remote sensing

Article

UAV Maneuvering Target Tracking in Uncertain Environments Based on Deep Reinforcement Learning and Meta-Learning

Bo Li [1,*], Zhigang Gan [1], Daqing Chen [2] and Dyachenko Sergey Aleksandrovich [3]

[1] School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China; ganzhigang@mail.nwpu.edu.cn
[2] School of Engineering, London South Bank University, London SE1 0AA, UK; chend@lsbu.ac.uk
[3] School of Robotic and Intelligent Systems, Moscow Aviation Institute, 125993 Moscow, Russia; dyachenkosa@mai.ru
* Correspondence: libo803@nwpu.edu.cn; Tel.: +86-133-5921-2759

Received: 20 October 2020; Accepted: 16 November 2020; Published: 18 November 2020

Abstract: This paper combines deep reinforcement learning (DRL) with meta-learning and proposes a novel approach, named meta twin delayed deep deterministic policy gradient (Meta-TD3), to realize the control of unmanned aerial vehicle (UAV), allowing a UAV to quickly track a target in an environment where the motion of a target is uncertain. This approach can be applied to a variety of scenarios, such as wildlife protection, emergency aid, and remote sensing. We consider a multi-task experience replay buffer to provide data for the multi-task learning of the DRL algorithm, and we combine meta-learning to develop a multi-task reinforcement learning update method to ensure the generalization capability of reinforcement learning. Compared with the state-of-the-art algorithms, namely the deep deterministic policy gradient (DDPG) and twin delayed deep deterministic policy gradient (TD3), experimental results show that the Meta-TD3 algorithm has achieved a great improvement in terms of both convergence value and convergence rate. In a UAV target tracking problem, Meta-TD3 only requires a few steps to train to enable a UAV to adapt quickly to a new target movement mode more and maintain a better tracking effectiveness.

Keywords: UAV; maneuvering target tracking; deep reinforcement learning; meta-learning; multi-tasks

1. Introduction

With the development of technology, various types of unmanned aerial vehicles (UAVs) have become available, such as fixed-wing and quadcopter. In particularly, quadcopters play an important role in a variety of scenarios. The advantages of using quadcopters include low-cost, low fault rates, light weight, fast response, high maneuverability, hovering capabilities, and vertical take-off and landing. In recent years, quadcopters have been extensively applied for different types of tasks, e.g., object tracking [1], wildlife protection [2], disaster rescue [3], 3D reconstruction [4], and maneuvering target tracking [5]. As an important application of UAV, maneuvering target tracking means that UAV observes a moving target through sensors, perceives the environment of the target, follows the target, and maintains the tracking range with the target. However, when dealing with the uncertain environment and the movement mode of the uncertain target, UAVs do not have the ability to autonomously complete decision-making and planning, and cannot achieve high-precision tracking of maneuvering targets. Accurate maneuvering target tracking under uncertain environments is the requirement of future development UAV.

Traditional methods of target tracking include genetic algorithms, Bayesian inference, and statistical theory, etc. These methods transform the target tracking problem into an optimization problem. By solving the optimization model, the tracking path can be obtained. However, these traditional algorithms have a poor real-time performance for uncertain environment problems and cannot realize real-time tracking decision.

In order to tackle these problems, researchers have employed artificial intelligence-based approaches, and one of such approaches is reinforcement learning (RL). RL is an effective approach to train an agent to make decisions in a complex and uncertain environment. Mnih [6] developed a novel artificial agent, named a deep Q-network (DQN), that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. Since then, deep reinforcement learning algorithms have been more widely used in UAV navigation and target tracking. Huang [7] combined DQN with UAV navigation to enable the agent to make decisions based on the received signal strength. Wu [8] proposed a search algorithm based on DQN, which can realize automatic search and path finding of UAV. Wang [9] developed a DRL framework for UAV navigation in large and complex environments for the partially observable Markov decision process (POMDP), and realized UAV navigation based on DDPG [10]. Wan proposed a novel DRL method to achieve robust control of UAV in dynamic uncertain environment [11]. Based on DQN, Bhagat proposed a deep reinforcement learning method for tracking targets in the presence of obstacles and target motion [12].

These researches have achieved good results in UAV navigation, but they all have some drawbacks. First of all, these researches have usually focused on the navigation of UAVs with fixed targets and the movement of targets is simple and single [13–15]. UAV maneuvering target tracking is different to navigation, and it requires a UAV to make autonomous decision in real time. If the navigation of a stationary target is applied to the tracking of a moving target directly, the tracking effect will not be guaranteed. Secondly, most of the current researches only consider to train DRL model in a single environment, and as such a UAV can only perceive a specific scenario, leading to a poor generalization capability of a learned policy. UAV cannot adapt or achieve better effects in a new environment. Meanwhile, recent research has found the existing DRL methods brittle [16,17]. If a DRL model is only trained for a single particular environment or task, when faced with uncertain environments and stochastic tasks, the model may be inefficient and unable to achieve a desired performance due to the model's limited applicability [18].

In this paper, we have conducted research on UAV maneuvering target tracking with different target maneuvering methods. We have combined the DRL algorithm TD3 [19] with meta-learning [20,21] to develop an improved TD3 algorithm, named meta twin delayed deep deterministic policy gradient (Meta-TD3), to be used to realize the tracking of UAV to maneuvering targets in uncertain environments.

The contributions of this paper are summarized as follows:

(1) A UAV motion model is constructed, and the UAV maneuvering target tracking is defined as an MDP. Based on TD3 algorithm, a decision-making framework is established to control the course and velocity of UAV. Through the decision-making framework, autonomous real-time maneuvering target tracking can be realized.
(2) In order to make the UAV quickly adapt to the tracking of multiple uncertain target motion modes, a deep reinforcement learning algorithm, meta-TD3, is developed by combining DRL and meta-learning. A multi-task experience replay buffer is proposed, which contains several replay buffers of different tasks. A multi-task meta-learning update method is developed which breaks the original DRL network update. In this way, DRL model can learn multiple tasks at the same time and perform meta-learning on weights obtained by training from multiple tasks. Through these, meta-TD3 can overcome the shortcomings of DRL and enable UAV to quickly adapt to an uncertain environment and realize target tracking quickly and efficiently.

(3) Through a series of experiments, compared with the state-of-the-art algorithms DDPG and TD3, the performance, training efficiency and the generalization capability of meta-TD3 have been verified. In the UAV target tracking problem, Meta-TD3 only requires a few steps to train to enable the UAV to adapt to the new target movement mode more quickly and maintain a better tracking effect.

The remainder of this paper is organized as follows. Section 2 introduces the UAV motion model and defines the UAV maneuvering target tracking as an MDP. In Section 3, we constructed a DRL framework for UAV maneuvering target tracking based on TD3 algorithm, and propose a reinforcement learning algorithm combined with meta-learning. The experiments in Section 4 demonstrate the effectiveness and adaptability of the proposed algorithm. Section 5 summarizes this paper and looks forward to future work.

2. Problem Formulation

In order to describe the UAV maneuvering target tracking, we construct the UAV motion model and target tracking model based on Markov decision process (MDP).

2.1. UAV Motion Model

The UAV motion model is the basis for completing navigation and target tracking missions. The UAV can be thought as a rigid body with forces and torques applied from the four rotors and gravity [22,23]. In navigation and target tracking scenario, we assume that UAV is flying at a fixed altitude. The experiments of this paper are set in a x-y plane of Cartesian inertial coordinates [24].

The UAV motion model in Cartesian inertial coordinates is expressed as

$$\begin{cases} \varphi_{t+1} = \varphi_t + \omega_t dt \\ v_{t+1} = v_t + n_t dt \\ x_{t+1} = x_t + v_{t+1} sin(\varphi_{t+1}) dt \\ y_{t+1} = y_t + v_{t+1} \cos(\varphi_{t+1}) dt \end{cases} \tag{1}$$

where x_t, y_t denote the position coordinates of the center of mass of UAV in Cartesian inertial coordinates at time t. x_{t+1}, y_{t+1} denote the position coordinates of the center of mass of UAV in Cartesian inertial coordinates at time $t+1$. v_t denotes the linear velocity of the UAV, n_t is the acceleration of the linear velocity of the UAV. φ_t denotes the heading angle, ω_t is the angular velocity of the heading angle and t $(t = 1, \ldots, n)$ t(t=1, … ,n)denotes each timestep, and dt is defined as the time interval between t and $t+1$.

The control of UAV is realized through the acceleration and the angular velocity of the heading angle, where acceleration represents the control of throttle and the angular velocity represents the control of steering. The UAV realizes the flight through the control of the throttle and the steering. The UAV motion model is shown as in Figure 1. ψ_t denotes the angle between the linear velocity of the UAV and the target line at time t, d_t is the distance between UAV and target at time t, x_t^{target}, y_t^{target} denote the position coordinates of the target in Cartesian inertial coordinates at time t.

