IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 January 2024, accepted 4 March 2024, date of publication 7 March 2024, date of current version 15 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3375083

RESEARCH ARTICLE

UAV Path Planning Based on the Average TD3 Algorithm With Prioritized Experience Replay

XUQIONG LUO[1], QIYUAN WANG[1], HONGFANG GONG[1], AND CHAO TANG[2]
[1]School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410114, China
[2]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China

Corresponding author: Hongfang Gong (ghongfang@126.com)

This work was supported in part by the Excellent Youth Project of Education Department of Hunan Province under Grant 21B0313, in part by the National Natural Science Foundation of China under Grant 61972055, and in part by Hunan Provincial Natural Science Foundation of China under Grant 2021JJ30734 and Grant 2021JJ30699.

ABSTRACT Path planning is one of the important components of the Unmanned Aerial Vehicle (UAV) mission, and it is also the key guarantee for the successful completion of the UAV's mission. The traditional path planning algorithm has certain limitations and deficiencies in the complex dynamic environment. Aiming at the dynamic complex obstacle environment, this paper proposes an improved TD3 algorithm, which enables the UAV to complete the autonomous path planning through online learning and continuous trial and error. The algorithm changes the experience pool of TD3 algorithm to priority experience replay, so that the agent can distinguish the importance of empirical samples, improve the sampling efficiency of the algorithm, and reduce the training time. The average TD3 is proposed, and the average value of Q_1Q_2 is taken when the target value is updated to solve the problem of overestimating the Q value while avoiding underestimating the Q value, so that the improved algorithm has better stability and can adapt to various complex obstacle environments. A new reward function is set up, so that each step of the UAV action can receive reward feedback, which solves the problem of sparse reward in deep reinforcement learning. The experimental results show that this method can train the UAV to reach the target safely and quickly in a multi-obstacle environment. Compared with DDPG, SAC and traditional TD3, the path planning success rate of this algorithm is higher than that of the other three algorithms, and the collision rate is lower than that of the comparison algorithm, which has better path planning performance.

INDEX TERMS UAV, path planning, deep reinforcement learning, prioritized experience replay, average TD3 algorithm.

I. INTRODUCTION

Unmanned aerial vehicle (UAV) are radio-controlled aircraft operated remotely or through self-contained program control devices. Because of its small size, low cost, easy to use and other advantages, it is widely used in various fields [1]. UAV path planning plays a vital role in establishing the UAV mission model and serves as a crucial guarantee for the successful completion of the UAV mission. Its purpose is to plan the optimal flight path in a given scenario, considering the path length, terrain environment, threat information, UAV maneuverability constraints and other related factors [2]. A well-designed path planning algorithm can enable UAV to accomplish tasks at a minimal cost, particularly in complex and dynamic environments. The path planning mentioned in this paper is a point-to-point planning method, which is characterized by obstacle avoidance, the shortest and smoothest running path. In contrast to coverage path planning, complete coverage path planning involves determining a path that traverses all points in a given region or spatial range while simultaneously avoiding obstacles [3].

In recent years, scholars have conducted a lot of research in the field of UAV path planning algorithms, and proposed a variety of path planning algorithms. These algorithms have

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandro Floris.

their own characteristics in application fields, advantages and disadvantages. According to the different research methods, UAV path planning algorithms can be divided into two categories: one is based on non-learning algorithms, including classical path planning algorithms and intelligent optimization algorithms; the other is learning-based algorithms, such as deep reinforcement learning algorithms [4].

The classical path planning algorithms, such as Artificial potential field (APF) [5], [6], Rapidly-exploring random tree (RRT) [7], [8], A* algorithm [9], [10], Voronoi diagram (VD) [11], [12], [13], Probabilistic road map (PRM) [14], and so on. The main reason why these classical path planning methods can be successful is that they are easy to implement. At the same time, they show good results in path optimization, fast solution generation and static environment with simple obstacles. However, the time complexity of these algorithms is relatively high, and the performance is easily reduced when dealing with high-dimensional space path planning. In addition, they are also easy to fall into local optimums, which may lead to large deviations in path planning results. Intelligent optimization algorithms, such as Genetic algorithm (GA) [15], Particle swarm optimization (PSO) [16], [17], [18], Gray wolf optimization (GWO) [19], Differential evolution (DE) [20], etc. These algorithms are simple to implement in UAV path planning, have global search ability, and show good robustness for large-scale optimization problems. However, in the path planning process, they may fall into local optimal solution. In addition, the computational complexity of these algorithms is high, the performance depends largely on the choice of parameters, and the convergence speed is relatively slow.

The above path planning algorithms are based on search-based and sample-based methods to generate viable paths within a given environment. However, with the increase of environmental complexity and uncertainty, the feasibility of these methods is greatly reduced. In addition, after the front-end path search, the above method also needs to optimize the back-end trajectory, which leads to a high time complexity of the algorithm. In practical applications, the utilization of the aforementioned methods faces significant limitations when the UAV needs to adapt to unfamiliar environments. Currently, enabling real-time collision-free path planning for UAVs from start to end in unknown environments remains a formidable challenge. In such unfamiliar environments, UAV lack knowledge of the environment and the environment is very likely to change all the time, which requires UAV to have the capacity to perceive, decide and act, as well as the ability to explore and learn. For this reason, it is particularly crucial to design a method that enables UAV to learn autonomous path planning in an unknown environment. In recent years, the Deep reinforcement learning (DRL) method with autonomous learning ability has successfully solved the path planning problem of UAV in unknown environments. As a decision-making control method different from traditional machine learning algorithms, DRL enables agents to adapt to the environment through online learning and continuous trial and error without any guidance signal in an unknown environment. Because DRL achieves the saliency of the target effect through training, it has attracted the attention of many researchers and has begun to be applied in the field of UAV path planning.

Han et al. [21] proposed an improved Deep Q-network (DQN) that utilizes priority and exponential sampling methods, enhancing sampling algorithm stability and performance by adjusting the random uniform sampling of UAV flight experience samples. Xie et al. [22] introduced an improved Deep recurrent Q-Network (DRQN) that combines reward and Q values using a novel action selection policy to mitigate inaccurate neural network predictions during early-stage training. The improved DRQN algorithm exhibits low computational complexity, significantly improving learning efficiency and stability. Runjia et al. [23] proposed a multi critic-delayed Deep deterministic policy gradient (DDPG) method that utilizes average estimation of multi evaluation networks to decrease the DDPG's reliance on the evaluation network. The method employs delayed learning to mitigate overestimation and target network error accumulation, resulting in superior path planning performance compared with traditional DDPG. Hu et al. [24] proposed the REL-DDPG algorithm in their research, which is a DDPG algorithm based on the concept of relevant experience learning. Compared to the traditional DDPG algorithm, this algorithm shows significant improvements in terms of convergence speed and effectiveness. Bohao and Wu [25] proposed an enhance DDPG. This algorithm guides the UAV to track targets by designing a new reward function and smoothens the trajectory of the UAV using penalty terms. Additionally, the algorithm approximates the environmental state using a long short-term memory network, thereby enhancing the algorithm's approximation accuracy and data utilization. Zhang et al. [26] proposed an improved Twin-delayed deep deterministic policy gradient (TD3) algorithm, which utilizes a twin stream actor-critic network architecture. This algorithm extracts environmental features from observations and their variations to handle the stochasticity and dynamics of obstacles in the environment. Experimental results demonstrate that the algorithm exhibits good path planning performance in dynamic environments. Lee et al. [27] proposed a new Soft actor-critic (SAC) algorithm called SACHER. Experimental results show that SACHER is capable of generating optimal paths for UAV. Yan et al. [28] set up a dual deep Q network (D3QN) algorithm based on global situation information. This method uses a set of situation diagrams as input to approximate the Q value corresponding to all candidate actions. In addition, it combines ε-greedy strategy and heuristic search rules to select actions. Experiments show that the algorithm shows good performance under both static and dynamic task settings. Peng et al. [29] studied a UAV-assisted mobile edge computing network, and adopted a DRL framework for the problem of size explosion. A Dual

deep Q-learning network (DDQN) algorithm is proposed to realize the path planning of UAV. The simulation results verify the effectiveness of the path planning scheme.

