x_9, x_{10} and x_{11} are affected by rewiring and need to be updated to the timestamps of $x_3^{'}$, $x_4^{'}$, $x_5^{'}$ and $x_6^{'}$, respectively. In addition, because of the change in the timestamp, the collision checking of these update nodes must be reperformed to ensure that the nodes will not collide with the moving threats under the new timestamp. If collision checking cannot be passed, all branches formed by this node and all its child nodes need to be deleted. This process brings many additional computing costs. Therefore, in the HPO-RRT* algorithm, we delete the rewiring process of the time-based RRT* to reduce the computational consumption and meet the requirements of efficient and fast tree structure expansion within a fixed time interval. In addition, we keep the process of x_{new} reselecting the new parent node. This process is retained because it is a forwards expansion process, which will only affect the relevant parameters of the new nodex_{new}. Therefore, it will not bring additional computing consumption. Moreover, it can optimize the tree structure to a certain extent.

The planning space in this paper contains static and moving threats. For static threats, the position remains fixed at each time step. However, the coordinates of moving obstacles change with time. During the planning of the current time interval, the UAV can only refer to the threat prediction trajectory before the current planning to expand the tree by N_l time steps Δt (Algorithm 1, line 12). However, as the predicted threat trajectory may not be accurate enough, the path obtained in this partial planning may become infeasible because of the new prediction of the threat trajectory in the next planning. An intuitive idea is to set N_l very small to improve the accuracy of obstacle prediction and reduce the number of potentially invalid and repeated collision checking calls to improve the calculation efficiency. However, this approach makes the expansion range of the tree in each cycle very small, which is not suitable for the

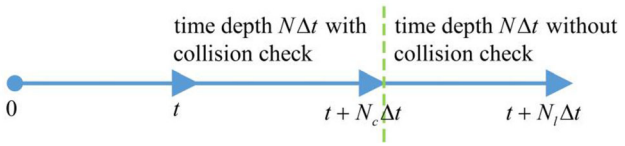


Fig. 3 Illustration of lazy collision checking

large-scale 3D planning space of UAVs. Therefore, to ensure a certain extension range of the forwards tree and reduce the computational loss, this paper proposes a method of lazy collision checking, which delays the collision checking process until it must be called. Figure 3 illustrates lazy collision checking. This method defines a new extended time depth $N_c < N_l$. In $N_c\Delta t$, all extended branches perform collision checking. The step length of each UAV execution is also set in $N_c\Delta t$. However, collision checking will not be performed in the time depth from $N_c\Delta t$ to $N_l\Delta t$. Because of the parallel planning and execution, the planning of the next time interval will start with a new root node, and the time depth for collision checking will continue to expand backwards. Therefore, all nodes on the final expansion tree will complete collision detection. Note that N_c greatly influences the performance of HPO-RRT*, so the calculation complexity and planning depth should be fully weighed.

Algorithm 3: **Extend**$\left(\mathrm{T}_{new}^{*}, x_{apfrand}, \mathbf{X}_{obs}, N_c, N_l\right)$

```
Notation x* is the position of x
x*_nearest ← Nearest(x* ∈ T.V, x_apfrand)
x*_new ← Steer(x*_nearest, x_apfrand)
X_near ← Near(T.V, r, x*_new)
X_parent ← OrderParent(x*_new, X_near)
while i=1,...,size[X_parent] do
   x_new ← (x*_new, t_new=t_parent,i + Δt)
   if t_new ≤ N_c Δt
      CollisionCheck(x_new, x_parent,i)
      if CollisionCheck = true
         return x_parent = x_parent,i
      else
         continue
      end
   else if N_c Δt < t_new ≤ N_l Δt
      CollisionCheck = true
      return x_parent = x_parent,i
   else if t_new > N_l Δt
      break
   endif
endwhile
T.V ← T.V ∪ x_new, T.E ← T.E ∪ (x_new, x_parent)
return T_new
```

The pseudocode of the **Extend** process is shown in Algorithm 3. After the **SampleAPF** process returns the biased sample $x_{apfrand}$, the forward expansion process begins. Similar to the time-based RRT*, the process of expanding the new node x_{new}^{*} is realized by the **Nearest**, **Steer** and **Near** functions. In the expansion process of x_{new}, only its position is considered temporarily (i.e. let x_{new} be x_{new}^{*}). First, the

Nearest function returns the closest node $x^*_{nearest}$ between $x_{apfrand}$ and the tree node set $T.V$, and then **Steer** generates a new node on x^*_{new}. Next, the algorithm searches the neighbourhood formed by the sphere $\mathfrak{B}_{x^*_{new},r}$ whose centre is x^*_{new} and radius is r (where $r = \gamma\left(\frac{\log n}{n}\right)^{1/d}$ [32]) to obtain the node set X_{near} whose distance from x^*_{new} in $\mathfrak{B}_{x^*_{new},r}$ is less than r. After connecting x^*_{new} to X_{near}, the timestamp t_{new} of x_{new} is arranged from small to large to form a potential parent node set X_{parent}. First, the algorithm judges whether t_{new} is within the time depth $N_c\Delta t$. If this criterion is met, the algorithm performs collision detection to select the optimal parent node of x_{new}; if $t_{new} \in (N_c\Delta t, N_l\Delta t]$, it directly selects without collision detection. If $t_{new} > N_l\Delta t$, x_{new} exceeds the maximum time depth limit of this planning. Finally, x_{new} and its edge formed with the optimal parent node x_{parent} are added to the tree T.

Planning path finding

Because the growth of the tree can only occur in the fixed planning time of a single cycle, and the maximum time depth of the node is limited to $N_l\Delta t$, for the 3D large-scale space X_{goal} in which the UAV flies, it is almost impossible to generate an expansion tree in which the node falls in the target area SD after only one or several cycles. Therefore, this paper introduces a cost function composed of the true distance cost and the estimated distance cost to find the heuristic partial path in the single planning spanning tree, namely, the **HeuristicPathFinding** process.

In a planning interval, tree expansion is limited by the maximum expansion depth N_l. That is, in the nodes of tree T_{new}, the maximum extension time depth is $N_l\Delta t$. Based on this limitation, the node set of the true path cost can be defined as X_l, including n_l nodes x_l with a time depth of $N_l\Delta t$, i.e. $X_l := \left\{(x^*_{l,1}, N_l\Delta t), ...(x^*_{l,i}, N_l\Delta t), ..., \left(x^*_{l,nl}, N_l\Delta t\right)\right\}$. The set of true costs of all paths that can reach $x_{l,i}$ in the current tree is defined as $g(X_l)$. The set of estimated costs from any node in X_l to the target region X_{goal} is defined as $h(X_l)$. Then, the cost $f(X_l)$ of the path is:

$$f(X_l) = g(X_l) + h(X_l). \tag{15}$$

We aim to find the node $x_{l,i}$ that can minimize $f(X_l)$, i.e.

$$x_{l,\min} = \underset{x_{l,\min}\in X_l}{\arg\min}\ f(X_l). \tag{16}$$

For the time-based tree, the exact value of $g(X_l)$ can be obtained by calculating the path length, while $h(X_l)$ can only be estimated. Therefore, an acceptable estimation cost calculation method should be selected so that the real cost of reaching x_{goal} is not overestimated. The following formula is used for calculating the estimated cost $h\left(x_{l,i}\right)$ of any node $x_{l,i}$:

$$h\left(x_{l,i}\right)_{x_{l,i}\in X_l} = k_l\rho\left(x_{l,i}, x_{goal}\right), \tag{17}$$

where k_l is the heuristic parameter.

