


## Article

# A State-Compensated Deep Deterministic Policy Gradient Algorithm for UAV Trajectory Tracking

Jiying Wu , Zhong Yang \*, Luwei Liao, Naifeng He, Zhiyong Wang and Can Wang

College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; wujiying@nuaa.edu.cn (J.W.); llw@nuaa.edu.cn (L.L.); nfhe@nuaa.edu.cn (N.H.); wangzhiyong@nuaa.edu.cn (Z.W.); wangcan@nuaa.edu.cn (C.W.)

\* Correspondence: yangzhong@nuaa.edu.cn

**Abstract:** The unmanned aerial vehicle (UAV) trajectory tracking control algorithm based on deep reinforcement learning is generally inefficient for training in an unknown environment, and the convergence is unstable. Aiming at this situation, a Markov decision process (MDP) model for UAV trajectory tracking is established, and a state-compensated deep deterministic policy gradient (CDDPG) algorithm is proposed. An additional neural network (C-Net) whose input is compensation state and output is compensation action is added to the network model of a deep deterministic policy gradient (DDPG) algorithm to assist in network exploration training. It combined the action output of the DDPG network with compensated output of the C-Net as the output action to interact with the environment, enabling the UAV to rapidly track dynamic targets in the most accurate continuous and smooth way possible. In addition, random noise is added on the basis of the generated behavior to realize a certain range of exploration and make the action value estimation more accurate. The OpenAI Gym tool is used to verify the proposed method, and the simulation results show that: (1) The proposed method can significantly improve the training efficiency by adding a compensation network and effectively improve the accuracy and convergence stability; (2) Under the same computer configuration, the computational cost of the proposed algorithm is basically the same as that of the QAC algorithm (Actor-critic algorithm based on behavioral value Q) and the DDPG algorithm; (3) During the training process, with the same tracking accuracy, the learning efficiency is about 70% higher than that of QAC and DDPG; (4) During the simulation tracking experiment, under the same training time, the tracking error of the proposed method after stabilization is about 50% lower than that of QAC and DDPG.

**Keywords:** trajectory tracking; deep reinforcement learning; deep deterministic policy gradient algorithm; state compensation network



**Citation:** Wu, J.; Yang, Z.; Liao, L.; He, N.; Wang, Z.; Wang, C. A State-Compensated Deep Deterministic Policy Gradient Algorithm for UAV Trajectory Tracking. *Machines* **2022**, *10*, 496. <https://doi.org/10.3390/machines10070496>

Academic Editor: Dan Zhang

Received: 20 May 2022

Accepted: 14 June 2022

Published: 21 June 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Trajectory tracking is related to time series and the mobile robot needs to reach the original set position through the control of the trajectory tracking system within a specified time. It is dedicated to estimating the motion state and motion trajectory of the tracked object in a continuous spatiotemporal sequence. Therefore, a controller with high-performance trajectory tracking capability is required for the mobile robot [1]. With the development of control technology, the learning-based robot control method is the latest research hotspot in the field of control [2–4], and researchers have proposed methods based on reinforcement learning (RL). It ignores the dynamic model of the robot and learns the control method through a large amount of motion data, which has received extensive attention in the field of automatic control. Due to its data-driven approach, reinforcement learning can reduce the need for complex engineering theory. Model-based reinforcement learning algorithms generally target simple dynamic environments or agents with few actual interactions. Without knowledge of the environmental dynamics model, model-free

reinforcement learning algorithms can directly evaluate the quality of a policy or find the optimal value function and optimal policy through the actual interaction between individuals and the environment. Internationally, a lot of research has been done on this problem from theory to experiment, and fruitful results have been achieved in theoretical analysis, numerical calculation and experimental verification [5–13]. Among them, the value-based deep reinforcement learning (DRL) algorithms, such as Q-learning, Sarsa, and the deep Q-learning algorithm (DQN) can only realize the control of discrete action space, so they can only realize the discrete direction control of the robot. However, in the case of a large-scale behavior space or continuous behavior, it will be difficult for value-based reinforcement learning to learn a good result. For trajectory tracking, value-based control methods can hardly achieve accurate tracking by simply using discrete action spaces. In this case, policy learning can be performed directly.

The policy gradient algorithm proposed by Peters [14] realizes continuous control, and the deterministic policy gradient (DPG) proposed by Silver [15] significantly outperforms the stochastic policy gradient in high-dimensional behavior spaces. Lillicrap [16] introduced DQN on the basis of DPG, combined with the training characteristics of the DQN neural network, and proposed the DDPG algorithm. Due to the good performance of DDPG, many researchers applied DDPG to UAV navigation and target tracking, and achieved good results [17]. Gan Z et al. [18] established a UAV tracking model based on deep reinforcement learning, and studied the target tracking problem of UAV. Modares et al. [19] used an off-policy reinforcement learning algorithm to learn and track the solution of the Hamilton-Jacobi-Isaac (HJI) equation online, and studied the design of an H-tracking controller for nonlinear continuous-time systems with completely unknown dynamics. Ye L et al. [20] proposed reinforcement learning tracking control (RLTC) for unknown continuous dynamic systems to solve the traditional iterative learning control (ILC) problem.

The DDPG algorithm is a deterministic policy algorithm using deep learning technology and based on the Actor-critic algorithm, which can better solve the trajectory tracking problem in continuous behavior space [8]. Among the latest research results, Luy NT et al. [21] proposed a reinforcement learning-based design method for mobile robot kinematics and dynamic tracking control algorithms, in which the Actor-critic structure uses only one neural network to reduce computational costs and storage resources. In response to the time-consuming RL training, Wang GF et al. [22] proposed a TL framework based on transfer learning, where the agent extracts knowledge from the human-demonstration trajectories of the source task, and reuses the knowledge in RL in the target task, which shows remarkable results in experiments. Levine S et al. [23] proposed a deep reinforcement learning method based on uncertainty perception. By estimating the probability of collision, the robot can keep “vigilance” in the face of unfamiliar and unknown environments, reduce the running speed and reduce the possibility of collision. In order to reduce the number of trials and errors in the interaction between the UAV and the environment, they also proposed a guided policy search algorithm which uses the optimized data for learning to realize the obstacle avoidance control of the UAV in the simulation environment. Hwangbo J et al. [24] proposed a new deep reinforcement learning algorithm under the Actor-critic framework to achieve UAV path tracking control under any initial condition. However, William F K et al. [25] compared three deep reinforcement learning algorithms, namely, deep deterministic policy gradient, confidence region policy optimization, and near-end policy optimization, with the traditional PID (Proportion Integration Differentiation) algorithm. The experiment shows that in the UAV attitude control simulation environment, the near-end policy optimization algorithm is superior to the other three control algorithms in terms of overshoot, rise time and tracking error. B Rubí et al. [26] proposed the DDPG algorithm for the path tracking problem of the UAV aircraft. Through training and testing in the RotorS-Gazebo environment, it was proved that the tracking performance of the method was better than that of the NLGL (Nonlinear Navigation Logic) method. Zheng Q et al. [27] used the PPO algorithm of reinforcement learning to adjust the PID controller gain, and achieved good stability of the aircraft in control, anti-jamming and

flying height. In addition, Zhen Y et al. [28,29] proposed a hybrid DDPG (Mi-DDPG) algorithm. Levine S et al. [30] formulated policy search as optimization of trajectory distribution. Yang B et al. [31] used a deep deterministic policy gradient algorithm to solve a closed-loop tracking method for tracking trajectories, and proved that it achieved a high tracking rate. Most of the above-mentioned previous studies on DDPG focus on the optimization problem of DDPG algorithm itself, or use a DDPG algorithm to adjust the parameters of traditional controllers to increase the stability of the system.

