Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9017079, 12 pages
https://doi.org/10.1155/2022/9017079

Research Article

UAV Path Planning Based on Multicritic-Delayed Deep Deterministic Policy Gradient

Runjia Wu,[1] Fangqing Gu,[1] Hai-lin Liu,[1] and Hongjian Shi[2]

[1]*School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou, China*
[2]*Beijing Normal University-Hong Kong, Baptist University United International College, Zhuhai, China*

Correspondence should be addressed to Fangqing Gu; fqgu@gdut.edu.cn

Received 3 January 2022; Accepted 17 February 2022; Published 14 March 2022

Academic Editor: Xingsi Xue

Deep deterministic policy gradient (DDPG) algorithm is a reinforcement learning method, which has been widely used in UAV path planning. However, the critic network of DDPG is frequently updated in the training process. It leads to an inevitable overestimation problem and increases the training computational complexity. Therefore, this paper presents a multicritic-delayed DDPG method for solving the UAV path planning. It uses multicritic networks and delayed learning methods to reduce the overestimation problem of DDPG and adds noise to improve the robustness in the real environment. Moreover, a UAV mission platform is built to train and evaluate the effectiveness and robustness of the proposed method. Simulation results show that the proposed algorithm has a higher convergence speed, a better convergence effect, and stability. It indicates that UAV can learn more knowledge from the complex environment.

1. Annotation Demo Section

In recent years, unmanned aerial vehicles (UAVs) have been widely applied, and their high maneuverability and rapidly deployable UAVs have been applied to search and rescue [1], multi-UAV cooperation [2], formation flight [3], remote surveillance [4], and other fields [5–7]. UAV faces a variety of complex challenges and complicated tasks. Among them, path planning is the first problem faced by UAV. How to make the UAV safely fly to the destination in an unknown working environment becomes a hot topic for researchers. Faced with complex and uncertain environments, many algorithms have been proposed for solving the UAV navigation problems. The most common method is the motion control problem in the unknown environment such as A-Star [8], artificial potential fields [9], rapidly exploring random tree (RRT) algorithm [10], and so on [11–15]. However, due to the constraints of model mismatch, insufficient measurement means, high accurate cost, and model migration, it is difficult to obtain an accurate dynamic model of aircraft in practical engineering. These model-based strategies can hardly be applied to practice in a complex uncertain environment.

In order to overcome the limitations of model-based strategies in uncertain environments, some researchers introduce learning-based methods to overcome these shortcomings. The first is supervised learning that uses large amounts of data to simulate real situations [16, 17]. However, supervised learning needs enough data; it cannot simulate a lot of changes in the real world [18]. The other is reinforcement learning (RL), which uses the interactive learning of the mapping from environment to behavior to seek the most accurate or optimal action decision by maximizing the state-value function and the action-value function [19]. Reinforcement learning has been applied in UAV path planning problem [20]. It transforms the UAV online path planning problem into a decision problem. The next time series of UAV actions are decided according to the current environment and its own state determined by sensors or external information. In the unknown complex environment, UAV has little prior knowledge related to the environment. Therefore, it is required to have strong

adaptive ability to such uncertainty. RL provides a better idea for this kind of problem by using historical data to obtain the nonlinear function relationship between approximate fitting state and overall performance [21–24].

Reinforcement learning has been extensively studied in recent years. For example, DeepMind innovatively proposed a deep reinforcement learning (DRL) through the combination of deep learning (DL) and RL. DRL transforms high-dimensional input into a lower-dimensional state. It achieved a promising result. Currently, Mnih et al. [25] proposed a deep Q-network (DQN) algorithm, which utilized the powerful function fitting ability of deep neural network to avoid the huge storage space of Q table. DQN enhances the stability of training process by using experiential replay memory and target network. Double DQN [26] and dueling DQN [27] are proposed gradually along with DRL research to overcome the defect of overestimation. DQN algorithm has achieved great success in discrete space. However, in high-dimensional continuous space, DQN will increase exponentially with the increase of discrete degree of action segmentation, resulting in training difficulties. Actor-critic (A-C) algorithm [28] adopts a method similar to policy gradient and uses actor network to output the probability value of the action, while critic network is responsible for evaluating the output action. A-C algorithm uses critic network approximate value function to guide agent update and provide low variance learning knowledge [29, 30]. Lillicrap et al. [31] proposed a deep deterministic policy gradient (DDPG) algorithm to improve the stability of A-C algorithm evaluation by using the target network and empirical replay of DQN. DDPG can be applied in the applications with continuous action space and achieve great success [32]. However, the performance of DDPG in practical applications is not very stable.

Reinforcement learning is independent of environmental models and prior knowledge. Thus, it can effectively solve the UAV path planning problem in unknown environments. The research of reinforcement learning in UAV path planning has received extensive attention from scholars. UAV navigation was modeled as a reinforcement learning problem and validated autonomous fight in unknown environments in [33]. Junell et al. [34] used reinforcement learning method to solve the flight test of quadrotor aircraft in an unknown environment. The continuity of DDPG is widely used in path planning, but its convergence is often unstable in complex environments. Model-free reinforcement learning algorithms are based on time division or Monte Carlo [35]. It suffers from the problem of overestimation. In large state space, the application of policy gradient method will bring a high variance of estimation results. It makes policy learning more sensitive and even leads to training failure.

Numerous researchers improved the accuracy of numerical estimation by improving the neural network. For example, double DQN [26] is guaranteed not to overestimate Q value via two critic networks. Twin-delayed DDPG (TD3) [36, 37] algorithm solves the overestimation problem by introducing three key technologies. In a real UAV flight, paper [38] solved the problem of slow convergence caused by sparse rewards by introducing the reward function of an artificial potential field. Paper [39] started with DDPG experience base combined with simulated annealing algorithm, and accelerated the learning process of DRL through multiexperience pool (MEP). Papers [38, 40] use LSTM to approximate the critic network by combining the current training observation sequence with the historical observation sequence, so that the UAV can break away from the U-shaped obstacle in large path planning. Paper [41] presented three improvements, environmental noise, delay learning, and hybrid exploration techniques, to enhance the robustness of DDPG. Nevertheless, robustness is still a great challenge for UAV path planning. In the TD3 algorithm, the algorithm solves the critic's overestimation problem by the method of clipped double Q-learning for actor-critic. However, only using low estimation often leads to slow convergence.