Algorithm 4: **HeuristicPathFinding**$\left(\mathrm{T}_{new}, x_{goal}\right)$

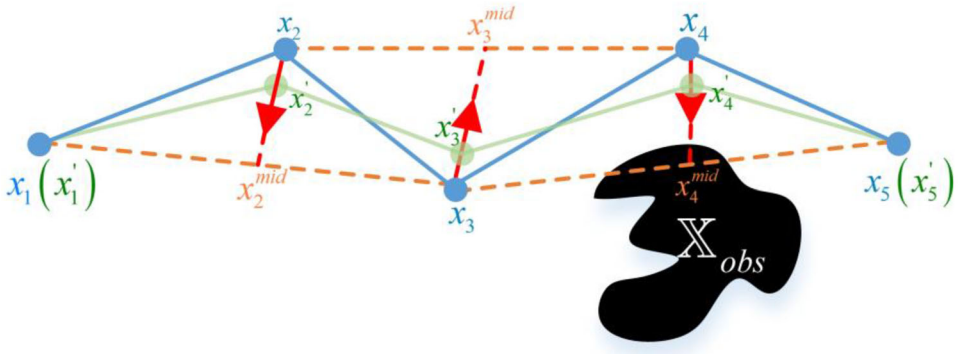
```
X_l ← find(t_l = t + N_l Δt, T_new.V)
f(x_i) ← ∞
for all x_i ∈ X_l do
    if g(x_i) + k_l ρ(x_i, x_goal) < f(x_i)
        x_i,min ← x_i
    endif
endfor
σ_heu ← RetroRoot(g(x_i,min))
```

On the basis of the above calculation, we ensure that in each partial planning, we can find a heuristic partial path towards the target from the current time-based tree. Algorithm 4 gives the pseudocode of the **HeuristicPathFinding** process. First, the algorithm finds all nodes with time depth t_l in tree T_{new} and forms a set X_l. Then, the node $x_{i,\min}$ that minimizes the estimation cost is found by $f(X_l)$. Finally, the heuristic path σ_{heu} generated by the planner is found by $x_{i,\min}$.

Low-cost path optimizer

The path planner using the HPO-RRT* algorithm efficiently generates heuristic paths without guaranteeing the optimality of the paths. Although the time-based Risk-RRT* algorithm can obtain the optimal path, as an online path-planning algorithm used to address moving threats, it takes a long time in the optimization process and sometimes cannot optimize the path well in a short planning time. Qi et al. [33] uses RRT* to find the initial path and then uses the ant colony (ACO) algorithm to reoptimize the path. This hierarchical optimization is interesting, but the ACO algorithm brings high computational complexity. The Bellman–Ford (BF) algorithm [34], as an algorithm for solving the shortest path, provides a method for searching the shortest path in a weighted graph. Therefore, inspired by [33, 34], combined with the requirements of online dynamic path planning, this paper proposes a low-cost fast path optimization method. This method directly optimizes the heuristic path σ_{heu} generated by the path planner without changing the tree structure. Therefore, additional complex calculations are not needed to optimize the path more quickly and effectively.

Fig. 4 Bias optimization



The BF algorithm estimates the distance from any node to a specified node by relaxing the trigonometric inequality constraint of the connection between nodes in the weighted graph. In the path optimization process of HPO-RRT*, we use this algorithm to offset the path points to optimize the heuristic path. As shown in Fig. 4, the blue path is the heuristic path σ_{heu} that has not been optimized, and the green translucent path is the path σ^* that has been optimized once. In the cycle, bias optimization is performed on the nodes in σ_{heu} (excluding the initial point and the target point) in turn. After all the path nodes are optimized once, path B is obtained. Obviously, $cost(\sigma^*) < cost(\sigma_{heu})$.

To clearly illustrate the optimization process, the offset to a is taken as an example. For the three nodes x_1, x_2 and x_3 in the tree, the path cost from x_1 directly to x_3 is minimal. However, if x_1 and x_3 are directly connected (that is, connected by the orange dotted line), the timestamps of x_3 and all its descendants need to be updated, resulting in many extra computations. In addition, if $\mathbb{X}_{obs}$ exists nearby, such as the handling of node x_4, the directly connected edge (x_3, x_5) will collide with the threat, causing the optimization path to fail. Therefore, we try to optimize the path by biasing the position of node x_2 towards edge (x_1, x_3). The specific method is to find the midnode x_2^{mid} of edge (x_1, x_3). Then, move x_2 along the vector $\overrightarrow{x_2 x_2^{mid}}$ by a step λ_{opt} to obtain the optimized node. For any node x_i in the path, the optimization step λ_{opt} is calculated as follows:

$$\lambda_{opt} = k_{opt}\left(\frac{(x_{i-1}+x_{i+1})/2 - x_i}{\rho((x_{i-1}+x_{i+1})/2,\ x_i)}\right), \tag{18}$$

where $k_{opt} \in \mathbb{R}_{>0}$ a is the bias factor. Then, the optimized node x_i of any node x_{new} in the path can be expressed as:

$$x_{new} = x_i + \lambda_{opt}. \tag{19}$$

Algorithm 5: **Low-costOptimizer**(σ_{heu})

```
σ* ← σ_heu
while FinishOptimization=false do
  σ_opt = ∅; σ_opt ← σ*(0)
  for i=1,...,size[σ*]−1 do
    x_{i−1} = σ*(i−1)
    x_i = σ*(i)
    x_{i+1} = σ*(i+1)
    x_new = x_i + k_opt ( ((x_{i−1}+x_{i+1})/2 − x_i) / ρ((x_{i−1}+x_{i+1})/2, x_i) )
    if CollisionCheck(x_{i−1}, x_new, x_{i+1})
      ∧ MaxAngleCheck(x_{i−1}, x_new, x_{i+1}) == true
        σ_opt(i) ← x_new
    else
        FinishOptimization=true
        σ_opt ← σ*
        return σ_opt
    endif
  endfor
  x_end ← σ_opt(size[σ_heu])
  if cost(σ*) − cost(σ_opt) < ε
    FinishOptimization=true
    σ_opt ← σ*
    return σ_opt
  else
    σ* ← σ_opt
    FinishOptimization=false
  endif
endwhile
```

Algorithm 5 gives the pseudocode of the **Low-costOptimizer** process. Define a function to determine whether the optimization process is completed and assign it as false at the beginning of the algorithm. The function

CollisionCheck$(x_{i-1}, x_{new}, x_{i+1})$ means collision checking for the path segment from x_{i-1} through x_{new} to x_{i+1}. If the result is true, then $(x_{i-1}, x_{new}, x_{i+1})$ is a collision-free path segment. The function **MaxAngleCheck**$(x_{i-1}, x_{new}, x_{i+1})$ indicates the UAV maximum angle constraint detection for the path segment $(x_{i-1}, x_{new}, x_{i+1})$. Note that line 18 compares the cost between the path before the current cycle and the optimized path to complete the current cycle. If the cost difference between the two paths before and after optimization is less than the threshold ε, where $\varepsilon \in \mathbb{R}$, then the path is hardly changed in the optimization process. When this result is obtained, the improvement of path quality in the optimization process is no longer obvious, i.e. the optimal path is essentially determined. Therefore, assign **FinishOptimization** to true.

In any optimization cycle, first, the initial position is added to the optimization path σ_{opt}. Next, the nodes in heuristic path σ_{heu} are biased optimized. If the optimized path meets the constraints of collision detection and UAV maximum angle, then the optimized path of this cycle planning is feasible. Finally, if the generated path can improve the path quality, start the next cycle with this path as the path to be optimized. The termination conditions of the optimization process shown in Algorithm 5 include the following criteria: (1) the process is terminated when collision detection or the maximum angle constraint of the UAV cannot be met; (2) in the same cycle, if the costs of the optimized path σ_{opt} and the nonoptimized path σ^* differ by less than ε, the process terminates.

Obviously, throughout the optimization process, the original tree structure is unchanged, and the timestamps of all nodes are unmodified. We can obtain a better path σ_{opt} than heuristic path σ_{heu} by performing bias optimization on the nodes of the heuristic path. Despite additional computations, the computational cost of our method is negligible relative to methods that optimize tree structures to optimize paths.