Aiming at the shortcomings of the UAV trajectory tracking control algorithm based on the deep deterministic policy gradient, such as low training efficiency and unstable convergence in unknown environments, a new UAV trajectory tracking control algorithm was proposed on the basis of DDPG, that is, state-compensated deep deterministic policy gradient (CDDPG) algorithm which uses deep reinforcement learning and fuses different state space networks. The process is shown in Figure 1. The CDDPG algorithm takes improving the learning efficiency without reducing the training accuracy as a breakthrough point. In practice, the state compensation network is used to compensate the dynamic loss in real time to improve the following effect. In the subsequent process, the role of the compensation network is constantly weakened, and reinforcement learning is the dominant factor to achieve fast and accurate follow-up effects. For example, before learning a certain skill, students first roughly learn most of the content; the instructor will correct the study direction and guide them precisely to speed up the learning progress and accuracy; the cycle goes back and forth. The students' self-study gradually becomes dominant, while weakening the tutor's guidance, and they finally achieve a quick and accurate mastery of this skill. It can be seen that the advantages of CDDPG are the stable convergence and the short training period.

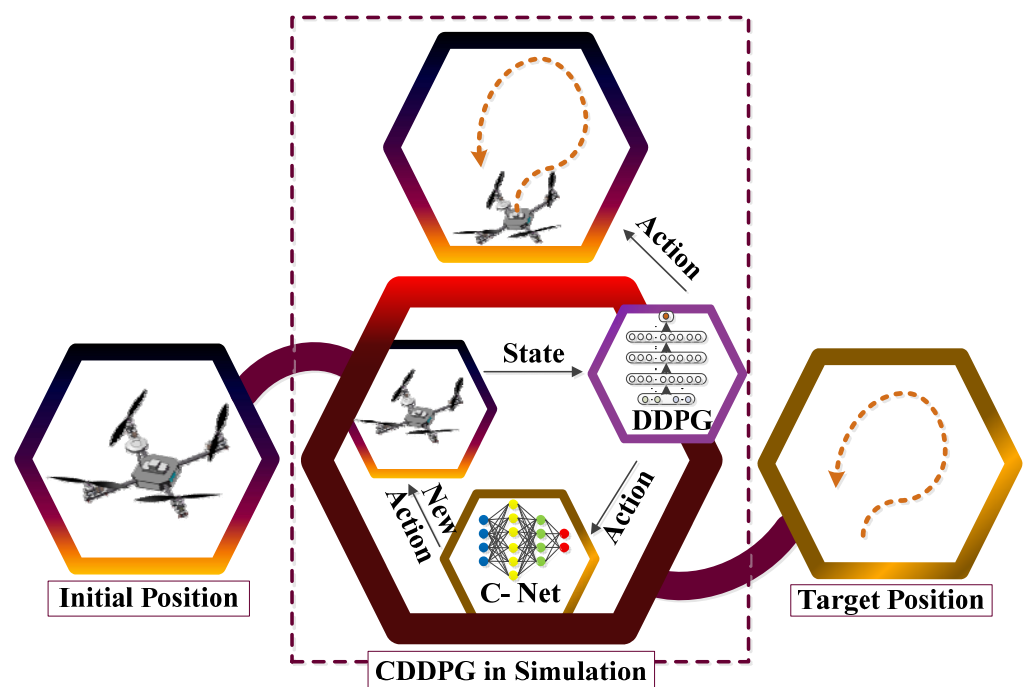


Figure 1. Principle of CDDPG algorithm.

The rest of this article is organized as follows. The second section presents the proposed control method. The third section verifies the performance of the method through simulation results. The fourth section is devoted to concluding remarks.

## 2. Models and Methods

Section 2.1 establishes the Markov decision process model for UAV trajectory tracking and describes the problem formulation of policy search in a non-model-based RL frame-

work. Section 2.2 presents the proposed method. In this paper, the policy is the controller learned by RL, which is updated after the learning iteration. Also, the policy formulation is undertaken in conjunction with the state compensation.

2.1. Problem Formulation

Consider a UAV dynamic tracking system with state  $s \in R^s$  and system input  $a \in R^a$ , given by

$$s_{t+1} = f(s, a) + o_t \tag{1}$$

where  $f(s, a)$  is an unknown function and  $o_t$  is a noise function. Model-free reinforcement learning learns the dynamics  $f(s, a)$  by continuously updating  $\{(s, a), s_{t+1}\}$ , where  $t$  represents the time steps recorded in all previous steps during training. The dataset is continuously updated during learning, assuming that all states are measurable.

In reinforcement learning, a trajectory tracking Markov decision process (MDP) is defined. The UAV tracking MDP consists of a tuple  $\langle S, A, P, R, \gamma \rangle$ , where:

- S is a finite state set;
- A is a finite behavior set;
- P is the behavior-based state transition matrix in the set:

$$P_{ss_{t+1}}^a = E[R_{t+1} | S_t = s, A_t = a] \tag{2}$$

R is the state and behavior-based reward function:

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a] \tag{3}$$

$\gamma$  is a decay factor:  $\gamma \in [0, 1]$ .

According to the Bellman equation, two Bellman equations of the state-value function  $v_\pi(s)$  and behavior-value function  $q_\pi(s, a)$  based on the policy  $\pi$  are obtained:

$$v_\pi(s) = E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \tag{4}$$

$$q_\pi(s, a) = E[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \tag{5}$$

As we know, the value of an action is expressed in terms of the value of subsequent states that the action can reach:

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s_{t+1} \in S} P_{ss_{t+1}}^a \sum_{a_{t+1} \in A} \pi(a_{t+1} | s_{t+1}) q_\pi(s_{t+1}, a_{t+1}) \tag{6}$$

The purpose of reinforcement learning is to find an optimal policy that allows individuals to obtain more gains than other policies in the process of interacting with the environment. This optimal policy is represented by  $\pi^*$ .

Here, given the current state  $s \in S$ , predict the probability of taking the next action  $a \in A$ , and then obtain the prediction for the next state  $s_{t+1}$ :

$$s_{t+1} = f(S_t = s, A_t = a | \pi_\theta(s, a)) + o_t \tag{7}$$

$$\pi_\theta(s, a) = P[A_t = a | S_t = s, \theta] \tag{8}$$

where  $\pi_\theta$  is a policy function,  $\theta$  is the parameter of  $\pi_\theta$ ,  $P[A_t = a | S_t = s, \theta]$  represents the probability of taking any possible action  $a$  under the setting of a given state  $s$  and certain parameters  $\theta$ . The Markov reward process is expressed as:

$$R_s^\pi = \sum_{a \in A} \pi_\theta(a | s) R_s^a \tag{9}$$

In (7), the training input state is  $S = [s_1, s_2, \dots, s_n]$ , and the output action is  $A = [a_1, a_2, \dots, a_n]$ . The purpose of the article is to design the objective function  $J(\theta)$ , find the

optimal parameters  $\theta$  through the gradient ascent method, and find the optimal policy  $\pi^*$  to achieve accurate target tracking.

## 2.2. Proposed Policy

In a DDPG algorithm, the parameters of the policy are randomly initialized to small values. In our experience, the DDPG algorithm is easily biased towards exploration, and it takes an extremely long time to complete each learning process with random actions in the early stage, making a minor or meaningless policy update. Since multiple iterations of learning are unavoidable, it is inefficient to apply the pre-learning DDPG algorithm in practice.

Let the policy of the original RL be  $\pi_{RL}$ , and the proposed policy be  $\pi_{C-RL}$ . Then, the hybrid control policy is:

$$\pi_{C-RL} = \pi_{RL} + \pi_C \quad (10)$$

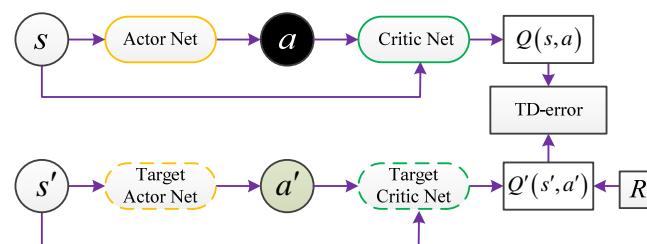
where  $\pi_C$  is the compensatory control policy.

The DDPG algorithm is a typical representative of algorithms commonly used in the reinforcement learning problem in continuous action space. The proposed policy chooses to combine the DDPG algorithm and the state compensation network. As the DDPG algorithm is based on the Actor-critic algorithm, two hybrid algorithms, CQAC and CDDPG, are discussed below.

