Journal of Applied Analysis and Computation
Volume 12, Number 3, June 2022, 868–894
Website:http://www.jaac-online.com
DOI:10.11948/20210442

DEEP DETERMINISTIC POLICY GRADIENT WITH GENERALIZED INTEGRAL COMPENSATOR FOR HEIGHT CONTROL OF QUADROTOR

Anlin Liu[1], Lei Liu[1,†] , Jinde Cao[2,3] and Fawaz E.Alsaadi[4]

Dedicated to Professor Jibin Li on the occasion of his 80th birthday.

Abstract This paper is corned with the desired height control of the quadrotor under the framework of deep deterministic policy gradient with prioritized experience replay (PER-DDPG) algorithm. The reward functions are designed based on an out-of-bounds plenty mechanism. By introducing a generalized integral compensator to the actor-critic structure, the PER-DDPG-GIC algorithm is proposed. The quadrotor is controlled by a neural network trained by the proposed PER-DDPG-GIC algorithm, which maps the system state to control commands directly. The simulation results demonstrate that introduction of generalized integral compensator mechanism can effectively reduce the steady-state error and the reward has been greatly enhanced. Moreover, the generalization ability and robustness, with respect to quadrotor models with different weights and sizes, have also been verified in simulations.

Keywords Height control, quadrotor vehicle, deep reinforcement learning, deterministic policy gradient, neural network.

MSC(2010) 93C10, 93C40.

1. Introduction

As a kind of unmanned aerial vehicles (UAVs), quadrotor aircrafts play a wide role in agricultural plant protection [33], industrial inspection [24], air transportation [9], and military fields [15] owing to their lightweight, small scale, low cost, and flexibility. The flight control systems are crucial in accomplishing various tasks efficiently. Therefore, many researchers focus on the intelligent technology training of unmanned flight systems, which has received more and more attention [6,17,30].

Since the quadrotor aircraft is a nonlinear, strong coupling, and underactuated system [4], and the external interference and model uncertainties are unavoidable in the actual control process, which makes the controller design becomes more difficult and complicated. In order to deal with the difficulties above, a series of control

†The corresponding author.
Email: liulei_hust@163.com(L. Liu), liulei_hust@hhu.edu.cn(L. Liu)
[1]College of Science, Hohai University, Nanjing 210098, China
[2]School of Mathematics, Southeast University, Nanjing 210096, China
[3]Yonsei Frontier Lab, Yonsei University, Seoul 03722, South Korea
[4]Department of information Technology, Faculty of Computing and IT, King Abdulaziz University, Jeddah, Saudi Arabia

strategies have been proposed. The principle of the proportional-integral-derivative (PID) controller [11] is the negative feedback to the system error, which has been successfully applied in a stable environment. However, the PID gains are manually adjusted by trial and error, which requires solid professional experience. In order to deal with the nonlinearities and uncertainties of the model, a number of non-linear control strategies have emerged, such as sliding mode control [23], backstepping [2], adaptive control [32], and so on. What is more, some studies combine these methods to improve the robustness of control strategies [20]. Nevertheless, a more sophisticated model usually brings about a more complex control strategy, which is not conducive to design and implement.

With the development of artificial intelligence, many control strategies based on neural networks and reinforcement learning have emerged. Neural network control is a model-free method, which has become an important branch of intelligent control. Dierks [6] proposed a new nonlinear controller using neural networks and output feedback, which can guarantee the UAV tracks the desired trajectory. Reinforcement learning algorithm make the agent interact with the environment and learn the control policy directly from the neural network. Thus, there is no need to assume and simplify the dynamic model. The control strategy with reinforcement learning is a relatively new idea, which can trace back to 2005. Waslande [30] achieved accurate tracking of the quadrotor with the standard strategy iteration method. In recent years, William [17] used deep deterministic policy gradient (DDPG), trust region policy optimization (TRPO), and proximal policy optimization (PPO) algorithms for internal loop attitude control respectively, which achieved better performances than PID controller. Similar approaches based on reinforcement learning have been proposed for quadrotor control, please refer to [7,8,10,18,21]. Moreover, most of the existing reinforcement learning control strategies just focus on attitude and speed control. Tiwari [26] used the augmented random search (ARS) algorithm to make the quadrotor reach the target position successfully. Therefore, the reinforcement learning method for position control of the quadrotor is the focus of our research. In recent years, Deepmind has proposed two effective algorithms, DQN and DDPG. DQN is a method based on value function, which is difficult to deal with large action space, especially in the case of continuous action. However, DDPG is based on actor-critic method. In the aspect of action output, a network is used to fit the policy function and directly output the action, which can deal with the output of continuous action and large action space. Based on the above reasons, this paper takes DDPG algorithm as the framework.