Analysis

In this section, we analyse the probability completeness and homotopy optimality of HPO-RRT*.

Probabilistic completeness

Probabilistic completeness is mainly used to analyse the ability of algorithms to find feasible solutions. The HPO-RRT* algorithm finds the path in a single plan through the path planner and then optimizes the path directly generated by the path planner. Therefore, the probabilistic completeness analysis of HPO-RRT* only needs to consider some paths obtained through the path planner in a single plan.

Let V_n^{ALG} represent the set of tree nodes generated by algorithm ALG after iteration n. Definition 1 gives the definition of probabilistic completeness.

Definition 1 (*Probabilistic completeness*). Given the initial node x_{init} and the goal region X_{goal}, if algorithm ALG can find the feasible path from x_{init} to $x_{goal} \in X_{goal}$ for any path-planning problem with a feasible solution, i.e.

$$\lim_{n\to\infty} \inf \| \left(V_n^{ALG} \cap P_{goal} \neq \emptyset \right) = 1, \tag{20}$$

then the algorithm is considered probability complete.

The RRT algorithm has been proven to be probability complete. HPO-RRT* covers all the key processes of RRT in the bottom planner. Similar to RRT, HPO-RRT* has probabilistic completeness, which is described in Theorem 1.

Theorem 1 (Probabilistic completeness of HPO-RRT*). *Given a path-planning problem, if there is a feasible solution, then*,

$$\lim_{n\to\infty} \| \left(V_n^{PTL-RRT*} \cap P_{goal} \neq \emptyset \right) = 1. \tag{21}$$

Proof of Theorem 1. The proof of Theorem 1 is based on the following three arguments.

(1) As shown in Algorithm 1, the vertex set $V^{PTL-RRT*}$ of the random tree T generated by HPO-RRT* includes node x_{init}, and $V_0^{PTL-RRT*} = x_{init}$, which is the same as RRT.
(2) Similar to RRT, the tree generated by HPO-RRT* is connected. In other words, any random sampling node can be connected to the tree.
(3) HPO-RRT* only plans the path within a fixed time interval of each cycle, and the time depth limits its expansion range. However, for any cycle i, HPO-RRT* is the same as the RRT algorithm and has probability completeness. That is, in cycle i, when sample n_i tends to infinity, the probability of HPO-RRT* finding a path from the current planning starting point (also the subtarget point of the last cycle planning, i.e. $x_{subgoal,i-1}$) to the subgoal node $x_{subgoal,i}$ is one, i.e.

$$\lim_{n_i\to\infty} \|_i \left(V_{n_i}^{PTL-RRT*} \cap x_{subgoal,i} \neq \emptyset \right) = 1. \tag{22}$$

When the number of cycles is infinite, the subgoal node $x_{subgoal,i}$ can surely fall into the goal region X_{goal}, and all partial paths are connected. Therefore, the probability that HPO-RRT* finds a path from x_{init} to X_{goal} can be expressed as:

$$\lim_{n_i\to\infty, i=1,\ldots N} \| \left(V_n^{PTL-RRT*} \cap X_{goal} \neq \emptyset \right) = \prod_{i=0}^{N} \lim_{n_i\to\infty} \|_i = 1. \tag{23}$$

Thus, Theorem 1 is proved.

Homotopy optimality

The asymptotic optimality of RRT* and many of its derivative algorithms has been proven. This section proves that HPO-RRT* has similar properties to these algorithms, which is called homotopy optimality.

Note that HPO-RRT* performs partial path planning within a fixed time interval of each planning. Therefore, the path that ensures the homotopy optimality of HPO-RRT* is obtained by optimizing the heuristic path in a fixed time interval planning, rather than the entire path from the initial position to the goal region. The relevant definitions of homotopy optimality and the proof process are as follows.

Definition 2 (*Homotopy class of feasible paths* [35]). For two arbitrary paths σ_1 and σ_2 with a fixed initial node and goal node, if one path can be continuously deformed into the other without intersecting any threat, then σ_1 and σ_2 are considered to belong to the same homotopy class $[\sigma]$.

When multiple paths belong to the same homotopy class $[\sigma]$, there must be a homotopy optimal path σ^*_{opt}, which is the least costly path in $[\sigma]$. Thus, Lemma 1 is given as follows.

Lemma 1 (Homotopy optimality of σ_{opt}). *Given that the homotopy class containing heuristic path σ_{heu} is $\left\{ [\sigma]_{\mathrm{PTL-RRT}*} \middle| \sigma_{heu} \in [\sigma]_{\mathrm{PTL-RRT}*} \right\}$, the optimization path σ_{opt} returned by the path optimizer of HPO-RRT* is the homotopy optimal path of the homotopy class, which is the path with the lowest cost.*

Proof of Lemma 1. In process re-optimization, we use a low-cost path optimization algorithm to quickly optimize heuristic path σ_{heu}. When the path meets the termination conditions of the process, an optimized path σ_{opt} is generated. Therefore, the proof of Lemma 1 can be divided into three arguments according to the conditions of path termination optimization.

(1) After the optimization process, the path cost is almost unchanged, i.e. $\Delta cost < \varepsilon$. As we analysed in "Low-cost path optimizer", if $\Delta cost < \varepsilon$, then the path has reached optimality, so the generated path σ_{opt} is the homotopy optimal path in the homotopy class.

(2) When the return value of $CollisionCheck$ is false, path σ_{opt} is obtained. First, we assume that the termination condition is triggered because edge cannot pass $CollisionCheck$. As shown in Fig. 5a, there are two kinds of optimized path (i.e. σ_{opt} is the blue path and σ'_{opt} is the green path). Suppose the σ'_{opt} whose cost is $cost\left(\sigma'_{opt}\right) < cost\left(\sigma_{opt}\right)$ and $cost\left(\sigma_{opt}\right) - cost\left(\sigma'_{opt}\right) > \varepsilon$, that is, σ'_{opt} must be closer to edge (x_1, x_4) than σ_{opt}. However, σ'_{opt} cannot pass the collision checking, and σ'_{opt} will collide with obstacles. This causes the $CollisionCheck$ procedure returns a false value. Then, process re-optimization stops and returns the optimized path σ_{opt} before the start of the current cycle as the optimal path. Therefore, no path costs less than σ_{opt}. Then, σ_{opt} can be claimed to be homotopy optimal in the homotopy class.

(3) When the return value of $MaxAngleCheck$ is false, path σ_{opt} is obtained. First, we assume that the termination condition is triggered because edge cannot pass $MaxAngleCheck$. Similar to Proof 2), Fig. 5b shows two kinds of optimized path (i.e.σ_{opt} is the blue path and σ'_{opt} is the green path). The edge (x_0, x_1) is the last segment of the optimized path obtained from the previous cycle planning It is assumed that there is a path σ'_{opt}, and $cost\left(\sigma'_{opt}\right) < cost\left(\sigma_{opt}\right)$. Then, σ'_{opt} must be closer to edge (x_1, x_4) than σ_{opt}. However, the steering angle of σ'_{opt} is ϕ', which is larger than the steering angle $\phi_{\max}$(i.e. the maximum steering of UAV) of σ_{opt}. If the angle ϕ' does not meet the UAV angle constraint, $MaxAngleCheck$ returns false. Obviously, ϕ' does not meet the constraint. Then,σ'_{opt} is infeasible, and re-optimization returns to path σ_{opt}. Therefore, no path costs less than σ_{opt}. It can be proven that σ_{opt} is homotopy optimal. Based on the above three arguments, σ_{opt} is proven to be the homotopy optimal path in the homotopy class containing σ_{heu}.