### 2.2.1. DDPG Algorithm

The Actor-critic algorithm consists of a policy function and a behavior value function, where the policy function acts as an actor, generating behavior and interacting with the environment; the behavior value function acts as a critic, which is responsible for evaluating the actor's performance and guiding the actor's follow-up actions. The QAC algorithm does not require a complete state sequence, but since the introduced critic is still an approximate value function, there is a possibility of introducing bias.

The DDPG algorithm is derived from the DQN algorithm, which is a typical reinforcement learning algorithm for solving continuous control problems. It is based on the Actor-critic algorithm but is different from it, and it adds a noise function to the deterministic behavior in the learning process to realize small-scale behavior exploration. Moreover, backup parameters are added to the actor network and the critic network as the target network, respectively, so as to achieve a dual parameter setting and increase the convergence probability (Figure 2).



**Figure 2.** Transformation process in DDPG algorithm.

The conversion process is outlined below: (1) The current state  $s$  generates specific behavior  $a$  through the actor network; (2) The critic network calculates the behavior value  $Q(s, a)$  corresponding to  $s$  and  $a$ ; (3) The target actor network generates  $a'$  used to estimate the value according to the subsequent state  $s'$  given by the environment; (4) The target critic network generates the behavior value  $Q'(s', a')$  used to calculate the target value  $Q_{Target}$  according to  $s', a'$  and the reward  $R$ .

The task of the actor in the DDPG algorithm is to find the optimal behavior to maximize the value of the output behavior, where the reward obtained by executing  $a$  is  $R$ , and this transformation process is stored as  $(s, a, R, s)$ . After the system stores enough

transformation process, it randomly selects  $X$  samples  $(s_i, a_i, R_{i+1}, s_{i+1})$  for calculation, where  $i = 1, 2, \dots, X$ , the calculation process of the objective function  $Q_{Target}$  is [16]:

$$Q_{Target} = R_{i+1} + \gamma Q'(s_i, A'(s_i | \theta^{A'})) | \theta^{Q'} \tag{11}$$

The loss of the Critic network is TD-error (temporal-difference error):

$$Loss = \frac{1}{X} \sum_i (Q_{Target} - Q(s_i, a_i | \theta^Q))^2 \tag{12}$$

Updating actor network using gradient ascent:

$$\nabla_{\theta^A} J \approx \frac{1}{X} \sum \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=A(s_i)} \nabla_{\theta^A} A(s | \theta^A) |_{s_i} \tag{13}$$

Update the target network:

$$\begin{aligned} \eta \theta^Q + (1 - \eta) \theta^{Q'} &\rightarrow \theta^{Q'} \\ \eta \theta^A + (1 - \eta) \theta^{A'} &\rightarrow \theta^{A'} \end{aligned} \tag{14}$$

where  $\gamma$  is the decay factor,  $\eta$  is the update rate parameter;  $\theta^Q$  and  $\theta^A$  are the weights to the Critic network value function  $Q(s, a | \theta^Q)$  and the actor network  $A(s | \theta^A)$ ;  $\theta^{Q'}$  and  $\theta^{A'}$  are the weights to the target Critic network value function  $Q'(s, a | \theta^{Q'})$  and the target actor network  $A'(s | \theta^{A'})$ .

### 2.2.2. CDDPG Algorithm

The state compensation network is added to DDPG algorithm to form a CDDPG algorithm controller. The control policy of the hybrid controller  $\pi_{CDDPG}$  is:

$$\pi_{CDDPG}(\theta, \omega) = \pi_{DDPG}(\theta) + \pi_C(\omega) \tag{15}$$

where  $\theta$  is the parameter of the DDPG control policy  $\pi_{DDPG}$ ,  $\omega$  is the parameter of the state compensation control policy  $\pi_{DDPG}$ . In this algorithm, all parameters are updated iteratively, as  $\theta$  of  $\pi_{DDPG}$  is determined by the gradient ascent method and  $\omega$  of  $\pi_C$  is determined by the gradient descent method.

Rewrite Equations (11) and (12) to get the following deduction equation.

$$Q_{Target} = R_{i+1} + \gamma Q'(s_i, A'(s_i | \theta^{A'}, \omega^{C'})) \tag{16}$$

$$Loss = \frac{1}{X} \sum_i (Q_{Target} - (Q(s_i, a_i | \theta^Q, \omega^C)))^2 \tag{17}$$

Therefore, the objective gradient function can be obtained.

$$\begin{aligned} \nabla_{\omega^C} J &\approx \frac{1}{X} \sum \nabla_a Q(s, a | \theta^Q) |_{s=s_i, a=C(s_i)} \nabla_{\omega^C} C(s | \omega^C) |_{s_i} \\ \nabla J &= \nabla_{\theta^A} J + \nabla_{\omega^C} J \end{aligned} \tag{18}$$

$$\begin{aligned} \Delta \omega &= -\varepsilon \nabla_{\omega^C} J \\ \omega^{C'} + \Delta \omega &\rightarrow \omega^{C'} \end{aligned} \tag{19}$$

Above formula, where  $\varepsilon$  is a learning rate of the state compensation network (C-Net)  $C(s | \omega^C)$ ;  $\omega^C$  is the weight to C-Net;  $\omega^{C'}$  is the weight to the target state compensation network (TC-Net)  $C'(s | \omega^{C'})$ . Figure 3 and Algorithm 1 outlines the proposed algorithm, where  $\delta_a$  and  $\delta_{a'}$  are the compensated actions calculated by C-Net and TC-Net.



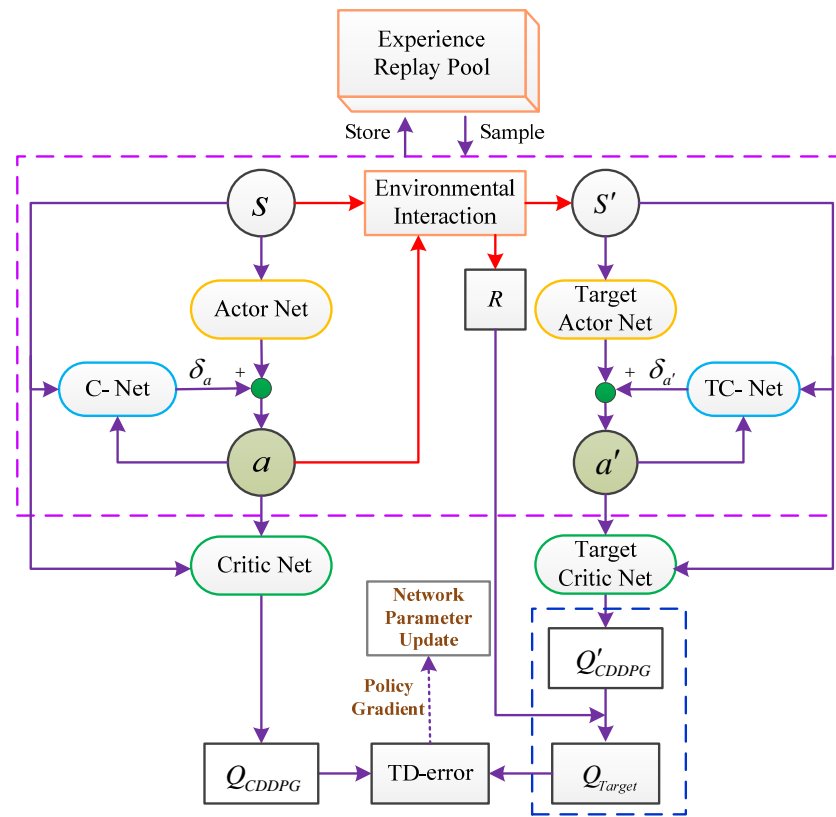


Figure 3. CDDPG algorithm schematic diagram.

