DOI: 10.19884/j.1672-5220.202407004

A Hybrid of RRT* and TD3 Deep Reinforcement Learning Algorithm for UAV Path Planning in 3D Partially Unknown Environments

HE Yanxi[1], QI Jie[1,2*], WU Nailong[1,2]
1. College of Information Science and Technology, Donghua University, Shanghai 201620, China
2. Engineering Research Center of Digitized Textile and Fashion Technology, Ministry of Education, Donghua University, Shanghai 201620, China

Abstract: To guide an unmanned aerial vehicle (UAV) flying in complex three-dimensional (3D) environments with unknown obstacles, a novel UAV path planning algorithm named IRRT*-C2TD3 is proposed. The algorithm combines the rapidly-exploring random tree star (RRT*) algorithm with the twin delayed deep deterministic policy gradients (TD3) algorithm (a deep reinforcement learning algorithm). By employing exploration strategies from reinforcement learning, IRRT*-C2TD3 improves the RRT* algorithm. IRRT*-C2TD3 is a two-stage path planning algorithm comprising pre-planning and real-time planning. It performs pre-planning of paths by generating paths based on geometric connections toward the goal and smoothing them using cubic B-spline curves. By designing the network architecture and reward function of the TD3 algorithm, real-time planning in unknown environments is achieved based on the pre-planned path from the first stage. Simulation results show that IRRT*-C2TD3 demonstrates better path planning performance in 3D partially unknown environments than RRT*-C2TD3, M-C2TD3 and MOD-RRT* algorithms.

Keywords: 3D path planning; deep reinforcement learning; rapidly-exploring random tree (RRT); UAV
CLC number: TP181 **Document code**: A
Article ID: 1672-5220(2025)06-0639-11

Open Science Identity (OSID)

0 Introduction

The advent of intelligent robots has significantly enhanced human life[1]. These machines not only excel in executing repetitive and tedious tasks but also serve as capable substitutes for humans in undertaking perilous missions. An unmanned aerial vehicle (UAV), with its notable advantages of high-altitude accessibility and swift flight speeds, has found extensive applications across various sectors, including military[2], agriculture[3], and ecology[4]. The implementation of high-performance path planning enables the UAV to perform its assigned tasks efficiently. Real-world environments often present unforeseen obstacles, necessitating rapid decision-making and placing heightened demands on the real-time path planning capabilities of UAVs. Thus, research on UAV path planning remains of paramount significance.

Path planning algorithms can be classified into non-learning-based and learning-based algorithms. Non-learning-based algorithms include grid-based algorithms, sampling-based algorithms, and so on. Tang et al.[5] modeled the environment as a grid map. To optimize the paths generated by the conventional A* algorithm[6], they designed a filtering function and incorporated cubic B-spline curves. This approach effectively improved the algorithm's performance by reducing both the turning angle and the path length. Xu et al.[7] improved the global search capability of the ant colony algorithm and combined it with a dynamic window approach for local path planning. Rapidly-exploring random tree star (RRT*)[8] is an optimal variant of the original rapidly-exploring random tree (RRT)[9], ensuring asymptotic optimality by adding steps for parent node selection and rewiring. To improve the convergence speed of RRT*, various studies have gradually focused on optimizing the sampling strategy, leading to improvements in the quality of initial solutions and faster convergence rates[10-12]. Eshtehardian et al.[13] combined B-spline curves with RRT* to smooth the paths generated by RRT*. The above non-learning-based algorithms have relatively simple and transparent principles, stable performance, and predictable results, making them suitable for many standardized application scenarios. However, during actual navigation, unexpected obstacles often arise, requiring constant path re-planning based on environmental information. To achieve dynamic planning, some researchers have proposed improvements to non-learning-based algorithms. Sun et al.[14] introduced a stochastic optimization strategy to handle uncertainties

Received date: 2024-07-16
Foundation items: National Natural Science Foundation of China (No. 62173084); Foundation of Shanghai Committee of Science and Technology, China (Nos. 23ZR1401800 and 22JC1401403)
* Correspondence should be addressed to QI Jie, email: jieqi@dhu.edu.cn
Citation: HE Y X, QI J, WU N L. A hybrid of RRT* and TD3 deep reinforcement learning algorithm for UAV path planning in 3D partially unknown environments [J]. *Journal of Donghua University (English Edition)*, 2025, 42(6): 639-649.

in path planning tasks. Qi et al.[15] proposed a path re-planning algorithm based on Pareto theory, which selects alternative nodes from the stored state tree to achieve obstacle avoidance. Due to the reliance of non-learning-based algorithms on predefined rules and heuristic methods, they still have limitations when handling path planning in complex dynamic environments, such as incomplete state estimation leading to sensor errors and control inaccuracies.

To improve the adaptability of path planning algorithms in complex environments, learning-based algorithms have gradually gained attention[16]. Li et al.[17] used network parameters from a simple environment as prior knowledge for transfer learning, aiming to accelerate convergence in complex environments. Lee et al.[18] combined the soft actor-critic algorithm with hindsight experience replay (HER)[19], effectively enhancing UAV path planning performance and learning speed. Since learning-based algorithms have demonstrated strong adaptability in path planning, some researchers have used learning-based methods to enhance the adaptability of non-learning-based algorithms. The experience from successful planning was used to train models, thereby improving planning efficiency[20-23]. Although these methods do not rely on labeled data, they still require generating a large number of samples for model training.

Therefore, we propose a hybrid algorithm based on RRT* and twin delayed deep deterministic policy gradient (TD3)[24], named IRRT*-C2TD3. This hybrid algorithm uses deep reinforcement learning (DRL), which learns directly through interaction with the environment. Because it does not depend on an environmental model or require labeled datasets, it is particularly effective for solving path planning problems in environments with unknown obstacles. Moreover, DRL-based path planning algorithms can achieve multi-objective optimization by incorporating constraints such as path safety and path length into the algorithm[25-27]. We decompose the entire path planning task into two stages. In the first stage, the pre-planning stage, we improve the conventional RRT* random sampling strategy with an ε-greedy strategy[28] and optimize the initial path planned by the improved RRT* for three-dimensional (3D) environments. In the second stage, the real-time planning stage, based on the pre-planned path from the first stage, we achieve multi-objective optimization for path length and path smoothness through the design of a reasonable network framework and reward functions. This hybrid algorithm leverages the advantages of both non-learning-based algorithms and learning algorithms. Compared to algorithms solely relying on DRL, the transparent principles of the RRT* algorithm reduce the model design complexity of DRL algorithms and accelerate the convergence of DRL. Compared to non-learning-based algorithms, the proposed algorithm demonstrates superior adaptability and real-time planning capabilities. The main contributions of this paper are summarized as follows.