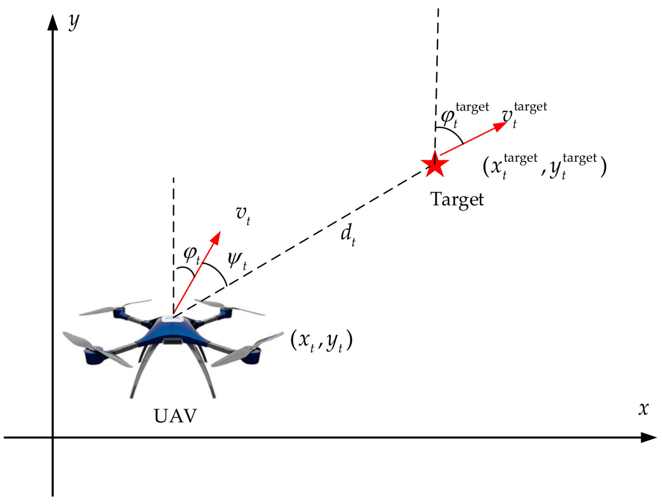

Figure 1. Unmanned aerial vehicle (UAV) motion model.

2.2. Target Tracking Model Based on Reinforcement Learning

In order to realize the robust control of UAV in complex and uncertain environments, we focus on reinforcement learning. Reinforcement learning is a machine learning method wherein an autonomous agent learns to find a near-optimal behavior through trial-and-error interactions with its environment [2]. The specific implementation process is defined as the Markov decision process [25,26].

Markov decision process can be described by a five-tuple (S, A, P, R, γ). State space S is the set of all states of the UAV. Action space A is the set of all the actions that the UAV can perform. is denoted as $P(s'|s, a)$, which indicates the transition probability of the agent taking action $a \in A$ from state $s \in S$ to the next state $s' \in S$. R is denoted as $R(s, a)$, indicating the reward that can be obtained by taking action $a \in A$ in state $s \in S$. γ represents the discount factor. The implementation process of reinforcement learning is as follows.

$$Q(s, a) = E_{\pi(\mathrm{s})}[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1})|s_t = s, a_t = a] \tag{2}$$

The agent's policy $\pi(\mathrm{s})$ provides the guideline on what is the optimal action to take in a certain state with the goal to maximize the total rewards. denotes the expected reward when performing actions in state and following policy $\pi(\mathrm{s})$. The state value function of the judgement strategy is defined as:

$$V_\pi(\mathrm{s}) = \sum_{a \in A} \pi(a|\mathrm{s}) Q_\pi(s, a) \tag{3}$$

The reinforcement learning model framework of the UAV navigation and target tracking is shown in Figure 2. The state of the UAV and the state of the target are integrated into the Agent. During the interaction process, the agent outputs the action to the environment to change the state of the UAV, and the target changes the state according to its own movement mode. For the action taken by the UAV, the environment feedbacks the corresponding reward to the agent. The policy is dynamically updated in the process, so that the action tends to be optimal, thus realizing the target tracking.

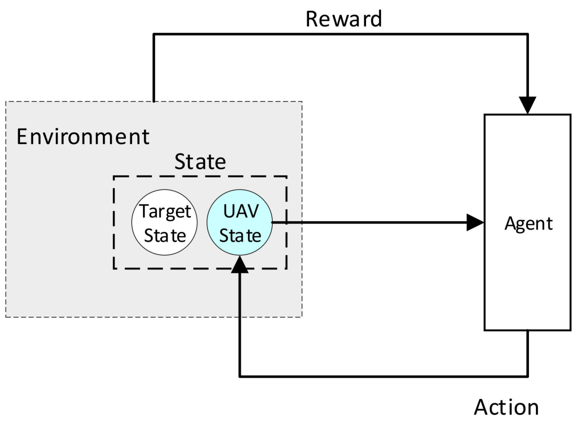

Figure 2. UAV navigation and target tracking model based on reinforcement learning.

2.2.1. State Space

The state space describes the collection of information that the UAV can obtain from the environment. The position of the center of mass of the UAV relative to the inertial coordinate system and the motion variables of UAV are defined as state. In this paper, we assume that the UAV can get the position, attitude information and target distance through GPS and sensors. Therefore, the state space consists of the following variables, $x, y, \varphi, v, \psi, d$. x, y denote the coordinate of UAV, φ denotes the heading angle, v is the linear velocity of the UAV, ψ denotes the angle between the linear velocity of the UAV and the target line and d is the Euclidean distance between the UAV and the target. In order to unify the range of each state variable, eliminate errors caused by the variable scale, and improve the efficiency of neural network training, normalization operation is carried out for each state variable.

In addition, every state s is defined as:

$$s = [x, y, \varphi, v, \psi, d]^T \tag{4}$$

2.2.2. Action Space

In the Markov decision process, the agent selects actions from the action space. Considering the actual situation, the speed of the UAV cannot be changed abruptly. We set the acceleration and the angular velocity of the heading angle as actions. In other words, with the acceleration and the angle change, the state of UAV will change simultaneously.

The action of the model is defined as:

$$a = [\omega_t, n_t]^T \tag{5}$$

where n_t denotes the acceleration of the linear velocity of the UAV, ω_t is the angular velocity of the heading angle. The action space is defined as a continuous interval $A = [a_{\min}, a_{\max}]$ from which the actions of the reinforcement learning are selected.

2.2.3. Reward Shaping

The reward function is the assessment of the agent's actions by the environment. The reward function can determine the performance of the agent in the reinforcement learning. In this paper, the reward function is composed of three parts: distance reward, angle reward and extra distance penalty and reward.

In order to improve the convergence speed and accuracy of the reinforcement learning model, this paper normalizes the distance reward function. The normalized distance reward function is defined as:

$$r_d = -\delta \frac{d_t}{d_{\max}} \tag{6}$$

where δ is a positive constant, d_t denotes the distance between UAV's current position and the target at timestep t. $d_{\max}$ is the max distance that UAV can detect. To ensure the UAV flying in the direction of the target at all times, UAV would obtain an angle reward. The normalized angle reward function $r_\theta(s_t, a_t)$ is defined as:

$$r_\psi = -\varepsilon \left| \psi_t - \psi_{t-1} \right| \tag{7}$$

where ε is a positive constant and ψ_t denotes the angle between the linear velocity of the UAV and the target line at timestep t.

In order to achieve faster tracking of a target, we set extra distance penalty and reward. The extra distance penalty and reward is defined as:

$$r_e = \begin{cases} p, & if \quad d_t \geq D \\ r, & if \quad d_t < D \end{cases} \tag{8}$$

where D denotes the optimal tracking distance, p is a negative constant and r is a positive constant. To summarize, the final reward function can be formulated as:

$$r = r_d + r_\psi + r_e \tag{9}$$

3. Meta-TD3 for Target Tracking Model

In this section, we introduce the UAV target tracking based on the actor-critic reinforcement learning framework and propose a reinforcement learning algorithm combined with meta-learning.

3.1. UAV Motion Model

In Section 2.2, we define the target tracking model based on reinforcement learning, and action space is defined as the continuous space A. Because of the continuous space, some reinforcement learning algorithms such as Q-learning and DQN cannot be applied [27–29]. These algorithms are more suitable for reinforcement learning in the discrete action space. For the target tracking problem to be solved in this paper, these algorithms require optimize all the values in the action space [10]. It's difficult to effectively explore such a large action space and ensure the ideal effect of training. In this section, we introduce the model architecture based on TD3 (twin delayed deep deterministic policy gradient). The reinforcement learning model architecture we created consists of an actor module, a critic module and an experience replay buffer.

In order to increase the stability of training, actor module is composed of two neural networks and critic module is composed of four neural networks with the same structure. Actor module consists of an online policy neural network and a target policy network, critic module consists of two online Q-value neural networks and two target Q-value neural networks.

The structure and implementation process of target tracking model is shown in Figure 3. At each step, the agent selects an action by $a_t = \pi_\phi(s_t) + N_t$, where N_t is exploration noise. Based on the reward function in Section 2.2.3, the environment returns the reward $r(s_t, a_t)$ to the UAV. The samples $(s_t, a_t, r(s_t, a_t), s_{t+1})$ from the interaction between the agent and the environment are stored in the experience replay buffer. When training, a mini batch of samples $N * (s_j, a_j, r_j, s_{j+1})$ are randomly selected to update the network parameters by gradient descent. This random sampling method greatly reduces the correlation between samples, thereby improving the stability of the training process.