---

#### Algorithm 1: CDDPG Algorithm

---

**Input:**  $\gamma, \eta, \epsilon, \theta^Q, \theta^A, \omega^C$

**Output:** optimized  $\theta^Q, \theta^A, \omega^C$

randomly initialize  $Q(s, a | \theta^Q), A(s | \theta^A), C(s | \omega^C)$  with weights  $\theta^Q, \theta^A, \omega^C$

initialize target network  $Q'(s, a | \theta^{Q'}), A'(s | \theta^{A'}), C'(s | \omega^{C'})$  with weights  $\theta^{Q'} \leftarrow \theta^Q, \theta^{A'} \leftarrow \theta^A, \omega^{C'} \leftarrow \omega^C$

initialize the experience cache space  $\Gamma$

**for** episode from 1 **to** Limit **do**

Initialize a noise function and receive the initial state

**for**  $t = 1$  **to**  $T$  **do**

Perform action  $a_t = A(s_t | \theta^A) + C(s_t | \omega^C) + O_t$ , get reward  $R_{t+1}$

next state  $s_{t+1}$ , store transition  $(s_t, a_t, R_{t+1}, s_{t+1})$  in  $\Gamma$

sample a batch of random  $X$  transition  $(s_i, a_i, R_{i+1}, s_{i+1})$  in  $\Gamma$ ,  
where  $i = 0, 1, 2, \dots, X$

set target value function via equation (16)

update critic by minimizing the loss via equation (17)

update actor policy by policy gradient via equation (18)

update networks via equation (14), (19)

**end for**

**end for**

---

The above is the inference of the CDDPG algorithm based on the DDPG algorithm, but the above logic analysis is also applicable to the QAC algorithm, and the QAC algorithm combined with the state compensation network is recorded as CQAC in the same way. We compare the performance of using the QAC and DDPG algorithm with the CQAC and CDDPG algorithm in the third section.

### 3. Simulations and Results Discussion

The algorithm is solved with the UAV position and velocity parameters  $s = [p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z]^T$  known. Assuming that the attitude control is stable, this paper focuses on the driving force control of the UAV in three-dimensional space of the Cartesian coordinate system.

In the algorithm simulations, the current state feature  $(p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z)$  of the UAV and the following target position  $(t_x, t_y, t_z)$  are used as the state input of the algorithmic framework algorithm, and its drive acceleration  $(a_x, a_y, a_z)$  in the X, Y and Z directions of the three-dimensional space is used as the output. The learning process is to repeat the policy function solution process. During the trajectory tracking process, the measurement and control system will transmit the current position information, speed information and coordinate data of the following target to the UAV in each time step, and at the same time determine the reward value. The larger the following error, the lower the reward value, and the maximum reward value is zero.

#### 3.1. Training Framework and Results

In CDDPG, the critic network evaluates the value of the UAV in the current state to guide the policy generation action, and the actor network is responsible for generating specific actions according to the current state. The input accepted by the critic network is the number of features observed by the UAV and the number of features of the action, and the output is the value of the state-action. Considering the need of the UAV for feature extraction of the observed state, we design a total of 3 hidden layers in critic. Since the number of elements in the input layer is the sum of the number of state elements and the number of behavior elements, the number of hidden layers is determined by the effect and delay of the algorithm. If the number of hidden layers is too large, the delay of the algorithm will be higher, and if the number of hidden layers is too small, the accuracy of the algorithm will be reduced, and the effect will be worse. Taking the above factors into consideration, the input of the critique network designed in this paper contains nine states and three actions, and the output is a Q value. After several simulations, to maintain a good training effect without too much delay, the hidden layer in this paper is set to three layers. The hidden layer that processes the state and the action is operated separately first, and is fully connected through the last hidden layer, and then outputs the state-action value together. The input to the actor network is the state observed by the UAV, and the output is the action the UAV will perform. The actor network is designed with a total of the hidden layers, and the layers are fully connected. The architecture of this network is shown in Figure 4. In order to realize the exploration, the algorithm adopts the Ornstein-Uhlenbeck process as the noise model, and adds a random noise on the basis of the generated action so that it can realize a certain range of exploration around the exact action.

In the compensation simulation, the method performed is to train C-Net with a neural network and optimize the network with its tracking average error as the objective function. The input of the network is the state after the UAV performs the action, and the output is the target difference, which will be weighted into the output action to interact with the environment. The C-Net is designed with a total of three hidden layers, and the layers are fully connected (Figure 5). After this neural network is embedded in the controller, trajectory tracking training is performed.

In the simulation, the tracking accuracy is set as  $r$ , that is, if the UAV is located within the range of the tracking target as the center and within the radius of  $r$ , it is a successful follower, and a larger reward can be obtained. The simulation is carried out using a deep reinforcement learning framework incorporating a state compensation network. The proposed method is simulated with OpenAI Gym, and the simulated computer configuration is an Intel(R) Core(TM) i5-7300HQ. The UAV flight range is a three-dimensional space of  $10 \times 10 \times 10$  (m). Taking the tracking accuracy  $r$  as 0.3 m, the proposed method is used to





that from the 11th episode, the rewards after that are all close to zero, but judging from the previous episodes rewards, the CQAC algorithm and CDDPG algorithm during the training process have increased significantly compared to QAC and DDPG. It can also be seen from the curve of the iteration steps experienced by each episode in Figure 7. In the early stage of training, the QAC and DDPG algorithms have to go through tens of thousands of iterations for each complete episode, while the iteration steps required by CQAC and CDDPG is only 30% of the original algorithm. Since the calculated time cost of the four algorithms are basically the same (about 0.998 milliseconds, Figure 8) under the same computer configuration, it fully shows that the training time is improved by about 70% with the same training accuracy. The same conclusion can also be confirmed by the comparison of total iterations of the four algorithms (Figure 9).

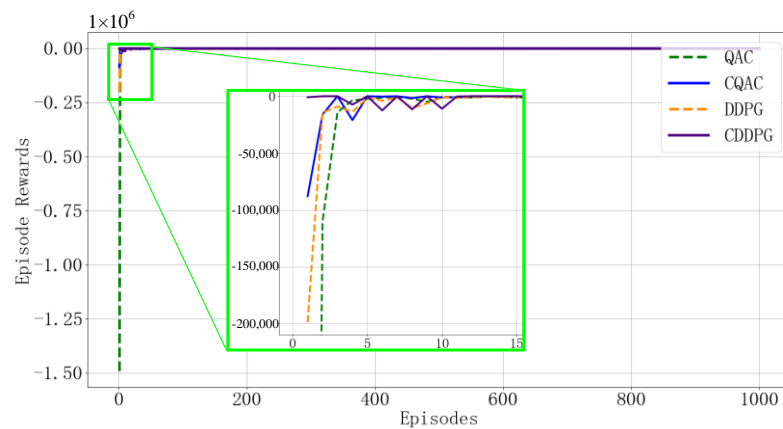


Figure 6. Training episode rewards of four algorithms.

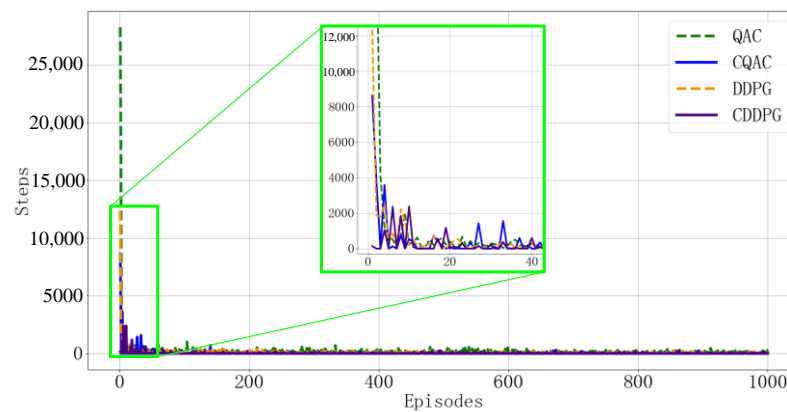


Figure 7. Training steps of four algorithms.