1) We propose a novel path planning algorithm for 3D partially unknown complex environments. The algorithm is composed of an improved RRT* and a DRL-based real-time obstacle avoidance algorithm. It uses the strengths of conventional RRT* and DRL-based path planning approaches simultaneously, and thus improves the performance at convergence speed, adaptability and ability to generalize.

2) This is a two-stage path planning algorithm, comprising pre-planning and real-time planning. In the pre-planning stage, an adaptive sampling approach is implemented by using an ε-greedy strategy with a triangular distribution to accelerate convergence. Geometric optimization guided by the goal direction and cubic B-spline[29] is applied to optimize and smooth the path's inflection points. This process effectively reduces path length and enhances smoothness.

3) We achieve the optimization of path length and smoothness in real-time planning through the design of reasonable reward functions, and the proposed network architecture can efficiently process high-dimensional image data during navigation. Simulations demonstrate that the proposed algorithm can be applied to different environments, and the planned paths exhibit promising performance.

This paper is organized as follows. Section 1 describes the application environment model of the algorithm and the indicators for evaluating algorithm performance. Section 2 elaborates on the specific improvements to the RRT* algorithm and the state space, action space, reward function, and network framework of the DRL algorithm. Section 3 describes the implementation details of training and testing, and analyzes the simulation results. Finally, Section 4 summarizes the main findings and suggests future work.

1 Problem Description

This section presents the 3D partially unknown environmental model and evaluation indicators for the proposed path planning algorithm. The environment consists of free space and cuboid obstacles of varying dimensions. Known variables include the UAV's start, current, and goal positions, as well as certain obstacles' dimensions and centers of mass. Some obstacles remain unknown until detected by the UAV's onboard sensors during flight. The UAV model features two side-mounted laser ranging sensors and a front-facing depth camera with a 64×32 resolution. The laser ranging sensors measure the minimum distance to obstacles, while the depth camera provides per-pixel depth and color information. The environmental model and UAV model applied are illustrated in Fig. 1.

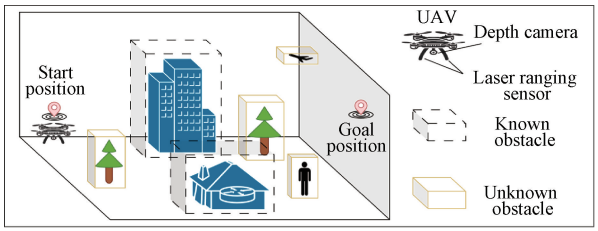


Fig. 1 Model of UAV and 3D partially unknown environment

The path planning in this paper is to find a feasible path for a UAV from a start position to a goal position. The proposed algorithm includes two stages: the improved RRT* pre-planning and C2TD3-based real-time planning. We introduce three evaluation indicators for these stages: 1) path length; 2) path smoothness; 3) algorithm success rate. These indicators measure the algorithm's effectiveness, with path length and smoothness also serving as optimization indicators. In Fig. 2, taking the path connected by a black arrow as an example, the path consists of three waypoints (p_{start}, p_1, and p_{goal}).

The defined indicators are as follows.

Indicator 1: path length L_{P}

It is represented by the sum of the Euclidean distances between the waypoints (p_{start}, p_1, and p_{goal}).

$$L_{\text{P}} = \sqrt{\sum_{k=x,y,z} (k_{\text{start}} - k_1)^2} + \sqrt{\sum_{k=x,y,z} (k_1 - k_{\text{goal}})^2}. \quad (1)$$

Indicator 2: horizontal turning angle sum (HTAS) S_{HTA}

HTAS reflects the total deviation from the previous path at each turn. As shown in Fig. 2, the angle formed between the solid red line and the dashed red line represents the horizontal turning angle. ϕ_1 and ϕ_2 are the horizontal turning angles from p_{start} to p_1 and from p_1 to p_{goal}, respectively. Therefore, HTAS is expressed as

$$S_{\text{HTA}} = \phi_1 + \phi_2. \quad (2)$$

Indicator 3: climbing angle sum (CAS) S_{CA}

CAS reflects the total angle deviation from the previous path at each ascent. As shown in Fig. 2, the angle between the solid yellow line and the path represents the climbing angle. θ_1 and θ_2 are the climbing angles from p_{start} to p_1 and from p_1 to p_{goal}, respectively. Therefore, CAS is expressed as

$$S_{\text{CA}} = \theta_1 + \theta_2. \quad (3)$$

We address path smoothness quantitatively through HTAS and CAS.

Indicator 4: pre-planning success rate (PR) R_{P}

We perform a total of N two-stage path planning processes, each including pre-planning and real-time planning. Among them, m pre-planning attempts are defined as those situations in which a path is successfully found within the given number of iterations, and q real-time planning attempts are defined as those situations in which a collision-free path is generated during real-time navigation. The fourth indicator R_{P} is defined as

$$R_{\text{P}} = m/N \times 100\%. \quad (4)$$

Indicator 5: real-time planning success rate (RR) R_{R}

$$R_{\text{R}} = q/m \times 100\%. \quad (5)$$

Indicator 6: both stages of planning success rate (OR) R_{O}

$$R_{\text{O}} = q/N \times 100\%. \quad (6)$$

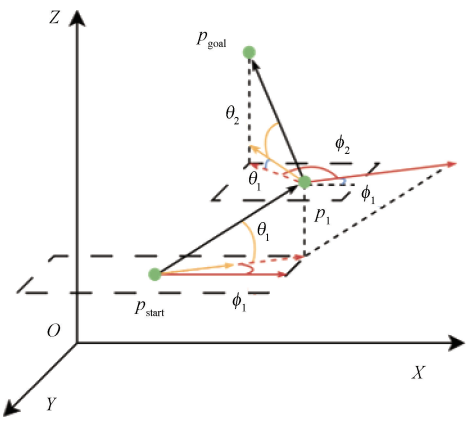


Fig. 2 Illustration of performance indicators

2 Two-Stage Path Planning

This section first presents the algorithm framework, and then introduces the improved RRT* (IRRT*) and C2TD3-based path planning algorithms. The proposed IRRT*-C2TD3 algorithm framework is illustrated in Fig. 3. Based on the known environmental information, the IRRT* pre-plans an initial path. When the UAV navigates this path and detects obstacles within a safe distance, the C2TD3 algorithm performs real-time planning based on the data from the onboard sensors and the key waypoints in the pre-planned path.