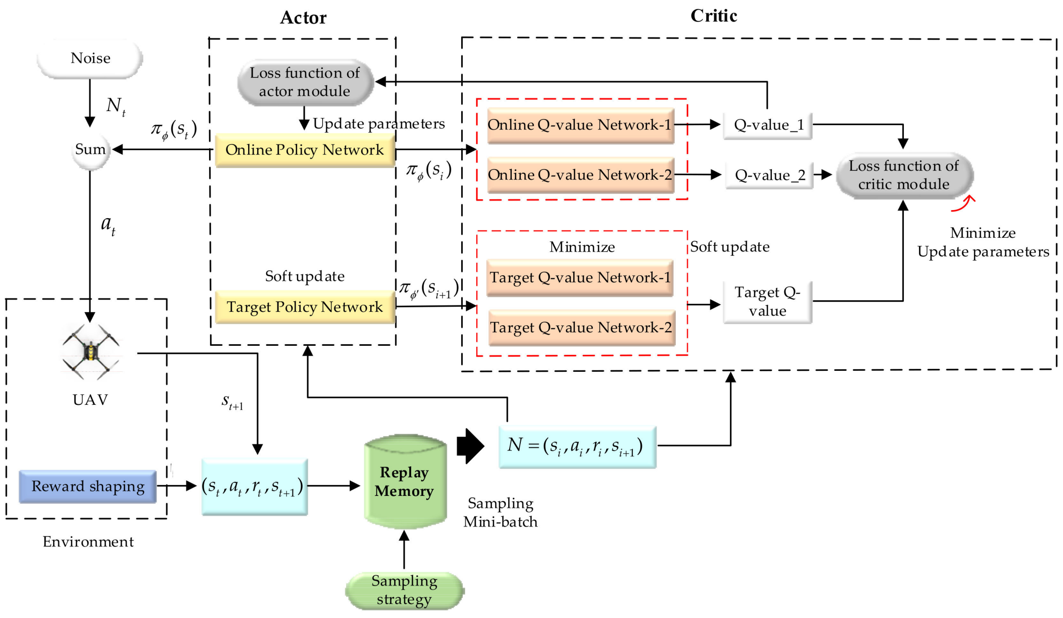

Figure 3. Target tracking model based on twin delayed deep deterministic policy gradient (TD3).

The main purpose of actor module is to generate reasonable actions according to the current state of the UAV to approach the target and finally realize the target tracking. The input of the actor module is the state of the UAV, and the output is the action of the UAV. The training process of actor module is to find the optimal policy π_ϕ.

The loss function of actor module is defined as:

$$J = E[Q_{\theta_1}(s,a)\big|_{s=s_j, a=\pi_\phi(s_j)}] \tag{10}$$

where θ_1 denotes the neural networks parameters of the online Q-value Networks, ϕ is the neural networks parameters of online policy networks. $Q_{\theta_1}(s,a)\big|_{s=s_j, a=\pi_\phi(s_j)}$ is the reward that the agent can obtain in state s_j by choosing action according to policy $\pi_\phi(s)$. During the training process, Adam gradient descent is performed on the network parameters ϕ. The online policy networks parameters ϕ are updated to maximize the loss function J as much as possible.

Through the actor module, the agent can get action according to the current state. In order to evaluate the agent's decision, we introduce the critic module to guide the policy to be optimal. The input of the critic module is the state and the action of the UAV, the output is the Q-value. During the training process, the networks parameters of the critic module are updated along the direction of increasing the Q value. In order to obtain the maximum Q-value and reduce the over-estimation of the Q-value, according to the Bellman equations and [30], the target optimal Q-value function y_t is defined as:

$$y_j = r(s_j, a_j) + \gamma \min_{i=1,2} Q_{\hat{\theta}_i}(s_{j+1}, \pi_{\hat{\phi}}(s_{j+1})) \tag{11}$$

where ϕ' denotes the neural networks parameters of target policy network, $\hat{\theta}_i (i = 1, 2)$ are the neural networks parameters of target Q-value Networks. $r(s_j, a_j)$ denotes the immediate reward the agent can receive in state s_j by taking action a_j. γ denotes the discounting factor of future reward. $Q_{\hat{\theta}_i}(s_{j+1}, \pi_{\hat{\phi}}(s_{j+1}))$ is the future reward that the agent can receive in state s_{j+1} according to policy $\pi_{\hat{\phi}}$.

The loss function of critic module is defined as (12). L is similar to the form of MSE (mean square error) in supervised learning, where θ_i , $i = 1, 2, \ldots,$ denotes the neural networks parameters of online Q-value networks:

$$L = \frac{1}{N} \sum_{j=1,2\ldots}^{N} \sum_{i=1,2} (y_j - Q_{\theta_i}(s_j, a_j))^2 \tag{12}$$

During training, the parameters of online policy neural networks are updated by maximizing the loss function J, the parameters of online Q-value neural networks are updated by minimizing the loss function L. The parameters of target neural networks are updated by soft update. The updating method is shown in (13), where τ is a constant coefficient, used to the regulate the update rate:

$$\begin{cases} \hat{\phi} \leftarrow \tau\phi + (1-\tau)\hat{\phi} \\ \hat{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\hat{\theta}_i \end{cases} (i = 1, 2) \tag{13}$$

3.2. *Multi-Tasks Experience Replay Buffer and Meta-Learning Update*

In Section 3.1, we introduce the UAV target tracking model based TD3, which can realize the target tracking in a single motion form. However, the target movement mode may be more complex and changeable. Aiming at the problem of insufficient generalization ability of reinforcement learning, we consider a combination of reinforcement learning and meta-learning to improve the generalization ability of reinforcement learning.

In order to find a more general tracking strategy, it is necessary to train for multi-tasks. In Section 3.2.1, we introduce the setting of multi-tasks and the experience replay buffer for multi-tasks. In Section 3.2.2, we propose a new reinforcement learning updating method, which breaks the original TD3 update method through meta-learning, thereby obtaining a general tracking strategy.

3.2.1. Multi-Tasks Setting

Generalization ability refers to the predictive ability of the model learned by the method to unknown data, which is an essential and important property of the learning method. As for reinforcement learning, we expect a good reinforcement model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time. In traditional reinforcement learning, the agent is usually trained in a specific environment. The exploration is usually employed to ensure the generalization of the policy. However, it remains to explore more implementation possibilities for a specific environment and task. In this way, the generalization capability of reinforcement learning has not improved much. Taking the target tracking problem as an example, the tracking policy trained for a specific target motion (speed and motion direction change in a specific way) is difficult to directly adapt to the task of tracking other target motion modes.

In this section, we propose the multi-task setting. Agent needs to learn a variety of different tasks at the same time, and finally learn a policy that can adapt to many tasks well. We construct task experience replay buffer B_{T_j} for task $T_j(j = 1, 2, \ldots, n)$ and combine all task experience replay buffers to form multi-task experience replay buffer$B$. The structure of multi-tasks experience replay buffer is shown in Figure 4. For target tracking, different motion modes of target, different initial relative positions of the UAV and target are defined as different target tracking tasks. In task T_j, $(s_t^{T_j}, a_t^{T_j}, r^{T_j}(s_t, a_t), s_{t+1}^{T_j})$ generated by the agent interacting with the environment are stored in the corresponding task experience replay buffer B_{T_j}. During training, an equal amount of experience is randomly selected from each task experience replay buffer for training, thus realizing the learning of a variety of different tasks.

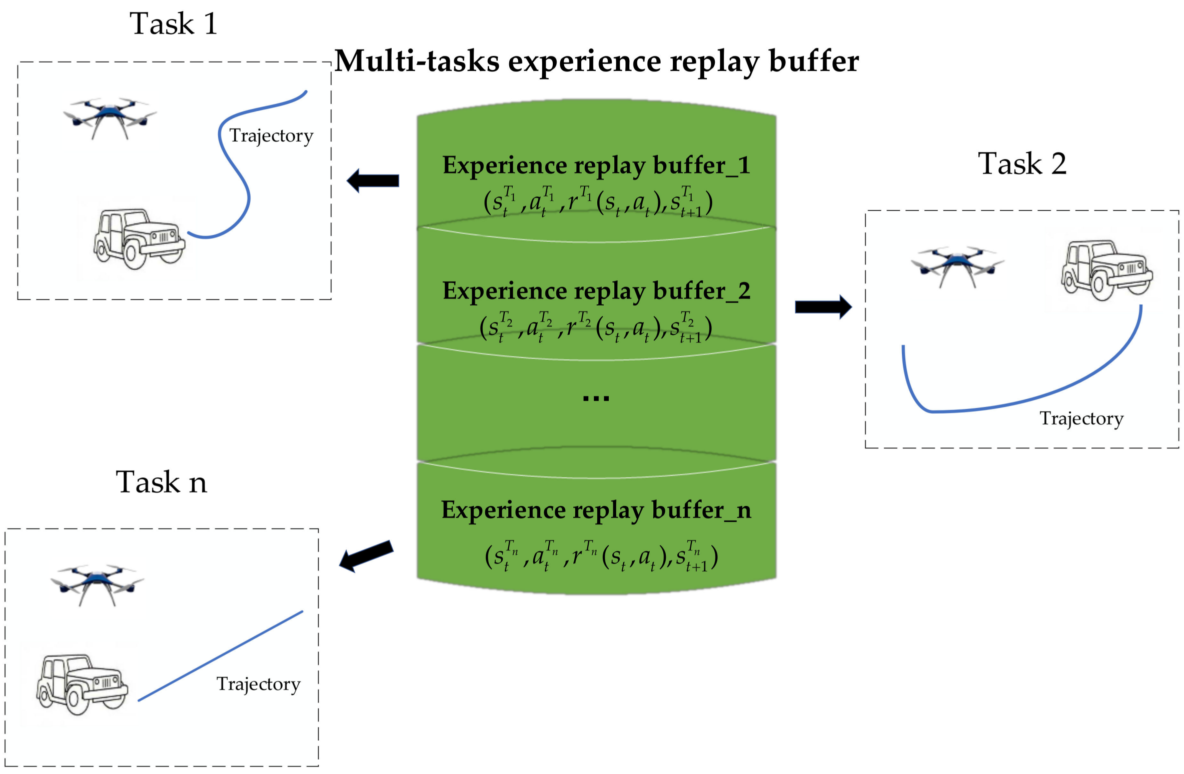
Task 1
Multi-tasks experience replay buffer
Trajectory
Experience replay buffer_1
$(s_t^{T_1}, a_t^{T_1}, r^{T_1}(s_t, a_t), s_{t+1}^{T_1})$
Task 2
Experience replay buffer_2
$(s_t^{T_2}, a_t^{T_2}, r^{T_2}(s_t, a_t), s_{t+1}^{T_2})$
...
Trajectory
Task n
Experience replay buffer_n
$(s_t^{T_n}, a_t^{T_n}, r^{T_n}(s_t, a_t), s_{t+1}^{T_n})$
Trajectory