DRL algorithms have been applied to tackle the autonomous path planning problem of UAV, yielding improved outcomes. However, existing algorithms such as DQN, DRQN, DDPG have overestimation of Q value, while TD3 algorithm has underestimation of Q value. And in practice, there are some problems to be solved and optimized in path planning using DRL in complex environments, such as long exploration period, sparse rewards, low sample utilization, and convergence stability. Aiming at these problems, this paper proposes an Improved TD3 (I-TD3) algorithm that enables UAV to exhibit better path planning performance in complex and dynamic obstacle environments. The difference of this paper is that the traditional TD3 algorithm takes the minimum value of Q_1Q_2 when updating the target value, and the algorithm of this paper takes the average value of Q_1Q_2 when updating the target value, which enhances the stability of the algorithm and enables the UAV to adapt to various obstacle environments; for the low utilization of samples, this paper changes the experience pool of the TD3 algorithm into the Priority experience replay (PER), which improves the algorithm's utilization of samples and reduces the training time; in the face of the DRL reward sparsity problem, a new reward function is set so that each step of UAV action can receive reward feedback.

The main contributions of this paper are as follows:

(1) Using the OpenAI Gym of DRL as a simulation platform, a three-dimensional continuous simulation environment is customized. The established simulation environment can visualize the training process and help analyze the behavior of the UAV. The simulation results show that the proposed algorithm has strong stability and a high success rate, and can effectively solve the path planning problem of UAV in a dynamic environment.

(2) The priority experience replay is used as the experience replay pool of the TD3 algorithm, so that the agent can distinguish the importance of the experience sample, reduce the training time, improve the sample utilization rate, and improve the efficiency of the experience pool extraction experience.

(3) This paper proposes an average TD3 algorithm. Different from the traditional TD3 algorithm, when updating the target value, the algorithm in this paper takes the average value of Q_1Q_2 instead of the minimum value. On the basis of solving the problem of overestimating Q value, the situation of underestimating Q value is avoided, so that the improved algorithm has better stability and can adapt to various complex obstacle environments.

(4) We re-set the new reward function to ensure that the UAV receives reward feedback at every step of the action. This improvement not only solves the problem of sparse feedback in DRL, but also greatly improves the convergence speed of the algorithm. This allows the UAV to complete its mission in an efficient manner.

The rest of this paper is organized as follows: The second part introduces some background knowledge of Markov decision process (MDP) and DRL. In the third part, the algorithm proposed in this paper is described in detail. In the fourth part, we describe the state space, action space and reward function of UAV path planning. The path planning process of UAV based on this algorithm is also described. The fifth part describes the experimental environment, experimental details, experimental settings and parameter settings, and analyzes the simulation results. The sixth part summarizes and prospects this paper.

II. BACKGROUND

A. MARKOV DECISION PROCESS

MDP refers to a random process with Markov property, which is a sequential decision model. The model was proposed by Bellman in 1957 to solve problems with uncertain and dynamic characteristics, such as robot navigation problems and asset portfolio problems. An agent is a machine learning agent in MDP. It can perceive the state of the external environment and make corresponding decisions accordingly, and constantly adjust its own decisions by applying actions to the environment and relying on the feedback from the environment. The environment in the MDP model covers everything outside the agent, and its state changes due to the influence of the agent 's behavior, and these changes can be fully or partially perceived by the agent. After each decision, the environment will provide corresponding rewards to the agent [30], [31], [32]. The MDP is shown in Fig. 1.

MDP can be expressed as: $MDP = (S, A, P, R, \gamma)$, Where S is the set of all possible states in the problem, and A is the set of all actions that the agent can take in the problem. P is a state transition probability function, which is used to calculate the probability of taking an action in a state to the next state. R is a reward function, which is used to measure the reward obtained by taking an action agent in a certain state. γ is the discount factor, also known as the attenuation factor, $\gamma \in [0, 1]$ which is used to weigh the impact of future rewards on cumulative rewards. As shown in the Fig. 1, in the learning phase, the agent receives the state S_t from the environment and performs the action A_t according to the learning policy π; then the environment returns a reward value R_t to the agent, and the purpose of the agent is to learn the policy of maximizing the reward from the environment. Repeating this process, the final agent update policy maximizes the cumulative reward return G_t.

The policy function:

$$\pi^*(s|a) = P[s = S_t|a = A_t]. \tag{1}$$

The reward function is defined as follows:

$$R(s|a) = E[R_{t+1}|S_t = s|A_t = a]. \tag{2}$$

The cumulative reward value:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma R_{t+k}. \tag{3}$$

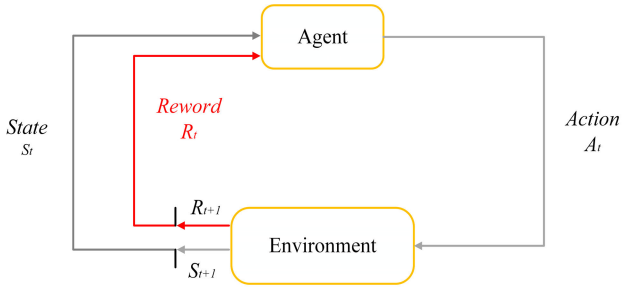


FIGURE 1. Markov decision process.

The state-value function:

$$V_\pi = E_\pi[G_t|S_t = s]. \tag{4}$$

B. DEEP REINFORCEMENT LEARNING

DRL is an algorithm that seamlessly combines Reinforcement learning (RL) and Deep learning (DL). By utilizing MDP, it effectively characterizes the interaction between the agent and the environment. The primary goal of DRL, akin to MDP, is to determine the optimal policy that permits the agent to achieve its objective within the current environment, while maximizing the rewards obtained from executing that policy. Throughout the training process, DRL allows the agent to actively interact with the environment, make informed decisions by selecting actions, and subsequently receive informative feedback using a reward mechanism. Continuously exploring and pursuing greater rewards, it ultimately achieves an outstanding action selection policy [33].

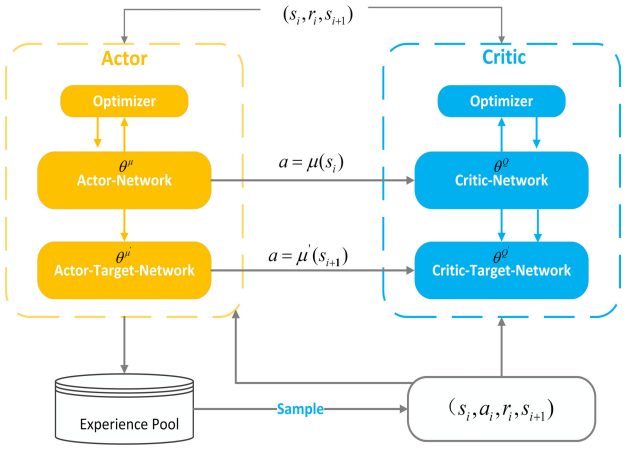


FIGURE 2. The framework of DDPG.

Taking DDPG as an example, it is a DRL algorithm based on AC framework, utilized for addressing problems in continuous action spaces. DDPG combines deep neural networks with deterministic policy gradients, enabling the learning of continuous action policies. The DDPG algorithm primarily consists of two networks: the actor network and the critic network. The actor network functions as a deterministic policy function, taking the state as input and producing the corresponding action as output. The critic network serves as a Q-value function that evaluates the value of the current state and action [34]. The DDPG framework is illustrated in Fig. 2.

III. IMPROVED TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENTS

At present, the traditional UAV path planning algorithm has certain limitations. When the UAV is in an unknown environment, it is necessary to plan the map globally every time, resulting in slow planning and time-consuming, and it is difficult to find a safe path. Therefore, how to make the UAV have the ability of autonomous learning and adapting to environmental changes in planning is particularly important. The DRL algorithm has the advantages of model-free, online learning, and offline policy, which breaks the limitations of traditional algorithms and enables UAV to perform autonomous path planning in unknown environments. Simultaneously, DRL has the capability to govern the continuous actions of UAV, aligning it more closely with the practical requirements of UAV path planning.

To address the UAV path planning problem, the TD3 algorithm, a DRL approach built upon policy gradients, has been employed in this study. Its advantage lies in the fact that it is updated with the policy as the target, and it is directly fitted to the policy during the training process, which realizes the output of the continuous action space, and it can reduce the training time and speed up the convergence of the algorithm because it does not need to calculate the action values. If the DRL method based on the value function is used, although it can solve the continuous or high-dimensional state space problem well, its action space is discrete, the planned path is not smooth, and it may be necessary to optimize the trajectory of the back-end, which is still a great limitation in UAV path planning. Meanwhile, encountering the random strategy problem, the value function-based method may produce large changes in each update during training, and is not easy to converge.