With the development of reinforcement learning, several scholars optimized some aspects of the reinforcement learning algorithm to further improve the performance of the algorithm in recent years. DeepMind firstly developed a framework for prioritizing experience and applied it to Deep Q-Networks (DQN) algorithm, which was proved to have better performance than the DQN algorithm, especially in learning speed and final performance [22]. In fact, experience replay enables reinforcement learning to make full use of the past experiences. It is a technique to a certain extent. However, the stability and speed of convergence of reinforcement learning, as well as the eventual performance of the learned policy, are strongly dependent on the expeperiences being replayed [5]. Based on the above ideas, Hou [13] introduced the idea of prioritized experience replay into the DDPG algorithm and found that it greatly shortened the training time and improved the stability of the learning process. In addition, compared with the DDPG algorithm, the DDPG with priori-

tized experience replay (PER-DDPG) algorithm is less affected by hyperparameters, which enhances the robustness of the algorithm. For a detailed understanding of this, please refer to [3, 16, 29, 31, 34]. Besides, the integral compensator has also received attention in recent years. This technique not only considers the state error at the current moment, but also considers the state error at the past moment. Wang [28] and Hu [14] introduced integral compensators into DDPG and PPO algorithms for robust quadrotor control, which effectively reduced the steady-state error.

Inspired by the above-mentioned papers, this paper further improves the PER-DDPG algorithm to obtain better performance on quadrotor control via a generalized integral compensator mechanism. It is worth mentioning that the integral compensator proposed in [28] and [14] treats errors in the past state equally. In fact, the closer the error is to the current moment, the more valuable it is. Based on these ideas, this paper gives weight to the state error in the past moment according to the distance to the current moment, and forms a generalized integral compensator. By taking the generalized integral compensated error as the input to the neural network of actor-critic structure, the PER-DDPG with generalized integral compensated (PER-DDPG-GIC) algorithm has been proposed. The proposed algorithm effectively eliminates the steady-state error and has remarkable robustness on different weights and sizes of the quadrotor. The main contributions of this paper are concluded as follows:

1) By virtual of a reward function with out-of-bounds penalty, the desired height control of the quadrotor is implemented under the framework of the PER-DDPG algorithm.

2) The performance of PER-DDPG algorithm has been improved via a novel reward function that embodies the reward of out-of-bounds plenty and the generalized integral compensator. Compared with the algorithm without the generalized integral compensator, the proposed algorithm can improve the tracking accuracy.

3) In the actor-critic structure, a generalized integral compensator is embedded and the PER-DDPG-GIC algorithm is proposed. By taking the generalized integral compensated error as the input to the neural network, the steady-state error can be effectively eliminated. Furthermore, the remarkable robustness on different weights and sizes of the quadrotor is verified.

The rest of this paper is organized as follows. The dynamic of the quadrotor is established in section 2. The proposed reinforcement learning policy is presented in section 3. In section 4, the simulation details, results, and discussions are provided to indicate the effectiveness of the proposed controller. Finally, the conclusion is given in section 5.

Notations: The following notations are utilized throughout this paper: the superscript $'T'$ represents matrix transposition. diag denotes diagonal matrix. ∇ refers to gradient calculation.

2. Quadrotor modeling

In this section, the creation process of the dynamic model of a quadrotor aircraft is given. The basic structure and frames of a quadrotor are shown in Fig. 1.

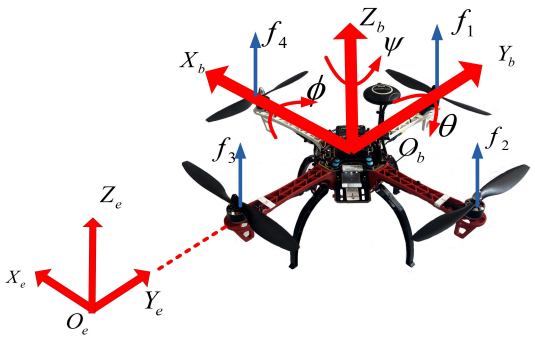

Figure 1. Structure and frames of the quadrotor.