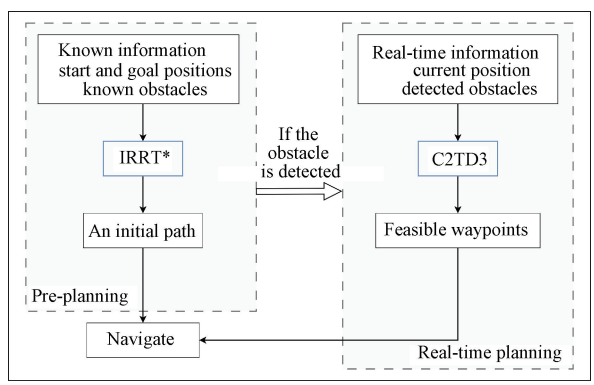


Fig. 3 Framework of IRRT*-C2TD3

2.1 IRRT*-based pre-path planning

To address the impact of random sampling in RRT* on path length and path smoothness, we made two improvements. The first step generates an initial path based on the ε-greedy strategy, and the second step optimizes the path generated in the first step. The complete IRRT* algorithm is shown in Algorithm 1. In the algorithm, n represents the number of iterations, which determines how many times the planning loop executes to explore the path; G represents the graph modeling the entire network or environment; V denotes the set of vertices, each indicating a specific location or node; E signifies the set of edges, defining the connections or paths between vertices; Ø represents the empty set, meaning that the set of edges E is initialized with no elements. The primitive procedures involved in IRRT* are briefly explained as follows.

Triangular (a,b,c): generating a random value from a triangular distribution with the parameters (a,b,c), where ε represents the threshold parameter for the ε-greedy strategy.

Rand (a,b): generating a random value $\tilde{r}$ uniformly distributed in the interval (a,b).

Random Node: generating a new random node p_{rand} from the search space.

Nearest (G,p): finding the nearest node p_s to p in the graph G.

Steer (p_s,p): returning the line segment connecting p_s to p.

ObstacleFree (σ): checking if the generated trajectory σ is collision-free, ensuring that the path does not intersect with any obstacles.

Near (G,p_s): finding a set of nodes P_{near} within a certain radius of p_s in the graph G.

ChooseParent $(P_{\text{near}}, p_s, \sigma)$: choosing an appropriate parent node p_{parent} from the set P_{near}, based on minimizing the path length.

Rewire (G,P_{near},p_s): if the path from p_s to any element of $P_{\text{near}}, P_{\text{near}}$ is collision-free, and the path is shorter than the current path, then the parent of P_{near} is replaced by p_s.

Path Optimization(G): optimizing the existing path in the graph to generate P_{new}.

Algorithm 1 IRRT*

Initialize $V \leftarrow p_{\text{start}}, E \leftarrow \emptyset, G = (V, E)$
for $i = 1,2,...,n$ **do**
  $\varepsilon \leftarrow \text{Triangular}(a,b,c)$
  $\tilde{r} \leftarrow \text{Rand(a,b)}$
  if $\tilde{r} < \varepsilon$ **then**
    $p_{\text{rand}} \leftarrow$ Random Node
    $p_s \leftarrow \text{Nearest}(G, p_{\text{rand}})$
    $\sigma \leftarrow \text{Steer}(p_s, p_{\text{rand}})$
  else
    $p_s \leftarrow \text{Nearest}(G, p_{\text{goal}})$
    $\sigma \leftarrow \text{Steer}(p_s, p_{\text{goal}})$
  end if
  if ObstacleFree(σ) **then**
    $P_{\text{near}} \leftarrow \text{Near}(G, p_s)$
    $p_{\text{parent}} \leftarrow \text{ChooseParent}(P_{\text{near}}, p_s, \sigma)$
    $V \leftarrow V \cup \{p_s\}$
    $E \leftarrow E \cup (p_{\text{parent}}, p_s)$
    $G \leftarrow \text{Rewire}(G, P_{\text{near}}, p_s)$
  end if
end for
$P_{\text{new}} = \text{Path Optimization}(G)$
Return P_{new}

2.1.1 Growth strategy based on the ε-greedy strategy

The ε-greedy strategy balances exploration and exploitation by performing greedy selection with a probability of $1-\varepsilon$ and random selection with a probability of ε. To improve the RRT* sampling strategy, we define ε using a triangular distribution, commonly used in simulations lacking detailed data[30]. We define environment complexity o_n as the ratio of the volume of all known obstacles to the entire search area. Thus, we use a triangular distribution with a minimum value $A = 0$, a maximum value $B = 1$, and a mode $C = o_n$. This allows for dynamic adjustment of ε. In simple environments, a smaller ε increases the probability of growing toward the goal. In complex environments, a larger ε enhances randomness and ensures convergence.

2.1.2 Path optimization

This optimization step addresses redundant waypoints from the previous algorithm's randomness. Due to the higher degree of freedom in a 3D environment, we propose an optimization based on geometric connections toward the goal direction. Starting from both the start and goal points, the waypoints are directly connected where possible, replacing the initial path. This process is repeated iteratively until all waypoints are optimized. Figure 4 illustrates the local effect of path optimization.

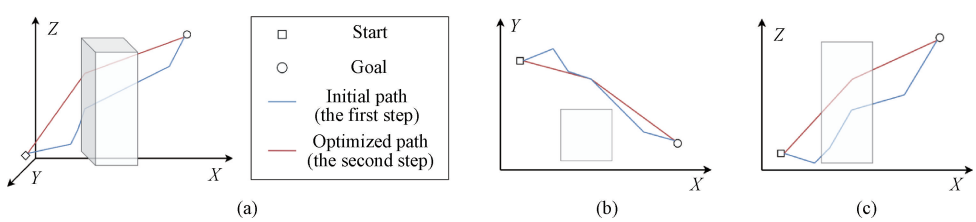


Fig. 4 Local effect of path optimization: (a) 3D comparative result; (b) top view; (c) side view

Although the path has been optimized, the unconstrained sampling method results in noticeable sharp turns, making it difficult to directly apply to a UAV. Therefore, a cubic B-spline curve is used to smooth the path, ensuring the final path is feasible for the flight path.

A B-spline is defined by a set of basis functions and control points. Assuming there are $n+1$ control points p_i, the cubic B-spline curve can be expressed as

$$C(u) = \sum_{i=0}^{n} p_i N_{i,3}(u), \tag{7}$$

where $C(u)$ is the point on the cubic B-spline curve corresponding to the parameter value u; $N_{i,3}(u)$ is the cubic B-spline basis function, which is defined by a recursive formula. The basis function is defined as

$$N_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \leqslant u < u_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

$$N_{i,j}(u) = \frac{u - u_i}{u_{i+j} - u_i} N_{i,j-1}(u) + \frac{u_{i+j+1} - u}{u_{i+j+1} - u_{i+1}} N_{i+1,j-1}(u). \tag{9}$$

The parameter interval set of the cubic B-spline is defined by the knot vector $\boldsymbol{u} = [u_0, u_1, \cdots, u_w]$, where $w = n + j + 1$. For a cubic B-spline, $j = 3$. Figure 5 shows the path before and after cubic B-spline smoothing.