In order to solve the problem that actor network relies heavily on critic network, which makes DDPG performance very sensitive to critic learning, this paper proposes a multicritic-delayed DDPG method for solving UAV path planning. It uses the average estimation of multicritics network to reduce DDPG's dependence on critic network and delayed learning method to reduce the overestimation problem of DDPG and reduce the error accumulation of the target network. Considering the sensitivity of the UAV to parameters in the real environment, adding Gaussian noise to action and state increases the robustness of the UAV. The main contributions of this paper are as follows:

(1) We propose a multicritic-delayed DDPG method, which includes two improvement techniques. The first is to add state noise and regularize it, which increases the robustness training network. The second is to use multicritic to average error and solve the error accumulation caused by overestimation

(2) We apply the proposed multicritic-delayed deep deterministic policy gradient method for solving UAV path planning. A nonsparse reward mode is designed

(3) A UAV mission platform is built to train and evaluate the effectiveness and robustness of the proposed method. Simulation results show that the proposed algorithm is effective with strong robust and adaptive capability for solving the path planning of the UAV flying destination under complex environment

The remainder of this paper is organized as follows: In Section 2, we briefly review reinforcement learning methods, i.e., DDPG, DT3, and MCDDPG, for solving UAV path planning. Section 3 gives a detailed description of the proposed multicritic-delayed deep deterministic policy gradient method. Section 4 provides the simulation results and analyses the empirical results. Finally, we draw a conclusion and future work in Section 5.

2. Reinforcement Learning for Solving UAV Path Planning

2.1. UAV Motion Model. Motion model of UAV is a basis of path planning problem. The UAV system is usually controlled by six degrees of freedom, representing three

coordinates of the UAV position $[x, y, z]$ and controlling the three freedoms of the yaw angle ψ, the roll angle δ, and the pitch angle ϕ. Six degrees of freedom kinematics mode $[x, y, z, \psi, \delta, \phi]$ is used to describe the internal state of the UAV.

For the sake of brevity and without loss of generality, we adopt the kinematic model of three degrees of freedom instead of six degrees of freedom. Assume that the UAV is fixed at a horizontal altitude, so that UAV's activity is confined to the $x-y$ plane. Ignoring the momentum impact of the UAV during fight, assuming that the UAV adopts a constant velocity v, the vector $\zeta = [x, y, \psi]$ is used to simplify the description of the position and motion of the UAV. Therefore, the vector ζ can be expressed as:

$$\begin{cases} x_{t+1} &= x_t + v_t \times \cos \psi_t, \\ y_{t+1} &= y_t + v_t \times \sin \psi_t, \\ \psi_{t+1} &= \psi_t + \Delta\psi. \end{cases} \tag{1}$$

where Δv is the change in velocity and $\Delta\psi$ is the change in yaw angle.

The state obtained by the environment of UAV is composed of three parts, i.e., the internal state, the interactions with the environment, and the location of the target. There are six coordinates $\xi_u = [x, y, x_v, y_v, v, \psi]$ representing the information about the their internal state, where (x, y) is the absolute position of the UAV, (x_v, y_v) is the velocity component of the corresponding coordinate, v is the speed of flight, and ψ is the yaw angle. $(v, \psi)(v, \psi)$is used to control the internal information of the UAV flight. The surrounding environment state is determined by radar, range finder, and other tools in the real-time interaction between UAV and environment. In this paper, we use range finders to receive the environment state $\xi_f = [d_0, \cdots, d_N]$, where d_i is the ith range finder mounted on the UAV as shown in Figure 1(b). Besides, the target position of the UAV is expressed as $\xi_T = [x_T, y_T]$ as shown in Figure 1(a). Through the combination of the three observation methods, we can obtain the final description of the state s by combining ξ_u, ξ_f, and ξ_T.

$$\mathbf{s} = [x, y, x_v, y_v, v, \psi, d_0, \cdots, d_N, x_T, y_T]. \tag{2}$$

The control of UAV is complicated in the actual situation, which requires multiple commands to achieve the motion of the UAV. In this paper, we appropriately selected the UAV's speed and roll as the motion control commands. The control vector of UAV is $\mathbf{a} = [a_v, a_\psi]$, where $a_v \in [-1, 1]$ denotes the ratio of the current speed to the maximum speed and $a_\psi \in [-1, 1]$ is a steering signal that can be selected to turn the UAV to the desired roll angle.

2.2. Reinforcement Learning. In reinforcement learning, the agent changes its state through interaction with the environment so as to obtain returns and achieve the optimal strategy. The model is usually expressed using five tuples S, A, P, R, γ of a Markov decision process (MDP), where S is a collection of environmental state descriptions of $\mathbf{s}$ and A is a set of all possible actions $\mathbf{a}$. $P : S \times A \times S \longrightarrow [0, 1]$ represents the transition probability of taking an action from S to the next S. $R = S \times A$ represents the immediate reward after the agent takes action. Reinforcement learning is designed to maximize future rewards, and a set of rewards can be expressed as $R_t^\gamma = \sum_{i=t}^{T} \gamma^{i-t} r(\mathbf{s}_i, \mathbf{a}_i)$. Based on the reward, reinforcement learning introduces two functions, the state-valued function when an agent adopts the policy π:

$$V_\pi(\mathbf{s}_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l}) | \mathbf{s}_t\right], \tag{3}$$

where π is to map system states to a probability distribution over the actions.

And the action value function:

$$Q_\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r(\mathbf{s}_{t+l}, \mathbf{a}_{t+l}) | \mathbf{s}_t, \mathbf{a}_t\right], \tag{4}$$

where $\gamma \in [0, 1]$ is the discounting factor which represents the difference in importance between future rewards and present rewards.

The value functions are used to measure the advantages and disadvantages of a certain state or action state, that is, whether it is worth an agent to select a certain state or execute an action in a certain state. Figure 2 illustrates the control of the agent under reinforcement learning model.

2.3. Nonsparse Reward Model. The reward function of traditional RL uses a simple sparse reward model, that is, an agent gets the reward only when the agent reaches the destination. This paper utilizes a nonsparse reward method to provide guidance for model learning. Obviously, nonsparse rewards provide more navigation domain knowledge than sparse rewards and do not change the policy invariance of rewards.

The nonsparse reward consists of four constructions:

$$r(\mathbf{s}, \mathbf{a}) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 + \lambda_4 r_4, \tag{5}$$

where λ_1, λ_2, λ_3, and λ_4 are the contribution rates of the four items, and

$$\begin{aligned} r_1 &= d_{\text{pre}} - d_{\text{cur}}, \\ r_2 &= -r_{\text{step}}, \\ r_3 &= -\Delta\psi + r_{\text{free}}, \\ r_4 &= -e^{-\omega d_{\text{obs}}}, \end{aligned} \tag{6}$$

where r_1 represents the change in distance between the current position and the destination and d_{pre} and d_{cur} are the previous and current relative distance between UAV and the target. When $d_{\text{pre}} > d_{\text{cur}}$, r_1 is a reward that is related to speed, guiding the UAV to its destination quickly. r_2 is a constant penalty advance to the UAV reaching its destination with a minimum number of steps. The UAV should