Figure 4. The structure of Multi-tasks experience replay buffer.

3.2.2. Meta-Learning Update Method

In this section, based on meta-learning, we propose Meta-TD3 to learn a variety of different tasks and obtain a more generalized policy meta-policy. The meta-policy obtained through meta-learning can enable agent to use only a small number of learning steps to acquire excellent strategies for new tasks.

During the training of reinforcement learning, agent only considers the current task and environment, trying to find a policy that can adapt to the current task and environment [31,32]. Although the reinforcement learning algorithm adopts exploration to ensure that the policy has a certain generalization ability, the previous policy cannot be used directly when dealing with different tasks and environments [33–35]. Aiming to resolve the problem of insufficient generalization ability for different tasks in the implementation process of reinforcement learning, we introduce meta-learning into reinforcement learning (Meta-TD3) and optimize the initial parameters of reinforcement learning to improve generalization ability.

The purpose of Meta-TD3 is to maximize the expected generalization ability of reinforcement learning algorithms in all the training tasks. Meta-TD3 is used to learn many different types of parameters to obtain the initial parameters of reinforcement learning for different type of tasks [36]. Meta-TD3 obtains prior knowledge from many related tasks as a guide to learning new tasks. When responding to a new task, based on the previous initial parameters learned through Meta-TD3, only a few steps of fine tuning are needed to meet the requirements of task.

Meta-TD3 includes update processes: internal TD3 reinforcement learning for a single task and external meta learning update for multiple different tasks. These processes can be viewed from the perspective of feature learning as building an internal representation that is widely applicable to many tasks. In Meta-TD3, internal TD3 training and external meta-learning update are performed alternately, which meets a certain update frequency. TD3 model learns multiple tasks separately to obtain different parameters, Meta-TD3 obtains the initial parameters of reinforcement learning by optimizing different parameters.

For the internal TD3 reinforcement learning update process, it can be described as a learning process for a specific task T_j. The agent will optimize the policy to achieve target tracking for the current task. The specific update process is the same as that introduced in Section 3.1. After completing the internal TD3 reinforcement learning update for the task T_j, the parameters $\phi^{T_j}, \hat{\phi}^{T_j}$ of the policy network about the task T_j and the parameters $\theta_1^{T_j}, \hat{\theta}_1^{T_j}, \theta_2^{T_j} \hat{\theta}_2^{T_j}$ of the Q-value network about the task T_j can be obtained.

For external meta-learning update, it can be described as quadratic gradient optimization for a series of task network parameters. The external meta-learning update process is as follows:

$$\begin{aligned} \theta_1^{meta} &\leftarrow \theta_1^{meta} + \beta \sum_{j=1,2,...}^{n} (\hat{\theta}_1^{T_j} - \theta_1^{meta}) \\ \theta_2^{meta} &\leftarrow \theta_2^{meta} + \beta \sum_{j=1,2,...}^{n} (\hat{\theta}_2^{T_j} - \theta_2^{meta}) \\ \phi^{meta} &\leftarrow \phi^{meta} + \beta \sum_{j=1,2,...}^{n} (\hat{\phi}^{T_j} - \phi^{meta}) \end{aligned} \tag{14}$$

where ϕ^{meta} denotes the parameters of the policy network updated through Meta-TD3, θ_1^{meta} and θ_2^{meta} denote the parameters of the Q-value network updated through Meta-TD3, $\hat{\phi}^{T_j}$ denotes the parameters of the policy target network for internal TD3 update for task T_j, $\hat{\theta}_2^{T_j}$ and $\hat{\theta}_1^{T_j}$ are the parameters of the Q-value target network for internal TD3 update for task T_j, and β is the meta step size of outer meta-learning. n denotes the number of tasks.

The Meta-TD3 update process is shown in Figure 5. During training, we randomly select tasks from the multi-tasks experience replay buffer. Take the update of policy neural network parameters ϕ^{meta} as an example. When learning task T_j, the neural network parameters $\hat{\phi}^{T_j}$ are obtained by the internal TD3 reinforcement learning update. When all tasks sampled have completed the internal TD3 reinforcement learning update, the neural network parameters are further optimized through external meta-learning update. Internal and external updates are carried out constantly to obtain the neural network parameters as meta-policy.

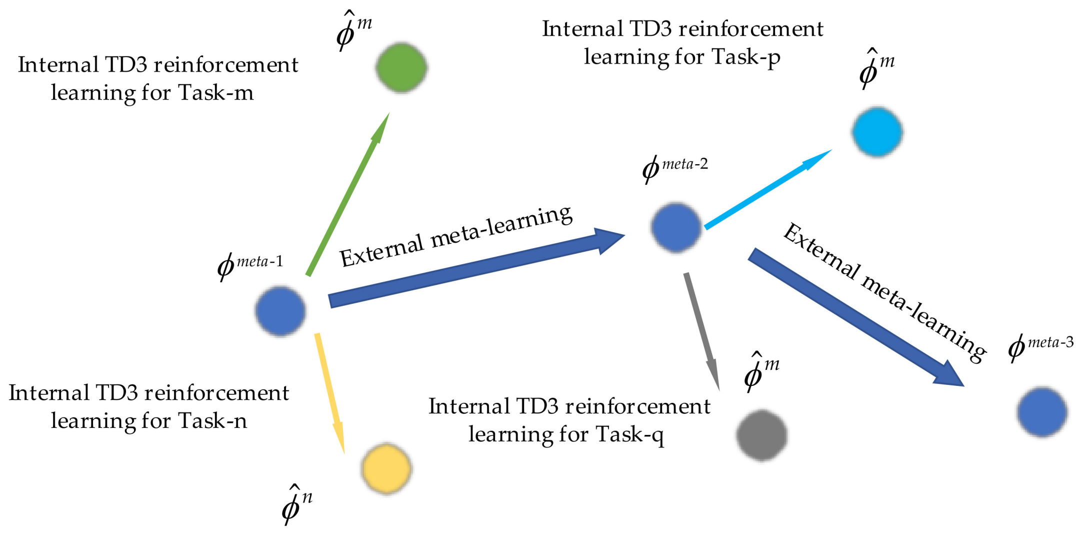
$\hat{\phi}^m$
Internal TD3 reinforcement learning for Task-m
Internal TD3 reinforcement learning for Task-p
$\hat{\phi}^m$
$\phi^{meta\text{-}2}$
External meta-learning
$\phi^{meta\text{-}1}$
External meta-learning
$\phi^{meta\text{-}3}$
$\hat{\phi}^m$
Internal TD3 reinforcement learning for Task-n
Internal TD3 reinforcement learning for Task-q
$\hat{\phi}^n$

Figure 5. Meta-TD3 update process.

Ultimately, we present Meta-TD3 by introducing the multi-tasks experience replay buffer and meta-learning update described in Sections 3.2.1 and 3.2.2 to TD3. Meta-TD3 is summarized in Table 1.