A. TWIN DELAYED DEEP DETERMINISTIC POLICY GRADIENTS (TD3)

The TD3 is a DRL algorithm that utilizes deterministic policies. It builds upon the DDPG algorithm and introduces three key techniques [35], [36]:

1) Dual network: Two sets of Actor-Critic frameworks are used, and the target value is calculated by taking the minimum value from the critic network, preventing overestimation of the network.

2) Target policy smoothing: When calculating the target value, adding noise perturbations to the outputs of the target policy to make the training more stable and facilitate convergence.

3) Delayed update: The actor network is updated after multiple updates to the critic network. This delayed update method can reduce error accumulation and make the training of the actor network more stable and reliable.

The TD3 algorithm consists of 2 actor networks and 4 critic networks. The critic target network evaluates the sampled

states s_{t+1} and actions $\tilde{a}_t$ to output $Q(s_{t+1}, \tilde{a}_t)$. During the updating process, the target values are updated based on the maximum Q-value, which can introduce some errors with each update. Over multiple updates, these errors can accumulate, resulting in overestimation of the values for certain states. To address this issue, the TD3 algorithm uses two critic networks to evaluate the Q-values. During the update process, select the smaller Q-value to update the target value.

$$y_{target} = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \tilde{a}_t). \tag{5}$$

The TD3 uses a delayed update method when updating network parameters. After every d updates to the critic network parameters, the actor network parameters are updated. The update formula for the critic network is as follows:

$$\nabla_{\theta_i} J(\theta_i) = \frac{1}{N} \sum_t \nabla_{\theta_i} (y_{target} - Q_{\theta_i'}(s_t, a_t))^2. \tag{6}$$

The update formula for the actor network is as follows:

$$\nabla_\phi J(\phi) = \frac{1}{N} \sum_i \nabla_{a_t} Q_{\theta_i}(s_t, a_t)|_{a_t=\pi_\phi(s_t)} \nabla \pi_\phi(s_t). \tag{7}$$

The update formula for the target network is as follows:

$$\begin{aligned} \phi' &= \tau\phi + (1-\tau)\phi' \\ \theta_i' &= \tau\theta_i + (1-\tau)\theta_i' \\ \tau &\in [0, 1]. \end{aligned} \tag{8}$$

B. IMPROVED TD3 (I-TD3)

When extracting samples from the experience pool for training, the TD3 currently utilizes a random sampling method, resulting in low learning efficiency. Additionally, in an attempt to address the issue of overestimated Q values, the TD3 algorithm updates the target value using the minimum value of Q_1Q_2, resulting in an underestimation of the Q value. This paper aims to enhance the existing algorithm by introducing priority experience replay to the TD3 algorithm's experience pool. This modification enables the agent to distinguish the importance of empirical samples, leading to improved learning efficiency and reduced training time. The proposed approach, referred to as average TD3, updates the target value by taking the average value of Q_1Q_2. This avoids underestimation and overestimation of the Q value, resulting in enhanced algorithm stability. Furthermore, the reward function is adjusted to provide feedback at each step of the UAV's action, effectively tackling the problem of reward sparsity in DRL.

1) PRIORITY EXPERIENCE REPLAY (PER)

In the training process of DRL, it is necessary to store the input and output data of the network, thus requiring the establishment of an experience replay buffer to store experience data. When the data is replayed, the agent updates the network parameters according to the previously observed empirical data. The form of the data is (s_t, a, r, s_{t+1}), and all the data in the experience pool are randomly sampled during the update. Priority experience replay (PER) is to extract the most valuable experience when extracting experience, but it cannot only extract the most valuable, otherwise it will cause over-fitting. It should be that the higher the value, the greater the probability of extraction, and the lowest the value, it will also be extracted with a certain probability. The key to the priority experience playback mechanism is to play back very successful or extremely failed experiences at a higher frequency, and these experience samples have higher learning value [37], [38], [39].

In DRL, TD-error represents the discrepancy between the current Q value and the target Q value, reflecting the degree of learning required by the agent. The larger the TD-error difference, the more the experience samples need to be updated, accelerating the agent's task completion. Therefore, TD-error is used to differentiate the importance level of experience samples, and TD-error is defined as follows:

$$\delta_j = r + \gamma Q_{\theta_i'}(s_{t+1}, \tilde{a}_t) - Q_{\theta_i}(s_t, a_t). \tag{9}$$

Sampling probability of experience samples:

$$P(j) = \frac{p_j^\alpha}{\sum_i p_i^\alpha, \alpha \in [0, 1]}, \tag{10}$$

among them, p_j is a priority index based on TD-error, α is a priority adjustment parameter. To maintain sample diversity, random factors are taken into consideration when selecting experience samples, which means that even experience samples with small TD-error values have the possibility of being chosen. When α takes 1, the TD-error value is directly used; when α takes 0, it is the original uniform random sampling. The priority indicator is based on a ranking approach.

$$p_j = \frac{1}{rank(j)}, p_j > 0. \tag{11}$$

The agent tends to update experience samples with high TD-error, which modifies the original probability distribution and introduces errors into the model. Consequently, the model may fail to converge during neural network training. To mitigate this issue, importance sampling is employed to correct weight changes:

$$W_j = (\frac{1}{M \cdot p_j})^\beta, \tag{12}$$

in the above equation, M is the number of experience replay pools, and parameter β is the degree of correction error. According to the above process, the data that interact with the environment can distinguish the importance of the experience sample and enhance the learning efficiency of the experience sample.

2) AVERAGE TD3

The TD3 algorithm solves the overestimation of DDPG, but there is also a case of underestimation. In order to

solve this problem, this paper proposes an average TD3 algorithm to solve the problem of overestimation and avoid underestimation.

The overestimation of the DDPG algorithm comes from two aspects: bootstrapping and maximization. If the overestimation is uniform, it will not affect the final decision of the agent; If it is non-uniform, the final decision of the agent will be significantly influenced by it. However, in fact, the overestimation of the network is usually non-uniform.

When updating the critic network, assuming that the data sampled from the experience pool is (s_t, a_t, r_t, s_{t+1}), Firstly, we will compute the target values y:

$$y_{target} = r + \gamma \min_{i=1,2} Q'_{\theta'_i}(s_{t+1}, a_{t+1}). \tag{13}$$

Owing to network overestimation, therefore:

$$Q'_{\theta'_i}(s_{t+1}, a_{t+1}) \geq Q^*(s_{t+1}, a_{t+1}), \tag{14}$$

where $Q^*_{\theta'_i}(s_{t+1}, a_{t+1})$ denotes the true optimal state action value of the state action to (s_{t+1}, a_{t+1}). We then let $Q_{\theta'_i}(s_t, a_t)$ approximate y, so that $Q_{\theta'_i}(s_t, a_t)$ is estimated, that is:

$$Q_{\theta'_i}(s_t, a_t) \geq Q^*(s_t, a_t). \tag{15}$$

When the critic network is updated, the state-action values get overestimated. To address the problem, the TD3 algorithm selects the minimum value from Q_1Q_2 to perform parameter updates. This ensures that the algorithm does not suffer from this issue. However, because the minimum value is chosen as the target value during each update, it may result in the underestimation of Q-values. Therefore, this paper selects the average value of the Q_1Q_2 to update the target values. This modification allows the improved algorithm to solve the problem of overestimated Q-values while avoiding the occurrence of underestimated Q-values.

$$y_{target} = r + \gamma average_{i=1,2} Q_{\theta'_i}(s_{t+1}, a_{t+1}). \tag{16}$$

IV. UAV PATH PLANNING BASED ON I-TD3 ALGORITHM

A. STATE SPACE

During the process of DRL, the UAV determines which actions to take based on the received state information from the environment. Therefore, designing a suitable state space is of utmost importance. The state space should accurately represent the current state of the UAV and provide information about significant environmental elements such as obstacle position, shape, and target position. Thus, we define the state space as a combination of sensor-detected environmental information and the UAV's own state.