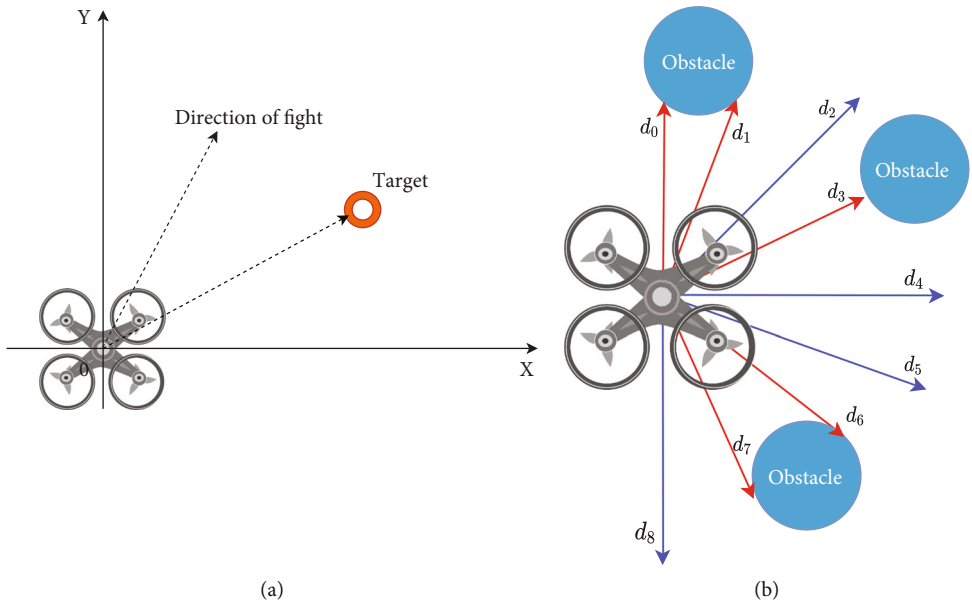


Figure 1: (a) The direction of the UAV on the coordinate axis. (b) The sensor of the UAV.

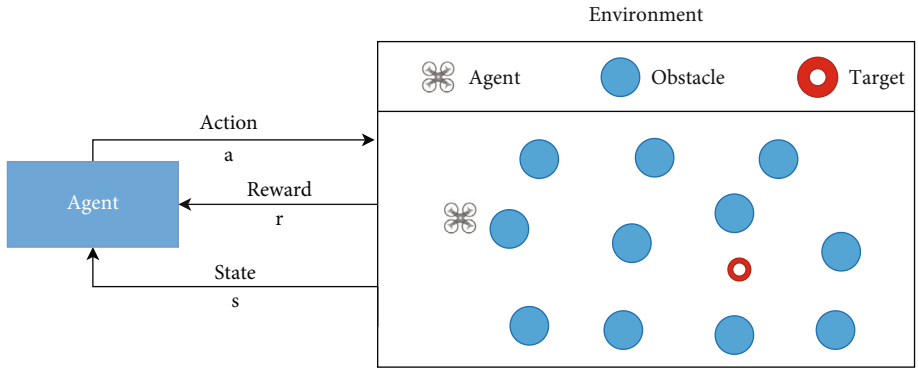


Figure 2: Control of agent under reinforcement learning model.

be encouraged to complete its missions as quickly as possible and punished after each transition. r_{free} represents a reward for flying without obstacles. Encourage UAV to fly to accessible places to explore more space. r_3 encourages UAV to shorten its range but at the same time ensure it can explore more space. The UAV should be able to fly toward its targets as soon as possible with penalties for deviations, but it should be encouraged to move towards free space if there are obstacles in the direction of flight. r_4 prevents the UAV from getting too close to the obstacle, and d_{obs} is the minimum distance between the UAV and the obstacle, and ω is a constant which is to control the size of the distance. Exponential function is used to prevent UAV from getting too close to obstacles, but it can also fly near obstacles. The UAV should actively avoid obstacles. If it gets close to obstacles, it will be punished greatly, so as to ensure that the UAV can stay away from obstacles.

2.4. Deep Deterministic Policy Gradient. Deep deterministic policy gradient (DDPG) adopts the network framework of actor-critic reinforcement learning, the critic can judge the value of the action based on the actor, and the actor can modify the probability of the action based on the value of the critic. The convolution neural networks Q network and μ network of DDPG method are used to approximate the state-action value function (3) and state-value function (4), respectively. θ^Q and θ^μ are the parameters of the Q network and μ network. The critic network learns the state-action value by minimizing time-difference (TD) errors:

$$L = \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q'\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}\middle|\theta^{Q'}\right) - Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\middle|\theta^{Q}\right)\right)^2. \quad (7)$$

In addition, deep Q-learning target network is used to remove the coupling during the update of formula (7). Figure 3 illustrates the DDPG motion control framework, where Q' represents the target critic network and μ' represents the target actor network.

Obviously, we know that the samples obtained by the agent in RL are highly correlated. The researcher uses the reply buffer to address this problem. The correlation of samples is broken by storing experiences $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$, and

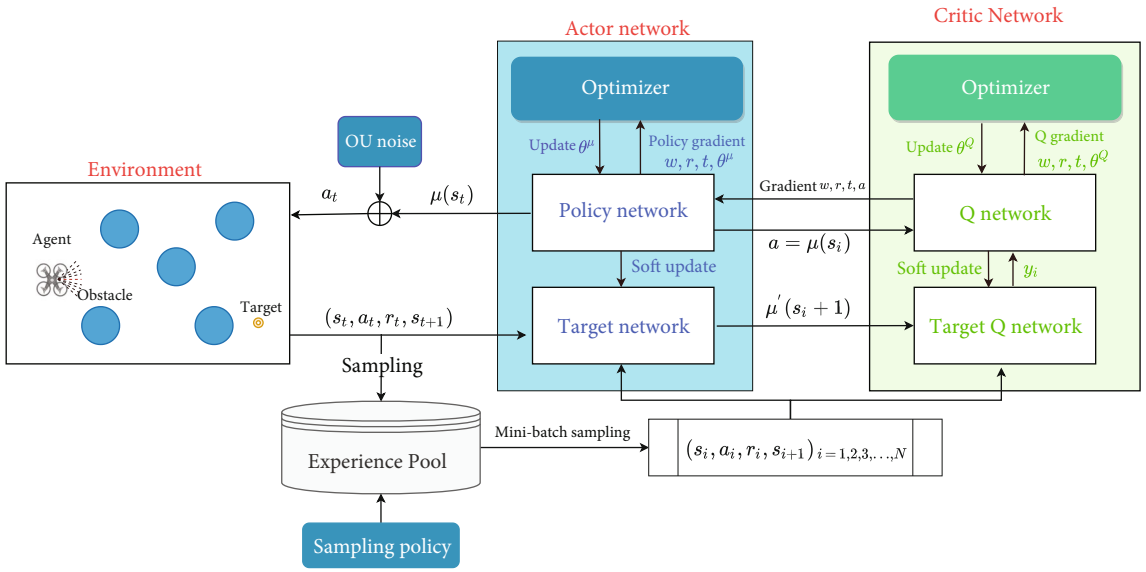


Figure 3: DDPG motion control framework.