Table 1. Meta-TD3 algorithm for target tracking model.

```
Algorithm: Meta-TD3
1: Require: distribution over tasks $p(T)$
2: Require: meta-learning steps size hyperparameters $\beta$
3: Initialize meta-critic networks $Q_{\theta_1^{meta}}$, $Q_{\theta_2^{meta}}$, and meta-actor network $\pi_{\phi^{meta}}$ with random parameters $\theta_1^{meta}$, $\theta_2^{meta}$, $\phi^{meta}$
4: Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network $\pi_\phi$ $\theta_1 \leftarrow \theta_1^{meta}$, $\theta_2 \leftarrow \theta_2^{meta}$, $\phi \leftarrow \phi^{meta}$
5: Initialize target networks $\hat{\theta}_1 \leftarrow \theta_1$, $\hat{\theta}_2 \leftarrow \theta_2$, $\hat{\phi} \leftarrow \phi$
6: Sample batch of tasks $T_j \sim p(T) j = 1, 2, \ldots, n$
7: Initialize replay buffer $B_{T_j}$
8: For meta_iteration $= 1, 2, \ldots$ do
9:     for t $= 1, 2, \ldots$ do
10:        Select action $a_{T_j} \sim \pi_\phi(s_{T_j}) + \varepsilon$ with exploration noise $\varepsilon \sim N(0, \sigma)$
11:        Observe reward $r_{T_J}$ and new state $s'_{T_j}$
12:        Store transition $(s_{T_J}, a_{T_J}, r_{T_J}, s'_{T_j})$ in $B_{T_j}$
13:       Sample mini-batch of N transitions $(s_{T_J}, a_{T_J}, r_{T_J}, s'_{T_j})$ from $B_{T_j}$
14:                   $\widetilde{a}_{T_j} \leftarrow \pi_{\hat{\phi}}(s'_{T_j}) + \varepsilon$, $y \leftarrow r_{T_J} + \gamma \min_{i=1,2} Q_{\hat{\theta}_i}(s'_{T_j}, \widetilde{a}_{T_j})$
15:       Update critics $\theta_i \leftarrow \mathrm{argmin}_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_1}(s_{T_J}, a_{T_J}))^2$
16:        if t mod update_policy_freq then
17:           Update $\phi$ by the deterministic policy gradient:
18:                   $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(s_{T_J}, a_{T_J})\big|_{a=\pi_\phi(s_{T_J})} \nabla_\phi \pi_\phi(s_{T_J})$
19:           Update the target networks:
20:               $\hat{\theta}_1 \leftarrow \tau\theta_1 + (1-\tau)\hat{\theta}_1$, $\hat{\theta}_2 \leftarrow \tau\theta_2 + (1-\tau)\hat{\theta}_2$, $\hat{\phi} \leftarrow \tau\phi + (1-\tau)\hat{\phi}$
21:        The neural network parameters of task $T_j$:$\hat{\phi}^{T_j} = \hat{\phi}$, $\hat{\theta}_1^{T_j} = \hat{\theta}_1$, $\hat{\theta}_2^{T_j} = \hat{\theta}_2$
22:        if t mod meta_update_freq then
23:            $\theta_1^{meta} \leftarrow \theta_1^{meta} + \beta \sum_{j=1,2,\ldots}^{n} (\hat{\theta}_1^{T_j} - \theta_1^{meta})$, $\theta_2^{meta} \leftarrow \theta_2^{meta} + \beta \sum_{j=1,2,\ldots}^{n} (\hat{\theta}_2^{T_j} - \theta_2^{meta})$
24:            $\phi^{meta} \leftarrow \phi^{meta} + \beta \sum_{j=1,2,\ldots}^{n} (\hat{\phi}^{T_j} - \phi^{meta})$
25:     end for
26:   $\theta_1 \leftarrow \theta_1^{meta}$, $\theta_2 \leftarrow \theta_2^{meta}$, $\phi \leftarrow \phi^{meta}$
27:   $\hat{\theta}_1 \leftarrow \theta_1$, $\hat{\theta}_2 \leftarrow \theta_2$, $\hat{\phi} \leftarrow \phi$
28:  end for
```

After completing the training of Meta-TD3, we will obtain the parameters of neural networks for tasks $\theta_1^{meta}, \theta_2^{meta}, \phi^{meta}$ and take them as prior knowledge. When dealing with new tasks, the model is trained on these parameters. New tasks can be achieved only through internal fine tune of Meta-TD3 algorithm.

4. Results

In this section, we set up comparative experiments to verify the implementation effect, training efficiency, and generalization ability of Meta-TD3 algorithm.

4.1. Experimental Platform and Environment Setting

The experimental environment of this paper is defined as 5000 m $\times$ 5000 m 2D plane in which the UAV can complete the target tracking task. Assuming that the UAV can the coordinates of the target through GPS in real time. The UAV is supposed to fly at a fixed altitude of 50 m and needs to approach the target as soon as possible and maintain the target within the distance range. Assuming that the target is moving on the ground, the motion mode of target will change in real time. At time t, if the horizontal distance d_t between the UAV and the target exceeds $L = 500$ m, the UAV will get a penalty $p = -1$. At time t, if the horizontal distance d_t between the UAV and the target is less than $D = 100$ m, it is considered that the UAV achieves the tracking at time t, the UAV will get a reward $r = 3$.

The parameters of the UAV motion model are set up as follows. Since the research object of this paper is the Quadrotor UAV, it is considered that UAV can complete the hovering. Therefore, the maximum speed of UAV is $v_{\max} = 20\,\text{m/s}$ and the minimum speed is $v_{\min} = 0\,\text{m/s}$. The maximum speed of target is 15 m/s and the minimum speed is 0. The acceleration of the UAV is $n_t \in [-2\ \text{m/s}^2, 2\ \text{m/s}^2]$ and the angular acceleration is $\omega_t \in [-3/s^2, 3/s^2]$. In order to unify the range of each variable and improve the efficiency of learning, each state variable is normalized into $[-1.0, 1.0]$. The coefficients in reward shaping are instantiated as $d_{\max} = 6000$ m, $\delta = 0.7$, $\varepsilon = 0.4$.

According to the definition of the state space and the definition of the action space, it is clear that the Meta-TD3 model has 6-dimensional input and 2-dimensional output. In Meta-TD3, all the policy neural networks have the same structure, and all the Q-value neural networks have the same structure. The policy neural networks and the Q-value neural networks are shown in Figure 6. The policy neural networks are constructed as $6 \times 256 \times 256 \times 2$ and the Q-value neural networks are constructed as $8 \times 256 \times 256 \times 1$. Except that the activation function of the last layer of the policy neural networks is tanh function and the last layer of the Q-value neural networks has no activation function, the remain layers are ReLU layers. The Adam optimizer is employed to optimize the actor module and the critic module, the learning rate is both 0.0003. Other hyper-parameters are set with soft update rate $\tau = 0.005$, the discounting factor $\gamma = 0.99$, the variance of gaussian noise $\sigma = 0.2$. When training, the parameters of actor module are updated as a frequency of 2. The maximum size of the task experience replay buffer R is set to 100,000, and the size of Minibatch is set to 256. In the simulation, the decision period dt is set to 1 s, and an episode contains 400 decision steps.

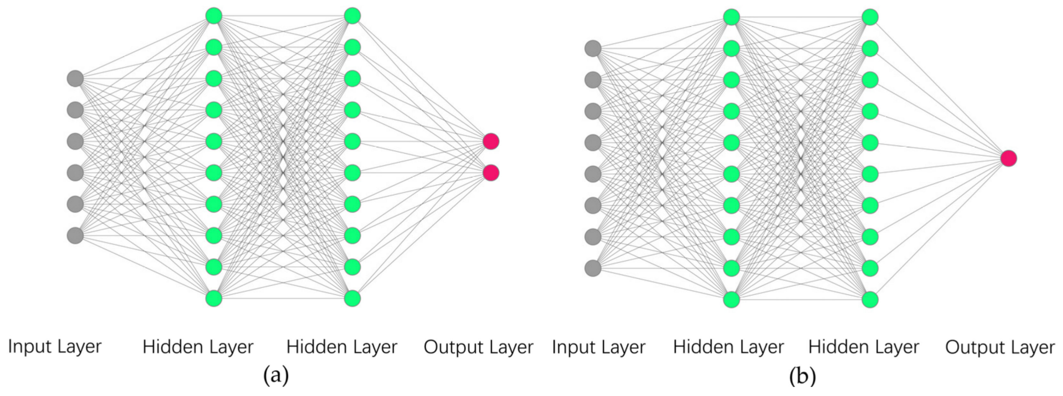

Figure 6. The structure of policy neural networks (**a**) and the Q-value neural networks (**b**).

4.2. Model Training and Testing

To verify the performance of Meta-TD3 in target tracking and the generalization in different environments, the experiment is divided into two parts: one is training in a series of tasks through Meta-TD3 algorithm to obtain a general tracking policy, and the other is to test the performance and the generalization of general policy in specific new tasks.

When training, the initial position of UAV is (0 m, −100 m), the initial position of target is (100 m, 100 m). The acceleration of target obeys the normal distribution $N(0.06, 0.1)$ with the value range of [−0.25,0.25], and the angular velocity of target obeys the value range of [–3,3] normal distribution $N(0, 3)$. Based on this, we generate five different target motion modes for training. The multi-task experience replay buffer contains five separate task experience replay buffer. The UAV is trained to track the target under the conditions of five different target motion modes, and stores the corresponding interactive data into the corresponding experience replay buffer. During training, for each task, samples of equal amount are randomly selected from the corresponding experience replay buffer for training. We train the Meta-TD3 algorithm for 1250 episodes and update the parameters of meta-learning every 10 steps.

