Hindawi
International Journal of Aerospace Engineering
Volume 2020, Article ID 8874619, 13 pages
https://doi.org/10.1155/2020/8874619

Research Article

Model-Free Attitude Control of Spacecraft Based on PID-Guide TD3 Algorithm

ZhiBin Zhang, XinHong Li, JiPing An, WanXin Man, and GuoHui Zhang

Department of Aerospace Science and Technology, Space Engineering University, Beijing 101416, China

Correspondence should be addressed to XinHong Li; lixinhong789@163.com

Received 11 July 2020; Revised 4 December 2020; Accepted 7 December 2020; Published 30 December 2020

Academic Editor: Marco Pizzarelli

This paper is devoted to model-free attitude control of rigid spacecraft in the presence of control torque saturation and external disturbances. Specifically, a model-free deep reinforcement learning (DRL) controller is proposed, which can learn continuously according to the feedback of the environment and realize the high-precision attitude control of spacecraft without repeatedly adjusting the controller parameters. Considering the continuity of state space and action space, the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm based on actor-critic architecture is adopted. Compared with the Deep Deterministic Policy Gradient (DDPG) algorithm, TD3 has better performance. TD3 obtains the optimal policy by interacting with the environment without using any prior knowledge, so the learning process is time-consuming. Aiming at this problem, the PID-Guide TD3 algorithm is proposed, which can speed up the training speed and improve the convergence precision of the TD3 algorithm. Aiming at the problem that reinforcement learning (RL) is difficult to deploy in the actual environment, the pretraining/fine-tuning method is proposed for deployment, which can not only save training time and computing resources but also achieve good results quickly. The experimental results show that DRL controller can realize high-precision attitude stabilization and attitude tracking control, with fast response speed and small overshoot. The proposed PID-Guide TD3 algorithm has faster training speed and higher stability than the TD3 algorithm.

1. Introduction

With the rapid development of space technology, the structure and composition of On-Orbit Servicing Spacecraft (OOSS) are becoming more and more complex, and the performance is constantly improving. The effectiveness of attitude control determines the success or failure of the service mission and the life of the spacecraft. On-Orbit Servicing Spacecraft is characterized by flexible multibody structure, liquid sloshing, and fuel consumption and needs to change the structure and parameters according to the mission requirements. For example, in the case of mass loss due to fuel consumption, the rate of mass change of the spacecraft is known to be a function of control application and actuator hardware characteristics. The motion of space manipulator will cause mass displacement, resulting in the change of inertial parameters, and cause disturbances to the attitude stability of the service spacecraft body. On the other hand, the spacecraft is affected by gravity gradient torque, aerodynamic torque, radiated torque, and other unknown disturbance torques. These peculiarities make the attitude of the OOSS present dynamic characteristics such as uncertainty, time-varying, strong nonlinearity, and high-order multivariable coupling [1]. Even in many complex space missions, it is difficult to establish an accurate mathematical model, which increases the difficulty of attitude control.

The classical attitude control methods include PID control [2], adaptive control [3], sliding mode control [4], Lyapunov control [5], optimal control [6], and robust H_∞ control [7]. These control algorithms have achieved good results in simulation experiments and practical applications. Traditional attitude control algorithms need to establish a relatively accurate system model and need to design the parameters of the controller accurately. When the system cannot be modeled completely or the environment changes greatly, the performance of the controller will degrade to some extent. For systems with unclear or completely unknown mathematical models, the intelligent control method with self-learning ability will be a promising choice.

Reinforcement learning (RL) is a kind of machine learning, which has a close relationship with dynamic programming and optimal control theory [8]. The basic idea of RL is to explore the optimal strategy through the interaction between agent and environment, so as to maximize the return [9]. Classical RL, such as Q-learning, discretizes the action and state space and uses the table method to solve the problem [10]. However, the actual control problem may have continuous action and state space, and it is difficult to discretize. High-dimensional continuous state and action space increase the computational burden and lead to the so-called Curse of Dimensions (CoD) problem. DRL solves the CoD problem by introducing neural networks to approximate value function and policy [11]. In 2015, Lillicrap et al. proposed the Deep Deterministic Policy Gradient (DDPG) algorithm to solve the control problem with continuous action space [12]. DRL became widely known in 2016 when Google AlphaGo defeated the top international Go player Lee Sedol in the Go competition.

Up to now, DRL has been used in robots [13], UAV [14, 15], energy [16, 17], transportation [18], and other control fields. DRL based on the Markov decision process (MDP) provides an effective way to realize intelligent control of spacecraft. In the process of self-learning, DRL optimizes the parameters of neural networks iteratively, which eliminates the trouble of design parameters, enables it to adapt to the changing software, hardware, and environment, and can continuously optimize the performance of the controller by changing the setting of reward function. As space exploration missions become more frequent and complex, spacecraft are getting farther and farther from the earth; DRL technology with fast learning ability and self-regulation ability will play an increasingly important role in spacecraft attitude control system.

Based on the above discussion, a model-free attitude control scheme based on DRL is proposed, and a kind of mixed reward function for spacecraft attitude control is designed to realize attitude stabilization control and attitude tracking control of spacecraft. Aiming at the problem that the TD3 algorithm is relatively slow to explore the optimal strategy without using any prior knowledge, the PID-Guide TD3 algorithm is proposed, which uses the PID controller to guide the exploration process of the DRL algorithm, so as to significantly speed up the training speed and improve the convergence accuracy of the algorithm. Aiming at the problem that reinforcement learning (RL) is difficult to deploy in the actual environment, this paper proposes that the algorithm should be pretrained on the ground and then fine-tune the parameters on orbit, so as to save training time and computing resources and achieve better results quickly.

The rest of the paper is organized as follows. The problem statements and control objectives are given in Section 2. The DRL controller of spacecraft attitude is designed, and the PID-Guide TD3 algorithm is proposed in Section 3. Section 4 proves the effectiveness of the proposed algorithm through three simulation cases. Conclusions are given in Section 5.

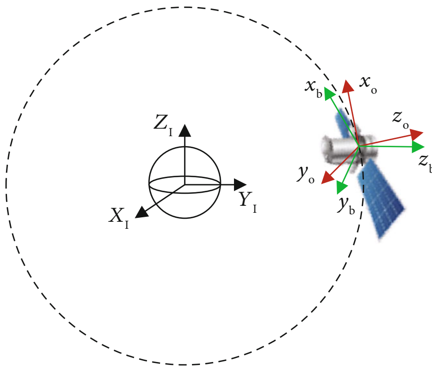

Figure 1: Coordinate definition.

2. Problem Statement and Preliminaries

2.1. Problem Statement. The paper mainly studies the attitude control problem of OOSS under complex situations such as uncertain inertia matrix, even unable to establish the motion model, and external disturbances. The control objective is to achieve attitude stabilization control under bounded disturbances, that is,

$$\begin{aligned} \lim_{t\to\infty} (p(t) - p_d) &= 0, \\ \lim_{t\to\infty} (\omega(t) - \omega_d) &= 0, \end{aligned} \tag{1}$$

where $p(t)$ is the attitude angle, $\omega(t)$ is the angular velocity, p_d is the target attitude angle, and ω_d is the target angular velocity.

The characteristics of this control problem are as follows: (A) it is difficult to establish accurate dynamic and kinematic models; (B) there are unknown external disturbances, such as perturbation and solar wind; (C) the actuator is saturated.