In this paper, we employ LIDAR for obstacle detection, and the environmental observations are depicted in Fig. 3 and 4. Fig. 3 shows the LIDAR's horizontal plane laser distance, while Fig. 4 shows the LIDAR's vertical plane laser distance. The scanning angle range is denoted by π, and the distance between the two laser beams at the angle is $\frac{\pi}{6}$. $(d_1, d_2, \cdots, d_7)$ and $(d_8, d_9, \cdots, d_{14})$ represent the ray lengths of the sensor on the horizontal and vertical planes, respectively. d_4 and d_{11} denotes the same laser line, so $d_4 = d_{11}$. If the LIDAR detector is unable to detect any obstacles within a specified range, the emitted ray's length will be equivalent to the maximum distance that can be detected.

Environmental information is defined as:

$$s_e = [\xi_i, d_i]^T, i = 1, \cdots, 14, \tag{17}$$

ξ_i is a hot code. If the sensor detects an object of limited distance, it is 1; otherwise, 0.

Taking a quadcopter with an X configuration as an example, the state of the UAV can be measured in real-time using GPS and gyroscopes.

$$s_u = [x, y, z, v_x, v_y, v_z, \beta, d_0]^T, \tag{18}$$

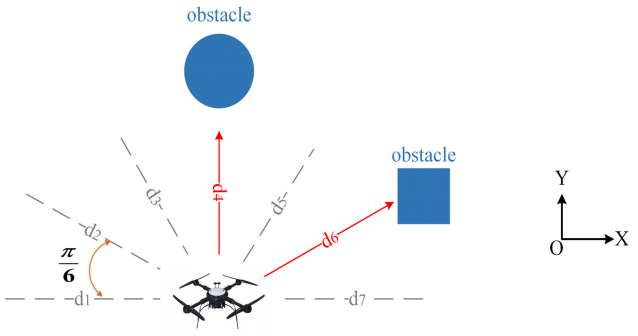


FIGURE 3. Laser distance on the horizontal plane of the UAV.

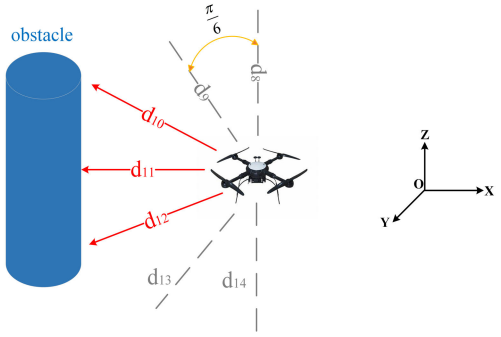


FIGURE 4. Laser distance on the vertical plane of the UAV.

where (x, y, z), representing the real-time position of the UAV, (v_x, v_y, v_z) denotes the speed of x, y, z the UAV along; d_0 is the straight-line distance between the UAV and the target, β represents the angle between the direction of d_0 and the y-axis.

In order to expedite the completion of the navigation task and improve convergence speed, we have changed the position of the UAV to its relative position with respect to the target. As a result, the state space of the UAV has been redefined.

$$s_u = [x_{target} - x, y_{target} - y, z_{target} - z, v_x, v_y, v_z, \beta, d_0]^T. \tag{19}$$

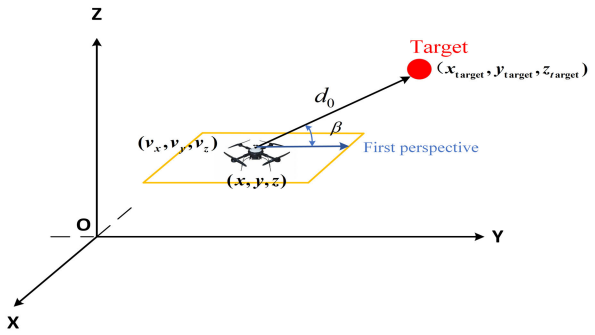


FIGURE 5. The state space of the UAV.

In summary, the state is defined as:

$$S = [s_e^T, s_u^T]. \tag{20}$$

B. ACTION SPACE

The propellers of the four-rotor UAV consist of two positive propellers and two negative propellers, symmetrically distributed in the four corners of the UAV frame [40]. During operation, the propellers generate a downward airflow through high-speed rotation, which provides an upward lift force to the UAV. By analyzing the forces acting on a quadcopter's basic flight attitude, we can conclude that the UAV achieves various flight attitudes by adjusting the different rotational speeds of its four motors [41]. Therefore, the flight control board can control the UAV's flight attitude and position by altering the lift and torque through different input voltages to the brushless motors, based on external demands.

This paper considers the forces in various directions of UAV as executable actions, enabling the UAV to achieve functions such as takeoff, landing, forward movement, backward movement, and lateral movement. Simultaneously, the UAV's steering is controlled by the rotation angle.

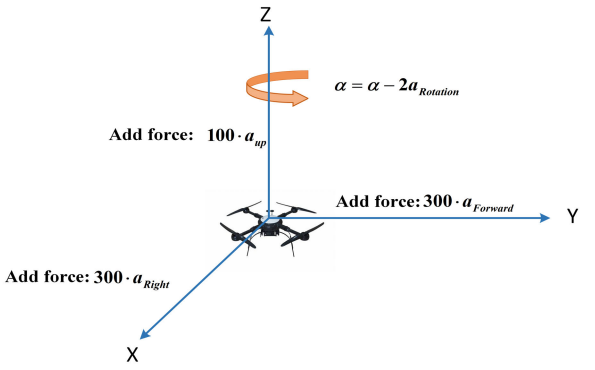


FIGURE 6. The action space of the UAV.

In the Fig. 6, $300 \cdot a_{Forward}$, $300 \cdot a_{Right}$, and $100 \cdot a_{Up}$ respectively represent the forces acting upon the UAV in the directions of the Y-axis, X-axis, and Z-axis. They can control the UAV's movement in the forward/backward, left/right, and upward/downward directions. $\alpha = \alpha - 2a_{Ratation}$ represents the variation in the UAV's rotation angle along the Z-axis. Therefore, the action space is represented as:

$$A = [a_{Forward}, a_{Right}, a_{Up}, a_{Ration}]^T$$
$$a_{Forward}, a_{Right}, a_{Up}, a_{Ration} \in [-1, 1]. \tag{21}$$

C. REWARD FUNCTION

The reward function is a crucial component of DRL. Designing a reasonable reward function not only improves the convergence speed of the training process but also enables the UAV to efficiently and safely accomplish its tasks [42], [43].

The reward function $r(s_t, a_t)$ represents the environmental feedback for taking action a_t in a state s_t, and it can be used to evaluate the quality of the action taken in the current state. If the reward $r(s_t, a_t)$ is large, it means that acting in the current state is good for achieving the task and the probability of acting in the next policy update will increase. Otherwise, the probability decreases. The reward function in this paper aims to guide the UAV to the target location while ensuring its safety. The reward function is set as follows:

$$R = r_{step} + r_{angle} + r_{obs} + r_{dis}, \tag{22}$$

$$r_{step} = -0.01, \tag{23}$$

$$r_{angle} = \begin{cases} 1, & \beta \in [0, \frac{\pi}{2}), \\ 0, & \beta = \frac{\pi}{2}, \\ -1, & \beta \in (\frac{\pi}{2}, \pi], \end{cases} \tag{24}$$

$$r_{obs} = \begin{cases} -0.1 \cdot (d_{safe} - d_i), & \text{if } d_i < d_{safe}, \\ -5, & \text{if collides with obstacles}, \\ 0, & else, \end{cases} \tag{25}$$

$$r_{dis} = \begin{cases} 5, & \text{if } d_0 < 5, \\ -0.1, & normalized(d_0), else, \end{cases} \tag{26}$$

The reward function consists of four components:

1) To expedite the UAV's navigation mission, a penalty of -0.01 is imposed on it at each step.

2) When $\beta \in [0, \frac{\pi}{2})$, the UAV is flying in the direction close to the target point, so it gets a positive reward. When $\beta = \frac{\pi}{2}$, the UAV is neither close to nor far from the target point, so There are neither rewards nor punishments. When $\beta \in (\frac{\pi}{2}, \pi]$, the UAV is moving away from the target, so it gets a penalty.

3) To make the UAV avoid colliding with an obstacle, when the distance between the UAV and the nearest obstacle is less than d_{safe}, the UAV will suffer a distance penalty. When the UAV collides with an obstacle, it will suffer a penalty of -5. When the distance between the UAV and the nearest obstacle is greater than d_{safe}, the obstacle poses no threat to the UAV, so it will not receive any penalty. The value of d_{safe} is 10.