In order to test the generalization performance of the parameters trained through Meta-TD3 algorithm, we set up three groups of different tasks for experiments. To be specific, we resort to other two state-of-the-art DRLs, DDPG and TD3, as baselines and re-implement them with almost the same hyper-parameter settings to the fine tune of Meta-TD3. We set up three tracking tasks according to increasing difficulty, and all three tasks are trained (fine-tuned) for 1000 episodes.

In Task 1, UAV is close to the target at the beginning, the initial speed direction of the UAV is basically consistent with the target's initial speed direction. The specific settings of Task 1 are as follows. The target initial position is (1000 m, 800 m), the initial speed is 3 m/s, the acceleration obeys the normal distribution $N(0.06, 0.1)$ with the value range of [−0.25, 0.25], and the angular velocity obeys the value range of [−3, 3] normal distribution $N(0.2, 3)$. The initial position of the UAV is (0 m, −300 m), and the initial speed of UAV is 4 m/s. After the start of the task, UAV will approach the target as soon as possible and achieve the distance to the target within the set range. Figure 7 shows the tracking trajectory with the Meta-TD3 weights for fine tune, TD3 and DDPG algorithms with randomly initialized network weights after 1000 episodes of training, and the distance between the UAV and the target at each decision step after 1000 episodes of training.

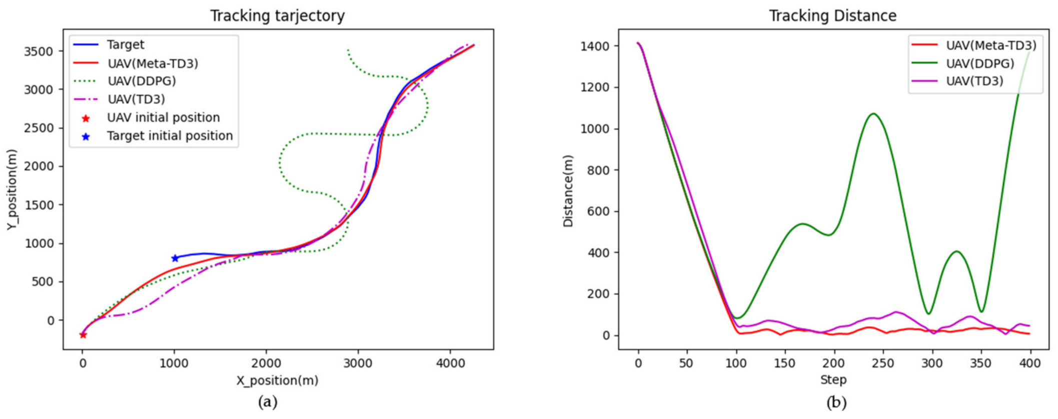

Figure 7. The tracking trajectory (**a**) and tracking distance (**b**) in Task 1.

In the of Figure 7a, the red star indicates the initial position of the UAV and the blue star indicates the initial position of the target. At the beginning of the Task 1, Meta-TD3, TD3 and DDPG can make the UAV quickly approach the target. It can be seen from the Figure 7a that the tracking trajectory of the fine tune using Meta-TD3 weights can be closer to the target trajectory, and the tracking effect is significantly better than that of the randomly initialized TD3 algorithm and the DDPG algorithm. As can be seen from the Figure 7b, both Meta-TD3 algorithm and TD3 algorithm can basically reach the tracking range at 100 decision steps and achieve stable tracking of the target while the DDPG algorithm cannot achieve stable tracking. Fine tune tracking with Meta-TD3 weights can ensure that the distance between the target is stable near 0 m, and the tracking stability and tracking effect are higher than the TD3 algorithm and the DDPG algorithm.

Figure 8 shows the average reward for Task 1. The average reward obtained by using Meta-TD3 weights for fine tune has been much higher than the other two algorithms within 1000 episodes, and the average reward can basically reach the end around 400 episodes and maintain convergence. But the average reward of TD3 algorithm and DDPG algorithm increases slowly during training, and the maximum average reward can basically be reached in 1000 episodes.

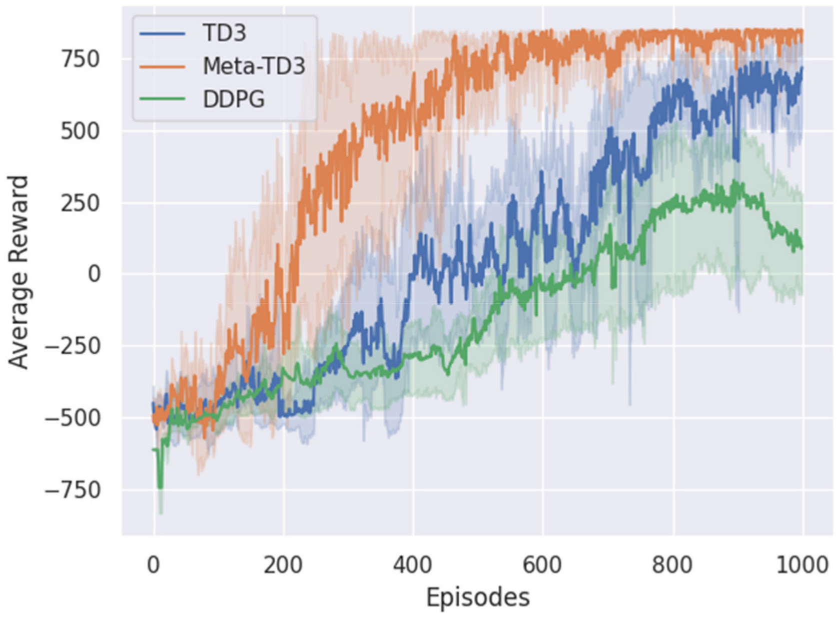

Figure 8. Average reward trends with respect to training episodes in Task 1.

In Task 2, the target initial position is (1000 m, 1000 m), the initial speed is 3 m/s, the acceleration obeys the normal distribution $N(0.06, 0.2)$ with the value range of [−0.25,0.25], and the angular velocity obeys the value range of [−3,3] normal distribution $N(0.2, 5)$. The initial position of the UAV is (500 m, 1300 m), and the initial speed of UAV is 4 m/s. Figure 9 shows the tracking trajectory with the Meta-TD3 weights for fine tune, TD3 and DDPG algorithms with randomly initialized network weights after 1000 episodes of training, and the distance between the UAV and the target at each decision step after 1000 episodes of training.

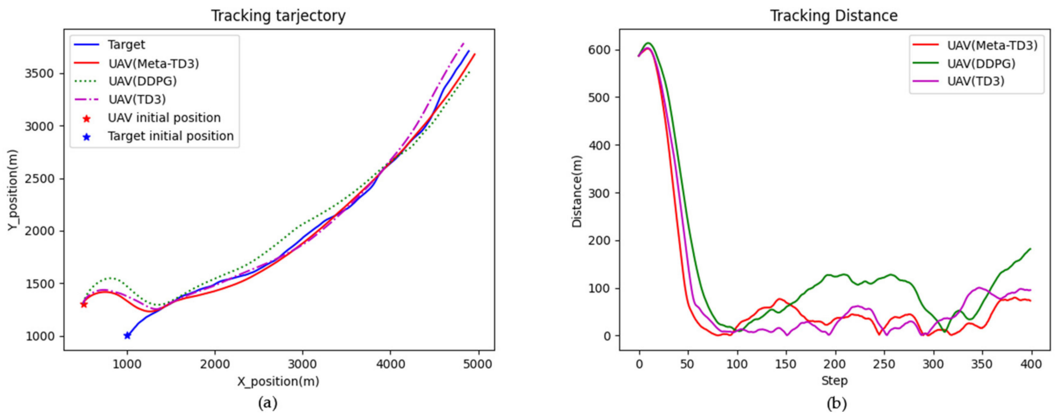

Figure 9. The tracking trajectory (**a**) and tracking distance (**b**) in Task 2.

As shown in Figure 9, all the three algorithms have achieved a good performance on target tracking. In the whole tracking process, fine tune with the parameters of Meta-TD3 algorithm can ensure that the distance between UAV and the target is smaller than the TD3 algorithm and DDPG algorithm. The average reward during training of the three algorithms is shown in Figure 10. For Task 2, the average reward of fine tune rises rapidly, reaching the maximum average reward in 300 episodes and maintaining convergence. The average reward obtained by TD3 and DDPG with randomly initialized training cannot reach the maximum value in 1000 episodes, and the average reward cannot converge.