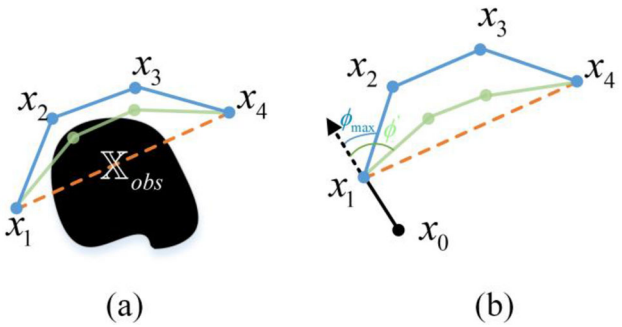


Fig. 5 Bias optimization

The results of simulation experiments

In this section, we give the simulation experiments of the proposed algorithm. The main purpose is to verify the feasibility

Table 1 Parameters of the moving threats in Scenario 1

	Starting moving time	Speed	Ending moving time
Radar 1	$t_{r\_start} = 0s$	$v_r = [2.6, 1.5, 0]$ m/s^2	$t_{r\_end} = 3$ s
Missile 1	$t_{m\_start} = 0s$	$v_m = [5, 0, 0]$ m/s^2	$t_{m\_end} = 5$ s
Anti-air gun 1	$t_{a\_start} = 11s$	$v_a = [4.5, 0.2, 0]$ m/s^2	$t_{a\_end} = 15$ s

and advantages of the algorithm in a UAV flight environment. Therefore, "Simulation setup" shows various threats in the considered UAV flight dynamic environment and the settings of some basic parameters to make the results as close to the real battlefield environment as possible to increase the reliability.

Simulation setup

Dynamic environment design

Constructing a battlefield environment is a prerequisite for path planning. The closer the construction of the battlefield environment is to reality, the better the verified path-planning algorithm will be implemented on UAVs. Therefore, to verify the feasibility and advantages of the HPO-RRT* algorithm in a battlefield environment, this paper simulates the real dynamic environment, including real terrain, radar, air defence missiles, anti-aircraft guns and a tower, as the configuration space for UAV path planning. We have already introduced the modelling and calculation of these threats in our previous work [36]. Since this paper mainly focuses on the planning of the HPO-RRT* algorithm, the modelling of threats will not be further described.

Based on the five threats in the above-mentioned real battlefield, this paper designs two 3D dynamic environments, including a threat with known moving trajectories and a random moving threat, to test the performance of the proposed algorithm and compare HPO-RRT* with related algorithms. The terrain threat in the two scenarios uses a unified elevation map. The two scenarios contain static or moving radar, missiles and anti-aircraft artillery threats.

Experiment Scenario 1 Dynamic flight scenario with known moving trajectory threats.

The environment size is 400*400*60 m^3. Radar 1, Missile 1 and Anti-aircraft gun 1 are moving threats, which start to move from their initial positions. The speed and moving time period are shown in Table 1. The remaining threats are static. When the moving threats touch the boundary of the configuration environment, they will rebound and then move in the opposite direction. There is no interaction between different moving threats.

Experiment Scenario 2 Dynamic flight scenario containing multiple random moving threats

The size of the environment is 400*400*60 m^3. The initial position of the terrain threat and the other three threats is the same as that of experimental Scenario 1. In Scenario 2, moving threats are randomly selected from all radars, air defence missiles and anti-aircraft guns in the environment. $\{1, 3, 5, 6\}$ threats can be selected to move at the same time. The movement speed is randomly selected from $\{1, 3, 5, 7\}$ m/s^2.The start time of the movement is any time during the planning period, and the time period of each movement is 3 s. All threats except those moving remain stationary. When the moving threats touch the boundary of the configuration environment, they rebound in the opposite direction and then continue to move. There is no interaction between different moving threats.

Basic parameter design

During the preparation of simulation experiments, the performance of the UAV, the location of threats in the environment, and the relevant basic parameters of the HPO-RRT* algorithm must be set. Table 2 shows the specific parameter names and values.

Simulation experiments

In this section, the HPO-RRT* algorithm is applied to the two dynamic scenarios with moving threats designed in "Simulation setup" to verify its feasibility, optimization, efficiency and success rate of the HPO-RRT* algorithm in the dynamic environment of UAV flight. Additionally, to further verify the performance of the HPO-RRT* algorithm, we compared it with three dynamic planning or replanning algorithms derived from RRT, including RRT^X, Risk-RRT and Risk-RRT*. In the two sets of scenarios, these algorithms are run many times to reduce the randomness of sampling-based algorithms. We evaluate the performance of each algorithm by taking the navigation time, planning path length and planning success rate as metrics.

Experiment Scenario 1 Figure 6 shows the UAV paths planned by the four algorithms in Scenario 1, and Table 3 shows some performance data in the experiment in Fig. 6. In general, compared with the other three algorithms, the path planned by HPO-RRT* is shorter and smoother. Moreover, in each partial planning, HPO-RRT* can quickly plan a longer path, so the planning efficiency and expansion range of this algorithm are also the best. In addition, HPO-RRT* has the

Table 2 Simulation experiment parameters

Parameter	Value			
k_{att}	20			
k_{rep}	30			
ρ_o	3 m			
k_{bias}	5			
Growing time Δt	0.5 s			
N_c	10			
N_l	15			
Maximum expansion step of a single node δ	15 m			
k_l	1			
$\phi_{\max}$	60°			
$\gamma_{\max}$	45°			
k_{opt}	1			
ε	0.1 m			
Start	(40, 40, 30) m			
Target	(350, 350, 50) m			
Experiment Scenario 1	Radar	**Centre**	**Radius**	
		1: (100, 80, 0) m	35 m	
		2: (100, 350, 5) m	35 m	
		3: (170, 230, 20) m	35 m	
		4: (280, 200, 20) m	35 m	
	Missile	**Centre**	**Radius**	**Height**
		1: (70, 170, 0) m	30 m	40 m
		2: (170, 140, 0) m	30 m	40 m
	Artillery	**Centre**	**Radius**	**Height**
		1: (300, 100, 0) m	30 m	40 m
		2: (260, 280, 0) m	25 m	40 m
	No-fly tower	**Centre**	**Radius**	**Height**
		1: (200, 290, 10) m	15 m	50 m
		2: (100, 275, 10) m	20 m	40 m
Experiment Scenario 2	Radar	**Centre**	**Radius**	
		1: (100, 90, 5) m	35 m	
		2: (170, 230, 20) m	35 m	
	Missile	**Centre**	**Radius**	**Height**
		1: (70, 170, 0) m	30 m	40 m
		2: (180, 140, 0) m	30 m	40 m
	Artillery	**Centre**	**Radius**	**Height**
		1: (300, 150, 0) m	30 m	40 m
		2: (270, 280, 0) m	25 m	40 m
	No-fly tower	**Centre**	**Radius**	**Height**
		1: (200, 290, 10) m	15 m	50 m
		2: (100, 275, 10) m	20 m	40 m