To describe the quadrotor kinematics model, two coordinate systems are established: the inertial coordinate system $E = \{o_e, x_e, y_e, z_e\}$ and the body-fixed coordinate system $B=\{o_b, x_b, y_b, z_b\}$. The mass center of the quadrotor in the inertial coordinate system is defined as $\boldsymbol{p} = [x, y, z]^T$. Correspondingly, $\dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}$ denotes the velocity and the acceleration. Besides, the attitude of the quadrotor is represented by $\boldsymbol{\eta} = [\phi, \theta, \psi]^T$, where ϕ, θ and ψ are roll, pitch and yaw angles respectively. Correspondingly, $\dot{\boldsymbol{\eta}}, \ddot{\boldsymbol{\eta}}$ describe the rotational angular velocity and acceleration.

2.1. Translational motion

According to Newton's second law of motion, the simplified dynamic model of the quadrotor can be obtained:

$$\boldsymbol{F_e} = \boldsymbol{R}\boldsymbol{F_l} + \boldsymbol{G} = m\ddot{\boldsymbol{p}}, \tag{2.1}$$

where $\boldsymbol{F_e}$ is the resultant force applied to the quadcopter, $\boldsymbol{R}$ is the transformation matrix, $\boldsymbol{F_l}$ is the lift force, G is the gravity of the quadrotor, m is the mass of the quadrotor, and $\ddot{\boldsymbol{p}}$ is the acceleration of the quadrotor in the inertial coordinate system. $\boldsymbol{F_l} = [0, 0, T_z]^T$ denotes the lift vector, where $T_z = \sum_{i=1}^{4} T_i$. T_i is defined in the body-fixed frame, while the gravity of the quadcopter and other external forces are defined in the inertial frame. Therefore, the translational matrix $\boldsymbol{R}$ is used to describe the transformation between the body-fixed frame and the inertial frame. Translational matrix $\boldsymbol{R}$ is given as follows:

$$\boldsymbol{R} = \begin{bmatrix} C_\phi C_\theta & C_\phi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ S_\psi C_\theta & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\phi C_\theta \end{bmatrix} \tag{2.2}$$

where $S_{\{\cdot\}}$, $C_{\{\cdot\}}$ represent $\sin(\cdot)$, $\cos(\cdot)$, Euler angle ϕ, θ, ψ are roll, pitch, and yaw angle respectively. From the above, the translational motion of the quadrotor dynamic can be expressed as:

$$\begin{cases} \ddot{x} = T_z \left(C_\phi S_\theta C_\psi + S_\phi S_\psi\right)/m, \\ \ddot{y} = T_z \left(C_\phi S_\theta S_\psi - S_\phi C_\psi\right)/m, \\ \ddot{z} = \left(T_z C_\phi C_\theta - mg\right)/m. \end{cases} \tag{2.3}$$

2.2. Rotational motion

For the rotational motion of the quadrotor, according to the law of rigid body rotation, the total external moment applied to the quadrotor can be expressed as:

$$\boldsymbol{M} = \boldsymbol{I}\dot{\boldsymbol{w}} + \boldsymbol{w} \times \boldsymbol{I}\boldsymbol{w}, \tag{2.4}$$

where $\boldsymbol{I}$ =diag(I_x, I_y, I_z) is the moment of inertia of the quadrotor, $\boldsymbol{w} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$ is the Euler angular velocity, and $\boldsymbol{M}$ is the control torque generated by different lifts. $\boldsymbol{M}$ is given as follows:

$$\boldsymbol{M} = \begin{bmatrix} L(T_4 - T_2) \\ L(T_1 - T_3) \\ K_\psi(T_1 - T_2 + T_3 - T_4) \end{bmatrix}, \tag{2.5}$$

where L is the distance between the center mass of the quadrotor and the rotor axis, and K_ψ is the reaction moment coefficient, T_i are the lift forces provided by different rotors. From the above, the quadrotor dynamic of the rotational motion can be expressed as:

$$\begin{cases} \ddot{\phi} = L\left(T_2 - T_4\right)/I_x, \\ \ddot{\theta} = L\left(T_3 - T_1\right)/I_y, \\ \ddot{\psi} = K_\psi\left(T_1 - T_2 + T_3 - T_4\right)/I_z. \end{cases} \tag{2.6}$$