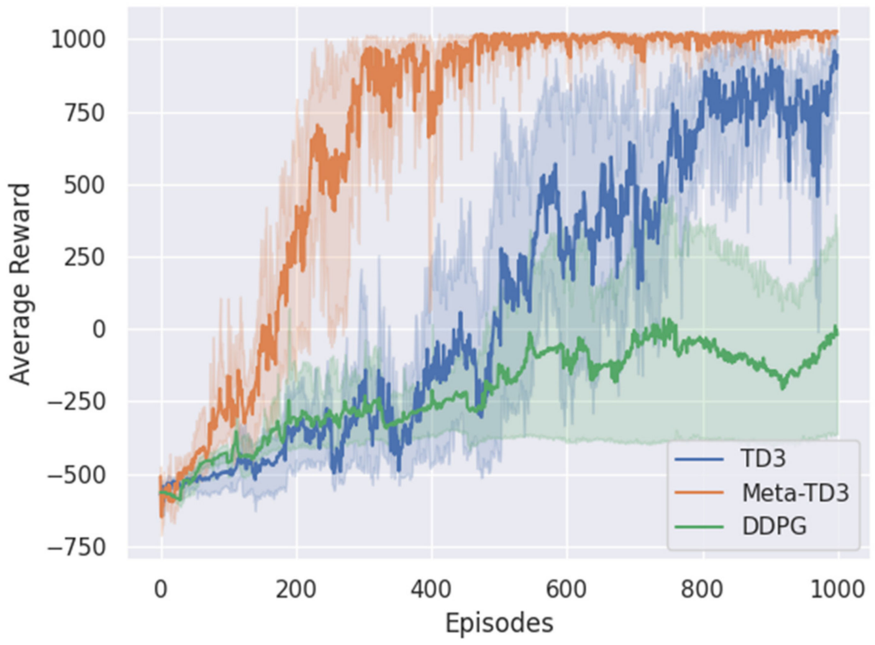

Figure 10. Average reward trends with respect to training episodes in Task 2.

In Task 3, the target initial position is (1000 m, 1000 m), the initial speed is 3 m/s, the acceleration obeys the normal distribution $N(0.06, 0.2)$ with the value range of [−0.25, 0.25], and the angular velocity obeys the value range of [−3,3] normal distribution $N(0.2, 5)$. The initial position of the UAV is (−800 m, 1300 m), and the initial speed of UAV is 4 m/s. Figure 11. shows the tracking trajectory with the Meta-TD3 weights for fine tune, TD3 and DDPG algorithms with randomly initialized network weights after 1000 episodes of training, and the distance between the UAV and the target at each decision step after 1000 episodes of training.

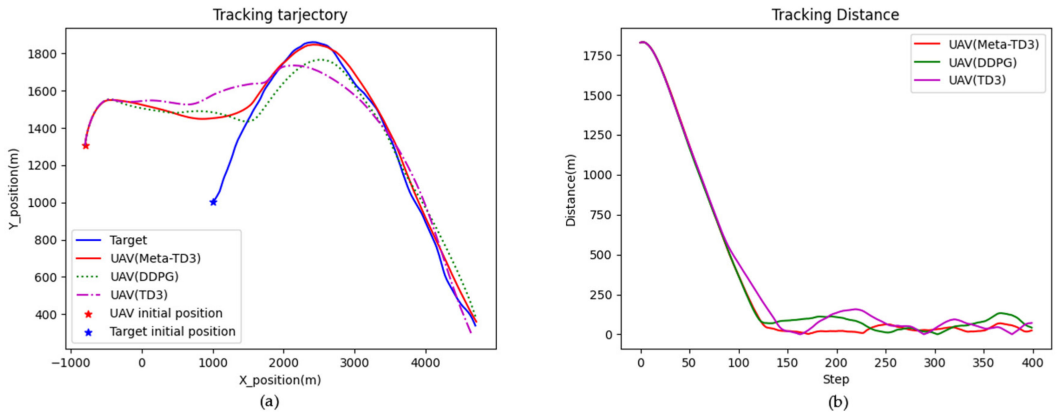

Figure 11. The tracking trajectory (**a**) and tracking distance (**b**) in Task 3.

In Figure 11, fine tune with the parameters of Meta-TD3 algorithm provides better effect to Task 3. It can be seen from the tracking distance that fine tune with the parameters of Meta-TD3 algorithm keeps a small tracking distance from the target. The average reward of Task 3 during training of the three algorithms is shown in Figure 12. Among the three algorithms, the average reward convergence speed of Meta-TD3 algorithm is much faster than other algorithms, and the maximum average reward of Meta-TD3 is higher than other algorithms.

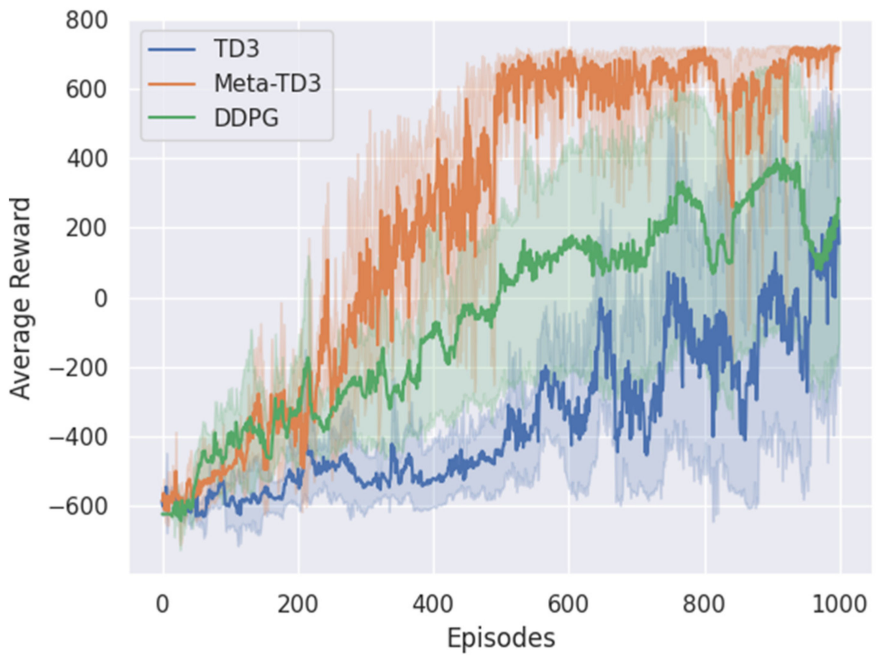

Figure 12. Average reward trends with respect to training episodes in Task 3.

5. Discussion

To evaluate the proposed Meta-TD3 algorithm, we conducted a large number of experiments in three different tasks by comparing with the TD3 and DDPG algorithms to verify the implementation effect, training efficiency and generalization ability of the Meta-TD3 algorithm.

For the target tracking effect, we compared the average reward, the average minimum tracking distance, and the average fluctuation range of the tracking distance (when UAV is close to the target) obtained by the three algorithms during target tracking in different environments. We conducted five comparative experiments under three different environments, and compared the average rewards obtained by different algorithms. The experimental results are shown in Table 2. It can be clearly seen that Meta-TD3 algorithm obtained the highest average reward in different environments, followed by TD3 algorithm, and DDPG algorithm obtained the lowest average reward. The results in Table 3 are the average minimum tracking distance, and the results in Table 4 are the fluctuation range of the average tracking distance. According to Tables 3 and 4, Meta-TD3 algorithm has the smallest average fluctuation range of the minimum tracking distance and the smallest average tracking distance under different tasks. Combined with Tables 2–4, the advantages of Meta-TD3 algorithm in tracking effect can be concluded. The great improvements come from the multi-tasks experience replay buffer and meta-learning update in Sections 3.2.1 and 3.2.2. The former ensures that multiple tasks data can be sampled during training, while the latter ensures that the algorithm learns multiple tasks during training. This enables the meta-TD3 algorithm to have better tracking effect in testing.

Table 2. Average reward over 5 trials. The best results from Meta-TD3 algorithm, for the three tasks, are highlighted in bold.

Environment	Meta-TD3	TD3	DDPG
Task 1	**847.24**	717.66	95.04
Task 2	**1027.45**	946.38	−20.42
Task 3	**715.84**	154.12	275.26

Table 3. Average minimum tracking distance at convergence over 5 trials. The best results from Meta-TD3 algorithm, for the three tasks, are highlighted in bold.

Environment	Meta-TD3	TD3	DDPG
Task 1	**0.79**	1.46	22.60
Task 2	**0.58**	4.11	45.77
Task 3	**2.02**	2.99	6.90

Table 4. The average fluctuation range at convergence over 5 trials. The best results from Meta-TD3 algorithm, for the three tasks, are highlighted in bold.