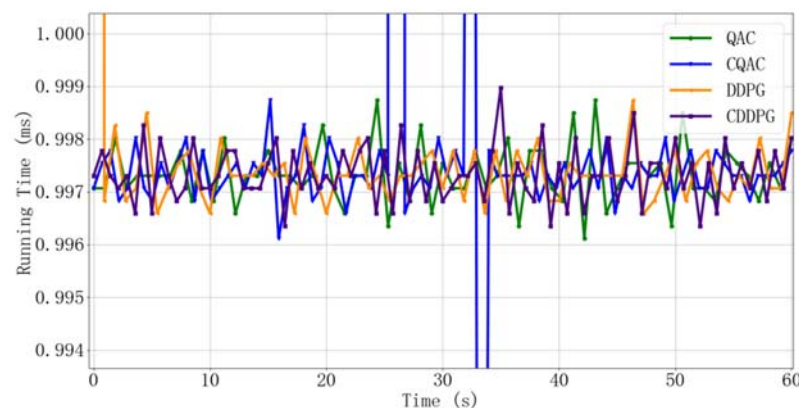
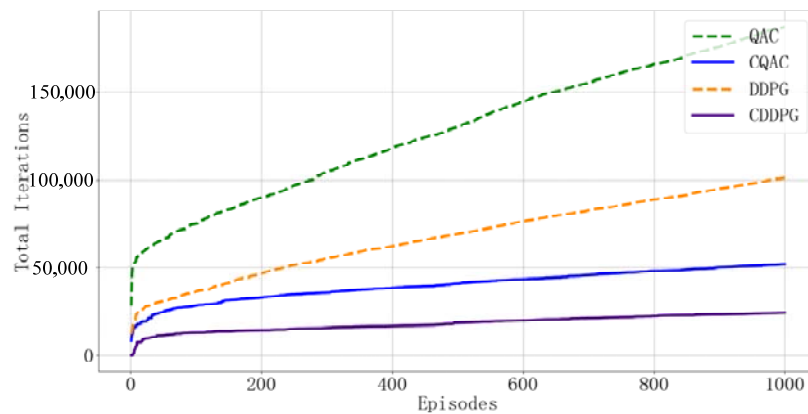


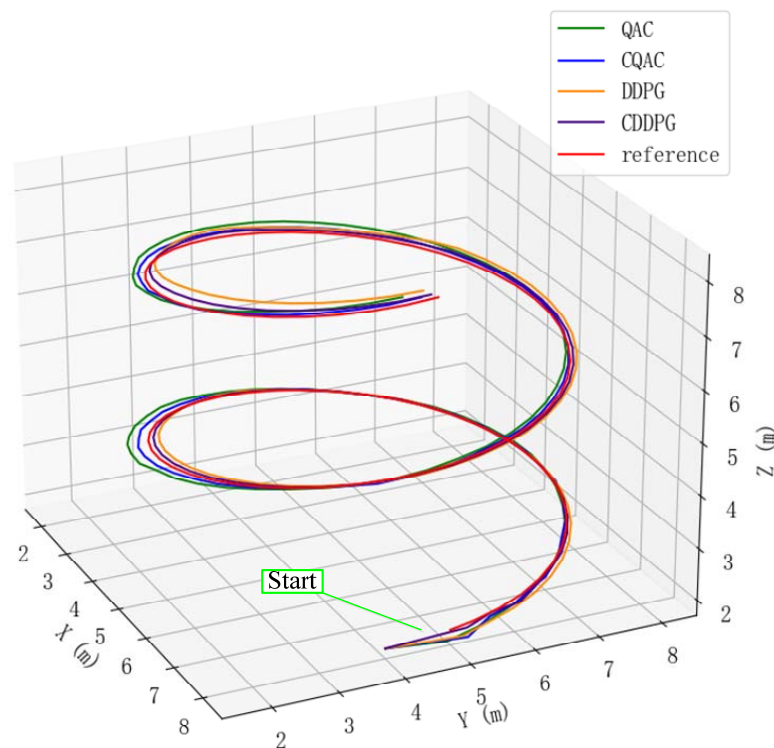
Figure 8. Calculate time cost of each single iteration step.



**Figure 9.** Total iterations of four algorithms.

### 3.2. Trajectory Tracking Simulation

To verify the dynamic tracking effect of the above training model, we designed a simulation experiment to track the target point as accurately as possible to complete the spiral trajectory motion in the Cartesian coordinate system. In this simulation, the target point moves at a constant speed to complete the preset trajectory tracking task. Since it has an angular velocity of  $\pi/15$  rad/s in the X-Y plane and a rising velocity of  $\pi/30$  m/s in the Z-axis, it is designed to complete two turns of helical motion with a radius of 3 m and a pitch of  $\pi$  m in space. It takes 60 s to complete that trajectory tracking. Through the adoption of four algorithm models, Figures 10–13 shows the tracking trajectories of the four algorithms in three-dimensional space and the tracking trajectories in the X, Y, and Z directions. Visually, the tracked trajectories under the CQAC and CDDPG algorithms are closer to the reference trajectories than those under the QAC and DDPG algorithms. In order to further clarify the superiority of the proposed algorithm, we conducted further analysis of the specific tracking error data.



**Figure 10.** Spiral trajectory tracking curve of four algorithms.

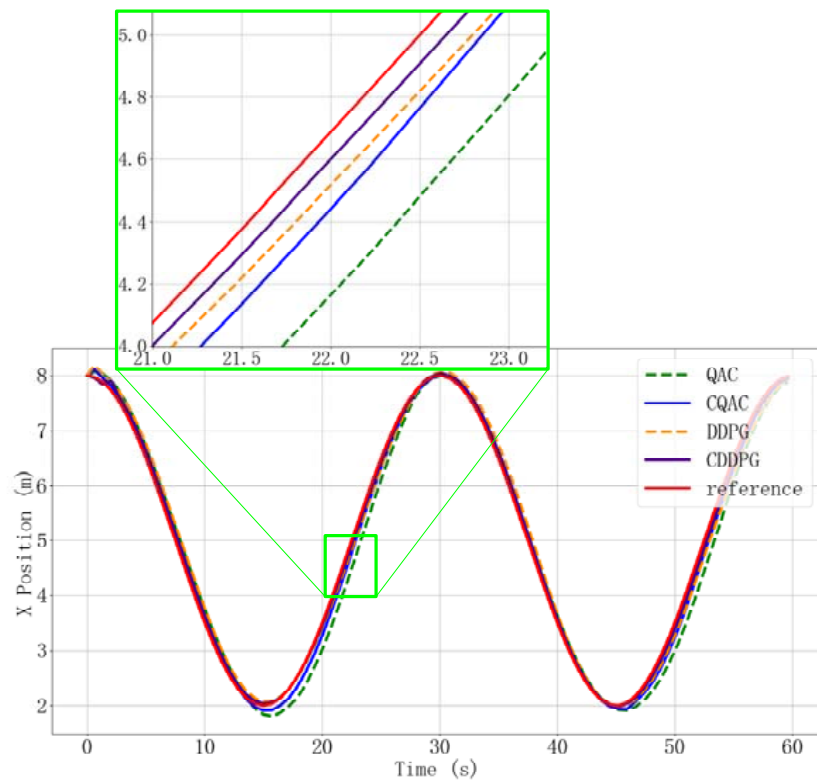


Figure 11. Tracking track of X position.

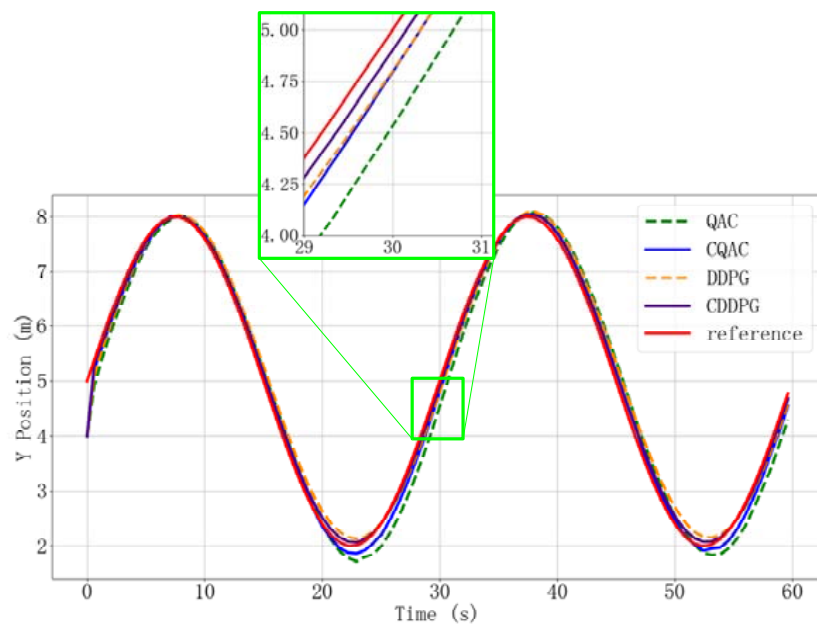


Figure 12. Tracking track of Y position.