then, random samples are taken from experience reply when the network trains. DDPG is derived from the deterministic policy gradient theorem for MDP. In this theorem, for MDP with continuous action space, the deterministic policy gradient exists. When the variance of probability policy approaches zero, it is deterministic action $\mathbf{a}_t = \mu(\boldsymbol{s}_t|\theta^{\mu})$, i.e.,

$$\nabla_{\theta^{\mu}} J(\theta^{\mu}) = \mathbb{E}_{S\sim\rho^{\pi}}\left[\nabla_{\theta^{\mu}}\mu(\boldsymbol{s}|\theta^{\mu})\nabla_{\mathbf{a}}Q\left(\boldsymbol{s}, \boldsymbol{a}\middle|\theta^{Q}\right)\middle|_{\mathbf{a}=\mu_{\theta}(\mathbf{s})}\right], \quad (8)$$

where ρ^{π} denotes the state distribution under a selected policy π. The actor network guides the choice of actions by maximizing performance objectives (8). DDPG trains the network with the stochastic gradient descent (SGD) algorithm with minibatch and then updates the target network with the soft update algorithm:

$$\begin{cases} \theta^{Q'} & \longleftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \\ \theta^{\mu'} & \longleftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}, \end{cases} \quad (9)$$

where τ is the updating rate.

2.5. Twin-Delayed Deep Deterministic Policy Gradient. Twin-delayed deep deterministic policy gradient (TD3) adopts an improved clipped variant of double Q-learning to reduce network overestimation problems. Following the idea of double Q-learning, TD3 uses two critics with the same pool of experience. Algorithm 1 contains the pseudocode of TD3. The minimum values of the two networks are used to update the critic networks:

$$L_{cd} = \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \min_{i=1,2} Q'\left(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}\middle|\theta_i^{Q'}\right) - Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\middle|\theta_i^{Q}\right)\right)^2, \quad (10)$$

where θ_1^Q and θ_2^Q are the parameters of two different critic networks and $\theta_1^{Q'}$ and $\theta_2^{Q'}$ are the parameters of the corresponding target networks, respectively. The action network is updated only according to θ_1^Q. Both θ_1^Q and θ_2^Q are updated by using Equation (10) to minimize the loss function. When the deterministic policy is updated, the value estimate of narrow peaks will occur. TD3 smooths the small area around the action and adds noise to the Q value of the target action:

$$\begin{aligned} \mathbf{a}_{t+1} &= \mu\left(\boldsymbol{s}_{t+1}\middle|\theta^{\mu'}\right) + \varepsilon, \\ \varepsilon &\sim \mathrm{clip}(\mathcal{N}(0,\sigma), -c, c), \end{aligned} \quad (11)$$

where $\mathcal{N}$ is a Gaussian distribution with mean 0 and variance σ. Noise ε can be seen as a regularization method, which makes value function updates smoother. c is for controlling the size of noise.

Deterministic policy is prone to errors caused by function approximation errors and increases the variance of the target. The regularization of Equation (11) can smooth the target policy. On the other hand, TD3 uses a stable target network approach to reduce error accumulation. The change is to update only the policy and target network after updating the target critic at a fixed frequency d. The method of delayed updating the target network can interrupt the accumulation of errors and ensure the TD error is small so as to slowly update the target network $\theta_i^{Q'} \longleftarrow \tau\theta_i^{Q} + (1-\tau)\theta_i^{Q'}$, $i = 1, 2$.

2.6. MCDDPG. Although DDPG is widely used in UAV path planning because it can well solve the continuous motion space, a large number of researchers have improved DDPG in UAV application due to the poor stability of the algorithm, and the convergence rate is slow. Because actor's learning ability depends on the judgment, paper [42] proposed a multicritic deep deterministic policy gradient (MCDDPG) to

Initialize the critic networks Q_1, Q_2 and actor network μ with parameters θ_1^Q, θ_2^Q, and θ^μ, separately.
Initialize the target critic networks $\theta_1^{Q'} \longleftarrow \theta_1^Q$, $\theta_2^{Q'} \longleftarrow \theta_2^Q$ and actor target network $\theta^{\mu'} \longleftarrow \theta^\mu$, separately.
Initialize the reply buffer R, maximum flight time T.
forepisode = 1 : M**do.**
◇ Reset environment and receive initial observation state $\mathbf{s}$.
for$t$ = 1 : T**do.**
◇ Select action $\mathbf{a}_t = \mu(\boldsymbol{s}_t|\theta^\mu) + \mathcal{N}(0, \sigma)$ and obtain the reward r_t and new state $\mathbf{s}_{t+1}$.
◇ Store transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in R.
◇ Sample a random minibatch of N transitions $(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_{i+1})$ from R.
◇$\mathbf{a}_{t+1} \longleftarrow \mu(\boldsymbol{s}_{t+1}|\theta^{\mu'}) + \varepsilon, \varepsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$.
◇ Update the critic networks by minimizing loss function in Equation (10).
if$t$ mod d**then**
◇ Update the actor network with policy gradient Equation (8).
◇ Update the parameters of target networks with updating rate τ.
end
end
end

Algorithm 1: TD3 method.

◇ Initialize the K critic networks Q_i and actor network μ with parameters θ_i^Q and θ^μ, separately.
◇ Initialize the K target critic networks Q_i' and actor target network μ' with parameters $\theta_i^{Q'} \longleftarrow \theta_i^Q$ and $\theta^{\mu'} \longleftarrow \theta^\mu$, separately.
◇ Initialize the reply buffer R, maximum flight time T, parameters α, β, η, updating rate τ.
forepisode = 1 : M**do**
◇ Reset environment and receive initial observation state $\mathbf{s}$.
for$t$ = 1 : T**do**
◇ Select action $\mathbf{a}_t = \mu(\boldsymbol{s}_t|\theta^\mu) + \mathcal{N}_t(0, \sigma)$ according to the current policy and exploration noise.
◇ Obtain the reward r_t and observe new state $\mathbf{s}_{t+1}$.
◇ Store transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in R.
◇ Sample a random minibatch of N transitions $(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_{i+1})$ from R.
◇ Update the K critic networks by minimizing loss function in Equation (14).
◇ Update the actor network with policy gradient Equation (8).
◇ Update the parameters of target networks with updating rate τ.
end
end

Algorithm 2: MCDDPG method.

overcome the sensitivity of training the critic network. MCDDPG is applied for solving UAV path planning in [20]. Algorithm 2 contains the pseudocode of MCDDPG. Specifically, it uses the average of the value of K critics to approximate instead of the action-value function.