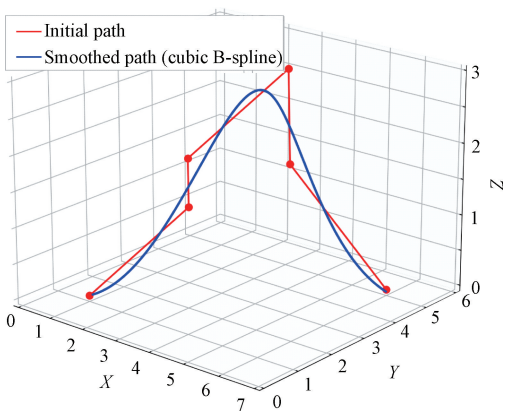


Fig. 5 Comparison of initial and cubic B-spline smoothed paths

2.2 TD3-based real-time path planning

When the UAV detects obstacles along the pre-planned path, the designed C2TD3 algorithm makes real-time decisions based on the current environmental information to plan a new path.

2.2.1 State space

The state space includes the information that the UAV obtains from the environment, helping it understand the surroundings and task requirements accurately. The UAV executes actions based on the acquired state information. If the UAV is moving toward the sub-goal point p_c in the pre-planned path, the states $\boldsymbol{s}_1$ related to the UAV's position information are defined as

$$\boldsymbol{s}_1 = [k_{\text{goal}} - k_c, k_{\text{goal}} - k, v_x, v_y, v_z]^{\mathrm{T}}, \quad k = x, y, z, \tag{10}$$

where $(x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})$ represents the coordinates of the goal; (x_c, y_c, z_c) represents the coordinates of point p_c, obtained from the pre-planned path; (x, y, z) represents the current position's coordinates obtained in real-time; (v_x, v_y, v_z) represents the current UAV's velocity along the three coordinate axes.

The UAV is equipped with laser ranging sensors on its left and right sides. These sensors measure the minimum distance to detected objects, contributing to the UAV's state space $\boldsymbol{s}_p$.

$$\boldsymbol{s}_p = [d_{\text{left}}, d_{\text{right}}]^{\mathrm{T}}, \tag{11}$$

where d_{left} and d_{right} are the minimum distances to detected obstacles on the UAV's left and right sides, respectively.

The depth camera at the front provides depth distance information $\boldsymbol{s}_d$ and color information $\boldsymbol{s}_c$ of detected objects, which are also included in the UAV's state space.

Furthermore, in order to accelerate the convergence of the algorithm and simplify the learning process, we normalize $\boldsymbol{s}_1$ and $\boldsymbol{s}_p$ as follows:

$$\boldsymbol{s}_1 = [k_{\text{goal}} - k_c, k_{\text{goal}} - k]^{\mathrm{T}} / k_{\text{map}}, \quad k = x, y, z, \tag{12}$$

$$\boldsymbol{s}_p = [d_{\text{left}}, d_{\text{right}}]^{\mathrm{T}} / d_{\max}, \tag{13}$$

where k_{map} is the maximum length along the k-axis in the given map; $d_{\max}$ is the maximum detection distance of the laser ranging sensor. Therefore, the complete definition of the UAV state space $\boldsymbol{s}$ is

$$\boldsymbol{s} = [\boldsymbol{s}_1^{\mathrm{T}}, \boldsymbol{s}_p^{\mathrm{T}}, \boldsymbol{s}_c, \boldsymbol{s}_d]^{\mathrm{T}}. \tag{14}$$

2.2.2 Action space

We aim to make the designed algorithm applicable to UAVs of any model, thus minimizing the reliance on specific details such as the type of UAV model and controller. The action space $\boldsymbol{a}$ is defined as the displacement along the coordinate axes:

$$\boldsymbol{a} = [a_x, a_y, a_z]^{\mathrm{T}}. \tag{15}$$

Thus, the coordinate information of the new sub-goal point p_{cn} corresponds to the current position's coordinates plus the displacement along the corresponding coordinate axes:

$$(x_{\text{cn}}, y_{\text{cn}}, z_{\text{cn}}) = (x + a_x, y + a_y, z + a_z). \tag{16}$$

2.2.3 Reward function

In DRL, an agent's actions are evaluated based on a reward function, which guides the agent in learning optimal strategies. The design of the reward function directly influences the agent's behavior and learning outcomes. The aims of real-time path planning are to ensure collision-free navigation, reduce path length, and improve path smoothness. Therefore, we design a reward function r consisting of four components: distance reward from the goal r_g, smoothness reward r_s, obstacle penalty r_o, and task completion reward r_c.

$$r = r_g + r_s + r_o + r_c. \tag{17}$$

To reduce path length, each planned path is expected to be closer to the goal. Therefore, r_g is defined as

$$r_g = -5d_i, \tag{18}$$

where d_i is the Euclidean distance between the current and the goal positions. We constrain the smoothness of the path based on the position states of the UAV at two points in time t and time t+1. Therefore, r_s is defined as

$$r_s = \sum_{k=x,y,z} (k_{goal} - k)^t - (k_{goal} - k)^{t+1}. \tag{19}$$

To ensure collision-free paths, a minor penalty is applied if the depth camera or laser ranging sensor detects an obstacle within the safe distance. If a collision occurs, a larger penalty is applied. Therefore, r_o is defined as

$$r_o = \begin{cases} -0.05, & \text{if detected in the safe distance,} \\ -5, & \text{if colliding,} \\ 0, & \text{else.} \end{cases} \tag{20}$$

The decision task is deemed complete when, after performing an action, the UAV is closer to the goal compared to its original sub-goal point p_c on the pre-planned path. Thus, r_c is defined as

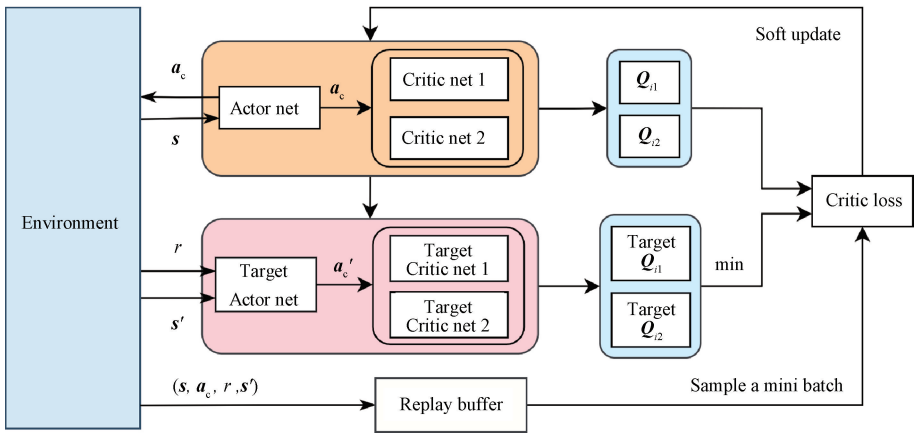
$$r_c = \begin{cases} 2, & \text{if } d(p_{cn}, p_{goal}) < d(p_c, p_{goal}), \\ 0, & \text{else.} \end{cases} \tag{21}$$