2.2. Attitude Representation. The definition of the earth-centered inertial coordinate system $F_I - O_I X_I Y_I Z_I$, spacecraft body-fixed coordinate system $F_b - O_b X_b Y_b Z_b$, and orbital coordinate system $F_o - O_o X_o Y_o Z_o$ is shown in Figure 1. When the attitude of the spacecraft is in stable state, the spacecraft body-fixed frame coincides with the orbital coordinate system. When attitude motion occurs, the orientation or pointing of the body-fixed frame relative to the orbital coordinate system is the attitude of the spacecraft. There are usually five ways to describe the attitude of the spacecraft, including Euler angles, Rotation Matrix, Quaternions, Rodrigues Parameters (RP), and Modified Rodrigues Parameters (MRP). Different description methods can be converted between each other. The characteristics of different description methods can be found in [19, 20].

2.3. Spacecraft Dynamics. The kinematic equation of spacecraft attitude described by MRP is given by

$$\dot{p} = \frac{1}{4} \begin{bmatrix} 1 - p^2 + 2p_1^2 & 2(p_1 p_2 - p_3) & 2(p_1 p_3 + p_2) \\ 2(p_2 p_1 + p_3) & 1 - p^2 + 2p_2^2 & 2(p_2 p_3 - p_1) \\ 2(p_3 p_1 - p_2) & 2(p_3 p_2 + p_1) & 1 - p^2 + 2p_3^2 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{2}$$

The matrix form of the above equation is as follows:

$$\dot{p} = \frac{1}{4}\left[\left(1 - p^T p\right) I_{3\times3} + 2p^\times + 2pp^T\right]\omega = \frac{1}{4}G(p)\omega, \quad (3)$$

where $p = [p_1 p_2 p_3]^T$ is the Modified Rodrigues Parameters and $\omega = [\omega_x \omega_y \omega_z]^T \in \mathbb{R}^3$ is the rotation angular velocity of the spacecraft relative to the earth-centered inertial, which is expressed in the body-fixed frame. $I_{3\times3}$ is the third-order identity matrix. $p^\times$ is the skew-symmetric matrix of p, defined as

$$p^\times = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}. \quad (4)$$

The attitude dynamics equation can be expressed by the following equation:

$$J\dot{\omega} + \omega^\times J\omega = M_c + M_d, \quad (5)$$

where $J \in \mathbb{R}^3$ is the inertia matrix of the spacecraft and $\omega^\times$ is the skew-symmetric matrix of ω. $M_c \in \mathbb{R}^3$ is the control torque. $M_d \in \mathbb{R}^3$ is the total disturbing torque, including the gravity gradient torque, aerodynamic torque, magnetic disturbing torque, and radiation torque.

Remark 1. The dynamic model mentioned here is only used as a simulation of the space environment and will not be used directly in the controller.

3. DRL Controller Design for Spacecraft

3.1. Background. Deep Deterministic Policy Gradient (DDPG) is an actor-critic method that uses neural networks as the approximation of policy function and Q-function, namely, the policy network and the Q network. Therefore, it can deal with the problem with continuous action space. At the same time, the use of empirical playback and double network solves the problem that actor-critic is difficult to converge [12].

While DDPG can achieve great performance sometimes, it is frequently vulnerable to hyperparameters and other kinds of tuning [21]. A common failure mode for DDPG is that the learned Q-function begins to overestimate Q values dramatically, which leads to the policy breaking, because the error in the Q-function is introduced into the training of the policy network. The Twin Delayed DDPG (TD3) algorithm is an improved version of the DDPG, using three techniques to solve the problems of DDPG [22].

The first one is Double Q Networks. TD3 learns two Q-functions and uses the smaller of the two Q values to form the targets in the Bellman error loss functions; thus, the overestimation of the approximation of the value function is reduced.

The second one is "Delayed" Policy Updates. TD3 updates the policy (and target networks) less frequently than the Q-function.

The last one is Target Policy Smoothing. TD3 adds noise to the target action to reduce sensitivity and instability, making it harder for the policy to exploit Q-function errors by smoothing out Q along changes in action. The combination of these three techniques significantly improves performance of TD3 relative to baseline DDPG.

3.2. End-to-End Attitude Control Based on TD3 Algorithm. The training goal of the DRL model is to maximize the total return when the agent interacts with the environment. DRL controller has a natural similarity with traditional control but uses different terms to represent the same concept, as shown in Figure 2. DRL is composed of agent, environment, state, action, and reward. The agent learns the optimal policy through interacting with the environment and takes the optimal action to maximize the long-term reward, while the traditional control is to design the controller (policy) through experts. The state feedback signal refers to the observation of the environment, and the reference signal is built into the reward function and observation.

3.2.1. Environment and State. In order to implement DRL, the simulation environment of spacecraft attitude motion should be established according to the attitude dynamic and kinematic equations of spacecraft. The simulation environment includes the environment dynamic model and the interface between the agent and the environment. The input of the environment is the action output by the agent, and the output is the observed state and reward signal after the action is performed. For the spacecraft attitude control problem, the system states are selected as attitude angle $p = (p_1 p_2 p_3)^T$ and angular velocity $\omega = (\omega_x \omega_y \omega_z)^T$. The reference signal includes the target state $p_T = (p_{T_1} p_{T_2} p_{T_3})^T$ and $\omega_T = (\omega_{Tx} \omega_{Ty} \omega_{Tz})^T$. The system state s and the error signal e are used as the observations, where $e = (p_T - p, \omega_T - \omega)^T = (dp_1 dp_2 dp_3 d\omega_x d\omega_y d\omega_z)^T$. The action space is continuous three-dimensional control torque $a \in [-1, 1]\,\mathrm{N}\cdot\mathrm{m}$.

3.2.2. Reward Function. Reward refers to the reward signal that the agent measures its performance according to the task goal. A reasonable reward function is the key for agent to learn effective policy, which determines the convergence speed and stability of RL. Typically, a positive reward is offered to encourage certain behaviors of the agent and a negative reward is offered to deter others [23]. The reward function can be divided into three types: continuous reward function, discrete reward function, and mixed reward function. The continuous reward function varies continuously with observation and action. In general, continuous reward signals can improve the convergence during training and simplify the network structure. The discrete reward function varies discontinuously with observation and action. This type of reward signals slow down the convergence rate and require a more complicated network structure. Discrete rewards are usually implemented as event that occurs in the environment. For example, if the agent exceeds a certain threshold, it may receive a positive reward; if a certain performance constraint is violated, it will be punished. The more commonly

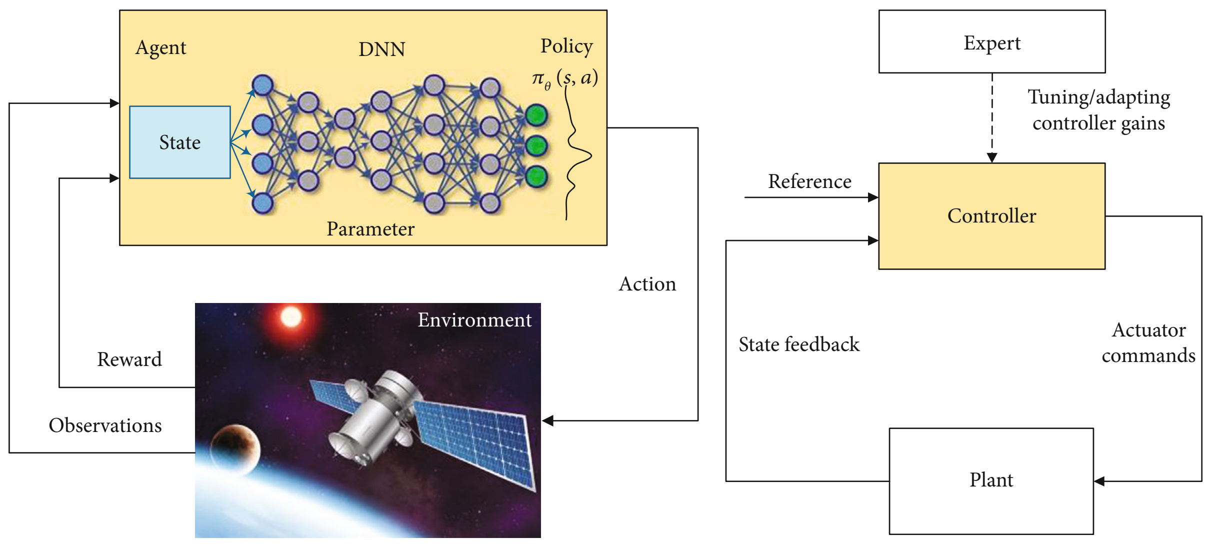

FIGURE 2: These are the schematics of the controllers: (a) controller based on deep reinforcement learning; (b) traditional controller.