Figure 14a plots the comprehensive tracking errors under each algorithm, and Figure 14b–d plot the tracking errors in the directions of three coordinate axes. The tracking error is then quantified and analyzed by the plot of simulation data. It can be seen from Figure 14 that the comprehensive tracking in the simulation is relatively stable. Since the tracking accuracy is set to 0.3 m during model training, the comprehensive tracking error of the QAC and DDPG algorithms in the figure basically fluctuates around 0.3 m, while the CQAC and CDDPG algorithms basically fluctuate around 0.15 m. Comparing the tracking error curves in the directions of the three coordinate axes, the tracking errors of the CQAC

and CDDPG algorithms are reduced to half of the original QAC and DDPG; the error fluctuation range is also greatly reduced, and the error trend is more stable compared with the original algorithm. Obviously, due to the addition of the compensation network, the tracking accuracy and convergence stability of the proposed method have been effectively improved; under the same training time, the stable tracking error of the proposed method is reduced by about 50% compared with the original algorithm. It shows that the compensation network added to the controller produces more active control to reduce the position error and improve the follow-up effect during the UAV flight.

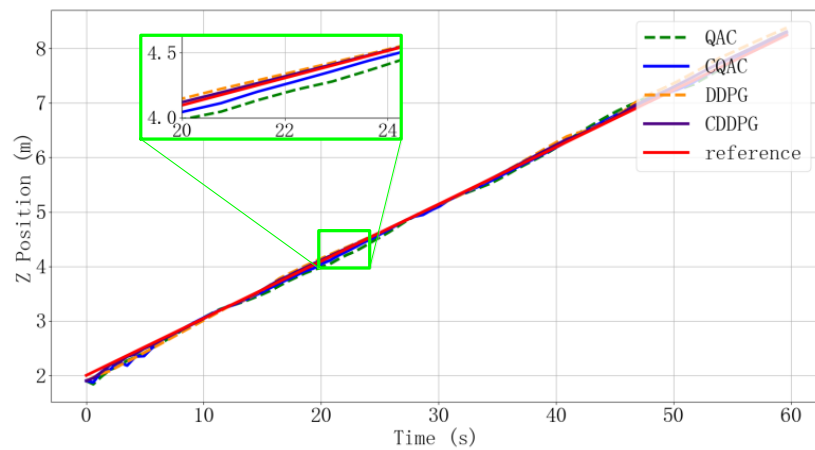


Figure 13. Tracking track of Z position.

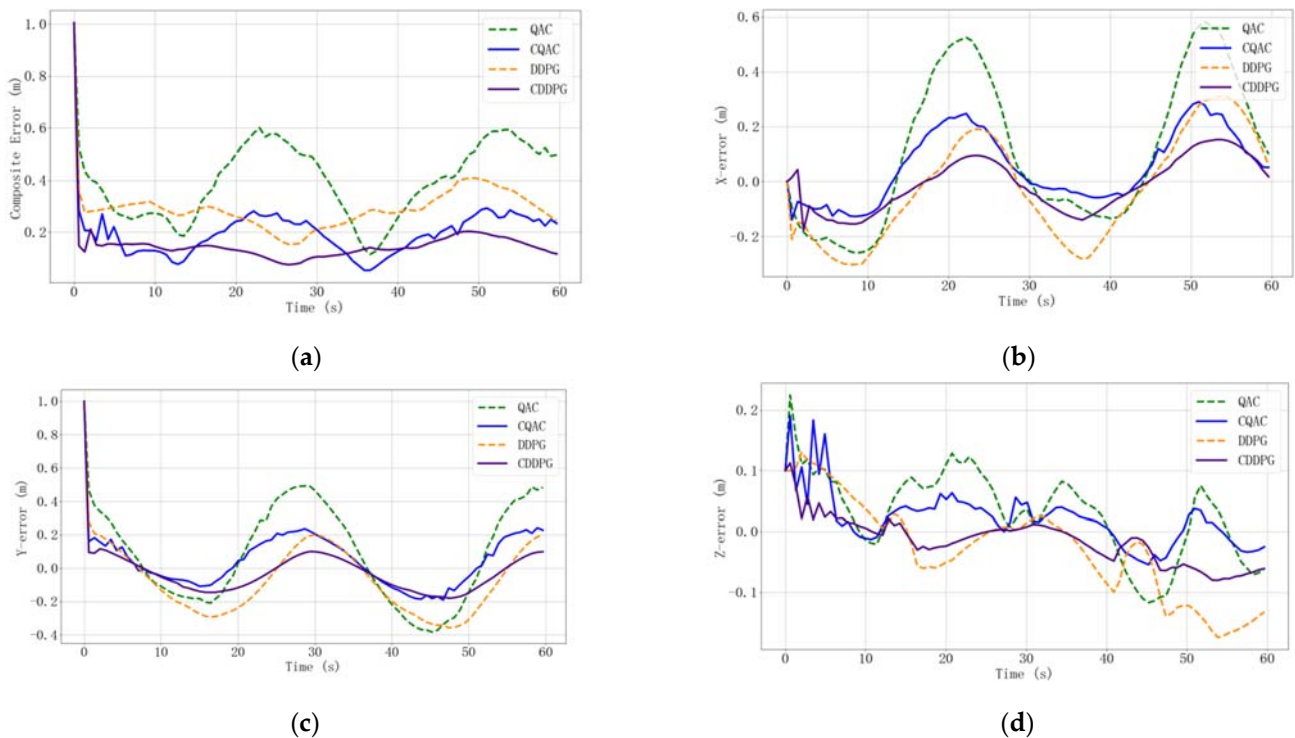
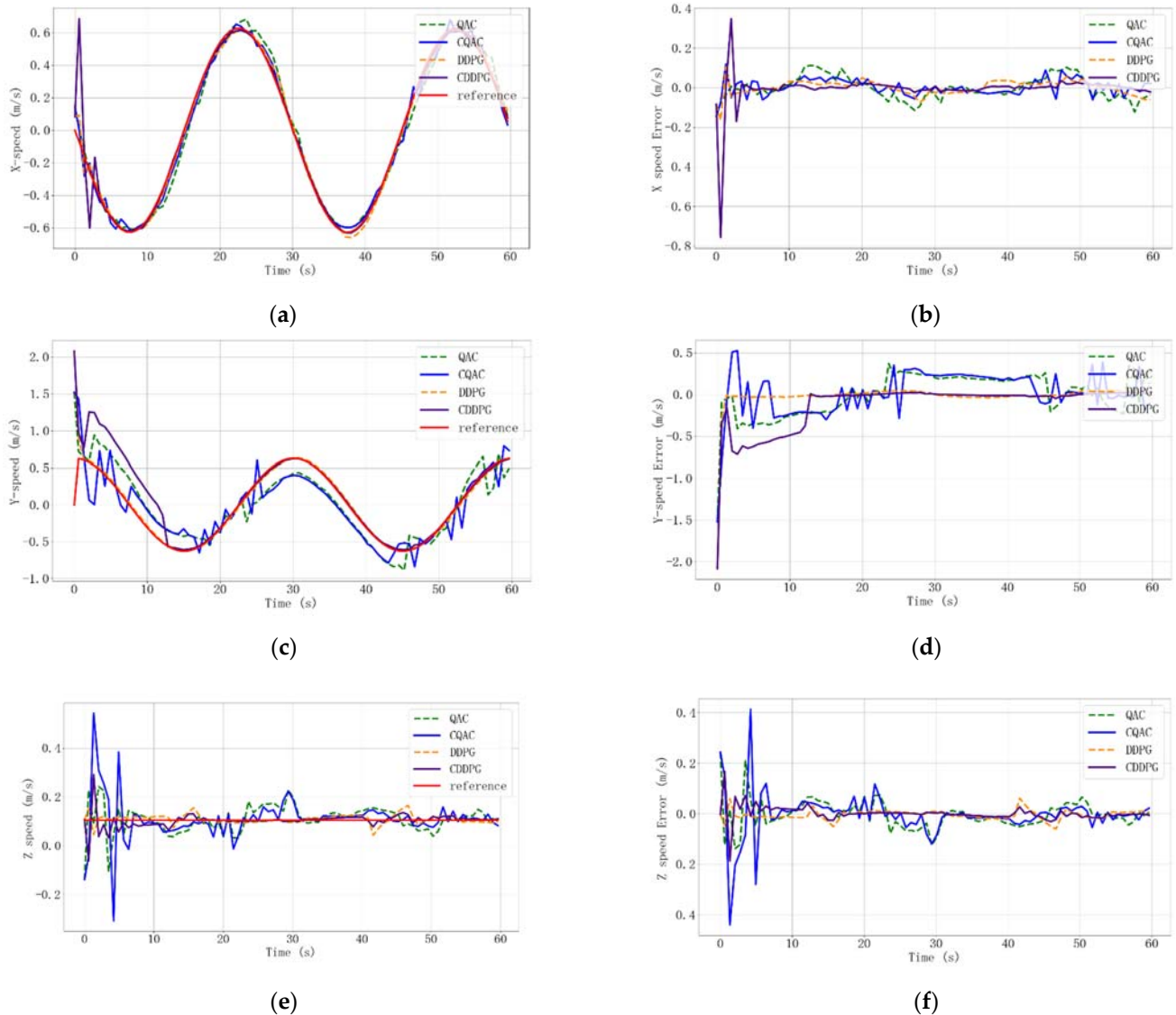