2.2.4 Network structure

The TD3 algorithm is a classic DRL algorithm, consisting of six networks: an actor network and its corresponding target actor network; two critic networks and their corresponding two target critic networks. The definition of the target value $\boldsymbol{Q}_{tar}$ is

$$\boldsymbol{Q}_{tar} = r + \gamma \min_{i=1,2} \boldsymbol{Q}'_i(\boldsymbol{s}', \boldsymbol{a}'_c), \tag{22}$$

where $\boldsymbol{s}'$ represents the state at the next time step; $\boldsymbol{a}'_c$ represents the action at the next time step; $\boldsymbol{Q}'$ is the output of the critic network; γ is a constant in $[0, 1]$. Since the actor network is updated by maximizing the cumulative expected return, any of the critic networks can be chosen to update the actor network. Additionally, the stability of the algorithm is improved by delaying the updates of the actor and critic networks. The network update process of TD3 is shown in Fig. 6.



$\boldsymbol{a}_c$—the action at the current time step.

Fig. 6 Network update process of TD3

The dimensions of $\boldsymbol{s}_d$ and $\boldsymbol{s}_c$ obtained from the environment are 1×64×32 and 3×64×32, respectively. Directly using them as states makes convergence difficult. Therefore, we propose two convolutional neural networks (CNNs) within the actor-critic network as the state feature extractors. $\boldsymbol{s}_d$ and $\boldsymbol{s}_c$ are each processed through a CNN for dimensionality reduction, and then concatenated with other states in the state space as inputs to the actor-critic network. The network structure of our algorithm is designed as Fig. 7. In Fig. 7, Conv represents convolution; ReLU represents rectified linear unit; MaxPool represents max pooling; FC represents fully connected layer.

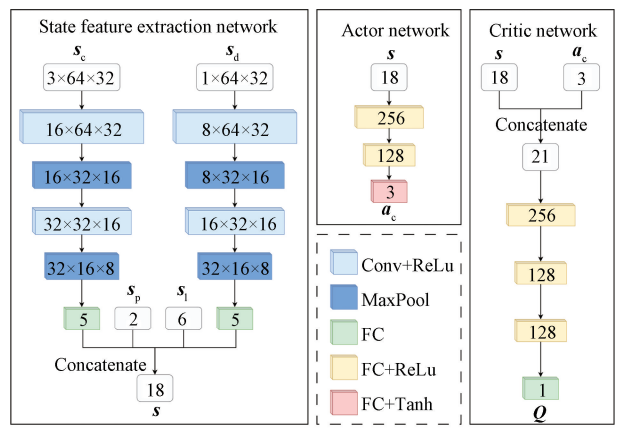


Fig. 7 Network structure of our algorithm

3 Experiments and Results

This section outlines two simulation experiments to evaluate the proposed algorithm. The first simulation validates the advantages of the ε-greedy strategy based on a triangular distribution in terms of environmental adaptability, and the second simulation demonstrates the training environment and test results of IRRT*-C2TD3. We implemented all algorithms on an i9-7900X Intel Core processor and NVIDIA TITAN Xp graphics processing unit.

3.1 Impact analysis of ε-greedy strategy

The ε-greedy strategy based on a triangular distribution in the pre-planned path is validated by a simulation experiment. The ε-greedy strategy in our approach is termed the adaptive ε-greedy strategy. To highlight the advantage of the adaptive ε-greedy strategy in terms of environmental adaptability, we compare it with two other strategies: one that uses a fixed ε value (ε_0) unrelated to environmental complexity, and a static path planning algorithm from MOD-RRT*[17]. The comparison is conducted in two scenarios (E1 and E2), both using a map size of 7.0 m × 7.0 m × 4.0 m. In E1 (low-complexity), $o_n \in [0.2, 0.3]$, and $\varepsilon_0 \in [0.6, 0.7]$; in E2 (high-complexity), $o_n \in [0.6, 0.7]$, and $\varepsilon_0 \in [0.2, 0.3]$. According to the definition in Section 2, a larger o_n indicates a more complex environment. Therefore, we select ε_0 values that are negatively correlated with o_n to demonstrate the advantages of the adaptive ε-greedy strategy in different environmental complexities. To further emphasize the convergence advantage of our adaptive ε-greedy strategy, we use iteration count (the number of iterations to convergence) as an evaluation indicator.

The results in Table 1 demonstrate the advantages of the adaptive ε-greedy strategy in both E1 and E2 environments. In E1, the adaptive ε-greedy strategy can avoid many cases of inefficient sampling, achieving the shortest average path length. Additionally, its mean iteration count is only 79, much fewer than that of the ε_0-strategy (112) and MOD-RRT* (125), indicating that our algorithm can find an initial path that is shorter and requires fewer iterations in less complex environments. In E2, the small ε_0 value results in a significantly higher iteration count because more randomness is needed to explore feasible paths in complex environments. Although the adaptive ε-strategy does not achieve the shortest path length in this case, it still maintains an advantage in terms of iteration count. In general, the adaptive ε-greedy strategy shows strong adaptability to varying environmental complexities, flexibly adjusting the balance between exploration and exploitation based on the complexity of the environment.

Table 1 Performance comparison of adaptive ε-greedy strategy

Environment	Algorithm	Length/m			Iteration count		
		Mean	Minimum	Maximum	Mean	Minimum	Maximum
E1	ε_0-strategy	11.9	10.0	15.6	112	56	153
	MOD-RRT*	12.6	9.8	16.4	125	68	179
	Adaptive ε-strategy	11.5	10.2	14.9	79	47	96
E2	ε_0-strategy	12.3	10.9	16.1	258	167	675
	MOD-RRT*	13.1	11.4	17.5	187	112	403
	Adaptive ε-strategy	12.8	10.3	16.8	143	97	368

3.2 Performance evaluation in partially unknown environments

3.2.1 Training settings

PyBullet is adopted as the simulation platform to minimize the differences between the simulation and real environments. As a physics engine, PyBullet offers accurate dynamics and collision detection, making it suitable for robotics and DRL applications.