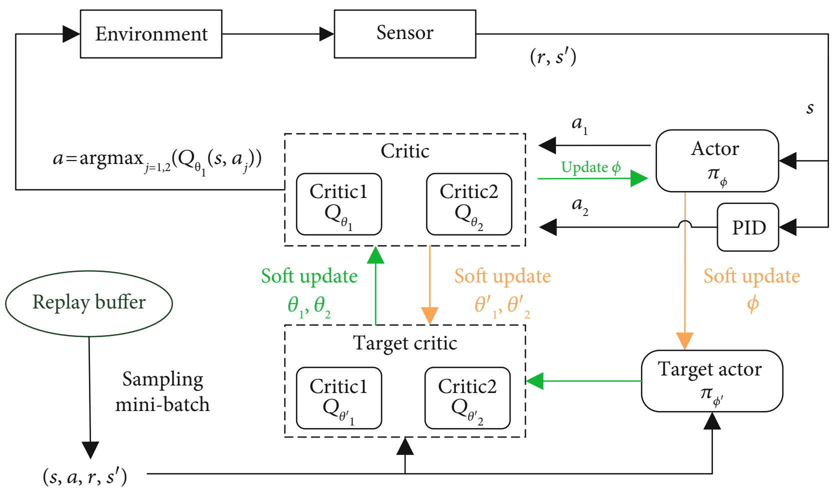

FIGURE 3: Structure of PID-Guide TD3.

used reward signal is mixed reward signal. Mixed rewards include continuous rewards and discrete rewards. Discrete reward signals keep the system away from bad states, and continuous reward signals improve convergence by providing smooth rewards near the target state.

The following mixed reward function is designed for spacecraft attitude control:

$$\begin{aligned} r_1 &= -(\|e\|_1) - a^2, \\ r_2 &= \frac{1}{\|p_T - p\|_1 + 0.01}\left(\|e\|_{-\infty} \le 0.1\right), \\ r_3 &= -100\left(\|p\|_\infty \ge 4 \text{ or } \|\omega\|_\infty \ge 4\right), \\ R &= r_1 + r_2 + r_3, \end{aligned} \tag{6}$$

where $\|\|_1$, $\|\|_{-\infty}$, and $\|\|_\infty$ represent vector norm. r_1 is a continuous reward, which can stabilize the system state and reduce fuel consumption. The smaller the error, the greater the reward of r_1. The second term in r_1 represents energy consumption. When the attitude of the spacecraft is very close to the target attitude, the error change is small, and the change of r_1 will be very insignificant. Therefore, the continuous reward r_2 is introduced to increase the reward gradient when the absolute value of each error component is less than 0.1, so as to guide the attitude angle to approach the target value quickly and accurately. r_3 is a discrete reward, which can control the attitude angle not to exceed the range and increase training speed.

3.2.3. TD3 Algorithm. The agent receives observations and rewards from the environment and sends actions to the environment. The TD3 algorithm is used as the learning method of agent in this paper. TD3 and DDPG are both off-policy algorithms. TD3 creates an experience replay buffer to store historical experiences and then randomly sample transitions

```
(1) Randomly initialize critic networks $Q_{\theta_1}$, $Q_{\theta_2}$ and actor network $\pi_\phi$ with weights $\theta_1, \theta_2, \phi$
(2) Initialize target networks $Q_{\theta_1'}, Q_{\theta_2'}, \pi_{\phi'}$ and with weights $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
(3) Initialize replay buffer $\mathscr{R}$
(4) for episode = 1 to $M$ do
  (1) Set the target state, randomly reset the environment, and get the initial state $s$
  (2) for $t = 1$ to $T$ do
    (i) Select action $a_1 = \pi_\phi(s) + \varepsilon$ according to the current policy and exploration noise, $\varepsilon \sim \mathcal{N}(0, \sigma)$
    (ii) Select another action according to PID controller $a_2 = K_p e(t) + K_i \int_0^t e(t)dt + K_d (de(t)/dt)$
    (iii) Execute action $a = \arg\ \max_{j=1,2}(Q_{\theta_1}(s, a_j))$ and observe reward $r$ and observe new state $s'$
    (iv) Store transition tuple $(s, a, r, s')$ in $\mathscr{R}$
    (v) If $s'$ is terminal, reset environment state
    (vi) If it is time to update, then randomly sample a mini batch of $N$ transitions from $\mathscr{R}$
    (vii) Compute target action $a' = \text{clip}(\pi_{\phi'}(s') + \text{clip}(\varepsilon, -c, c), -a_{\max}, a_{\max})$
    (viii) Compute target $Q$ value $y = r + \gamma\ \min_{i=1,2} Q_{\theta_i'}(s', a')$
    (ix) Update critic networks $\theta_i \leftarrow \arg\ \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
    (x) If $t$ mod $d$ then
      (a) Update $\phi$ by the deterministic policy gradient
$\nabla_\phi J(\phi) = 1/N \sum_{j=1}^{N} \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(\mathbf{s})$
      (b) Update target networks
$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$,
$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
    end for
  end for
```

Algorithm 1: PID-Guide TD3 Algorithm.

from it and feed those sample data to update actor and critic networks. The existence of experience replay buffer helps the agent to be able to learn previous experiences and improve the efficiency of sample utilization. Random sampling can break the correlation between samples and make the learning process of agents more stable [24].

TD3 uses a total of 6 neural networks, namely, actor network π_ϕ, actor target network $\pi_{\phi'}$, critic network Q_{θ_1}, critic network Q_{θ_2}, critic target network $Q_{\theta_1'}$, and critic target network $Q_{\theta_2'}$. The role and update rules of each network are as follows.

(1) Actor network π_ϕ: responsible for the iterative update of actor network parameters ϕ and selects the current action a according to the current state s, which is used to interact with the environment to generate the next state s' and reward R

(2) Actor target network $\pi_{\phi'}$: responsible for selecting the optimal next action a' according to the next state s' sampled in the experience replay buffer. Network parameters ϕ' are periodically copied from ϕ

(3) Critic networks Q_{θ_i}: responsible for the iterative update of Q network parameters and calculate the current Q value $Q_{\theta_i}(s, a)$. The target Q value is the smaller of the two value, namely, $y = r + \gamma\ \min_{i=1,2} Q_{\theta_i'}(s', a')$

Table 1: Actor and critic network architecture.

Layer	Actor network		Critic network	
	# units	Activation	# units	Activation
Input layer	12	Relu	15	Relu
Hidden layer 1	250	Relu	250	Relu
Hidden layer 2	250	Relu	250	Relu
Output layer	3	Linear	1	Linear

Table 2: Hyperparameter settings.