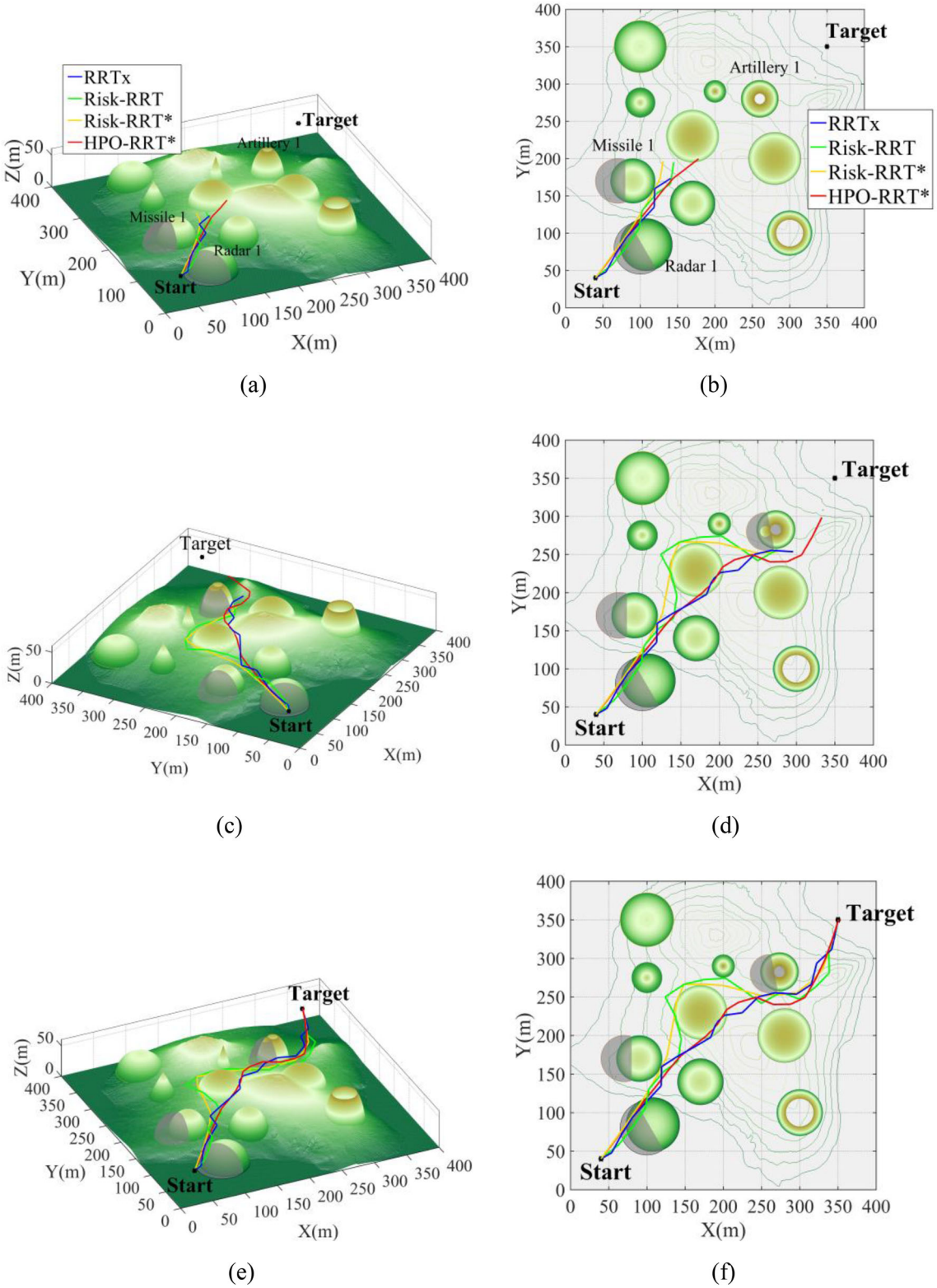


Fig. 6 Experiment Scenario 1. Planned path diagram of different algorithms at different times, in which the blue line represents RRT^X; the green line represents Risk-RRT; the yellow line represents Risk-RRT *; and HPO-RRT* is indicated by a red line. The grey translucent threat is the initial position of the three moving threats. **a**, **c** and **e** show path 3D diagrams at $t = 10s$, $20s$, $27.8s$, respectively; **b**, **d** and **f** show the contour top views of the above time path

Table 3 Length and navigation time of planned paths with the algorithms in Fig. 6

Algorithm	RRT^X	Risk-RRT	Risk-RRT*	HPO-RRT*
Length (m)	488.9	580.2	519.6	**477.9**
Time (s)	23.3	27.8	23.9	**21.2**

477.9 represents the shortest path length. 21.2 indicates the shortest running time

shortest planning and execution time, stable performance, and a high success rate.

RRT^X is a replanning algorithm. Compared with similar algorithms, RRT^X has faster information transmission speed and response speed and has better performance in the problem of path replanning in addressing moving threats. Although the tree structure needs to be pruned in RRT^X path planning, which greatly reduces the corresponding speed and planning efficiency of the algorithm, Fig. 6 shows that RRT^X can still maintain a good response speed and planning efficiency. In addition, when the tree structure is small at the initial stage of planning, the planned path quality is basically the same as that of the HPO-RRT* algorithm. However, when the sampling increases and the tree structure becomes larger, RRT^X takes a long time to prune the tree structure, thus reducing the time to optimize the path. Thus, the quality of the paths planned by RRT^X begins to lag behind that of HPO-RRT*. The path planned by RRT^X becomes less smooth, and the angle loss increases when the UAV flies along this path. Risk-RRT, Risk-RRT* and HPO-RRT* are time-based RRT algorithms. In other words, little time needs to be spent in tree structure pruning during planning, so they have a fast response speed. However, Risk-RRT is an algorithm whose path planner is time-based RRT, thus using Risk-RRT for planning will generally only obtain one feasible path. Furthermore, the optimality and planning efficiency of the planned path cannot be guaranteed. Therefore, we can see that the paths and navigation times obtained by Risk-RRT and HPO-RRT* are considerably different. Compared with Risk-RRT, the quality of the planned path of Risk-RRT* has been improved to some extent. Risk-RRT* is an algorithm that needs to rewire the tree structure to optimize the path, but Fig. 6 shows that the improvement of the path quality is not large. This result is obtained because part of the planning time is short, and the time allocated to the reconnection process is limited. However, Risk-RRT* requires a large number of rewiring calculations to optimize the path, so it cannot optimize the path better in a limited time. Similar to RRT^X, Risk-RRT* has good algorithm performance in the early stage due to its small tree structure. However, with increasing sampling, its performance gradually lags behind that of HPO-RRT*. It can be seen that HPO-RRT* always performs well in planning. HPO-RRT* uses APF bias sampling in the path planner to improve the quality and efficiency of sampling. At the same time, in the expansion of the tree,

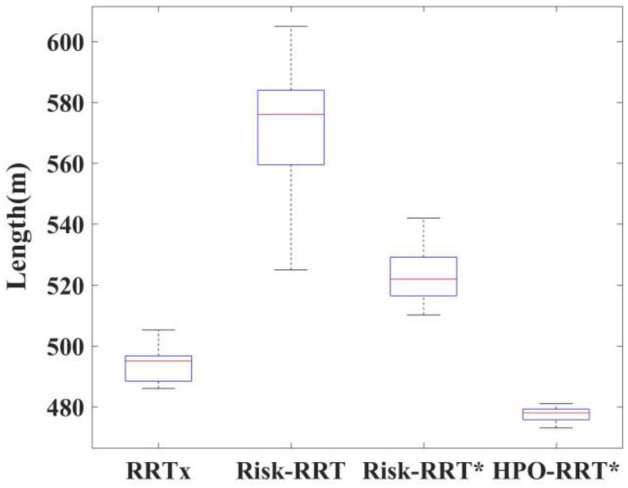


Fig. 7 Planned path length of 50 experiments of four algorithms

the lazy collision checking method is used to reduce the computational cost and expand the expansion range of the tree. After obtaining the heuristic path, HPO-RRT* uses the low-cost path optimizer to directly optimize the heuristic path to quickly obtain the optimized path. This optimization method does not change the structure of the time tree; that is, no additional calculation loss occurs in the optimization process. Therefore, compared with the other three algorithms, HPO-RRT* has the smallest navigation time, the shortest planned path and the highest smoothness.

Furthermore, in Scenario 1, each algorithm is repeated 50 times. We use the planned path length, navigation time and success rate to evaluate the performance of each algorithm. Figures 7, 8 and 9 compare the results. We find that the HPO-RRT* algorithm has good stability and algorithm performance. In the length comparison of 50 experiments, HPO-RRT* can use the low-cost path optimizer to fully and quickly optimize the path during planning so that the length of the planned path is always the shortest and the length variance is very small. Additionally, because HPO-RRT* maintains high planning efficiency, the navigation time can be quickly stabilized. Moreover, for the above reasons, the success rate of the algorithm is high. In other words, HPO-RRT* has good performance and stability when addressing UAV path planning in a dynamic environment with known moving threat trajectories.