Environment	Meta-TD3	TD3	DDPG
Task 1	**40.91**	334.75	660.72
Task 2	**85.63**	150.80	274.12
Task 3	**88.77**	321.42	160.01

For the training efficiency and generalization ability, we set the average reward and the convergence speed of training as indicators, carried out three groups of experiments, and obtained the average reward change curve. Figures 8–12 shows the average reward changes of Meta-TD3, TD3, and DDPG algorithms during experiments. According to the results shown in the Figures 8–12, it can be clearly seen that the reward curve of Meta-TD3 can converge more quickly and the maximum reward is higher than TD3 and DDPG algorithms. The average rewards and convergence steps of different environments are shown in Tables 2 and 5. According to Table 2, it can be clearly seen that the average reward of Meta-TD3 is the highest in different environments. In Table 5, it can be found that Meta-TD3 algorithm has the fastest convergence speed in different environments, which is nearly 50% faster than TD3, while DDPG algorithm can hardly converge in the three environments. Therefore, it can be determined that the training efficiency of Meta-TD3 is faster. Combining Tables 3 and 4, we can determine that the tracking distance and tracking distance fluctuation range of the Meta-TD3 algorithm are the smallest when converging. The policy obtained by Meta-TD3 has stronger generalization ability, which can maintain the training efficiency while improving Tracking effect in different environments. Stronger generalization ability means that Meta-TD3 can be promoted in different environments. This is promoted by the two improvements proposed in this paper. Learning a variety of tasks during training can ensure that the obtained policy converges quickly in the new environment and maintains a better tracking effect.

Table 5. The number of steps required for convergence of various algorithms over 5 trials. The best results from Meta-TD3 algorithm, for the three tasks, are highlighted in bold.

Environment	Meta-TD3	TD3	DDPG
Task 1	**400 ± 50**	950 ± 20	
Task 2	**350 ± 30**	900 ± 50	
Task 3	**500 ± 20**		

In the implementation process, this paper assumes that the height of UAV is fixed, and the computer performs Meta-TD3 update on the simulation scene in advance to obtain the meta-policy. Based on the meta-policy, UAV can quickly adapt to specific tracking scenarios to obtain more adaptive Tracking policy. From the experimental results, it can be seen that Meta-TD3 has great advantages in tracking effect, training efficiency and generalization ability, and can provide support for target tracking of UAV in uncertain environment. Although we have done some work on tracking effects and generalization capabilities, there is still challenges to deploy the algorithm to a real UAV. In a real situation, the external air pressure change as time passes. It is difficult for a UAV to maintain maneuvering at a fixed altitude. No matter how well UAV is trained in a virtual environment, there is a big gap with reality. But we can continuously increase the reality of the scene and model in the virtual environment, and get closer to the reality. For future research, we will consider realizing more detailed UAV control, including controlling a UAV with motor speed, taking the image information obtained by a UAV's vision sensor and the ranging information from Lidar as state information, so as to realize target tracking in a scenario closer to a real-life three-dimensional situation.

6. Conclusions

This paper discusses UAV maneuvering target tracking in uncertain environments. We construct UAV motion controlled by acceleration and angular acceleration on a two-dimensional plane and define the UAV maneuvering target tracking as MDP. To ensure that the UAV autonomously achieves target tracking in uncertain environments, this paper proposes Meta-TD3 based on DRL and meta-learning, which can realize fast target tracking under uncertain target motion.

Based on actor-critic framework and TD3 algorithm, we construct UAV tracking model that can realize maneuvering target tracking in a single deterministic environment. Considering the tracing policy for a deterministic environment cannot be directly applied in the new environment, we design the method to learn more general tracking policy by meta-learning from a variety of different environments. We propose multi-tasks experience replay buffer to provide data for multi-tasks learning of DRL algorithm. We combine meta-learning to propose a multi-task reinforcement learning update method to ensure the generalization of the policy. Through a series of experiments, compared with the state-of-the-art algorithms DDPG and TD3, the performance and the generalization capability of Meta-TD3 have been verified. Experiments show that the Meta-TD3 is superior to the DDPG and TD3 algorithms in terms of convergence value and convergence speed. In the UAV target tracking problem, Meta-TD3 only requires a few steps to train to enable the UAV to adapt to the new target movement mode more quickly and maintain a better tracking effect.

Some future work includes the realization of more detailed UAV control to 3D space and target tracking in more complex environments.

Author Contributions: Resources, B.L.; validation, Z.G. and D.S.A.; writing—original draft preparation, B.L. and Z.G.; writing—review and editing, supervision, D.C.; funding acquisition, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the Aeronautical Science Foundation of China under grant no. 2017ZC53021, the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University under grant no. ZZ2019159, and the Open Project Fund of CETC Key Laboratory of Data Link Technology under grant no. CLDL-20182101.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fu, C.; Carrio, A.; Olivares-Mendez, M.A.; Suarez-Fernandez, R.; Campoy, P. Robust real-time vision-based aircraft tracking from unmanned aerial vehicles. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–5 June 2014; IEEE; Piscataway, NJ, USA, 2014; pp. 5441–5446.
2. Olivares-Mendez, M.A.; Fu, C.; Ludivig, P.; Bissyandé, T.F.; Kannan, S.; Zurad, M.; Annaiyan, A.; Voos, H.; Campoy, P. Towards an autonomous vision-based unmanned aerial system against wildlife poachers. *Sensors* **2015**, *15*, 31362–31391. [CrossRef] [PubMed]
3. Birk, A.; Wiggerich, B.; Bülow, H.; Pfingsthorn, M.; Schwertfeger, S. Safety, security, and rescue missions with an unmanned aerial vehicle (UAV). *J. Intell. Robot. Syst.* **2011**, *64*, 57–76. [CrossRef]
4. Fu, C.; Carrio, A.; Campoy, P. Efficient visual odometry and mapping for unmanned aerial vehicle using ARM-based stereo vision pre-processing system. In Proceedings of the 2015 International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, 9–12 June 2015; pp. 957–962.
5. Li, B.; Wu, Y. Path Planning for UAV Ground Target Tracking via Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 29064–29074. [CrossRef]

6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
7. Huang, H.; Yang, Y.; Wang, H.; Ding, Z.; Sari, H.; Adachi, F. Deep reinforcement learning for UAV navigation through massive MIMO technique. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1117–1121. [CrossRef]
8. Wu, C.; Ju, B.; Wu, Y.; Lin, X.; Xiong, N.; Xu, G.; Li, H.; Liang, X. UAV autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access* **2019**, *7*, 117227–117245. [CrossRef]
9. Wang, C.; Wang, J.; Shen, Y.; Zhang, X. Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2124–2136. [CrossRef]
10. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019.
11. Wan, K.; Gao, X.; Hu, Z.; Wu, G. Robust Motion Control for UAV in Dynamic Uncertain Environments Using Deep Reinforcement Learning. *Remote. Sens.* **2020**, *12*, 640. [CrossRef]
12. Bhagat, S.; Sujit, P.B. UAV Target Tracking in Urban Environments Using Deep Reinforcement Learning. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020; pp. 694–701.
13. Hayat, S.; Yanmaz, E.; Brown, T.X.; Bettstetter, C. Multi-objective UAV path planning for search and rescue. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5569–5574.
14. Mukherjee, A.; Misra, S.; Sukrutha, A.; Raghuwanshi, N.S. Distributed aerial processing for IoT-based edge UAV swarms in smart farming. *Comput. Netw.* **2020**, *167*, 107038. [CrossRef]
15. Yang, B.; Cao, X.; Yuen, C.; Qian, L. Offloading Optimization in Edge Computing for Deep Learning Enabled Target Tracking by Internet-of-UAVs. *IEEE Internet Things J.* **2020**, 1. [CrossRef]
16. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. *arXiv* **2017**, arXiv:1709.06560.
17. Zhang, A.; Wu, Y.; Pineau, J. Natural environment benchmarks for reinforcement learning. *arXiv* **2018**, arXiv:1811.06032.
18. Liu, H.; Socher, R.; Xiong, C. Taming maml: Efficient unbiased meta-reinforcement learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 4061–4071.
19. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
20. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv* **2017**, arXiv:1703.03400.
21. Li, Z.; Zhou, F.; Chen, F.; Li, H. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv* **2017**, arXiv:1707.09835.
22. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525.
23. Imanberdiyev, N.; Fu, C.; Kayacan, E.; Chen, I.-M. Autonomous navigation of UAV by using real-time model-based reinforcement learning. In Proceedings of the 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, Thailand, 13–15 November 2016; pp. 1–6.
24. Zhou, D.; Schwager, M. Vector field following for quadrotors using differential flatness. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6567–6572.
25. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
26. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014.
27. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.

28. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2000; pp. 1057–1063.
29. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
30. Roderick, M.; MacGlashan, J.; Tellex, S. Implementing the deep q-network. *arXiv* **2017**, arXiv:1711.07478.
31. Yadav, A.K.; Gaur, P. AI-based adaptive control and design of autopilot system for nonlinear UAV. *Sadhana* **2014**, *39*, 765–783. [CrossRef]
32. Peters, J.; Schaal, S. Policy gradient methods for robotics. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 2219–2225.
33. Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.* **1992**, *8*, 293–321. [CrossRef]
34. Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; Lillicrap, T. Meta-learning with memory-augmented neural networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1842–1850.
35. Lake, B.M.; Ullman, T.D.; Tenenbaum, J.B.; Gershman, S.J. Building machines that learn and think like people. *Behav. Brain Sci.* **2017**, *40*, e253. [CrossRef] [PubMed]
36. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.