Hyperparameters	Symbol	Value
Random seed	—	2
Max episodes	M	400
Max steps per episode	T	200
Sample time	T_s	1
Replay buffer size	$\mathscr{R}$	10^6
Batch size	N	250
Policy network learning rate	rl_p	0.0003
Critic network learning rate	rl_c	0.001
Exploration noise scale	c	0.1
Delay update	d	3
Discount factor	γ	0.99
Soft update rate	τ	0.01

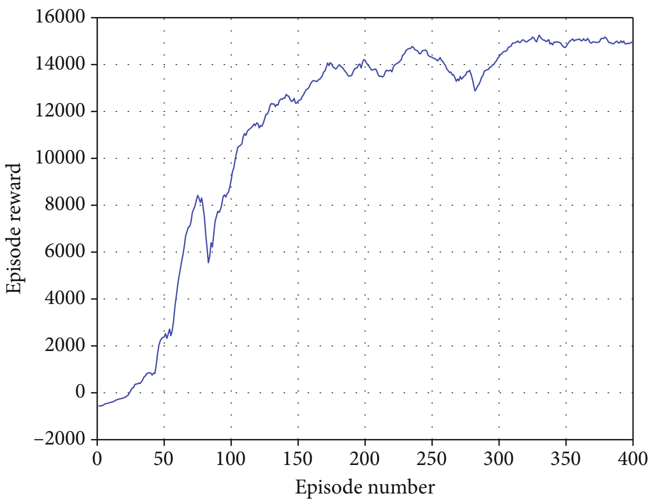

Figure 4: The learning process of TD3 agent under ideal environment.

(4) Critic target networks $Q_{\theta_i'}$: responsible for calculating $Q_{\theta_i'}(s', a')$ of the target Q value. Network parameters θ_i are periodically copied from θ_i'

For the critic networks, the loss function is defined as

$$J(\theta_i) = \frac{1}{N}\sum_{j=1}^{N}\left(y - Q_{\theta_i}(s, a)\right)^2. \tag{7}$$

For the actor network, the deterministic policy is used to optimize the parameters, and the loss function is defined as

$$\nabla_\phi J(\phi) = \frac{1}{N}\sum_{j=1}^{N}\nabla_a Q_{\theta_1}(s, a)\Big|_{a=\pi_\phi(s)}\nabla_\phi \pi_\phi(s). \tag{8}$$

The target networks are updated by soft update method:

$$\begin{aligned}\theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta_i',\\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi'.\end{aligned} \tag{9}$$

3.3. PID-Guide TD3 Algorithm. TD3 is a model-free algorithm, which does not use any prior knowledge and constantly explores through the interaction with the environment to obtain the optimal strategy. It is time-consuming for agent to find optimal policy without any prior knowledge due to the problems such as partial observability of environmental feedback, the sparsity of reward, and high-dimensional state and action space.

In order to speed up the training speed and improve the convergence stability of the algorithm, a PID-Guide TD3 algorithm is proposed in this section. The core idea of the PID-Guide TD3 algorithm is as follows. In the current state s, two actions are generated by the action network and PID controller, respectively. Then, the critical network is used to evaluate the two actions; the action with higher value will be actually executed. In fact, any model-free controller can guide TD3 to conduct policy search. Figure 3 describes the structure of PID-Guide TD3.

PID is a model-free controller based on the feedback of error. PID requires precise design of parameters, so the change of environment will lead to serious degradation of its performance. The formula of PID algorithm is

$$u = K_p e(t) + K_i\int_0^t e(t)dt + K_d\frac{de(t)}{dt}, \tag{10}$$

where K_p, K_i, and K_d are positive definite matrices containing the control parameters, which are, respectively, called proportional coefficient, integral coefficient, and differential coefficient.

The pseudocode of PID-Guide TD3 is given in Algorithm 1. The main steps of the PID-Guide TD3 algorithm are as follows:

(1) Randomly initialize critic networks $Q_{\theta_i'}$, Q_{θ_1}, Q_{θ_2} and actor network π_ϕ with weights θ_1, θ_2, ϕ and initialize target networks $Q_{\theta_1'}$, $Q_{\theta_2'}$, $\pi_{\phi'}$ with weights $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$

(2) Initialize replay buffer $\mathscr{R}$

(3) Start a new episode, set the target position, randomly reset the environment, and get the initial state s

(4) For every step, select an action according to the current policy with exploration noise:

$$a_1 = \pi_\phi(s) + \varepsilon, \tag{11}$$

where the exploration noise $\varepsilon \sim \mathcal{N}(0, \sigma)$

(5) Select another action according to PID controller:

$$a_2 = K_p e(t) + K_i\int_0^t e(t)dt + K_d\frac{de(t)}{dt}. \tag{12}$$

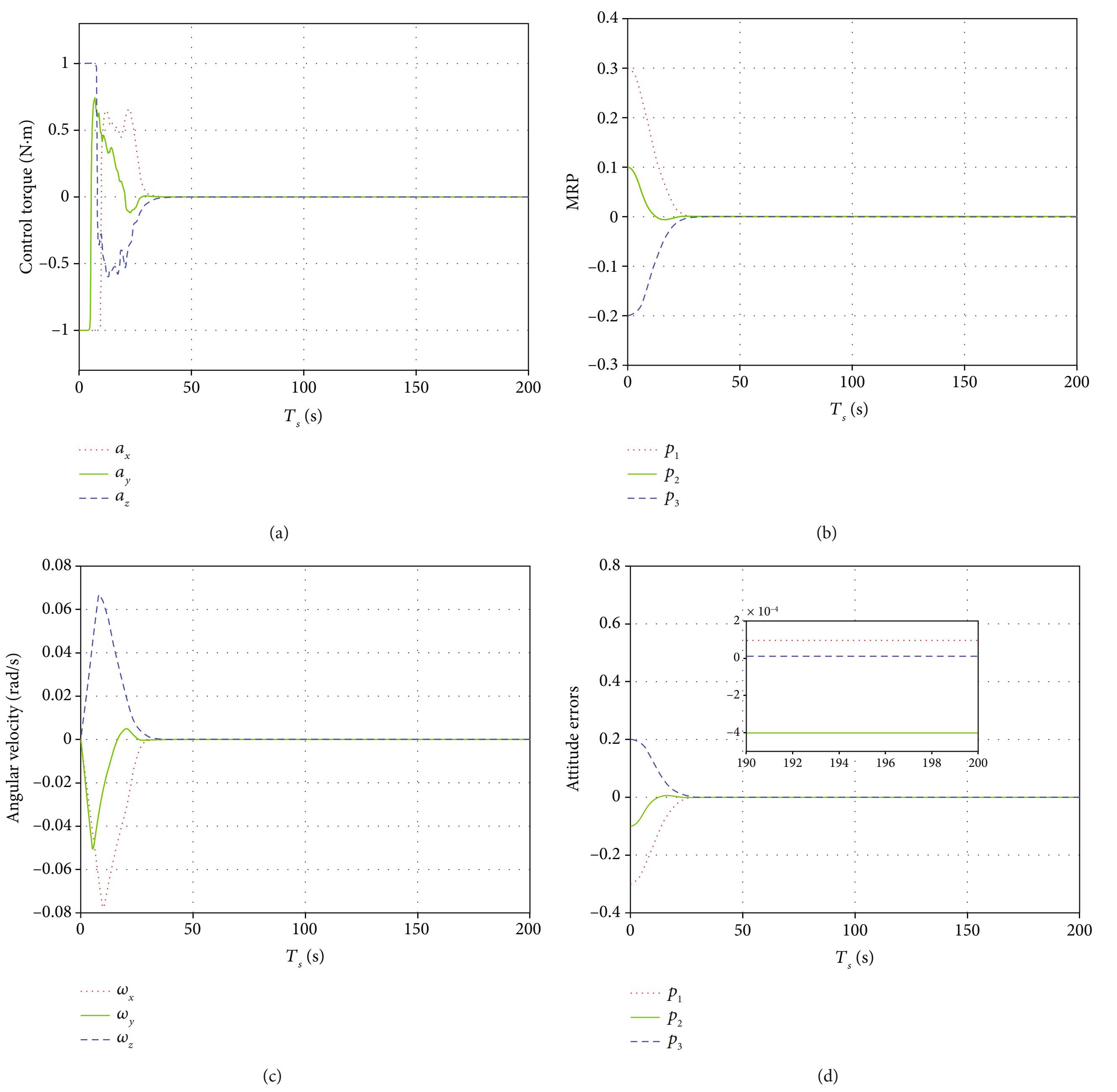

FIGURE 5: Attitude stabilization using the DRL controller under ideal environment: (a) time response of control torque; (b) time response of MRP; (c) time response of angular velocity; (d) the attitude error curve.

(6) Execute the action $a = \arg\ \max_{j=1,2}(Q_{\theta_1}(s, a_j))$ and observe the reward r and the new state s'

(7) Store transition tuple (s, a, r, s') in $\mathscr{R}$

(8) If s' is terminal, reset environment state

(9) If it is time to update the critic networks, then randomly sample a mini batch of N transitions from $\mathscr{R}$

(10) Compute target action:

$$a' = \text{clip}\left(\pi_{\phi'}\left(s'\right) + \text{clip}(\varepsilon, -c, c), -a_{\max}, a_{\max}\right), \quad (13)$$

where c is the scale of exploration noise and $a_{\max}$ is the maximum control torque

(11) Compute target Q value:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}\left(s', a'\right), \quad (14)$$

where γ is the discount factor

(12) Update critic networks by

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i}\left[\frac{1}{N}\sum_{j=1}^{N}\left(y - Q_{\theta_i}(s, a)\right)^2\right]. \quad (15)$$

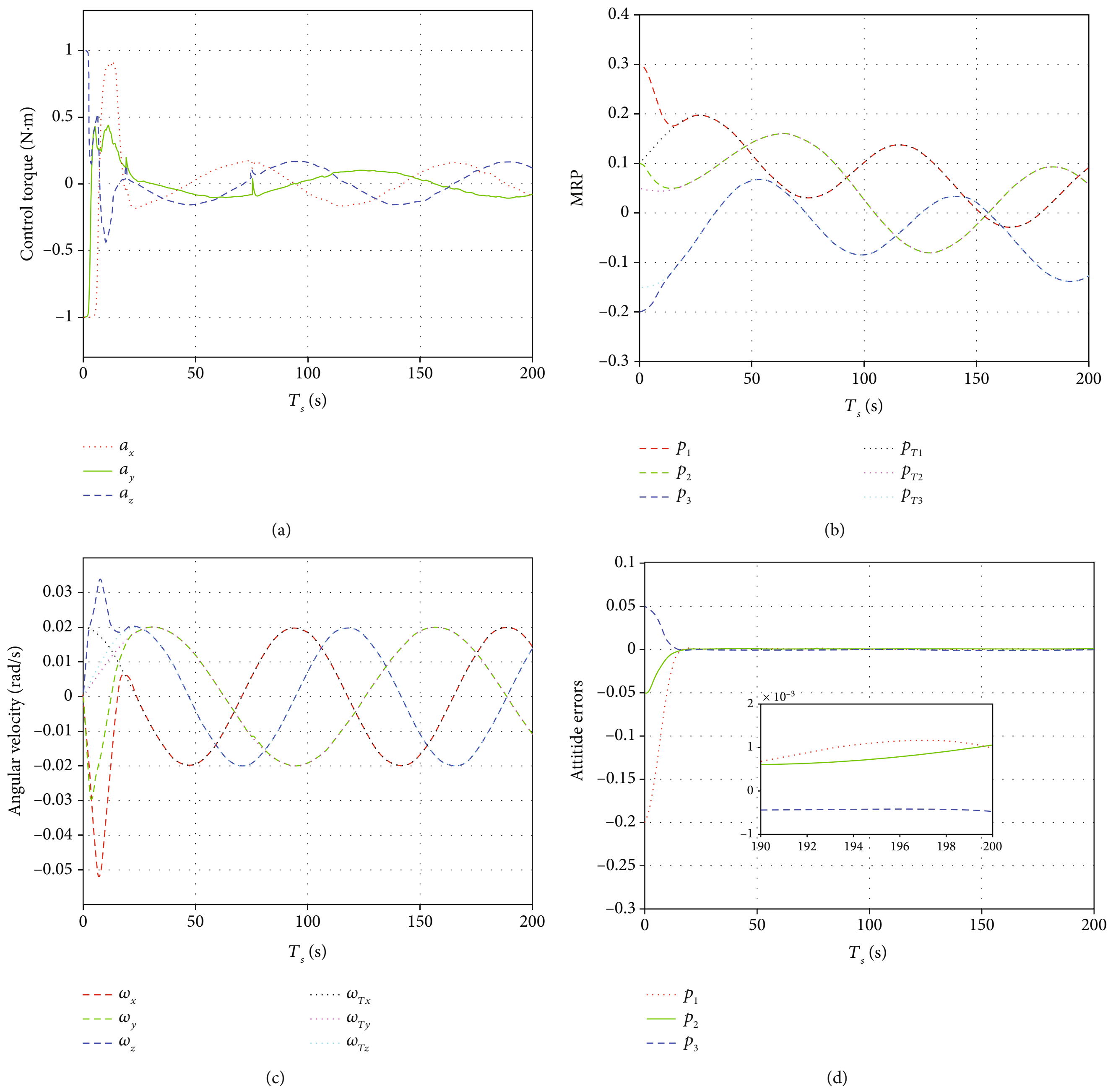

FIGURE 6: Attitude tracking using the DRL controller under ideal environment: (a) time response of control torque; (b) time response of MRP; (c) time response of angular velocity; (d) the attitude error curve.

(13) If it is time to update the actor network, update ϕ by the deterministic policy gradient:

$$\nabla_\phi J(\phi) = \frac{1}{N}\sum_{j=1}^{N} \nabla_a Q_{\theta_1}(s, a)\Big|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s). \tag{16}$$

(14) Update target networks by

$$\begin{aligned} \theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta_i', \\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi', \end{aligned} \tag{17}$$

where τ is the update rate for target model

3.4. Pretraining and Fine-Tuning. When the state space and action space are too large, the DRL algorithm is difficult to be applied in space tasks directly due to the low learning efficiency and the difficulty in training the networks. In order to shorten the training time and avoid dangerous states during the exploration, this paper proposes to use the pretraining and fine-tuning method in deep learning to further improve the learning efficiency.