The translational and rotational motions of the quadrotor are all the motions of the quadrotor in space. Therefore, the quadrotor kinematics model without considering the aerodynamic drag force, friction, and wind can be obtained:

$$\begin{cases} \ddot{x} = T_z\left(C_\phi S_\theta C_\psi + S_\phi S_\psi\right)/m, \\ \ddot{y} = T_z\left(C_\phi S_\theta S_\psi - S_\phi C_\psi\right)/m, \\ \ddot{z} = \left(T_z C_\phi C_\theta - mg\right)/m, \\ \ddot{\phi} = L\left(T_2 - T_4\right)/I_x, \\ \ddot{\theta} = L\left(T_3 - T_1\right)/I_y, \\ \ddot{\psi} = K_\psi\left(T_1 - T_2 + T_3 - T_4\right)/I_z. \end{cases} \tag{2.7}$$

3. Proposed Approach

In this section, firstly, the DDPG algorithm structure for quadrotor control is given. Secondly, the DDPG with prioritized experience replay (PER-DDPG) algorithm is applied to control the quadrotor aircraft. Then, a PER-DDPG-GIC is proposed on the basis of the first two sections.

3.1. DDPG control policy

For the problem of quadrotor control, the main goal is to find an appropriate control policy that can drive the quadrotor from the initial state to the desired state in a fast and stable manner. Algorithms based on policy gradient are most suitable for solving such continuous actions and states problems. In this paper, we select the DDPG algorithm and its improved algorithms for quadrotor control.

3.1.1. State and action specification

Reinforcement learning uses a Markov decision process (MDP) to model the controller, it can be described by a five-tuple $\langle S, A, P, R, \gamma \rangle$, where S is the states, A is the actions, P is the state transition probability function, R is the reward function, and γ is the discount factor. In this subsection, the state and action for the quadrotor control are specified. Theoretically, the state is supposed to be a collection of the information that the quadrotor can obtain. According to the quadrotor model established in section 2, the state in this experiment is a 12-dimensional vector $s_t = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$. It is worth mentioning that the dimension of the state space is very small, which makes it easy in function approximation and exploration. Besides, the small dimension of state space reduces the difficulty of network training and time consumption.

The quadrotor moves in translation and rotation through the lift force of four propellers. In our scenario, the control commands are represented by action $a = [r_1, r_2, r_3, r_4]$, where r_1, r_2, r_3, r_4 are the rotor speeds. In this paper, the same control law is applied to the four rotors, and the action is supposed to be $a = [r, r, r, r]$. In this way, the quadrotor can reach the specified height without drift.

The deterministic policy is a strategy that maps each state to the action with the greatest probability, so the action is uniquely determined. In order to achieve exploration, an Ornstein Uhlenbeck (OU) noise is added :

$$\bar{A}(s_t) = A^{\mu}(s_t) + N_t, \tag{3.1}$$

where $\bar{A}(s_t)$ is the final policy, $A^{\mu}(s_t)$ is the deterministic policy, N_t is the OU noise. In the case of one dimension, the OU process is defined by a stochastic differential equation:

$$dN_t = \theta(\mu - N_t)dt + \sigma dB_t, \tag{3.2}$$

where θ is a parameter, μ is mean of the noise, σ is the weight of noise, B_t is the standard Brownian motion.

3.1.2. Network structure

There are four networks used in the DDPG algorithm, namely an actor network, a target actor network, a critic network and a target critic network. The actor network provides the action that the quadrotor should take, and the critic network updates the Q value which is used by the DDPG algorithm to update the actor network later. The input of the actor network is the state, and the output is the action. The input of the critic network is the state along with the action, and the output is the Q value. There are two hidden layers of 400 relu nodes and 300 relu nodes in the actor network and critic network respectively. In fact, we have tried

many other network structures of different nodes, layers, and activation functions. Finally, the following network structure can satisfy the effect of quadrotor control. The actor-critic network structure of the DDPG algorithm for quadrotor control is shown in Fig. 2: In addition, the previous practice has proved that if only a single