4) To incentivize the UAV to reach the designated target zone quickly, a function that measures the distance between the UAV and the target has been set. If the distance is negative,

the UAV will incur a penalty of −0.1. Upon reaching the target point, the UAV will be rewarded with a value of 5.

D. STATE NORMALIZATION

The state space introduced in Section IV-A involves state values of different units and scales, so the state values of the input network need to be preprocessed. In this article, normalization method is employed to process the state values. In the state space, all state values except ξ_i require preprocessing.

$$k_{target} - k = \frac{k_{target} - k}{k_{max} - k_{min}}, k = x, y, z, \tag{27}$$

$$v_k = \frac{v_k}{v_{max,k}}, \tag{28}$$

$$\beta = \frac{\beta}{\pi}, \tag{29}$$

$$d_0 = \frac{d_0}{\sqrt{\sum(k_{max} - k_{min})^2}}, \tag{30}$$

$$d_i = \frac{d_i}{\text{max laster length}}, \tag{31}$$

Among them, k_{max} and k_{min} denote the upper and lower boundary values, respectively, along the k-axis for the task scene, and $v_{max,k}$ denotes the maximum achievable velocity of the UAV in the direction of the k-axis.

E. DESIGN OF UAV PATH PLANNING METHOD BASED ON I-TD3

When the UAV first interacts with the environment, it cannot distinguish between obstacles and targets. It adjusts its policy based on reward values and penalty values received from environmental feedback during the exploration process, ultimately accomplishing the path planning task. The framework of the path planning algorithm is illustrated in Fig. 7.

When the UAV explores the environment, the exploration of the action space can be increased by adding Gaussian noise. Simultaneously, the explored experiences are stored in the form of tuples and placed into an experience replay pool. During network training, PER is introduced to prioritize the learning of important experiences, thereby reducing training time. When updating the target values, we choose the average value of Q, which makes the algorithm more stable. In the end, the UAV is able to autonomously plan paths and successfully complete various tasks in complex environments.

The pseudo-code for the I-TD3 algorithm is as follows:

Algorithm 1 The I-TD3 algorithm

Initialize actor network π_ϕ,and critic networks $Q_{\theta_1}, Q_{\theta_2}$ with random parameters $\alpha, \beta, \phi, \theta_1, \theta_2, T$,minibatch k
Initialize target networks $\phi' \leftarrow \phi, \theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2$
Initialize replay buffer $\mathcal{M}$
for $t = 1$ to T **do**
  select action $a_t \sim \pi(s_t) + \epsilon, \epsilon \sim \mathcal{N}(0, \delta)$, reward r and new state s_{t+1}
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{M}$
  set the priority $P_t = \max_{i<t} P_i$
  if $t > M$ **then**
    for $j = 1$ to k **do**
      Based on $P(j)$ sampling empirical samples
      $j \sim P(j) = \frac{P_j^\alpha}{\sum_i P_i^\alpha}$
      Calculate the relevant importance sampling weights W_j
      $W_j = (\frac{1}{M \cdot P(j)})^\beta$
      Calculate δ_j
      $\delta_t = r + \gamma Q_{\theta_i'}(S_{t+1}, \tilde{a}_t) - Q_{\theta_i}(s_t, a_t)$
      Update the empirical sample priority based on the TD-error
      $P_j \leftarrow |\delta_j|$
    end for
    computation the critic network:
    $\tilde{a}_t \leftarrow \pi_{\phi'}(s_t) + \epsilon$
    $y_{target} \leftarrow r + \gamma average_{i=1,2} Q_{\theta_i'}(s_{t+1}, \tilde{a}_t)$
    Update critic $\theta_i \leftarrow \min_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$
    if t mod d **then**
      Update ϕ by the deterministic policy gradient
      $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \nabla \pi_\phi(s)$
      Update target network:
      $\phi' = \tau\phi + (1 - \tau)\phi'$
      $\theta_i' = \tau\theta_i + (1 - \tau)\theta_i'$
    end if
  end if
end for

V. EXPERIMENTS AND RESULTS

A. EXPERIMENT PLATFORM AND SETTINGS

OpenAI Gym is used as the simulation platform. OpenAI has developed and maintained a Python library called Gym, which serves as a toolkit for developing and comparing DRL algorithms. Gym allows testing and learning the performance of DRL algorithms and is compatible with other numerical calculation libraries such as TensorFlow and Torch. Gym provides commonly used DRL environments and also allows for customization. The simulation environment based on Gym customization is depicted in Fig. 8.

The simulation environment we have developed is a rectangular area measuring 400 × 400 × 100. Within this environment, the starting point of the UAV is represented by the blue area, while the destination is denoted by the green area. Additionally, random obstacles are scattered throughout the white areas of the environment. These obstacles are

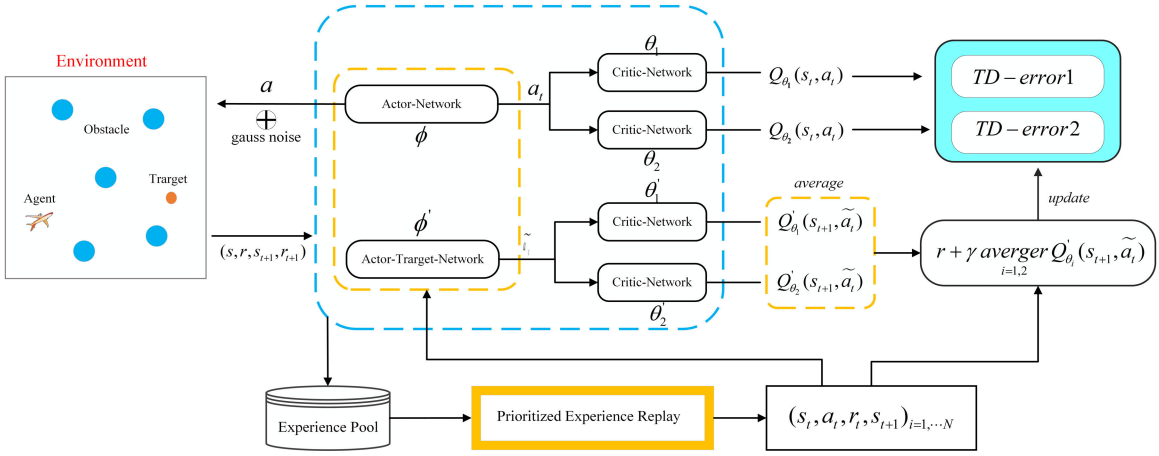


FIGURE 7. Framework diagram of UAV path planning based on I-TD3 algorithm.

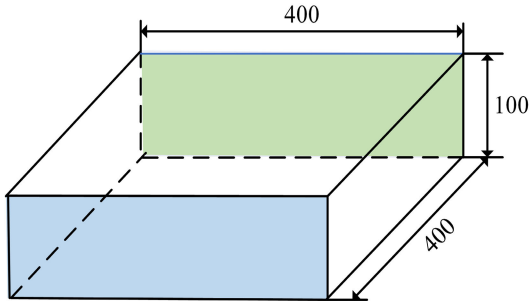


FIGURE 8. The simulation environment.

positioned in mid-air and are intended to train the UAV's ability to avoid collisions in the Z-axis direction. In our experiment, the UAV is capable of flying at a maximum speed of 20 and has a maximum stride of 1000. It maintains a flying height ranging from 20 to 100, while the target it aims to reach is a sphere with a diameter of 20. The task of the UAV is to start from the blue area without collision and eventually reach the green area.

B. SIMULATION ENVIRONMENT

For training and testing the performance of the I-TD3 algorithm, two experimental environments are established: Environment I and Environment II. Environment I consists of four static environments, and E1 is set up with five cylinders with dimensions of 15 × 100; E2 sets 5 cylinders of size 15 × 100 and 5 cubes of size 30 ×30× 50; E3 sets 5 cylinders of size 15×100,5 cylinders of size 15 × 50, and 5 cubes of size 50 ×30 ×50; E4 Set 5 cylinders of size 15 × 100, 5 cylinders of size 15 × 50, 5 cubes of size 30 × 30 × 50, and 5 cubes of size 50 × 30 × 30.