Figure 14. (a) Composite tracking error of four algorithms; (b) X-direction tracking error of four algorithms; (c) Y-direction tracking error of four algorithms; (d) Z-direction tracking error of four algorithms.

Through the above simulation process, the tracking speed and its errors during tracking are discussed. Figure 15 plots the comparison of the tracking speed and its error in three directions during the tracking process. According to the simulation settings, the target point performs a uniform circular motion on the X-Y plane and a uniform upward motion on the Z axis. That is to say, the speed along the X and Y axes conforms to the

law of trigonometric functions, which belongs to the variable acceleration motion. We can see from the comparison of the tracking speed in each axis direction that the speed in the tracking process is basically consistent with the reference speed, and the speed error is not much different. It can be seen from the above discussion that, the size of the tracking error can better represent the quality of the tracking effect.



**Figure 15.** (a) X-direction tracking speed of four algorithms; (b) X-direction velocity error of four algorithms; (c) Y-direction tracking speed of four algorithms; (d) Y-direction velocity error of four algorithms; (e) Z-direction tracking speed of four algorithms; (f) Z-direction velocity error of four algorithms.

In order to determine the effect of Z-axis uniform acceleration motion on the simulation effect, the following simulation adds an acceleration constant ( $0.033 \text{ m/s}^2$ ) along the Z axis based on the above simulation conditions. The target point does a half-circle helical motion (Figure 16), and its comprehensive error (Figure 17) and position error (Figure 18) on Z axis are observed, then the stability of each algorithm is analyzed.



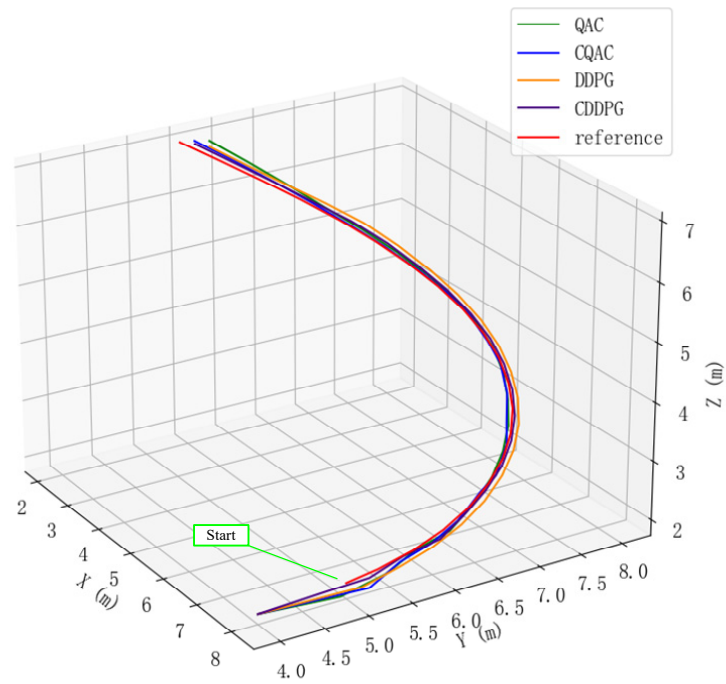


Figure 16. Spiral trajectory tracking of uniform acceleration.

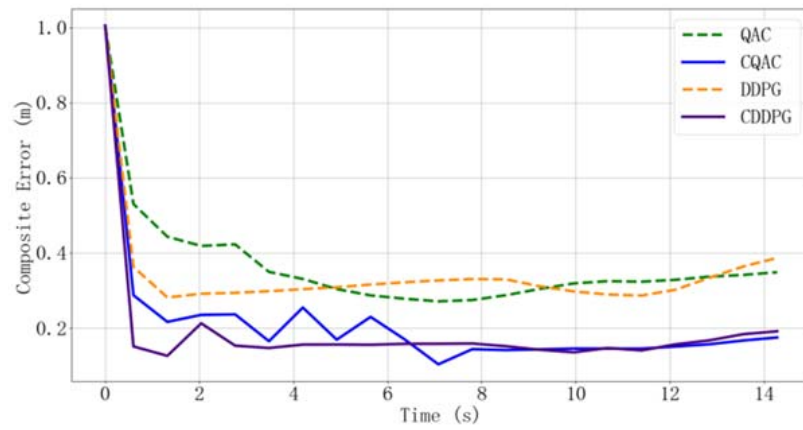


Figure 17. Composite tracking error of uniform acceleration.

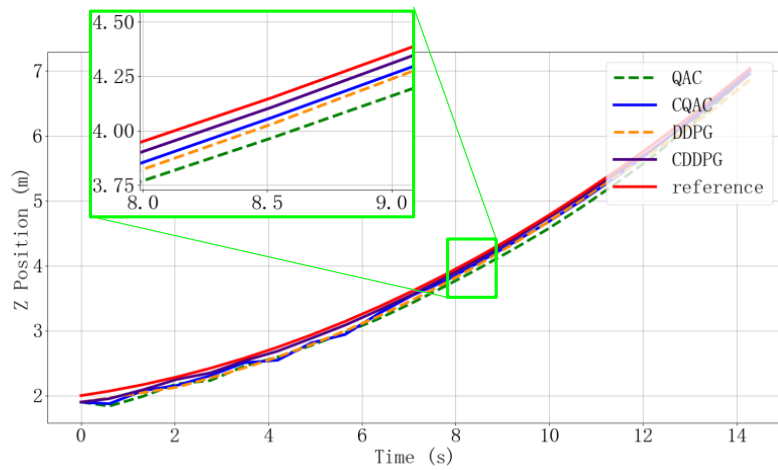


Figure 18. Position trajectory in the Z-axis direction under uniform acceleration.

After analysis, the simulation results are consistent with the previous results. The proposed algorithm still maintains good tracking characteristics, and uniform acceleration in the z-axis direction does not affect the following effect, which also proves that the proposed algorithm always has good accuracy and stability.

By comparing the simulation results of trajectory tracking, it is found that the tracking effect and error of the CDDPG algorithm are better than those of the DDPG algorithm. The designed hybrid controller for trajectory tracking based on CDDPG overcomes the disadvantage that it is difficult to form high-quality behaviors in a short time in reinforcement learning. It not only shortens the learning iteration period, but also solves the problem of large tracking error, which not only speeds up the convergence speed, but also has good convergence stability.