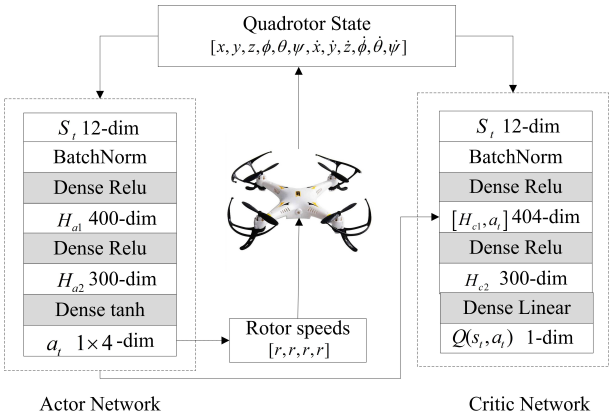

Figure 2. Actor-critic network of DDPG for quadrotor control.

neural network algorithm is used, the learning process is very unstable, because the network parameters are updated frequently and are used to calculate the gradient of the critic network and policy network. To address this bottleneck, the DDPG algorithm is applied, which adds a target actor network and a target critic network respectively. The parameters of the target network are updated by the soft update method, which is more stable and easy to converge. The soft update method is denoted as follows:

$$w' \leftarrow \tau w + (1-\tau)w', \tag{3.3}$$

$$\mu' \leftarrow \tau\mu + (1-\tau)\mu', \tag{3.4}$$

where w', μ' are the target critic network parameters and target actor network parameters respectively, τ is the soft update rate of the target network.

3.1.3. Policy optimization

To optimize the policy, parameters are adjusted based on the gradient of the expected returns. The idea of the above algorithm is the policy gradient theorem [25]:

$$\nabla_\mu J(A^\mu) = E_{s\sim\rho^{A^\mu}}[\nabla_\mu \log_{A^\mu}(s,a) Q^{A^\mu}(s,a)], \tag{3.5}$$

where $J(A^\mu)$ is the performance objective, ρ^{A^μ} is the state distribution of the policy A^μ, $Q^{A^\mu}(s,a)$ is the actual state-value function, $J(A^\mu)$ is the expected value of $Q^{A^\mu}(s,a)$ where $s \sim \rho^{A^\mu}$.

The DDPG algorithm adopts the actor-critic neural networks, in which the actor network is responsible for the policy iteration and the critic network is responsible for value estimation. The parameters in critic network are updated by minimizing the TD-error, where the TD-error is defined as:

$$\delta_{t+1} = r_t + \gamma Q_w(s_{t+1}, \mu(s_{t+1})) - Q_w(s_t, a_t), \tag{3.6}$$

where r_t is the reward, γ is the discount, $Q_w(s_t, a_t)$ is the Q value of the critic target network. The parameters in critic network are updated by minimizing the loss function:

$$\mathcal{L} = \frac{1}{N}\sum_i \left(r_i + \gamma Q_w(s_{i+1}, \mu(s_{i+1})) - Q_w(s_i, a_i)\right)^2, \tag{3.7}$$

$$w_{t+1} = w_t + \alpha_w \nabla_w \mathcal{L}(w), \tag{3.8}$$

where $\mathcal{L}$ is the loss function, N is the batch size, w is the critic network parameter, α_w is the learning rate of critic network. The goal of the actor network is to select the most suitable action and update it following the DPG theorem:

$$\nabla_\mu J(\mu) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\mu A^\mu(s_i) \nabla_a Q_w(s_i, a_i)|_{a_i = A^\mu(s_i)}, \tag{3.9}$$

$$\mu_{t+1} = \mu_t + \alpha_\mu \nabla_\mu J(\mu), \tag{3.10}$$

where $J(\mu)$ is the performance objective, μ is the actor network parameter, $A^\mu(s)$ is the policy that characterizes the actor network, α_μ is the learning rate of actor network. In the view of above theoretical support, the implementation framework of DDPG for quadrotor control is shown in Fig. 3.