Pretraining in deep learning is training the machines before they start performing a particular task. Pretraining imitates the way human beings process new knowledge. The weights saved from the previous network will be used

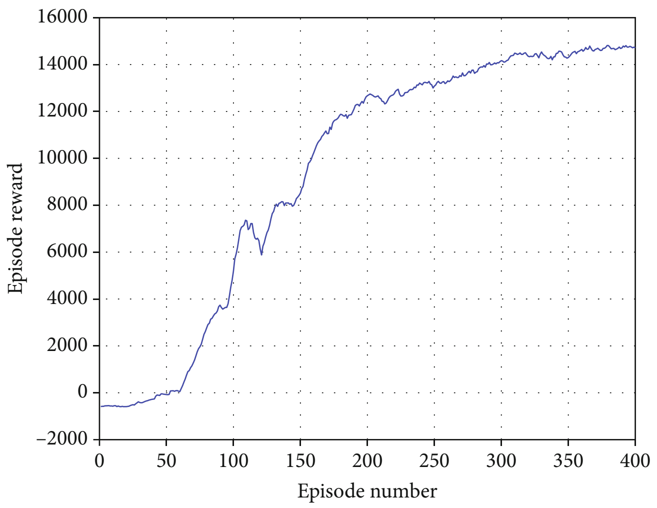

Figure 7: The learning process of TD3 agent under unknown external disturbances.

as the initial weight for the new experiment. In this way, the old knowledge helps new models successfully perform new tasks from old experience instead of from scratch.

A well-established paradigm is to pretrain models using large-scale data and then to fine-tune the models on target tasks that often have less training data. Pretraining has enabled state-of-the-art results on many tasks, including object detection, image segmentation, and action recognition [25].

The pretraining and fine-tuning method makes it possible to deploy the DRL controller on orbit. The real space environment is different from the simulation environment; there are some uncertain information such as unknown disturbances and unknown inertial parameters in space. For the spacecraft attitude control task, the agent which has been pretrained on the ground only needs a small amount of on-orbit training to fine-tune the parameters to obtain good performance.

4. Simulation and Results

In order to verify the effectiveness and superiority of the proposed PID-Guide TD3 algorithm, the simulations are carried out in this section. The following numerical simulations are organized.

Case 1: in the ideal environment without external disturbances, the agent is trained to realize the attitude stabilization control and attitude tracking control of spacecraft, respectively.

Case 2: on the basis of Case 1, the existence of unknown disturbance torques is considered.

Case 3: on the basis of Case 2, the PID-Guide TD3 algorithm is used to accelerate the training speed and convergence stability.

4.1. Simulation Setup. All training processes are done on the computer with an Intel Seon Silver 4210 CPU and an NVIDIA Quadro P2000 GPU. Python 3.7 is used as the project interpreter. The deep learning framework TensorFlow-2.0.0 is used for training the networks under Windows system.

The experiments are conducted in the OpenAI Gym simulation environment. The step size of the Gym simulator, which specifies the duration of each physics update step, is set to 0.1 s to develop highly accurate simulations. The inertia matrix is set as $J = \mathrm{diag}\,(120, 100, 120)\,\mathrm{kg}\cdot\mathrm{m}^2$. The initial states of system are selected as $p_0 = (0.3, 0.1, -0.2)^T$ and $\omega_0 = (000)^T$ rad/s. The target states are $p_d = (000)^T$ and $\omega_d = (000)^T$ rad/s. The maximum value of control torque is $a_{\max} = 1\,\mathrm{N}\cdot\mathrm{m}$.

The policy and value functions are approximated by four-layer neural networks with Relu activations on each hidden layer. The composition of the networks is shown in Table 1.

The hyperparameter settings during the implementation of the algorithm are given in Table 2. All network architecture and hyperparameters used in the three cases are the same.

4.2. Case 1: End-to-End TD3 Algorithm under Ideal Environment. In order to verify the performance of the End-to-End DRL controller and the effectiveness of the reward function proposed in Section 3.2, the agent is trained in an ideal environment without external disturbances to realize the attitude stabilization and attitude tracking control of spacecraft, respectively.

Figure 4 shows the learning process of the End-to-End TD3 algorithm. It can be seen that the algorithm basically achieves convergence after 100 episodes of training, and the rewards for each episode finally stabilized at 15000.

In this case, the control strategy output by the agent, namely, the control torque, is shown in Figure 5(a). Figures 5(b) and 5(c) show the parameter response of the attitude angle and attitude angular velocity under the above-mentioned control torque, respectively. After 30 seconds, the attitude angle has been basically controlled to the

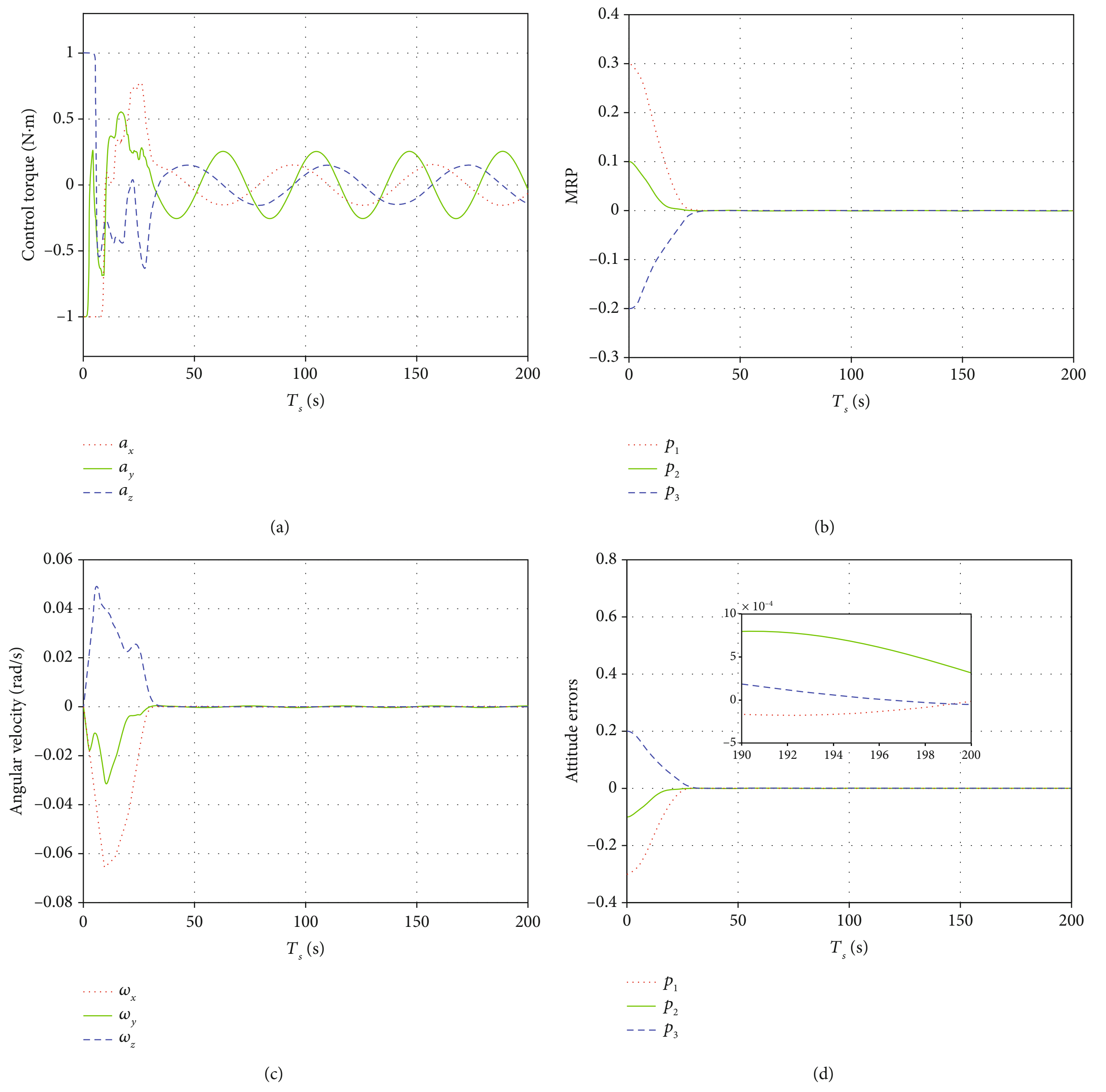

Figure 8: Attitude stabilization using the DRL controller under unknown external disturbances: (a) time response of control torque; (b) time response of MRP; (c) time response of angular velocity; (d) the attitude error curve.

target value, and the angular velocity is basically zero. The convergence speed is fast, and the amount of overshoot is small. The error curve of the attitude angle is shown in Figure 5(d). The error level drops to 10^{-4}, which shows that the control accuracy is relatively high.