Environment II consists of four dynamic environments, E5 sets five cylinders with dimensions of 15 × 100 and five cubes with dimensions of 30 × 30 × 50, where half of the obstacles move in the negative direction of the Y-axis at the speed of 10, and once the obstacle reaches the boundary of the task space, it initiates backward movement and proceeds to repeat the procedure; E6 differs from E5 in that obstacles

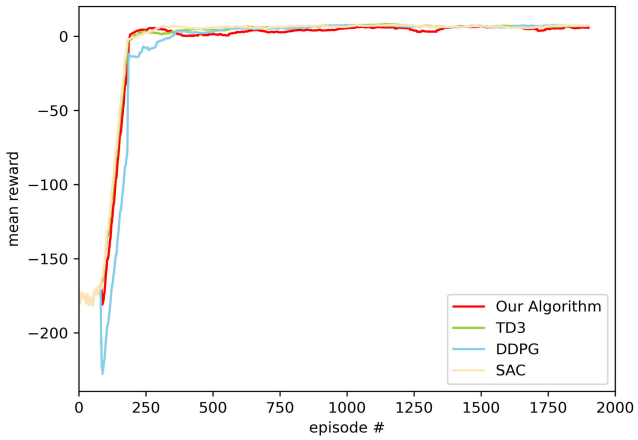


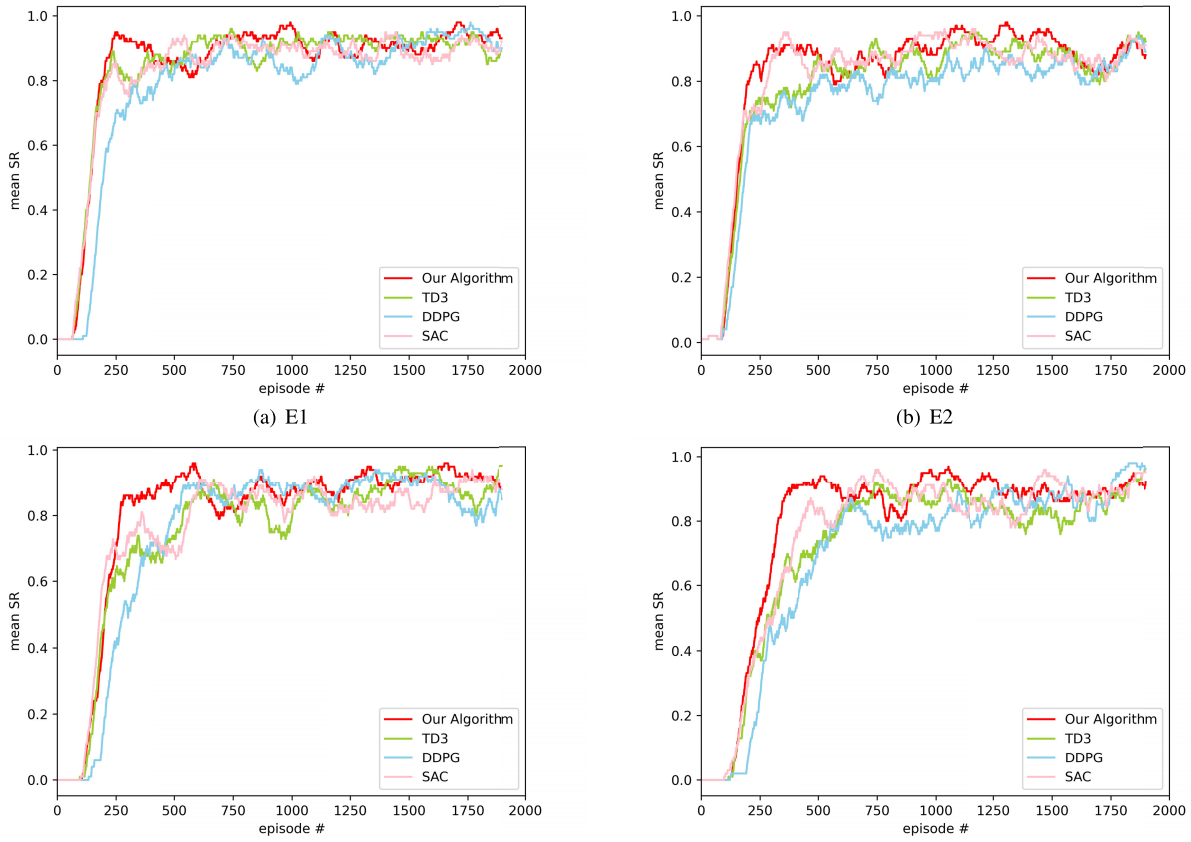
FIGURE 9. The convergence curve of average rewards obtained from the training environment E1.

move at a speed of 20; E7 Set 5 cylinders of size 15 × 100 and 5 cubes of size 30 × 30 × 50, with all obstacles moving at speed 10; E8 differs from E7 in that all obstacles move at speed 20.

C. TRAINING AND RESULTS

The training of the UAV involves exploring and adjusting its action policy based on environmental feedback to ultimately achieve path planning and obstacle avoidance. At the beginning of each training session, the network parameters are initialized, and start and end points are randomly generated within the corresponding region. Training will end if any of the following conditions occur: (1) the training step reaches 1000, (2) the UAV collides with an obstacle, or (3) the UAV reaches the destination. DDPG, TD3, SAC [44] and the proposed algorithm are used to train the UAV in the environment E1. In the training process of 2000 rounds, the average reward curves of the four algorithms are shown in Fig. 9.

We observe that during the initial training phase, the UAV explores the environment randomly, resulting in a very low average reward. As the UAV gathers more data, it starts



(a) E1 (b) E2

(c) E3 (d) E4

FIGURE 10. The average success rate of Environment I.

TABLE 1. Experiment parameters.

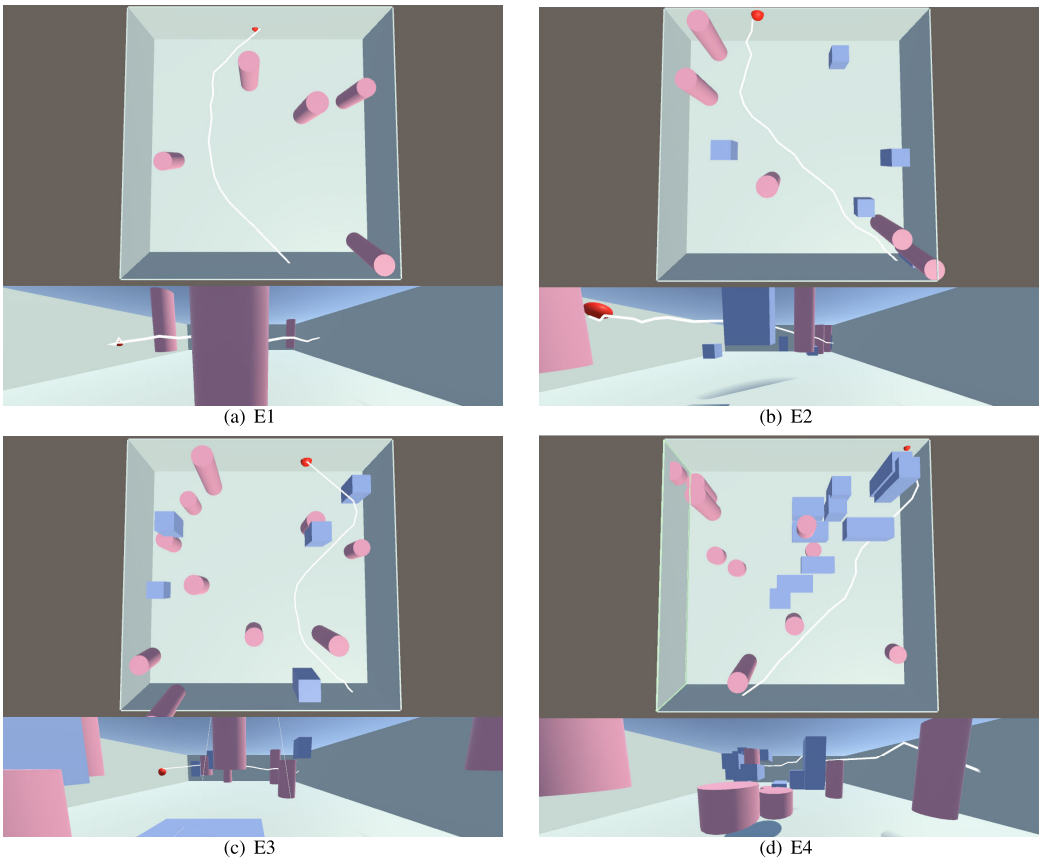
Parameters	Value
Policy Frequency	2
Discount Factor γ	0.99
Reply Buffer Size	2^{17}
Batch Size	128
Critic Learning Rate	0.001
Actor Leaning Rate	0.001
Soft Update Factor τ	0.01
Noise Clip	0.5
Policy Noise	0.2
Max Episodes	2000
Max Steps	1000

training the network to update its policy. With an increase in the number of training sessions, the average reward value gradually increases, and the average reward curves for all four algorithms converge at around 200 episodes. Toward the end of training, the average reward curve approaches approximately 0. Compared with DDPG, TD3 and SAC algorithms, the proposed algorithm has faster convergence speed and more stable convergence process.