$$Q_{\text{avg}}\left(\boldsymbol{s}, \boldsymbol{a}\middle|\boldsymbol{\theta}^{\boldsymbol{Q}}\right) = \frac{1}{K}\sum_{i=1}^{K} Q_i\left(\boldsymbol{s}, \boldsymbol{a}\middle|\boldsymbol{\theta}_{\boldsymbol{i}}^{\boldsymbol{Q}}\right), \tag{12}$$

where $\theta_i^Q \in \theta^Q$ is the parameter of the ith critic. The average of all critics can diminish the impact of the overestimation problem caused by the individual critic. We further rewrote the TD error according to Equation (7). Thus, the average TD error is:

$$L_{\text{avg}} = \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q'_{\text{avg}}\left(\boldsymbol{s}_{\boldsymbol{t+1}}, \boldsymbol{a}_{\boldsymbol{t+1}}\middle|\boldsymbol{\theta}^{\boldsymbol{Q}'}\right) - Q_{\text{avg}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\middle|\theta^Q\right)\right)^2, \tag{13}$$

where the Q'_{avg} is the average of the target critic networks. Using the same TD error update can cause critics to lose diversity, while a separate update can make a big difference. Therefore, different from DDPG critic network, the local error and global error must be considered when calculating the loss of critic network. According to Equation (13), the loss function for ith critic is defined as:

$$L_{\text{mc}} = \alpha L_{\text{avg}} + \beta L + \eta\left(Q_i(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta_i) - Q_{\text{avg}}(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta)\right), \tag{14}$$

where α, β, and η represent the weighting factor. The values of α, β, and η are between 0 and 1. And the sum of α and β is 1. When $K = 1$, L_{mc} should be the same as formula (7).

3. Multicritic-Delayed DDPG Method

The delayed updating of the critic network is very important in practical applications. The critic network of the traditional DDPG method is frequently updated in the training

Initialize the K critic networks Q_i and actor network μ with parameters θ_i^Q and θ^μ, separately.
Initialize the K target critic networks Q_i' and actor target network μ' with parameters $\theta_i^{Q'} \longleftarrow \theta_i^Q$ and $\theta^{\mu'} \longleftarrow \theta^\mu$, separately.
Initialize the reply buffer R, maximum flight time T, parameters α, β, η, updating rate τ.
forepisode = 1 : M**do**
Reset environment and receive initial observation state **s**.
for$t = 1 : T$**do**
Select action $\mathbf{a}_t = \mu(\boldsymbol{s}_t|\theta^\mu) + \mathcal{N}_t(0, \sigma)$ according to the current policy and exploration noise.
ifepisode > T_m**then**
$\mathbf{a}_t = \mu(\boldsymbol{s}_{\text{noise}}|\theta^\mu)$ according to Equation (15).
end
Obtain the reward r_t and observe new state $\mathbf{s}_{t+1}$, and Store transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in R.
Sample a random minibatch of N transitions $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}_{i+1})$ from R.
Update the K critic networks by minimizing loss function in Equation (14).
if$t$ mod d**then**
Update the actor network with policy gradient Equation (8).
Update the parameters of target networks with updating rate τ.
end
end
end

Algorithm 3: MCD method.

process. It may lead to increase the training steps and cause overestimation. If overestimated actions occur in the learning process, agents will get lost in the learning process, leading to training failure. From the perspective of UAV, delayed update strategy is equivalent to global guidance of UAV flight, while traditional DDPG guides UAV flight through small correction. The existing studies show that the delayed update strategy is more in line with actual UAV flight guidance. Thus, this paper proposes a multicritic-delayed DDPG method, named, MCD, for solving the UAV path planning problem. Algorithm 3 contains the pseudocode of MCD. It uses the delayed update strategy to improve the robustness of the algorithm. TD3 using clipped double Q network can effectively solve the overestimation problem caused by neural network, it also leads to underestimation at the same time. We adopt the error updating method formula (14) of multicritic to approximate the Q value of critic network in the proposed MCD.

MCD prevents network underestimation by retaining the global mean error of multicritic networks and preserving at the same time the error between the average and the individual guarantees diversity. Another improvement is to add noise to the state. DDPG increases agent's exploration of the environment by adding an Ornstein-Uhlenbeck (OU) noise. In fact, the acquisition of the real environment by UAV is often inaccurate. If the UAV is overly dependent on the training model, the deviation of state will often cause the UAV to crash. We simulate the deviation of the environmental state input in the real situation by adding a Gaussian noise:

$$\mathbf{s}_{\text{noise}} = \mathbf{s} + \mathcal{N}(0, \sigma_{\mathbf{s}}). \tag{15}$$

These noises obey the standard deviation σ_s of Gaussian distribution. When the model is trained to T_m, the robustness of the state noise training network is introduced. Introducing noise after the network has been trained to a certain extent can ensure that the initial network will not be trained to fail by noise.

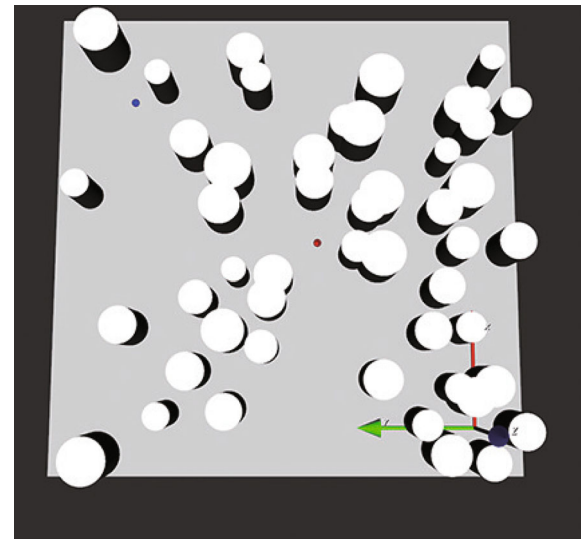


Figure 4: UAV's obstructed environment.

4. Experiments

In this section, in order to verify the performance of the proposed MCD, we compare it with three algorithms, i.e., DDPG, TD3, and MCDDPG, on a synthetic test problem.

4.1. Experimental Platform Setting. We built a random environment of different complexity. The terrain of each environment is a rectangular area of 1000×1000. As shown in Figure 4. The simulation environment is randomly generated by 49 cylindrical obstacles with diameters of $(30, 60)$ in the rectangular region (there may be overlapping of cylindrical obstacles). The UAV is fixed at horizontal altitude of