The objective of attitude tracking control is to track the desired attitude with $p_{T0} = (0.1, 0.05, -0.15)^T$ and $\omega_T(t) = [0.02 \cos(t/15), 0.02 \sin(t/20), 0.02 \sin(t/15)]^T$ rad/s. The simulation results are illustrated in Figure 6. It can be seen that both the attitude and angular velocity converge to the target attitude rapidly, which indicate that the tracking objective is accomplished with the proposed End-to-End DRL controller.

4.3. Case 2: End-to-End TD3 Algorithm under Unknown External Disturbances. In this case, the spacecraft suffers from external disturbances $d(t) = [0.15 \cos(0.1t), 0.25 \cos(0.15t), 0.15 \sin(0.1t)]$ N · m. The learning process is illustrated in Figure 7. The simulation results are illustrated in Figure 8. Compared with the ideal environment, the training

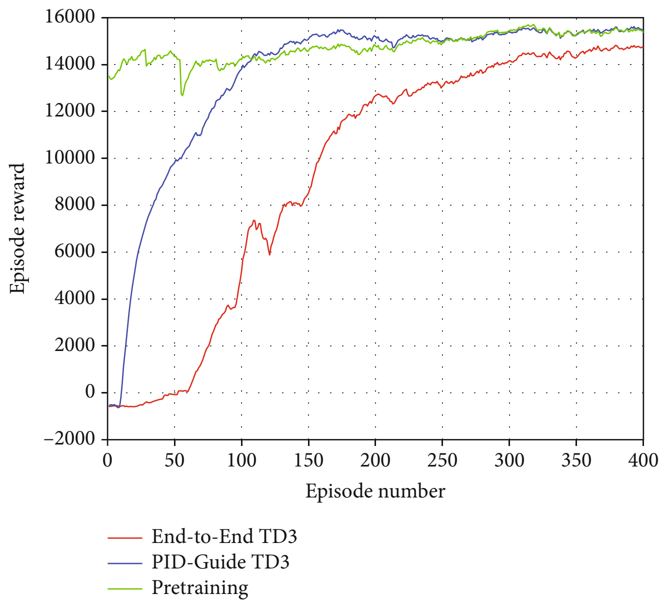

FIGURE 9: The training process of different algorithms.

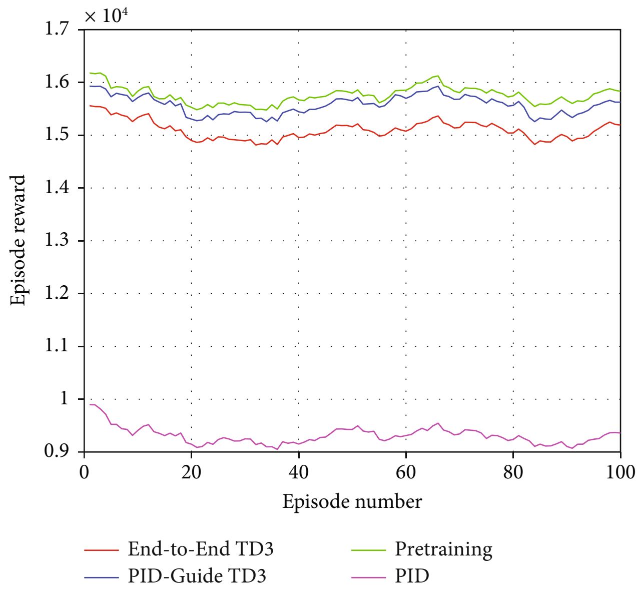

FIGURE 10: The testing process of different algorithms.

speed of the algorithm is slightly reduced, and it still has good convergence accuracy. It can be seen that the disturbance torques are completely compensated by the controller.

4.4. Case 3: PID-Guide TD3 Algorithm under Unknown External Disturbances. In order to verify the superiority of the PID-Guide TD3 algorithm proposed in Section 3.3, the training speed and stability performance of PID-Guide TD3, End-to-End TD3, and pretraining/fine-tuning method are compared, respectively. The training process and testing process are illustrated in Figures 9 and 10. The simulation results are illustrated in Figure 11.

A PID-like controller designed for spacecraft attitude stabilization is used as the guide controller [26] $u = G^T(p)K_p e + G^T(p)K_i\int_0^t e dt + K_d\omega$. The weighting coefficients are chosen as $K_p = 40I_{3\times3}$, $K_i = 0I_{3\times3}$, and $K_p = 60I_{3\times3}$.

From Figures 9 and 10, it can be discerned obviously that all the three algorithms got good convergence performance after training. However, compared with End-to-End TD3 algorithm, the PID-Guide TD3 has significantly faster training speed and higher convergence accuracy. The PID controller itself does not have very high performance, but it can guide the TD3 to produce better sample data at the beginning of training, thus speeding up the training process.

Further, it can also be observed that the pretrained agent only needs a few episodes of training to adapt to the new environment and at the same time avoid the occurrence of