Experiment Scenario 2 Experiment Scenario 2 focuses on the success rate and optimization of path-planning algorithms in a more complex dynamic environment. The algorithm

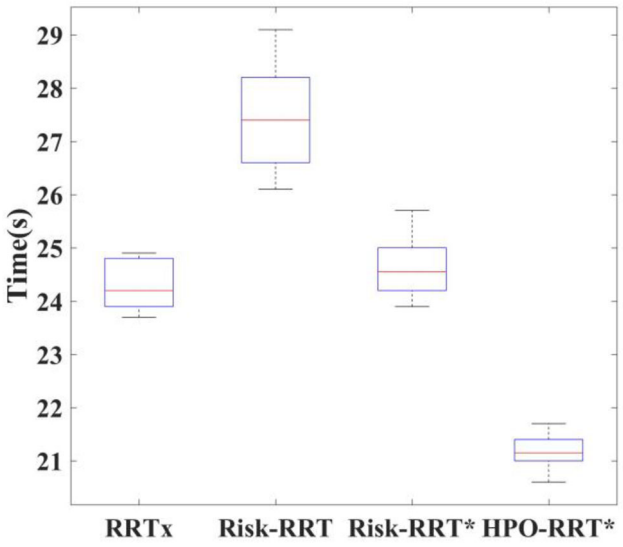


Fig. 8 Navigation time of 50 experiments of four algorithms

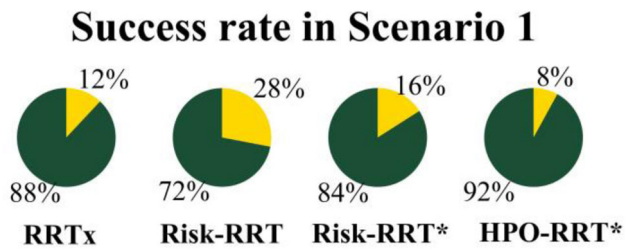


Fig. 9 Success rate

Table 4 Length and navigation time of planned paths with the algorithms in Fig. 9

Algorithm	RRT^X	Risk-RRT	Risk-RRT*	HPO-RRT*
Length (m)	526.7	616.0	546.3	**485.8**
Time (s)	35.1	38.4	36.2	**32.2**

485.8 represents the shortest path length. 32.2 indicates the shortest running time

performance is tested by setting the number and speed of moving threats in the UAV flight environment to a random generation state. Figure 10 and Table 4 show the performance of the four algorithms in one case with threats of random movement. Furthermore, for each combination of random threat number and speed in Scenario 2, each algorithm is tested in 6 trials. Therefore, for each algorithm, we conducted a total of 96 experiments on 16 combinations of random threats. We use the planned path length, navigation time and success rate to evaluate the performance of each algorithm. The performance of the algorithms is shown in Figs. 11 and 12. In general, compared with Scenario 1, the performance of the four algorithms declines in Scenario 2. This decline occurs because the four algorithms need to address more complex moving threats. However, compared with the other three algorithms, HPO-RRT* has the shortest and smoothest planned path length. The navigation time is also the shortest, and the success rate is also guaranteed. Therefore, the efficiency, effectiveness and stability of HPO-RRT* are demonstrated.

Figure 10 and Table 4 clearly show that the planning path quality of the HPO-RRT* algorithm remains superlative among the four algorithms. Because Risk-RRT has relatively low planning efficiency and no path optimization process, its planning path and navigation time are relatively poor. In more complex dynamic environments, Risk-RRT* does not have sufficient reconnection time, resulting in less obvious improvement of path quality. Under its ability of rapid replanning and optimization, RRT^X can obtain a faster navigation time and higher-quality paths than Risk-RRT and Risk-RRT*. With its efficient planner and optimizer, HPO-RRT* also maintains a high-quality planning path and has the shortest response time in a dynamic environment with random moving threats.

Furthermore, to verify the performance quality of HPO-RRT* more comprehensively, 96 simulations were conducted on 16 combined random moving threats. Figures 11 and 12 show the results. In the combination of a few moving threats and slow moving speed, the four algorithms easily plan the path, and all find the path with basically the same success rate, with the path length of HPO-RRT* being the shortest. However, with an increase in the number and speed of threats, the success rate of the algorithms declines. In addition, it can be seen that HPO-RRT* guarantees almost the same success rate as RRT^X, while the success rates of Risk-RRT and Risk-RRT * decrease significantly. Although HPO-RRT* as a time-based RRT algorithm has been proven to be inferior to the replanning algorithm (RRT^X) in addressing path-planning problems in highly dynamic environments (i.e. environments with fast-moving threats and more moving threats), HPO-RRT* can maintain good adaptability to dynamic environments with its high-quality sampling and efficient expansion, and the performance loss of this algorithm is low. In terms of path quality, HPO-RRT* maintains certain advantages in various combinations of random moving threats, which depend on the low-cost path optimizer of HPO-RRT* to quickly and fully complete path optimization at any time. Meanwhile, HPO-RRT* has the shortest and most stable navigation time with the good performance of the efficient path planner and the low-cost path optimizer in its hierarchical planning architecture. In other words, HPO-RRT* can adapt to more complex dynamic environments and has good performance and stability.

Conclusions and future work

In this paper, a time-based RRT named HPO-RRT* is proposed to solve the real-time path-planning problem of UAVs