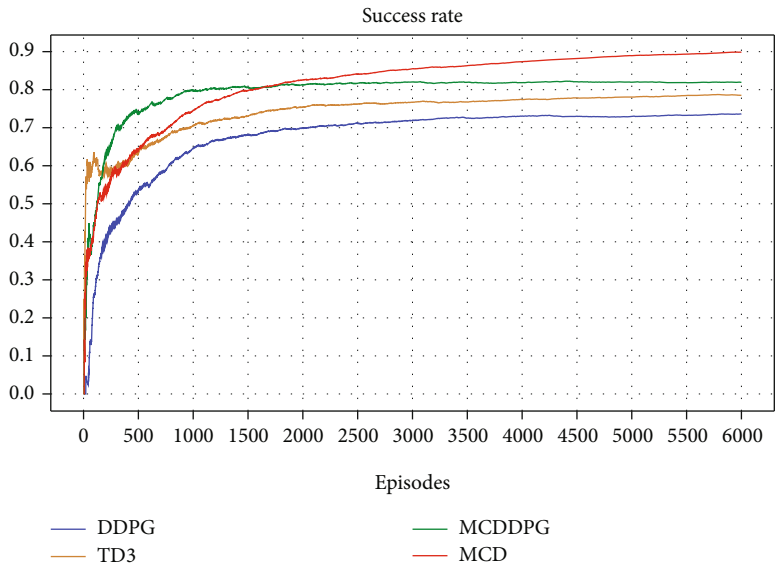
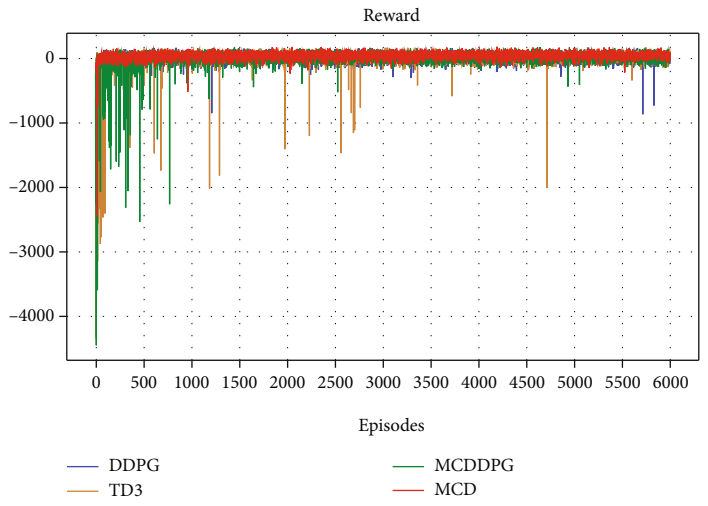
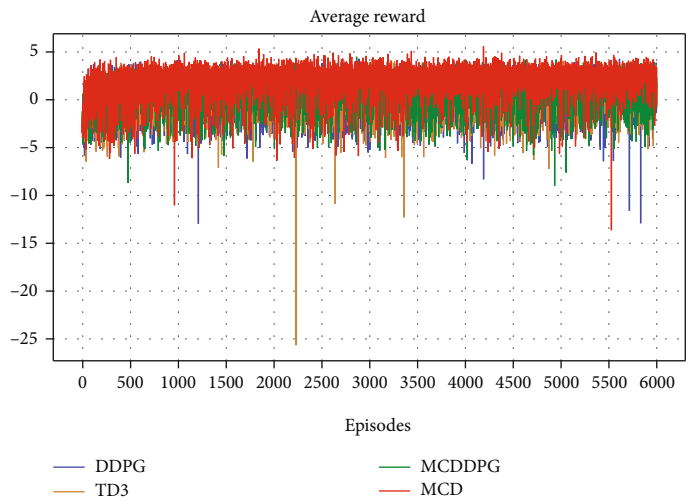


Figure 5: The success rate of reaching the target in training.



(a)

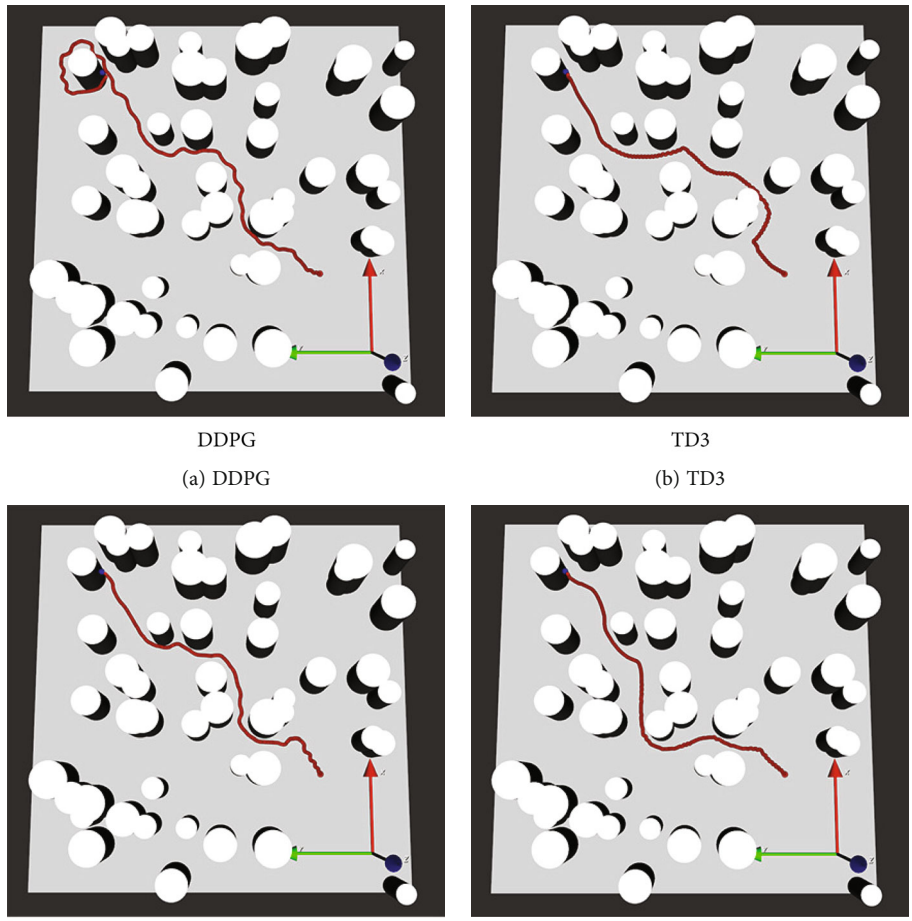


(b)

Figure 6: The 3000 set of the training.

Table 1: The result of algorithms.

	Learning stage			Exploiting stage		
	Success	Collision	Loss	Success	Collision	Loss
DDPG	73.6%	19.3%	7.1%	80.5%	10.1%	9.4%
TD3	78.5%	17.1%	4.4%	88.4%	5.6%	6.0%
MCDDPG	81.9%	15.8%	2.3%	92.1%	3.4%	4.5%
MCD	89.8%	10.1%	0.1%	94.3%	1.9%	3.8%



(a) DDPG (b) TD3

(c) MCDDPG (d) MCD

Figure 7: Performance of four algorithms in the same environment.

100 meters and equipped with nine sensors with a detection range of 100 meters. The UAV flies from its initial position to its designated target. The maximum speed of the UAV is limited to $a_{v,\max} = 45$ m/s, and the maximum yaw is $a_{\psi,\max} = \pi/2$.