The training environment is illustrated in Fig. 8, with a map size of 12.5 m×12.5 m×4.0 m. The start is randomly generated within a cuboid-shaped region with diagonally opposite corners at (−6.0, 6.0, 0.5) and (−5.0, −6.0, 3.5). Similarly, the goal is randomly generated within a cuboid-shaped region with diagonally opposite corners at (6.0, −6.0, 0.5) and (5.0, 6.0, 3.5). Gray cuboids represent known obstacles during the generation of the pre-planned path, and cuboids of other colors represent unknown obstacles. The two large yellow cuboids located at each end of the map represent the generation areas for the start and goal, respectively. The training parameters are detailed in Table 2.

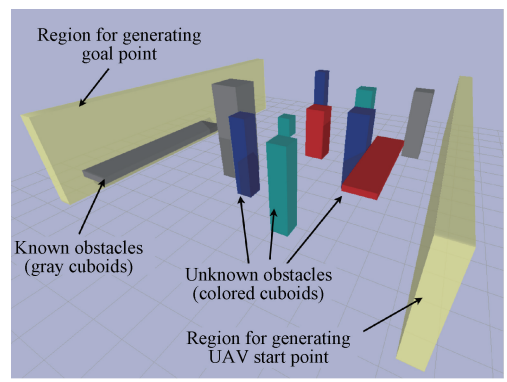


Fig. 8 Training environment of C2TD3

Table 2 Parameters of training experiment

Parameter	Value
Max episode	6 000
Max step	100
Max iteration	20
Batch size	512
Soft update factor	0.005
Actor learning rate	0.001
Critic learning rate	0.000 5
Weight decay	0.000 5
Discount factor	0.99
Reply buffer size	10^5
Target network update frequency	4
Safe distance/m	0.5

In each episode of the training process, a path is pre-planned by using IRRT*. If an obstacle is detected during navigation, C2TD3 is used for decision-making. If training starts at point p and the maximum step limit is reached before meeting the completion condition, it is considered the end of one iteration, and the training restarts at point p for a new iteration. The episode concludes if the maximum iteration limit is reached or the UAV successfully reaches the goal. Figure 9 illustrates the average reward curve during the training process.

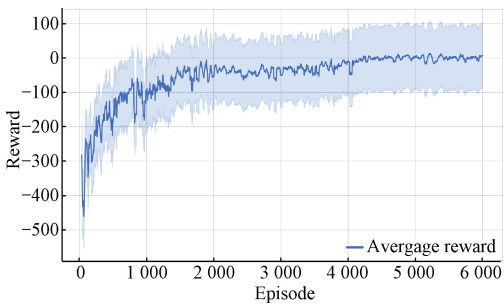


Fig. 9 Convergence curve of average reward

3.2.2 Testing and results

During the testing process, three environments (S1, S2, and S3) are set up. S1 has a map size of 12.5 m×12.5 m×4.0 m, with five known obstacles and 12 unknown obstacles. S2 has a map size of 20.0 m×20.0 m×4.0 m with 10 known obstacles and 25 unknown obstacles. S3 has a map size of 20.0 m×20.0 m×4.0 m with 15 known obstacles and 35 unknown obstacles.

In these three environments, we compare our IRRT*-C2TD3 algorithm with three other algorithms: RRT*-C2TD3, M-C2TD3, and MOD-RRT*. In the testing experiments, while the definition of the length in MOD-RRT* remains unchanged, the angle is revised to HTAS and CAS. All algorithms adopt a two-stage planning process, and the types of algorithms used in each stage are shown in Table 3. Additionally, to better compare algorithm performance, both MOD-RRT* and M-C2TD3 use the same pre-planned path in each test, and MOD-RRT* employs cubic B-spline interpolation to process dynamically generated path points. Each environment is tested 500 times, with start and goal positions randomly generated at opposite ends of the map. PR, RR, OR, path length, HTAS, and CAS serve as performance indicators for testing. The latter three indicators are data obtained from the successful completion of the two-stage path planning task.

Table 3 Types of algorithms used in two-stage path planning

Test algorithm	Pre-planning algorithm	Real-time planning algorithm
RRT*-C2TD3	RRT*	C2TD3
MOD-RRT*	MOD-RRT*	MOD-RRT*
M-C2TD3	MOD-RRT*	C2TD3
IRRT*-C2TD3	IRRT*	C2TD3

Figure 10 shows typical successful planning examples of four algorithms in three distinct scenarios. In our designed two-stage planning process, real-time planning is based on pre-planned waypoints. RRT*-C2TD3 (yellow line) uses the conventional RRT* for pre-planning, resulting in a final path with numerous turns. Both IRRT*-C2TD3 (red line) and M-C2TD3 (green line) use C2TD3 for real-time planning.

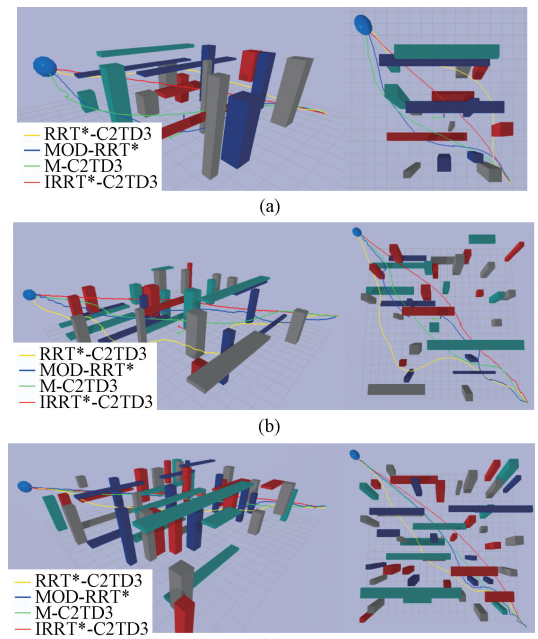


Fig. 10 Path planning results with corresponding top views (on the right) of typical examples in three scenarios: (a) S1; (b) S2; (c) S3

The differences in their paths are mainly due to the pre-planned paths. Compared to M-C2TD3, the paths planned by IRRT*-C2TD3 are superior in terms of length and smoothness, validating the effectiveness of our IRRT*. Additionally, a comparison between M-C2TD3 (green line) and MOD-RRT* (blue line) shows that C2TD3's decisions are smoother with no significant fluctuations, whether avoiding horizontal or vertical obstacles. In contrast, MOD-

RRT*'s decisions rely on the state tree from the pre-planned path, leading to some fluctuations in waypoints. Therefore, IRRT*-C2TD3 demonstrates better path planning capabilities in 3D environments.

Table 4 shows that while the OR of all algorithms decreases with increasing environmental complexity, our proposed IRRT*-C2TD3 consistently outperforms the comparison algorithms across all three scenarios. In the most complex case (S3), it achieves the highest RR (82.8%) and the highest OR (75.0%). This demonstrates its effective performance and better adaptability in complex environments.