#### 4. Conclusions

Reinforcement learning may require a series of iterative processes in the learning process, and the long initial policy training period results in a large amount of training time for controller development. This paper designs a CDDPG algorithm based on deep reinforcement learning to speed up training time, improve learning efficiency, and stabilize the balance between exploration and utilization. Aiming at the UAV trajectory tracking control problem of the model unknown system, a compensation control algorithm combined with RL is proposed. The simulation results show that: (1) The training efficiency can be significantly improved by adding a compensation network, and the accuracy and convergence stability are also effectively improved; (2) Under the same configuration, the computational cost of the algorithm in this paper is basically the same as that of the DDPG algorithm; (3) The training time is about 70% lower than that of QAC and DDPG; (4) The tracking error is about 50% lower than QAC and DDPG. The above factors are better than model-free reinforcement learning algorithms represented by the DDPG algorithm. It breaks the traditional idea of using reinforcement learning to adjust or optimize system parameters, and embeds the compensation network into the reinforcement learning method, which is also the innovation of this paper.

The work in this paper is a simulation performed under ideal conditions, confining the movement of the UAV to the same space, and does not involve the attitude stability and anti-jamming capability of the aircraft. In further studies, the above problems will be considered, comprehensive simulation will be carried out, real machine experiments will be increased as much as possible, and in-depth research will be carried out in real scenarios.

**Author Contributions:** Conceptualization, J.W. and Z.Y.; methodology, J.W.; software, J.W.; validation, N.H., L.L. and Z.W.; formal analysis, J.W.; investigation, C.W. and J.W.; resources, Z.Y.; data curation, J.W.; writing—original draft preparation, J.W.; writing—review and editing, J.W. and Z.Y.; visualization, J.W.; supervision, Z.Y.; project administration, Z.Y.; funding acquisition, Z.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Guizhou Provincial Science and Technology Projects, grant number Guizhou-Sci-Co-Supp[2020]2Y044. The authors fully appreciate their financial supports.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** The authors would like to thank the reviewers for their constructive comments and suggestions that may help improve this paper.

**Conflicts of Interest:** The authors declare that they have no conflict of interest.

## References

1. Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, *4*, eaau5872. [[CrossRef](#)] [[PubMed](#)]
2. Xie, Z.; Berseth, G.; Clary, P.; Hurst, J.; Panne, M. Feedback control for cassie with deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1241–1246.
3. Chen, Y.M.; He, Y.L.; Zhou, M.F. Decentralized PID neural network control for a UAV helicopter subjected to wind disturbance. *J. Cent. South Univ.* **2015**, *22*, 168–179. [[CrossRef](#)]
4. Xu, Y.; Li, X.; Yang, K.; Yang, Y. Design of UAV UAV control system based on deep learning. *Comput. Meas. Control* **2020**, *28*, 123–155.
5. 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](#)]
6. Efe, M.O. Neural network assisted computationally simple pid control of a UAV uav. *IEEE Trans. Ind. Inform.* **2011**, *7*, 354–361. [[CrossRef](#)]
7. Smart, W.D.; Kaelbling, L.P. Reinforcement learning for robot control. *Proc. SPIE* **2002**, *4573*, 92–103.
8. Li, Z.; Chen, X.; Xie, M.; Zhao, Z. Adaptive fault-tolerant tracking control of flying-wing unmanned aerial vehicle with system input saturation and state constraints. *Trans. Inst. Meas. Control* **2021**, *44*, 880–891. [[CrossRef](#)]
9. Millan-Arias, C.; Fernandes, B.; Cruz, F.; Dazeley, R.; Fernandes, S. A robust approach for continuous interactive Actor-critic algorithms. *IEEE Access* **2021**, *9*, 104242–104260. [[CrossRef](#)]
10. Iwata, T.; Shibuya, T. Adaptive modular reinforcement learning for robot controlled in multiple environments. *IEEE Access* **2021**, *9*, 103032–103043. [[CrossRef](#)]
11. Wang, M.; Zeng, B.; Wang, Q. Research on Motion Planning Based on Flocking Control and Reinforcement Learning for Multi-Robot Systems. *Machines* **2021**, *9*, 77. [[CrossRef](#)]
12. Yeh, Y.L.; Yang, P.K. Design and Comparison of Reinforcement-Learning-Based Time-Varying PID Controllers with Gain-Scheduled Actions. *Machines* **2021**, *9*, 319. [[CrossRef](#)]
13. Wada, D.; Araujo-Estrada, S.A.; Windsor, S. Unmanned Aerial Vehicle Pitch Control Using Deep Reinforcement Learning with Discrete Actions in Wind Tunnel Test. *Aerospace* **2021**, *8*, 18. [[CrossRef](#)]
14. Peters, J.; Schaal, S. Policy Gradient Methods for Robotics. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 2219–2225.
15. 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 (ICML-14), Beijing, China, 21–26 June 2014; pp. 387–395.
16. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
17. Kiumarsi, B.; Lewis, F.L. Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems. *IEEE Trans Neural Netw Learn. Syst.* **2015**, *26*, 140–151. [[CrossRef](#)]
18. Gan, Z.; Li, B.; Neretin, E.S.; Dyachenko, S.A. UAV Maneuvering Target Tracking based on Deep Reinforcement Learning. *J. Phys. Conf. Ser.* **2021**, *1958*, 012015. [[CrossRef](#)]
19. Chen, C.; Modares, H.; Xie, K.; Lewis, F.L.; Wan, Y.; Xie, S. H-infinity Tracking Control of Completely Unknown Continuous-Time Systems via Off-Policy Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2550–2562.
20. Ye, L.; Li, J.; Wang, C.; Liu, H.; Liang, B. Reinforcement Learning Tracking Control for Unknown Continuous Dynamic Systems. In Proceedings of the 2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS'21), Suzhou, China, 14–16 May 2021.
21. Luy, N.T.; Thanh, N.T.; Tri, H.M. Reinforcement learning-based intelligent tracking control for wheeled mobile robot. *Trans. Inst. Meas. Control* **2014**, *36*, 868–877. [[CrossRef](#)]
22. Wang, G.F.; Fang, Z.; Li, P.; Li, B. Transferring knowledge from human-demonstration trajectories to reinforcement learning. *Trans. Inst. Meas. Control* **2016**, *40*, 94–101. [[CrossRef](#)]
23. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *Int. J. Robot. Res.* **2017**, *37*, 421–436. [[CrossRef](#)]
24. Hwangbo, J.; Sa, I.; Siegwart, R.; Hutter, M. Control of a UAV With Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2096–2103. [[CrossRef](#)]
25. William, K. Flight Controller Synthesis via Deep Reinforcement Learning. Ph.D. Dissertation, Boston University, Boston, MA, USA, 2019.
26. Rubí, B.; Morcego, B.; Pérez, R. A Deep Reinforcement Learning Approach for Path Following on a UAV. In Proceedings of the European Control Conference (ECC 2020), Petersburg, VA, USA, 12–15 May 2020.
27. Qingqing, Z.; Renjie, T.; Siyuan, G.; Weizhong, Z. A PID Gain Adjustment Scheme Based on Reinforcement Learning Algorithm for a UAV. In Proceedings of the 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020.
28. Zhen, Y.; Hao, M. Research on Intelligent PID Control Method Based on Deep Reinforcement Learning. *Tactical Missile Technol.* **2019**, *5*, 37–43.

29. Zhen, Y.; Yuan, J.; Chi, Q.; Hao, M. Research on Application of Deep Reinforcement Learning Method in Aircraft Control. *Tactical Missile Technol.* **2020**, *4*, 112–118.
30. Levine, S.; Koltun, V. Learning Complex Neural Network Policies with Trajectory Optimization. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014.
31. Yang, B.; Liu, P.; Feng, J.; Li, S. Two-Stage Pursuit Strategy for Incomplete-Information Impulsive Space Pursuit-Evasion Mission Using Reinforcement Learning. *Aerospace* **2021**, *8*, 299. [[CrossRef](#)]