The critic networks adopt the same network structure of $19 \times 200 \times 300 \times 1$. The observed states as inputs are normalized to 19 dimensions, and the actor network composed of $19 \times 200 \times 300 \times 2$ uses 2 dimensions output action to control the UAV. The parameters α, β, η are set to 0.8, 0.2, 0.1 in MCDDPG and MCD. The number K of the critic networks in Equation (12) is set to be $K = 3$. When K is set too high, the overestimation ability of the algorithm will be greatly reduced but the operation efficiency will be too low.

Table 2: The length and the steps of the flight paths obtained by the compared algorithms.

	Path length	Step
DDPG	1432.2 m	112
TD3	78.5 m	242
MCDDPG	81.9 m	124
MCD	89.8 m	153

According to paper [42], $K = 3$ is a more appropriate value. UAV observation and action are normalized to $[-1, 1]$. Adam optimizer [43] is used to learn network parameters. The learning rates of the actor and critic networks are set

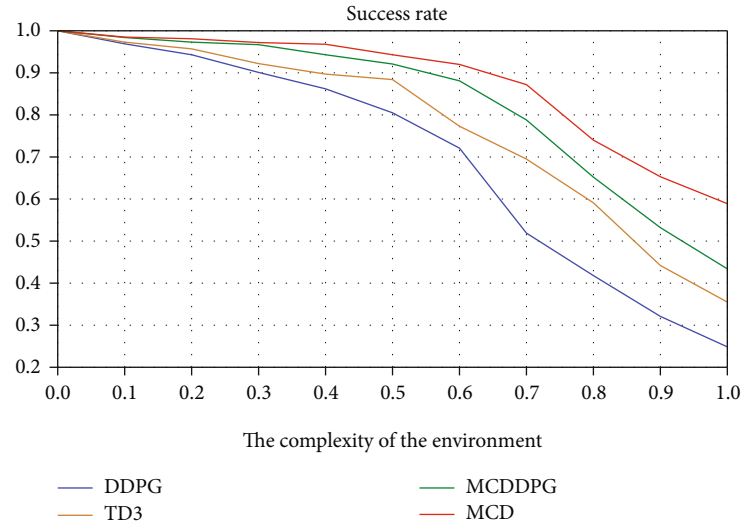


Figure 8: Success rates in different environmental complexities under 2000 sets of training.

to be 10^{-3}. In addition, the discount factor is $\gamma = 0.99$, and the soft update rate is $\tau = 0.001$. The other hyperparameters are given as follows: minibatch size $N = 64$, experience reply $R = 10000$. In addition, Gaussian distribution $\mathcal{N}(0, 0.25)$ is used to increase motion detection space, and the Gaussian distribution deviation σ_s of the state noise is 0.2. When the training number reaches $T_m = 4000$, the state noise will be enabled. In TD3, the action adds smoothing $\mathcal{N}(0, 0.2)$ and is clipped to $(-0.5, 0.5)$. The parameter of reward is set to $\lambda_1 = 1.2, \lambda_2 = 1.5, \lambda_3 = 32$, and $\lambda_4 = 25$, and the maximum value of iteration is $T = 1000$.

4.2. Performance of Multicritic Delayed. As shown in Figure 5, 3000 sets were used to train both models. MCD has a success rate of 89.8%, which is significantly better than that of the compared algorithms. In the training of MCD, the training effect is not as good as that of TD3 and MCDDPG in the early, which is due to the minimization of multicritic networks. However, it ensures that the MCD estimation is not too high and can grow steadily, and it obviously exceeds other algorithms in the later period. For more specific verification, we trained the model three times, intercepted 3000 episodes in 6000 sets and averaged them, and selected the number of the average reward per episode and the total reward per episode, as shown in Figure 6. From this figure, we can see that MCD fluctuates much less and rewards better than the other three algorithms.

In order to prove the generalization ability of the algorithms, we further calculated the success rate, collision rate, and loss rate of agents. Table 1 lists the results obtained by the compared algorithms. The results showed that the generalization of MCD is better than that of TD3. The success rate is 94.3% which is more than 88.4% of TD3. It is proved that the algorithm can effectively improve the success rate and obtain more environmental information by using the average critic network. We can note that the loss rate of model exploiting is greatly improved compared with the mode training. The model training can avoid the collision of UAV with obstacles, which is useful in practical situations.

4.3. Testing of Different Algorithms. In this part, we will use the actor network completed in the training in the last section to observe the influence of different algorithms on UAV flight. We loaded the actor network with DDPG, TD3, MCDDPG, and MCD algorithms training onto the UAV to guide the UAV flight. From the same starting position $(230, 277)$, the UAV flies through 49 cylindrical obstacles composed of radius $(30, 60)$ and reaches the target position $(820, 809)$. Figure 7 plots the flight paths obtained by the algorithms. Table 2 lists the length and the steps of the flight paths.

From this figure, we can see that UAV moves fast in a way that is close to the obstacle obtained by DDPG algorithm. UAV approaches the obstacle with fewer steps and faster speed, but at the same time, makes it easier to hit the obstacle obtained by DDPG. And because the target point is near the obstacle, DDPG tends to avoid the obstacle when planning the path, resulting in loss in the environment far from the obstacle. UAV flies in a safer manner and can correct the course if the target navigation goes wrong at a small cost obtained MCDDPG. TD3's path planning is more conservative, but it sacrifices time for longer path planning. During the first half of the flight, hesitancy resulted in a zigzag flight path. Multicritic delay absorbs the advantages of the above three algorithms to reach the destination with the shortest path and takes less turns than TD3, which is not far from the other two algorithms. Obviously, we can get that multicritic delay is superior to other algorithms because it enables the UAV to complete the task with minimal path cost and lower time.

4.4. Testing of Complex Environment. To further verify the robustness of MCD in more complex environments, we set up a more complex environmental threat test to investigate the robustness of MCD. We set up a series of environments

with different numbers of obstacles. As shown in Figure 4, it represents a complex environment with a density of 0.5. Density 1 represents 100 obstacles, and 10 obstacles are reduced for every decrease of 0.1. We repeated 2,000 episodes of the four algorithms in the same obstacle environment, redeploying drones and targets in each episode. The success rate for 2000 sets is shown in Figure 8. Obviously, as the number of obstacles rose, the success rate of all four algorithms began to decline. However, MCD algorithm declined the most slowly and still maintained a success rate of 58.9% in the most complex environment. The other three algorithms MCDDPG, TD3, and DDPG are reduced to 43.4%, 35.5%, and 24.8%, respectively. Therefore, MCD is highly adaptable to complex environments.

5. Conclusion

In this paper, we proposed a reinforcement learning method, named, MCD, for solving the UAV path planning problem under a complex environment. It uses multicritic networks and delayed learning methods to reduce the overestimation problem of DDPG and adds noise to improve the robustness in the real environment. Moreover, a UAV mission platform is built to train and evaluate the effectiveness and robustness of the proposed method. Simulation results show that the proposed algorithm is superior to the traditional DDPG in path planning. However, some issues remain to be resolved, such as MCD hyperparameter settings, improvements to nonsparse rewards, and experience reply settings.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

We declare that there is no conflict of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (62172110), in part by the Natural Science Foundation of Guangdong Province (2021A1515011839), and in part by the Programme of Science and Technology of Guangdong Province (2021A0505110004 and 2020A0505100056).