Table 4 Comparison of test rates in different scenarios

Algorithm	PR/%			RR/%			OR/%		
	S1	S2	S3	S1	S2	S3	S1	S2	S3
RRT*-C2TD3	96.2	81.8	78.2	74.0	73.1	67.2	71.2	59.8	52.7
MOD-RRT*	96.8	88.6	85.4	81.2	79.2	73.3	78.6	70.2	62.6
M-C2TD3	96.8	88.6	85.4	88.6	84.8	79.4	85.8	75.2	67.8
IRRT*-C2TD3	**98.4**	**95.2**	**90.6**	**93.2**	**88.4**	**82.8**	**91.8**	**84.3**	**75.0**

Figure 11 shows the boxplots of the performance indicators for the four algorithms. It can be observed that compared with other algorithms, the paths planned by IRRT*-C2TD3 are shorter and smoother across environments of varying complexity, especially in more complex scenarios.

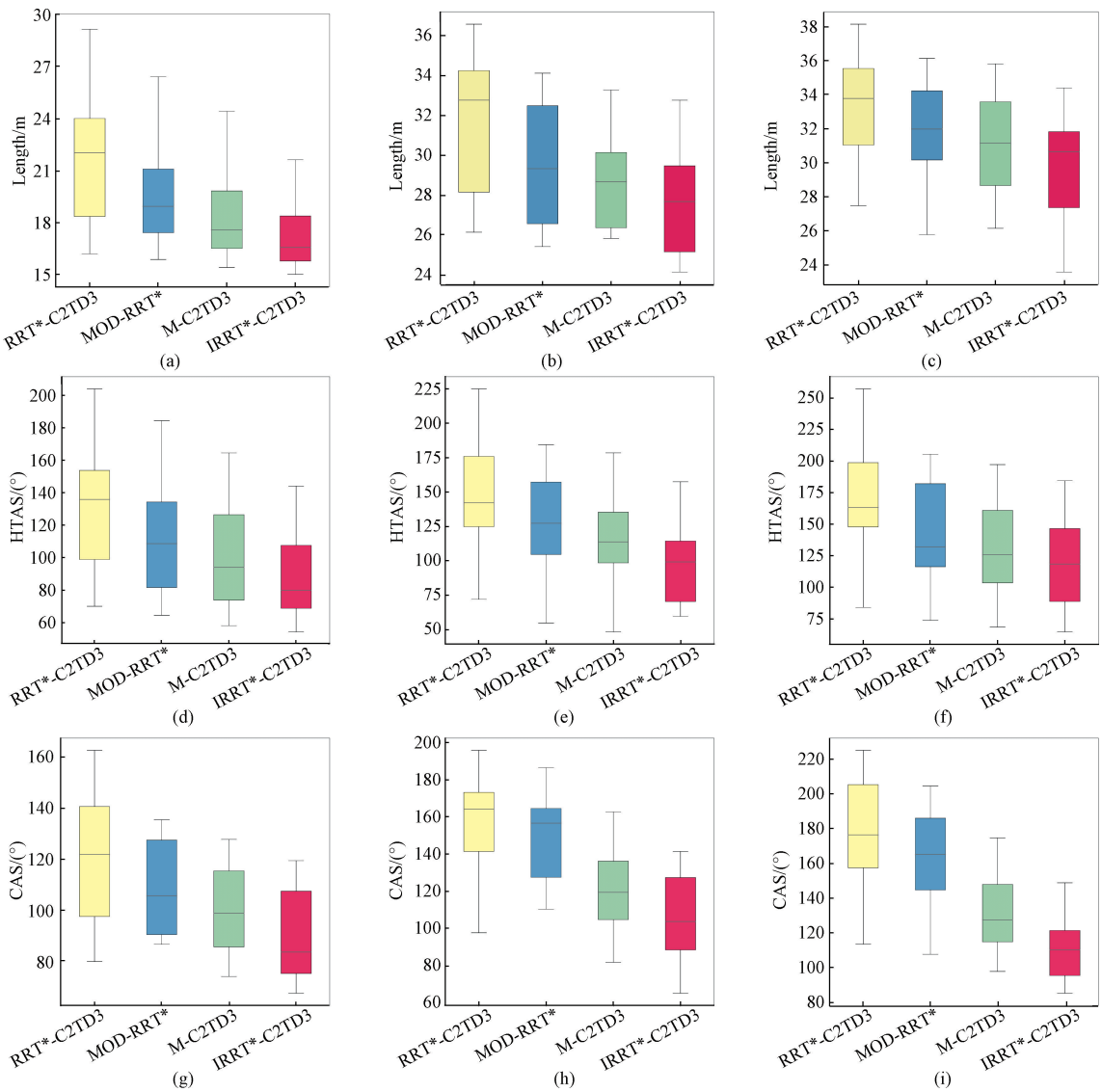


Fig. 11 Boxplots of performance indicators for four algorithms across three scenarios: (a) length indicators in S1; (b) length indicators in S2; (c) length indicators in S3; (d) HTAS indicators in S1; (e) HTAS indicators in S2; (f) HTAS indicators in S3; (g) CAS indicators in S1; (h) CAS indicators in S2; (i) CAS indicators in S3

4 Conclusions

This paper proposes a UAV path planning algorithm named IRRT*-C2TD3. It combines IRRT* with DRL to achieve UAV path planning in 3D partially unknown environments. The algorithm consists of two stages: pre-planning based on IRRT* and real-time planning based on C2TD3. The initial path provided by IRRT* and the ability of C2TD3 to extract high-dimensional environmental feature information jointly guide the real-time path planning of the UAV in complex environments. The test results show that in 3D partially unknown environments, the path length, smoothness, completion time, and success rate planned by IRRT*-C2TD3 are all superior to those of the comparison algorithms. Future research will focus on the environments that include moving obstacles, necessitating improvements in the network structure to adapt to such dynamic conditions. Additionally, the development of safe motion control for UAVs will be a consideration in the path planning process.

References

[1] LIU Z H, LIU Q, XU W J, et al. Robot learning towards smart robotic manufacturing: a review[J]. *Robotics and Computer-Integrated Manufacturing*, 2022, 77: 102360.

[2] HU Z J, GAO X G, WAN K F, et al. Relevant experience learning: a deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments[J]. *Chinese Journal of Aeronautics*, 2021, 34(12): 187-204.

[3] SENTHILNATH J, KANDUKURI M, DOKANIA A, et al. Application of UAV imaging platform for vegetation analysis based on spectral-spatial methods [J]. *Computers and Electronics in Agriculture*, 2017, 140: 8-24.