In DRL, parameters refer to various adjustable variables, which directly affect the structure, algorithm efficiency and training process of deep neural networks. The selection and adjustment of these parameters are very important for the performance, convergence speed, stability and final learning effect of the algorithm. Usually, a series of experiments are conducted to determine the best combination of parameters to achieve better performance and learning results. In this paper, through the adjustment of multiple rounds of training experiments, the experimental parameters used are determined, and the specific values are listed in Table 1. The setting of these parameters makes the algorithm converge rapidly in the training process, greatly shortens the training time, ensures the stability of the algorithm, and has superior generalization ability and strong adaptability, which can adapt to the simulation environment constructed in this paper.

D. TESTING AND RESULTS

After conducting 2000 episodes of training on DDPG, TD3, SAC, and the I-TD3, the policies are then evaluated in two experimental environments: Environment I and Environment II. In Environment I, where all obstacles are stationary, the algorithm's ability to sense the relative motion trend between the UAV and obstacles is evaluated. In Environment II, where all obstacles are dynamic, the UAV's real-time decision-making capability is tested, as the trained policies are fully



(a) E1 (b) E2 (c) E3 (d) E4

FIGURE 11. A typical case of I-TD3 algorithm successfully reaching the target area in Environment I.

TABLE 2. Test results under Environment I.

Env	DDPG				TD3				SAC				I-TD3(Ours)			
	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)
E1	-20.28	9.5	13.9	76.6	-10.05	6.45	11.75	81.8	-10.41	6.45	11.45	82.1	-11.89	6.85	**10.6**	**82.55**
E2	-16.09	7.4	20	72.6	-10.99	6.4	16.5	77.1	-10.76	6.45	15.25	78.3	-12.4	6.45	**13.65**	**79.9**
E3	-21.1	9.6	17.9	72.5	-12.38	6.55	20.05	73.4	-12.03	6.95	16.75	76.3	-13.6	6.75	**15.9**	**77.35**
E4	-21.27	9.45	22.35	68.2	-12.53	6.95	22	71.05	-11.52	6.45	17.6	75.95	-13.81	7	**16.45**	**76.55**

TABLE 3. Test results under Environment II.

Env	DDPG				TD3				SAC				I-TD3(Ours)			
	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)	AR	LR(%)	CR(%)	SR(%)
E5	-21.16	9.1	15.25	75.65	-11.02	6.85	15.25	77.9	-10.67	6.55	13.45	80	-11.67	6.75	**11.85**	**81.4**
E6	-18.43	7.55	18.25	74.2	-12.39	7.85	16	76.15	-10.99	6.45	15.65	77.9	-10.91	6.75	**13.35**	**79.9**
E7	-16.17	7.05	23.55	69.4	-13.36	7.1	19.85	73.05	-12.13	6.6	18.55	74.85	-16.24	6.75	**14.8**	**78.45**
E8	-19.26	7.35	24.9	67.75	-12.53	6.65	21.95	71.4	-10.82	6.65	19.7	73.65	-11.58	6.45	**15.3**	**78.25**

utilized. In addition, no random actions occur during the testing phase.

The evaluation metrics for assessing algorithm performance include Average Reward (AR), Loss Rate (LR), Collision Rate (CR), and Success Rate (SR). AR represents the average reward value over the entire testing period, indicating the average quality of the test; LR represents the percentage of rounds out of the 2000 episodes in which the UAV did not reach the target or collide with obstacles, while CR represents the percentage of collisions between the UAV and obstacles in 2000 episodes. SR indicates the percentage of successful target findings. The test results of DDPG, TD3, SAC and I-TD3 in Environment I are shown in Table 2.

We can see that the I-TD3 has the highest success rate in the four experimental environments of Environment I. With the increase of the complexity of the experimental environment, the advantages of the algorithm are more obvious. The number of obstacles gradually increases from E1 to E4, and the success rate decreases in turn. However, the success rate of the I-TD3 algorithm is the lowest (6%), while the SAC (6.15%), DDPG (8.4%) and TD3 (10.75%) are higher. Due to the limited training fragments and the randomness of the

environment, the success rate of the algorithm did not reach 100%, but still achieved good results. Finally, compared with TD3 and SAC algorithms, the I-TD3 has a higher success rate, but the average reward is lower. This is because in more cases, our algorithm makes the UAV neither find the target nor collide with obstacles, resulting in extremely low reward values. The results show that the I-TD3 is more adaptable to complex environments than DDPG, SAC and TD3.

Fig. 10 shows the success rate per 100 episodes in four static environments of Environment I. Fig. 11 displays a typical scenario in which the UAV, using the I-TD3 algorithm, successfully reaches the target point in Environment I.

Next, these algorithms will be tested in Environment II. The results of DDPG, TD3, SAC and I-TD3 in different dynamic environments are shown in the Table 3.

As evident from the table, the success rate of the four algorithms shows a decline as the presence of dynamic obstacles increases. While there may be slight deviations in selecting optimal behavior in a static environment, the impact is relatively minimal. However, in a dynamic environment, particularly when obstacles are moving at high speeds, it can lead to disastrous consequences. In these four algorithms, from environment E5 to E8, with the increase of dynamic obstacles, the success rate of DDPG decreased by 7.9%, SAC decreased by 6.35%, TD3 decreased by 6.5%, while the success rate of the I-TD3 decreased by 3.25%. In the table, we can see that the success rate of the algorithm in this paper is much higher than that of DDPG, TD3 and SAC in the environment E5 to E8, and the collision rate is also the lowest among the four algorithms. This is due to the proposed algorithm's ability to quickly perceive changes in the surrounding environment, enabling the UAV to avoid obstacles and make timely decisions. The results suggest that the I-TD3 demonstrates a high degree of adaptability in dynamic environments.

VI. CONCLUSION

This paper proposes a UAV path planning method based on DRL, which enables it to complete the path planning task autonomously in a multi-obstacle environment. We introduce priority experience playback as the experience replay pool of TD3 algorithm, which improves the utilization of sample data. At the same time, the average TD3 algorithm is proposed, which avoids the underestimation of Q value and improves the stability of the algorithm on the basis of solving the problem of overestimation of Q value. In addition, we design a new reward function so that the UAV can obtain reward feedback for each step of action, which solves the problem of reward sparseness in DRL. We tested the algorithm in a custom simulation environment. The results show that the algorithm can train the UAV to reach the target area safely and quickly in a multi-obstacle environment, and has good path planning performance. Compared with DDPG, TD3 and SAC, the proposed algorithm shows better stability and generalization ability in complex dynamic environments.

The experimental results of this algorithm are good, but there are still some shortcomings. When the simulation experiment is set up, the dynamic obstacles set up are moving at a uniform speed. In the actual flight environment, the obstacles encountered are not all moving at a uniform speed according to the prescribed line, but are random speeds and random routes. Therefore, the experimental environment closer to reality will be considered to test the performance of the I-TD3. Currently, this paper exclusively focuses on addressing the path planning issue for a single agent. Nonetheless, in our future research, we will delve into the path planning problem for multiple agents. Doing so will enable us to further explore and understand the collaborative path planning of UAV clusters, facilitating the achievement of more intricate tasks.