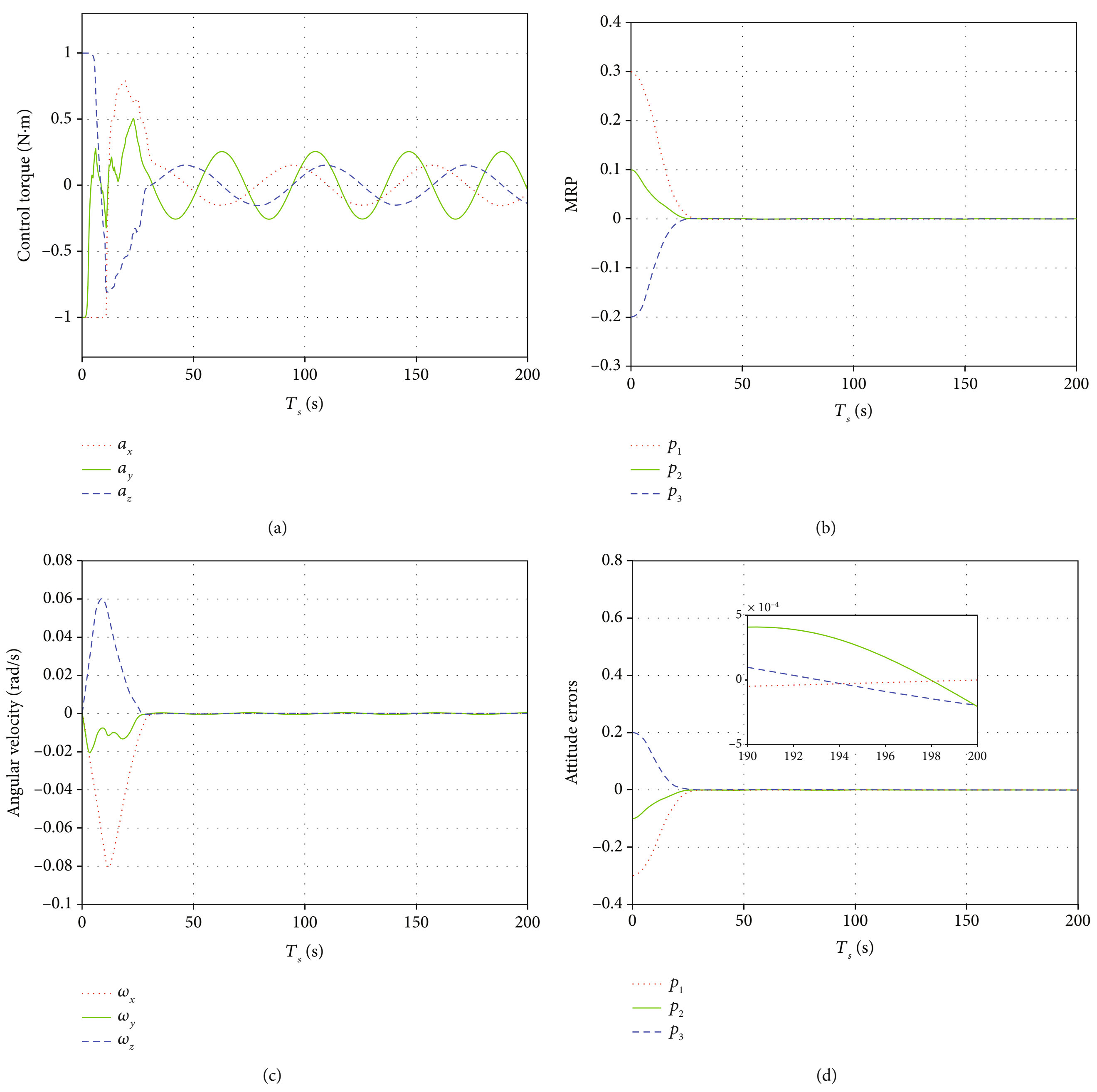

FIGURE 11: Attitude stabilization using the PID-Guide TD3 algorithm under unknown external disturbances: (a) time response of control torque; (b) time response of MRP; (c) time response of angular velocity; (d) the attitude error curve.

dangerous states in the process of exploration. The benefits of pretraining extend beyond merely quick convergence, since pretraining can improve model robustness and uncertainty. Therefore, the pretrained DRL controller can be deployed to the spacecraft and then fine-tune parameters on orbit, instead of training from scratch.

5. Conclusions

A DRL-based control approach is proposed to handle the model-free attitude control problem of OOSS under the guidance of the mixed reward system. The PID-Guide TD3 algorithm based on prior knowledge is proposed to increase the training speed and learning stability of the TD3 algorithm. In addition, the pretraining and fine-tuning method is proposed to realize the deployment of DRL controller in space. Simulation results show that the DRL controller can achieve high-precision attitude stabilization and attitude tracking control with fast response and small overshoot. The learning curves of the three DRL methods illustrate that the proposed PID-Guide TD3 algorithm has faster learning speed and convergence accuracy than the baseline TD3 algorithm. The pretraining and fine-tuning method can make the controller adapt to uncertain and unknown disturbances in the actual environment very quickly.

Data Availability

All data, models, or code generated or used during the study are available from the corresponding author by request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported by the National Defense Science and Technology Innovation Zone of China, grant number 00205501.

References

[1] W. Hu, *Fundamental spacecraft dynamics and control*, John Wiley & Sons, 2015.

[2] J. Su and K. Cai, “Globally stabilizing proportional-integral-derivative control laws for rigid-body attitude tracking,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1260–1264, 2011.

[3] S. J. Paynter and R. H. Bishop, “Adaptive nonlinear attitude control and momentum management of spacecraft,” *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 5, pp. 1025–1032, 1997.

[4] M. Jafari and S. Mobayen, “Second-order sliding set design for a class of uncertain nonlinear systems with disturbances: an LMI approach,” *Mathematics and Computers in Simulation*, vol. 156, pp. 110–125, 2019.

[5] D. Fragopoulos and M. Innocenti, “Stability considerations in quaternion attitude control using discontinuous Lyapunov functions,” *IEE Proceedings-Control Theory and Applications*, vol. 151, no. 3, pp. 253–258, 2004.

[6] Y. Park, “Robust and optimal attitude stabilization of spacecraft with external disturbances,” *Aerospace Science and Technology*, vol. 9, no. 3, pp. 253–259, 2005.

[7] F. Bayat, S. Mobayen, and T. Hatami, “Composite nonlinear feedback design for discrete-time switching systems with disturbances and input saturation,” *International Journal of Systems Science*, vol. 49, no. 11, pp. 2362–2372, 2018.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, 2018.

[9] R. S. Sutton, A. G. Barto, and R. J. Williams, “Reinforcement learning is direct adaptive optimal control,” *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 19–22, 1992.

[10] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[11] H. V. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning in thirtieth AAAI conference on artificial intelligence,” pp. 2094–2100, 2016, https://arxiv.org/abs/1509.06461.

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

[13] H. Xiong, T. Ma, L. Zhang, and X. Diao, “Comparison of end-to-end and hybrid deep reinforcement learning strategies for controlling cable-driven parallel robots,” *Neurocomputing*, vol. 377, pp. 73–84, 2020.

[14] D. Xu, Z. Hui, Y. Liu, and G. Chen, “Morphing control of a new bionic morphing UAV with deep reinforcement learning,” *Aerospace Science and Technology*, vol. 92, pp. 232–243, 2019.

[15] L. Gong, Q. Wang, C. Hu, and C. Liu, “Switching control of morphing aircraft based on Q-learning,” *Chinese Journal of Aeronautics*, vol. 33, no. 2, pp. 672–687, 2020.

[16] Q. Huang, R. Huang, W. Hao, J. Tan, R. Fan, and Z. Huang, “Adaptive power system emergency control using deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1171–1182, 2020.

[17] J. Duan, R. D. Di Shi, H. Li et al., “Deep-reinforcement-learning-based autonomous voltage control for power grid operations,” *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 814–817, 2020.

[18] Z. Ning, R. Y. K. Kwok, K. Zhang et al., “Joint computing and caching in 5G-envisioned Internet of vehicles: a deep reinforcement learning-based traffic control system,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020.

[19] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, “Rigid-body attitude control,” *IEEE Control Systems*, vol. 31, no. 3, pp. 30–51, 2011.

[20] S. Bandyopadhyay, S. J. Chung, and F. Hadaegh, “Attitude control and stabilization of spacecraft with a captured asteroid,” in *AIAA Guidance, Navigation, and Control Conference*, 2015no. 596.

[21] W. Shi, S. Song, C. Wu, and C. L. P. Chen, “Multi pseudo q-learning-based deterministic policy gradient for tracking control of autonomous underwater vehicles,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3534–3546, 2019.

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

[23] D. Dewey, “Reinforcement learning and the reward engineering principle,” *AAAI Spring Symposium Series*, pp. 13–16, 2014.

[24] Y. H. Wu, Z. C. Yu, C. Y. Li, M. J. He, B. Hua, and Z. M. Chen, “Reinforcement learning in dual-arm trajectory planning for a free-floating space robot,” *Aerospace Science and Technology*, vol. 98, article 105657, 2020.

[25] K. He, R. Girshick, and P. Dollár, “Rethinking imagenet pre-training,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4918–4927, Seoul, Korea (South), Korea (South), 2019.

[26] C. Li, K. L. Teo, B. Li, and G. Ma, “A constrained optimal PID-like controller design for spacecraft attitude stabilization,” *Acta Astronautica*, vol. 74, pp. 131–140, 2012.