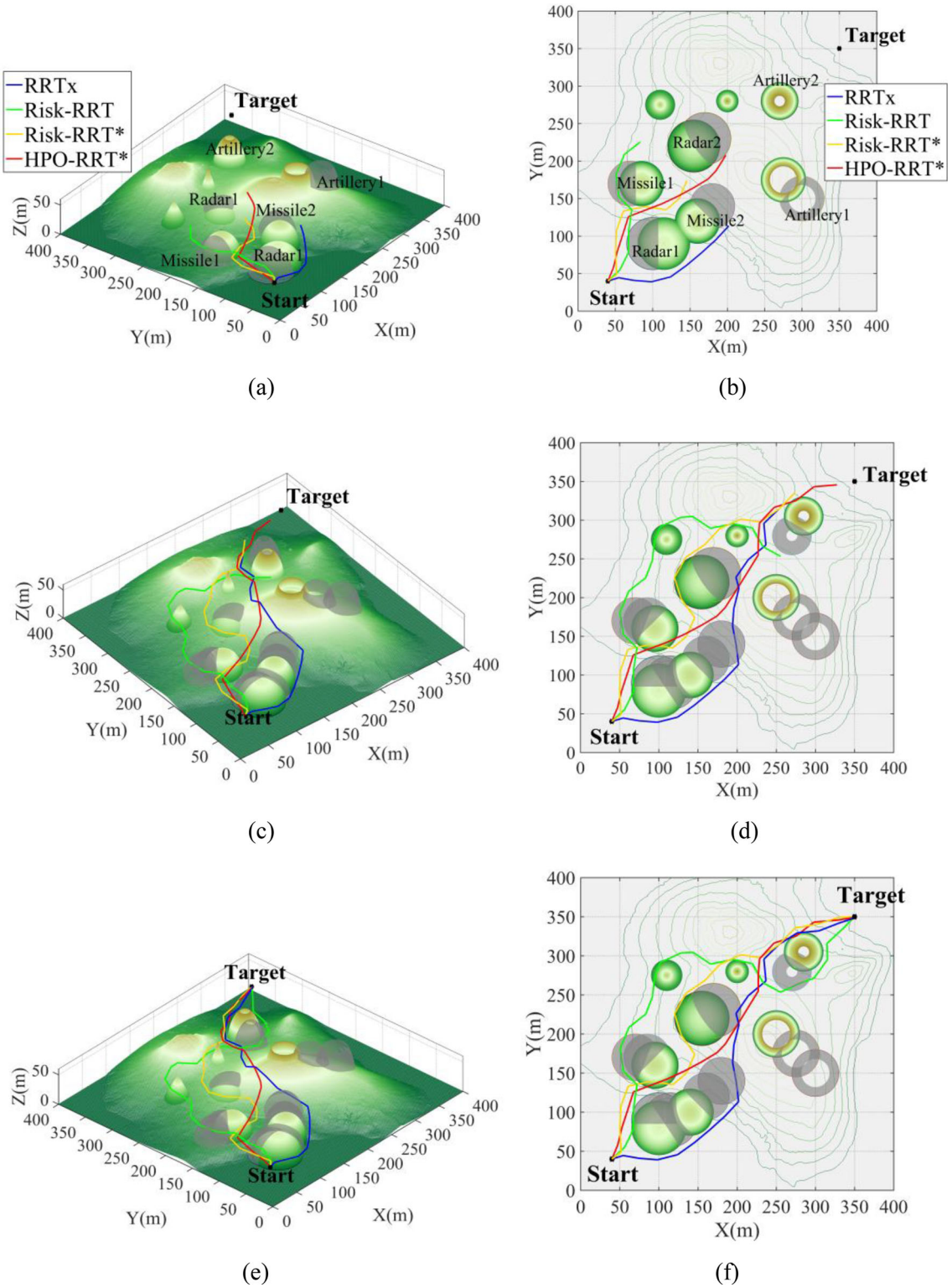


Fig. 10 Experiment Scenario 2. Planned path diagram of different algorithms at different times, in which the blue line represents RRT^X; the green line represents Risk-RRT; the yellow line represents Risk-RRT*; and HPO-RRT* is indicated by a red line. The grey translucent threat is the initial position of the three moving threats. **a**, **c** and **e** show the path 3D diagrams when t = 15 s, 30 s, 38.4 s, respectively; and **b**, **d** and **f** show the contour top views of the above time path

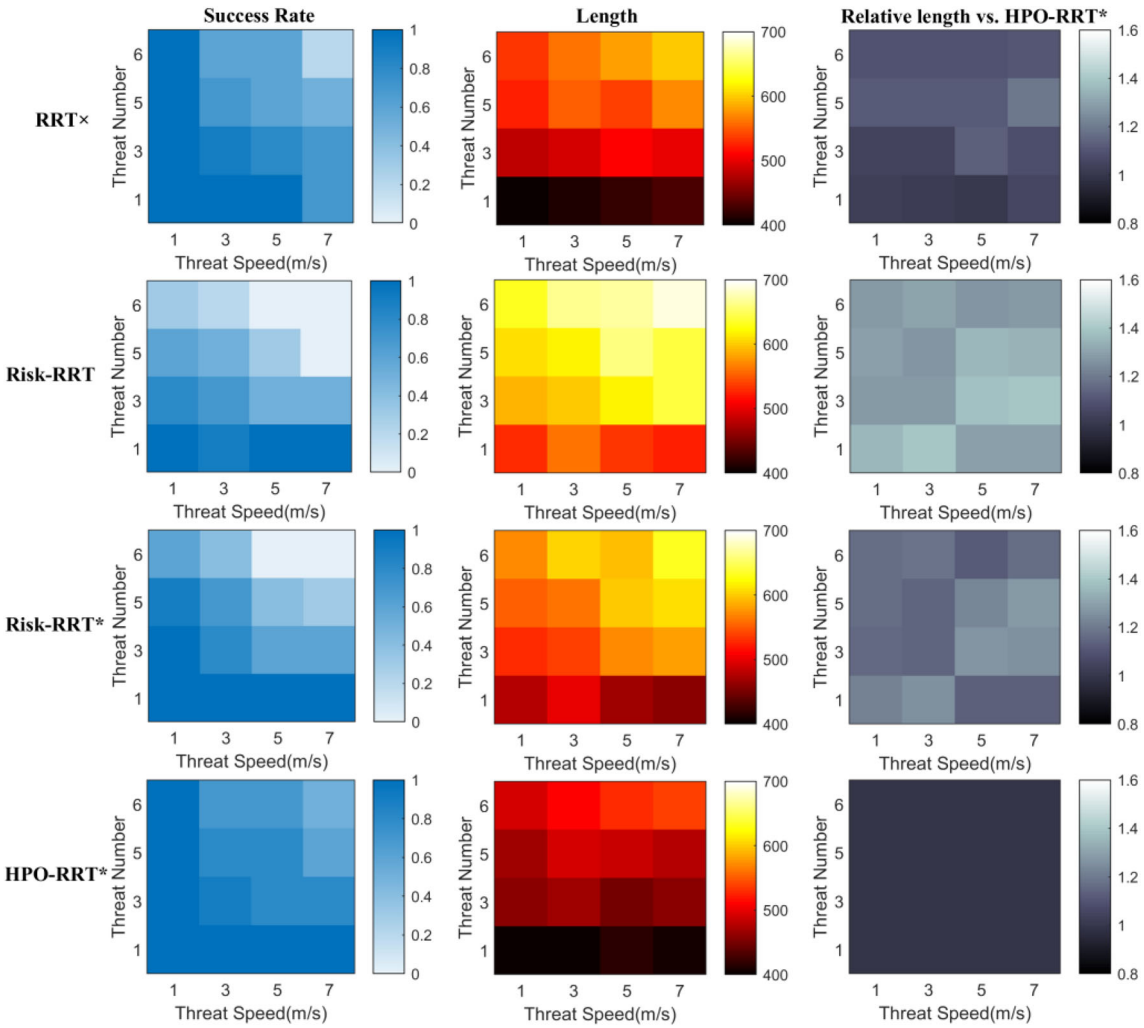


Fig. 11 Relationship between the success rate, average path length and obstacle speed and number

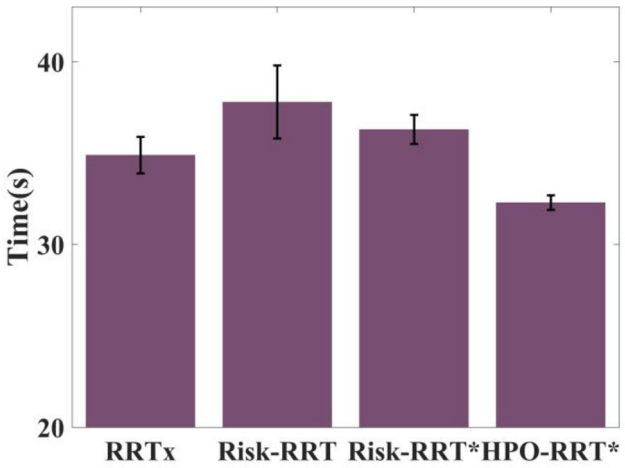


Fig. 12 Comparison of the mean and variance of the navigation time

in a dynamic environment. This algorithm adopts a hierarchical framework including the UAV perception system, path planner and path optimizer, which can optimize the path while improving efficiency. In the path planner, an improved time-based RRT* algorithm is used to obtain heuristic paths. The algorithm of the path planner improves the sampling process, tree expansion process and collision checking process to avoid additional calculation loss and improve planning efficiency. In the low-cost path optimizer, to avoid numerous calculations caused by the optimization tree structure, the heuristic path is directly optimized to quickly and efficiently obtain a homotopy optimal path. Simulations and comparisons show that the proposed algorithm has better performance.

In future work, we will conduct outdoor experimental verification on the independently developed quad-rotor platform to confirm the application value of the algorithm. In addition, we expect to extend HPO-RRT* to the formation and cooperation of UAVs in future work. An intuitive prospect is to use a formation framework to complete the cooperative path

planning of UAVs. Of course, a better prospect is to bring HPO-RRT* into the full autonomous cooperative architecture.