REFERENCES

[1] A. A. Saadi, A. Soukane, Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "UAV path planning using optimization approaches: A survey," *Arch. Comput. Methods Eng.*, vol. 29, no. 6, pp. 4233–4284, Apr. 2022.
[2] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based UAV path planning," *Knowl.-Based Syst.*, vol. 158, pp. 54–64, Oct. 2018.
[3] X. Yu and Y. Zhang, "Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects," *Prog. Aerosp. Sci.*, vol. 74, pp. 152–166, Apr. 2015.
[4] Z. Qadir, F. Ullah, H. S. Munawar, and F. Al-Turjman, "Addressing disasters in smart cities through UAVs path planning and 5G communications: A systematic review," *Comput. Commun.*, vol. 168, pp. 114–135, Feb. 2021.
[5] Z. Wu, S. Dong, M. Yuan, J. Cui, L. Zhao, and C. Tong, "Rotate artificial potential field algorithm toward 3D real-time path planning for unmanned aerial vehicle," *Proc. Inst. Mech. Eng., G, J. Aerosp. Eng.*, vol. 237, no. 4, pp. 940–955, Mar. 2023.
[6] H. Shen and P. Li, "Unmanned aerial vehicle (UAV) path planning based on improved pre-planning artificial potential field method," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 2727–2732.
[7] K. Yang, S. Keat Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Adv. Robot.*, vol. 27, no. 6, pp. 431–443, Apr. 2013.
[8] M. Kothari and I. Postlethwaite, "A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees," *J. Intell. Robotic Syst.*, vol. 71, no. 2, pp. 231–253, Aug. 2013.
[9] H. Liang, H. Bai, R. Sun, R. Sun, and C. Li, "Three-dimensional path planning based on DEM," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 5980–5987.
[10] L. Xia, X. Jun, C. Manyi, X. Ming, and W. Zhike, "Path planning for UAV based on improved heuristic A* algorithm," in *Proc. 9th Int. Conf. Electron. Meas. Instrum.*, Jul. 2009, pp. 3488–3493.
[11] J. Baek, S. I. Han, and Y. Han, "Energy-efficient UAV routing for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1741–1750, Feb. 2020.
[12] X. Feng and A. T. Murray, "Allocation using a heterogeneous space Voronoi diagram," *J. Geograph. Syst.*, vol. 20, no. 3, pp. 207–226, Jul. 2018.
[13] X. Chen and M. Zhao, "Collaborative path planning for multiple unmanned aerial vehicles to avoid sudden threats," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2019, pp. 2196–2201.
[14] W. Li, L. Wang, A. Zou, J. Cai, H. He, and T. Tan, "Path planning for UAV based on improved PRM," *Energies*, vol. 15, no. 19, p. 7267, Oct. 2022.
[15] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 132–141, Feb. 2013.
[16] Q. Geng and Z. Zhao, "A kind of route planning method for UAV based on improved PSO algorithm," in *Proc. 25th Chin. Control Decis. Conf. (CCDC)*, May 2013, pp. 2328–2331.

[17] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 2, pp. 511–526, Mar. 2012.
[18] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization," *Math. Problems Eng.*, vol. 2013, pp. 1–9, Aug. 2013.
[19] C. Qu, W. Gai, M. Zhong, and J. Zhang, "A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106099.
[20] X. Chai, Z. Zheng, J. Xiao, L. Yan, B. Qu, P. Wen, H. Wang, Y. Zhou, and H. Sun, "Multi-strategy fusion differential evolution algorithm for UAV path planning in complex environment," *Aerosp. Sci. Technol.*, vol. 121, Feb. 2022, Art. no. 107287.
[21] X. Han, J. Wang, J. Xue, and Q. Zhang, "Intelligent decision-making for 3-dimensional dynamic obstacle avoidance of UAV based on deep reinforcement learning," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2019, pp. 1–6.
[22] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments," *IEEE Access*, vol. 9, pp. 24884–24900, 2021.
[23] R. Wu, F. Gu, H.-L. Liu, and H. Shi, "UAV path planning based on multicritic-delayed deep deterministic policy gradient," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–12, Mar. 2022.
[24] Z. Hu, X. Gao, K. Wan, Y. Zhai, and Q. Wang, "Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments," *Chin. J. Aeronaut.*, vol. 34, no. 12, pp. 187–204, Dec. 2021.
[25] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.
[26] S. Zhang, Y. Li, and Q. Dong, "Autonomous navigation of UAV in multi-obstacle environments based on a deep reinforcement learning approach," *Appl. Soft Comput.*, vol. 115, Jan. 2022, Art. no. 108194.
[27] M. H. Lee and J. Moon, "Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor-critic with hindsight experience replay approach," *ICT Exp.*, vol. 9, no. 3, pp. 403–408, Jun. 2023.
[28] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments," *J. Intell. Robotic Syst.*, vol. 98, no. 2, pp. 297–309, May 2020.
[29] Y. Peng, Y. Liu, D. Li, and H. Zhang, "Deep reinforcement learning based freshness-aware path planning for UAV-assisted edge computing networks with device mobility," *Remote Sens.*, vol. 14, no. 16, p. 4016, Aug. 2022.
[30] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts, "Markov decision processes: A tool for sequential decision making under uncertainty," *Med. Decis. Making*, vol. 30, no. 4, pp. 474–483, Jul. 2010.
[31] T. Li, W. Liu, Z. Zeng, and N. N. Xiong, "DRLR: A deep-reinforcement-learning-based recruitment scheme for massive data collections in 6G-based IoT networks," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14595–14609, Aug. 2022.
[32] X. Guo and A. Piunovskiy, "Discounted continuous-time Markov decision processes with constraints: Unbounded transition and loss rates," *Math. Oper. Res.*, vol. 36, no. 1, pp. 105–132, Feb. 2011.
[33] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, Oct. 2021.
[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[35] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[36] M. Li, T. Huang, and W. Zhu, "Clustering experience replay for the effective exploitation in reinforcement learning," *Pattern Recognit.*, vol. 131, Nov. 2022, Art. no. 108875.
[37] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
[38] T. Li, D. Yang, and X. Xie, "Prioritized experience replay based reinforcement learning for adaptive tracking control of autonomous underwater vehicle," *Appl. Math. Comput.*, vol. 443, Apr. 2023, Art. no. 127734.
[39] Y. Zhang, X. Rao, C. Liu, X. Zhang, and Y. Zhou, "A cooperative EV charging scheduling strategy based on double deep Q-network and prioritized experience replay," *Eng. Appl. Artif. Intell.*, vol. 118, Feb. 2023, Art. no. 105642.
[40] M. Guanglei and P. Haibing, "The application of ultrasonic sensor in the obstacle avoidance of quad-rotor UAV," in *Proc. IEEE Chin. Guid., Navigat. Control Conf. (CGNCC)*, Aug. 2016, pp. 976–981.
[41] Z. Ma and S. M. Jiao, "Research on the attitude control of quad-rotor UAV based on active disturbance rejection control," in *Proc. 3rd IEEE Int. Conf. Control Sci. Syst. Eng. (ICCSSE)*, Aug. 2017, pp. 45–49.
[42] Y. Pan, Y. Yang, and W. Li, "A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV," *IEEE Access*, vol. 9, pp. 7994–8005, 2021.
[43] S. Lin, F. Li, X. Li, K. Jia, and X. Zhang, "Improved artificial bee colony algorithm based on multi-strategy synthesis for UAV path planning," *IEEE Access*, vol. 10, pp. 119269–119282, 2022.
[44] C. Banerjee, Z. Chen, and N. Noman, "Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3121–3129, May 2022.

XUQIONG LUO received the M.S. degree in computational mathematics from Guangxi University for Nationalities, in 2009, and the Ph.D. degree in computational mathematics from Nanjing Normal University, in 2013. He is currently an Associate Professor with Changsha University of Science and Technology. His research interests include numerical calculation of differential equations and multi-agent system control theory.

QIYUAN WANG received the B.S. degree in mathematics and statistics from Changsha University of Science and Technology, Changsha, China, in 2021, where she is currently pursuing the M.S. degree with the School of Mathematics and Statistics. Her research interests include UAV path planning and deep reinforcement learning.

HONGFANG GONG received the M.E. degree in computer application and the Ph.D. degree in computer science and technology from Hunan University, China, in 2004 and 2018, respectively. He is currently a Full Professor in mathematics and statistics with Changsha University of Science and Technology, Changsha, China. His research interests include machine learning, bigdata analysis, autonomous driving, and cyber-physical systems (CPS).

CHAO TANG received the B.S. degree in computer and communication process from Changsha University of Science and Technology, Changsha, China, in 2020. His research interests include recommendation systems and data mining.