References

[1] T. Tomic, K. Schmid, P. Lutz et al., "Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[2] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, "UAV air combat autonomous maneuver decision based on DDPG algorithm," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pp. 37–42, Edinburgh, UK, 2019.

[3] H. Sira-Ramirez, R. Castro-Linares, and G. Puriel-Gil, "An active disturbance rejection approach to leader-follower controlled formation," *Asian Journal of Control*, vol. 16, no. 2, pp. 382–395, 2014.

[4] R. Stevens, F. Sadjadi, J. Braegelmann, A. Cordes, and R. Nelson, "Small unmanned aerial vehicle (UAV) real-time intelligence, surveillance and reconnaissance (ISR) using onboard pre-processing," *Proceedings of SPIE*, vol. 6967, 2008.

[5] C. Wu, S. Shi, S. Gu, L. Zhang, and X. Gu, "Deep reinforcement learning-based content placement and trajectory design in urban cache-enabled UAV networks," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8842694, 11 pages, 2020.

[6] R. R. Murphy, E. Steimle, M. Hall et al., "Robot-assisted bridge inspection," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 1, pp. 77–95, 2011.

[7] D. Hausamann, W. Zirnig, G. Schreier, and P. Strobl, "Monitoring of gas pipelines – a civil UAV application," *Aircraft Engineering and Aerospace Technology*, vol. 77, no. 5, pp. 352–360, 2005.

[8] J. K. Howlett, T. W. Mclain, and M. A. Goodrich, "Learning real-time ∗ path planner for unmanned air vehicle target sensing," *Journal of Aerospace Computing Information and Communication*, vol. 3, no. 3, pp. 108–122, 2006.

[9] G. Luo, J. Yu, Y. Mei, and S. Zhang, "UAV path planning in mixed-obstacle environment via artificial potential field method improved by additional control force," *Asian Journal of Control*, vol. 17, no. 5, pp. 1600–1610, 2015.

[10] R. Kala and K. Warwick, "Planning of multiple autonomous vehicles using RRT," in *IEEE International Conference on Cybernetic Intelligent Systems*, pp. 20–25, London, UK, 2011.

[11] F. J. Rubio, F. J. Valero, J. L. Suñer, and V. Mata, "Simultaneous algorithm for trajectory planning," *Asian Journal of Control*, vol. 12, no. 4, pp. 468–479, 2010.

[12] A. E. Oguz and H. Temeltas, "On the consistency analysis of A-SLAM for UAV navigation," in *Unmanned Systems Technology XVI, vol. 9084 International Society for Optics and Photonics*, Baltimore, Maryland, United States, 2014.

[13] R. Strydom, S. Thurrowgood, and M. V. Srinivasan, "Visual odometry: autonomous UAV navigation using optic flow and stereo," in *Proceedings of Australasian conference on robotics and automation*, Melbourne, Melbourne, Australia, 2014.

[14] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 35–42, 2009.

[15] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[16] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) IEEE*, pp. 4241–4247, Vancouver, BC, Canada, 2017.

[17] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: policy learning using adaptive trajectory optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA) IEEE*, pp. 3342–3349, Singapore, 2017.

[18] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: learning to fly by driving," *IEEE*

Robotics and Automation Letters, vol. 3, no. 2, pp. 1088–1095, 2018.

[19] N. C. Luong, D. T. Hoang, S. Gong et al., "Applications of deep reinforcement learning in communications and networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[20] R. Wu, F. Gu, and J. Huang, "A multi-critic deep deterministic policy gradient UAV path planning," in *Proceedings of 2020 16th International Conference on Computational Intelligence and Security*, pp. 6–10, Guangxi, China, 2020.

[21] Y. Zhu, H. Liu, B. Ren, H. Duan, X. She, and Z. Wu, "A model-free flat spin recovery scheme for miniature fixed-wing unmanned aerial vehicle," in *2019 IEEE International Conference on Unmanned Systems (ICUS) IEEE*, pp. 623–630, Beijing, China, 2019.

[22] Y. Zeng, X. Xu, S. Jin, and R. Zhang, "Simultaneous navigation and radio mapping for cellular-connected UAV with deep reinforcement learning," *IEEE Transactions onWireless Communications*, vol. 20, no. 7, pp. 4205–4220, 2021.

[23] R. Xie, Z. Meng, Y. Zhou, Y. Ma, and Z. Wu, "Heuristic Q-learning based on experience replay for three-dimensional path planning of the unmanned aerial vehicle," *Science Progress*, vol. 103, no. 1, pp. 1–18, 2019.

[24] O. Walker, F. Vanegas, F. Gonzalez, and S. Koenig, "A deep reinforcement learning framework for UAV navigation in indoor environments," in *2019 IEEE Aerospace Conference IEEE*, pp. 1–14, Big Sky, MT, USA, 2019.

[25] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100, Arizona, USA, 2015.

[27] Z. Wang, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot, and N. D. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference onMachine Learning*, pp. 1995–2003, New York, USA, 2016.

[28] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, pp. 1008–1014, La Jolla, CA, 2000.

[29] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[30] X. Liu, X. Wang, and Y. M. Cheung, "FDDH: fast discriminative discrete hashing for large-scale cross-modal retrieval," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.

[32] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[33] S. Ross, N. Melik-Barkhudarov, K. S. Shankar et al., "Learning monocular reactive UAV control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*, pp. 1765–1772, Karlsruhe, Germany, 2013.

[34] J. L. Junell, E. J. Van Kampen, C. C. de Visser, and Q. P. Chu, "Reinforcement learning applied to a quadrotor guidance law in autonomous flight," in *AIAA Guidance, Navigation, and Control Conference*, p. 1990, Kissimmee, Florida, 2015.

[35] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, New York, New York, USA, 2016.

[36] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, https://arxiv.org/abs/1802.09477.

[37] X. Liu, Z. Hu, H. Ling, and Y. M. Cheung, "MTFH: a matrix tri-factorization hashing framework for efficient cross-modal retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 964–981, 2021.

[38] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.

[39] Z. Hu, W. Kaifang, X. Gao, Y. Zhai, and Q. Wang, "Deep reinforcement learning approach with multiple experience pools for UAV's autonomous motion planning in complex unknown environments," *Sensors*, vol. 20, no. 7, p. 1890, 2020.

[40] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[41] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning," *Remote Sensing*, vol. 12, no. 4, p. 640, 2020.

[42] J. Wu, R. Wang, R. Li, H. Zhang, and X. Hu, "Multi-critic DDPG method and double experience replay," in *Proceedings of 2018 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 165–171, Miyazaki, Japan, 2018.

[43] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *The 3rd International Conference for Learning Representations*, pp. 1–15, Arizona, USA, 2015.