Acknowledgements Gratitude is extended to the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

Funding This research work was funded by the National Natural Science Foundation of China, Grant no. 62073266, the Aeronautical Science Foundation of China, Grant no. 201905053003.

Data availability Most of the data has been thoroughly described in the paper. The other data that support the findings of this study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethics approval Not applicable.

Consent to participate Yicong Guo, Xiaoxiong Liu, Qianlei Jia, Xuhang Liu and Weiguo Zhang.

Consent for publication Yicong Guo, Xiaoxiong Liu, Qianlei Jia, Xuhang Liu and Weiguo Zhang.

References

1. Fu B, Chen L, Zhou Y et al (2018) An improved A* algorithm for the industrial robot path planning with high success rate and short length. Rob Auton Syst 106:26–37. https://doi.org/10.1016/j.robot.2018.04.007
2. Patle BK, Babu LG, Pandey A et al (2019) A review: on path planning strategies for navigation of mobile robot. Def Technol 15:582–606
3. Rasekhipour Y, Khajepour A, Chen SK, Litkouhi B (2017) A potential field-based model predictive path-planning controller for autonomous road vehicles. IEEE Trans Intell Transp Syst 18:1255–1267. https://doi.org/10.1109/TITS.2016.2604240
4. Yao P, Wang H, Su Z (2015) Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment. Aerosp Sci Technol. https://doi.org/10.1016/j.ast.2015.09.037
5. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl. https://doi.org/10.1007/s11042-020-10139-6
6. Thoresen M, Nielsen NH, Mathiassen K, Pettersen KY (2021) Path planning for UGVs based on traversability hybrid A*. IEEE Robot Autom Lett 6:1216–1223. https://doi.org/10.1109/LRA.2021.3056028
7. Djukanovic M, Raidl GR, Blum C (2020) Finding longest common subsequences: new anytime A∗ search results. Appl Soft Comput J. https://doi.org/10.1016/j.asoc.2020.106499
8. Meng Z, Huang P, Yan J (2008) Trajectory planning for hypersonic vehicle using improved sparse A* algorithm. In: IEEE/ASME international conference on advanced intelligent mechatronics, AIM
9. Majumder S, Prasad MS (2016) Three dimensional D∗ algorithm for incremental path planning in uncooperative environment. In: 3rd international conference on signal processing and integrated networks, SPIN 2016. pp 431–435
10. LaValle SM, Kuffner JJ, Donald B et al (2000) Rapidly-exploring random trees: progress and prospects. Algorithmic Comput Robot New Dir 5:293–308. https://www.taylorfrancis.com/books/9781439864135/chapters/10.1201/9781439864135-43
11. Kavraki LE, Švestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12:566–580. https://doi.org/10.1109/70.508439
12. Chen Y, He Z, Li S (2019) Horizon-based lazy optimal RRT for fast, efficient replanning in dynamic environment. Auton Robots 43:2271–2292. https://doi.org/10.1007/s10514-019-09879-8
13. Ryu H, Park Y (2019) Improved informed RRT* using gridmap skeletonization for mobile robot path planning. Int J Precis Eng Manuf 20:2033–2039. https://doi.org/10.1007/s12541-019-00224-8
14. Webb DJ, Van Den Berg J (2013) Kinodynamic RRT*: asymptotically optimal motion planning for robots with linear dynamics. In: Proceedings—IEEE international conference on robotics and automation. pp 5054–5061
15. Wang J, Meng MQH, Khatib O (2020) EB-RRT: optimal motion planning for mobile robots. IEEE Trans Autom Sci Eng 17:2063–2073. https://doi.org/10.1109/TASE.2020.2987397
16. Shome R, Solovey K, Dobson A et al (2020) dRRT*: scalable and informed asymptotically-optimal multi-robot motion planning. Auton Robots 44:443–467. https://doi.org/10.1007/s10514-019-09832-9
17. Chandler B, Goodrich MA (2017) Online RRT∗ and online FMT∗: rapid replanning with dynamic cost. In: IEEE international conference on intelligent robots and systems. pp 6313–6318
18. Otte M, Frazzoli E (2016) RRTX: asymptotically optimal single-query sampling-based motion planning with quick replanning. Int J Rob Res 35:797–822. https://doi.org/10.1177/0278364915594679
19. Petti S, Fraichard T (2005) Safe motion planning in dynamic environments. In: 2005 IEEE/RSJ international conference on intelligent robots and systems, IROS. pp 2210–2215
20. Fulgenzi C, Spalanzani A, Laugier C, Tay C (2010) Risk based 1225 motion planning and navigation in uncertain dynamic environment. Res Rep 1–14. https://hal.inria.fr/inria-00526601
21. Chi W, Meng MQH (2017) Risk-RRT∗: a robot motion planning algorithm for the human robot coexisting environment. In: 2017 18th international conference on advanced robotics, ICAR 2017. pp 583–588
22. Thomas S, Morales M, Tang X, Amato NM (2007) Biasing samplers to improve motion planning performance. In: Proceedings—IEEE international conference on robotics and automation. pp 1625–1630

23. Gammell JD, Barfoot TD, Srinivasa SS (2018) Informed sampling for asymptotically optimal path planning. IEEE Trans Robot 34:966–984. https://doi.org/10.1109/TRO.2018.2830331
24. Gammell JD, Srinivasa SS, Barfoot TD (2015) Batch Informed Trees (BIT∗): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: Proceedings—IEEE international conference on robotics and automation. pp 3067–3074
25. Gasilov N, Doğan M, Arici V (2011) Two-stage shortest path algorithm for solving optimal obstacle avoidance problem. IETE J Res 57:278–285. https://doi.org/10.4103/0377-2063.83650
26. Otte M, Correll N (2013) C-FOREST: parallel shortest path planning with superlinear speedup. IEEE Trans Robot 29:798–806. https://doi.org/10.1109/TRO.2013.2240176
27. Yi J, Yuan Q, Sun R, Bai H (2022) Path planning of a manipulator based on an improved P_RRT* algorithm. Complex Intell Syst. https://doi.org/10.1007/s40747-021-00628-y
28. Qureshi AH, Ayaz Y (2016) Potential functions based sampling heuristic for optimal path planning. Auton Robots 40:1079–1093. https://doi.org/10.1007/s10514-015-9518-0
29. Pharpatara P, Herisse B, Bestaoui Y (2017) 3-D trajectory planning of aerial vehicles using RRT∗. IEEE Trans Control Syst Technol 25:1116–1123. https://doi.org/10.1109/TCST.2016.2582144
30. Zhang X, Duan H (2015) An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. Appl Soft Comput J 26:270–284. https://doi.org/10.1016/j.asoc.2014.09.046
31. Sintov A, Shapiro A (2014) Time-based RRT algorithm for rendezvous planning of two dynamic systems. In: Proceedings—IEEE international conference on robotics and automation. pp 6745–6750
32. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Robot Res 30(7):846–894. http://journals.sagepub.com/doi/10.1177/0278364911406761
33. Qi J, Yang H, Sun H (2021) MOD-RRT*: a sampling-based algorithm for robot path planning in dynamic environment. IEEE Trans Ind Electron 68:7244–7251. https://doi.org/10.1109/TIE.2020.2998740
34. Mo Y, Dasgupta S, Beal J (2019) Robustness of the adaptive bellman-ford algorithm: global stability and ultimate bounds. IEEE Trans Automat Control 64:4121–4136. https://doi.org/10.1109/TAC.2019.2904239
35. Liu Y, Zheng Z, Qin F (2021) Homotopy based optimal configuration space reduction for anytime robotic motion planning. Chinese J Aeronaut 34:364–379. https://doi.org/10.1016/j.cja.2020.09.036
36. Jiang W, Lyu Y, Li Y et al (2022) UAV path planning and collision avoidance in 3D environments based on POMPD and improved grey wolf optimizer. Aerosp Sci Technol. https://doi.org/10.1016/j.ast.2021.107314