[4] SHAHSAVANI H. An aeromagnetic survey carried out using a rotary-wing UAV equipped with a low-cost magneto-inductive sensor[J]. *International Journal of Remote Sensing*, 2021, 42(23): 8805-8818.

[5] TANG G, TANG C Q, CLARAMUNT C, et al. Geometric A-star algorithm: an improved A-star algorithm for AGV path planning in a port environment[J]. *IEEE Access*, 2021, 9: 59196-59210.

[6] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths[J]. *IEEE Transactions on Systems Science and Cybernetics*, 1968, 4(2): 100-107.

[7] XU H Q, XING H X, LIU Y. Path planning of UAV by combining improved ant colony system and dynamic window algorithm[J]. *Journal of Donghua University (English Edition)*, 2023, 40(6): 676-683.

[8] KARAMAN S, FRAZZOLI E. Sampling-based algorithms for optimal motion planning[J]. *The International Journal of Robotics Research*, 2011, 30(7): 846-894.

[9] LAVALLE S M, KUFFNER J J Jr. Randomized kinodynamic planning[J]. *International Journal of Robotics Research*, 2001, 20(5): 378-400.

[10] GAMMELL J D, SRINIVASA S S, BARFOOT T D. Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic[C]//2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. New York: IEEE, 2014: 2997-3004.

[11] LI Y J, WEI W, GAO Y, et al. PQ-RRT*: an improved path planning algorithm for mobile robots[J]. *Expert Systems with Applications*, 2020, 152: 113425.

[12] WANG J K, LI T G, LI B P, et al. GMR-RRT*: sampling-based path planning using Gaussian mixture regression [J]. *IEEE Transactions on Intelligent Vehicles*, 2022, 7(3): 690-700.

[13] ESHTEHARDIAN S A, KHODAYGAN S. A continuous RRT*-based path planning method for non-holonomic mobile robots using B-spline curves[J]. *Journal of Ambient Intelligence and Humanized Computing*, 2023, 14(7): 8693-8702.

[14] SUN Z Y, SHEN B, PAN A Q, et al. A modified self-adaptive sparrow search algorithm for robust multi-UAV path planning[J]. *Journal of Donghua University (English Edition)*, 2024, 41(6): 630-643.

[15] QI J, YANG H, SUN H X. MOD-RRT*: a sampling-based algorithm for robot path planning in dynamic environment[J]. *IEEE Transactions on Industrial Electronics*, 2021, 68(8): 7244-7251.

[16] VASHISTH A, RÜCKIN J, MAGISTRI F, et al. Deep reinforcement learning with dynamic graphs for adaptive informative path planning [J]. *IEEE Robotics and Automation Letters*, 2024, 9(9): 7747-7754.

[17] LI W J, YUE M, SHANGGUAN J Y, et al. Navigation of mobile robots based on deep reinforcement learning: reward function optimization and knowledge transfer [J]. *International Journal of Control, Automation and Systems*, 2023, 21(2): 563-574.

[18] LEE M H, MOON J. Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: a soft actor-critic with hindsight experience replay approach[J].

ICT Express, 2023, 9(3): 403-408.
[19] ANDRYCHOWICZ M, WOLSKI F, RAY A, et al. Hindsight experience replay [EB/OL]. (2018-02-23)[2024-07-20]. https://arxiv.org/abs/1707.01495.
[20] WANG J K, CHI W Z, LI C M, et al. Neural RRT*: learning-based optimal path planning[J]. *IEEE Transactions on Automation Science and Engineering*, 2020, 17(4): 1748-1758.
[21] WANG J K, JIA X, ZHANG T Y, et al. Deep neural network enhanced sampling-based path planning in 3D space[J]. *IEEE Transactions on Automation Science and Engineering*, 2022, 19(4): 3434-3443.
[22] URAIN J, LE A T, LAMBERT A, et al. Learning implicit priors for motion optimization [C]//2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). New York: IEEE, 2022: 7672-7679.
[23] LIU B L, JIANG G D, ZHAO F, et al. Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot[J]. *IEEE Robotics and Automation Letters*, 2023, 8(11): 7082-7089.
[24] FUJIMOTO S, HOOF H, MEGER D. Addressing function approximation error in actor-critic methods [C]//International conference on machine learning. [S.l.]: PMLR, 2018: 1587-1596.
[25] MIENYE I D, SUN Y X. A survey of ensemble learning: concepts, algorithms, applications, and prospects[J]. *IEEE Access*, 2022, 10: 99129-99149.
[26] YANG C P, ZHAO Y Q, CAI X, et al. Path planning algorithm for unmanned surface vessel based on multiobjective reinforcement learning [J]. *Computational Intelligence and Neuroscience*, 2023, 2023(1): 2146314.
[27] HUANG S Q, WU X R, HUANG G M. Deep reinforcement learning-based multi-objective 3D path planning for vehicles[C]//Proceedings of 2023 Chinese Intelligent Systems Conference. Singapore: Springer, 2023: 867-875.
[28] LIU X F, ZHANG P, FANG H, et al. Multi-objective reactive power optimization based on improved particle swarm optimization with ε-greedy strategy and Pareto archive algorithm[J]. *IEEE Access*, 2021, 9: 65650-65659.
[29] QU C Z, GAI W D, ZHONG M Y, et al. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning [J]. *Applied Soft Computing*, 2020, 89: 106099.
[30] JOHNSON D. The triangular distribution as a proxy for the beta distribution in risk analysis [J]. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 1997, 46(3): 387-398.

融合 RRT* 与 TD3 深度强化学习的无人机三维部分未知环境路径规划算法

何彦熹[1]，齐　洁[1,2*]，吴乃龙[1,2]

1. 东华大学 信息科学与技术学院，上海 201620
2. 东华大学 数字化纺织服装技术教育部工程研究中心，上海 201620

摘　要：为完成无人机在含有未知障碍物的三维环境中的导航任务，提出了一种无人机路径规划算法，IRRT*-C2TD3。该算法结合了快速探索随机树星（RRT*）算法和双延迟深度确定性策略梯度深度强化学习（TD3）算法。利用强化学习中的探索策略，IRRT*-C2TD3 改进了 RRT* 算法。IRRT*-C2TD3 的路径规划包含预规划与实时规划两个阶段。它先基于指向目标的几何连接生成路径，然后采用三次 B 样条曲线进行平滑处理，从而实现路径的预规划；通过设计 TD3 算法的网络架构和奖励函数，结合第一阶段的预规划路径，实现在未知环境中的实时规划。仿真结果表明，IRRT*-C2TD3 与 RRT*-C2TD3、M-C2TD3 和 MOD-RRT* 算法相比，在三维部分未知环境中的路径规划性能更佳。

关键词：三维路径规划；深度强化学习；快速探索随机树；无人机