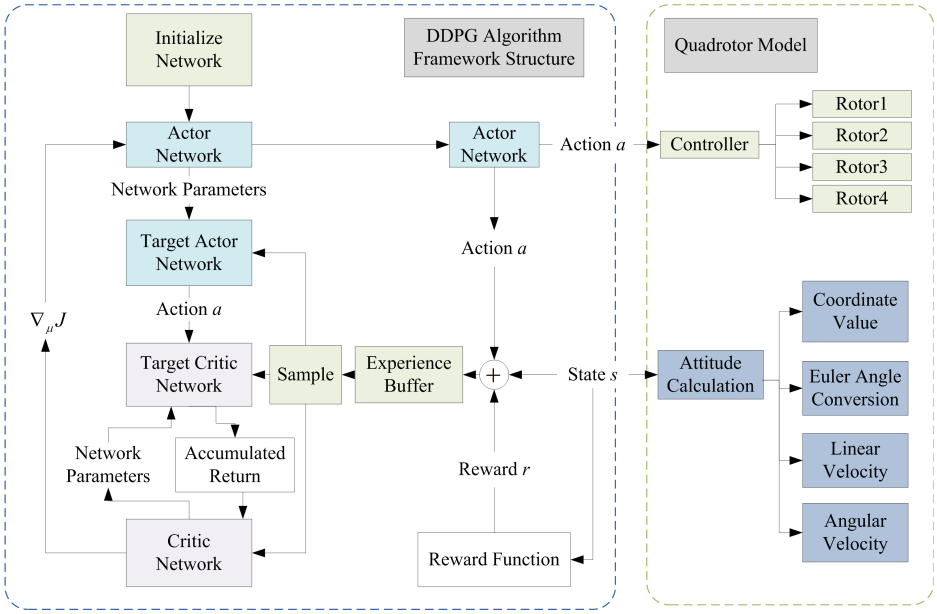

Figure 3. System network framework structure.

3.2. PER-DDPG control policy

Consider that in the DDPG algorithm, experience transitions are uniformly sampled from the replay buffer. In fact, we certainly expect that high-value transitions to be sampled more frequently. So, we use a framework for prioritizing experience. The main reason is to replay important transitions more frequently and learn more efficiently. Firstly, store the experience of each time step $e = (s_t, a_t, r_t, s_{t+1})$ in the

experience buffer D. The uniform experience replay is to randomly sample a batch of experience B from the replay buffer during each update, and use the experience transitions to update the network parameters. However, the priority experience replay is to replay transitions with high expected learning progress. According to (3.6), the larger the temporal-difference (TD) error, the larger the gap between the current Q value and the target Q value, and the more worth learning. Due to the large TD-errors, we give the transition a higher priority value, that is, the probability of sampling transition is large. The probability of sampling transition i is defined as:

$$P(i) = \frac{p_i^\alpha}{\sum p_i^\alpha}, \tag{3.11}$$

where P is the probability of sampling transition, p_i is the priority of transition, where $p_i = |\delta_t + \varepsilon|$, δ_t is the TD error, α represents the magnitude of the priority. In this paper, $\alpha = 0.4$ and $\varepsilon = 10 - 5$.

Remark 3.1. To avoid overfitting, the priority experience replay adopted in this paper only increases the sampling probability of transition with large TD errors, not just only sample a batch of transitions with high expected learning progress. This ensures the diversity of transitions and avoids overfitting to a certain extent.

3.3. Generalized Integral Compensator

In many cases, the steady-state error cannot be eliminated no matter what training tricks and hyperparameters are used owing to the inaccurate estimation of the action-value function in critic network [12,27]. In the traditional DDPG algorithm, the actor network is to learn a policy that can maximize the expected value of action-value function $J(\mu)$. Therefore, inaccurate estimation of the action-value function will lead to deviations when updating the policy, and cannot track the desired point accurately. Considering that the PID controller introduces an integral part which is the integral of the tracking error concerning time to eliminate the steady-state error, an integral compensation is introduced to the DDPG algorithm for quadrotor control. The error with integral compensation is expressed as:

$$S_c^t = S_e^t + \beta \sum\nolimits_{i=1}^{t} S_e^i, \tag{3.12}$$

where S_e is the error state, for instance, $x_e = |x - x_d|$, $y_e = |y - y_d|$, $z_e = |z - z_d|$ are the differences between the current coordinate position and the desired coordinate position, S_c is compensated error, β is the integral gain, superscript is used to indicate the time step. The core idea is to construct a set of errors with generalized integral compensation instead of the direct error. Compared with the integral compensation proposed in [28], the generalized integral compensation proposed in this paper adds discounts on β. The error with generalized integral compensation is as follows:

$$S_g^t = S_e^t + \sum\nolimits_{i=1}^{t} \beta^{N+1-i} S_e^i, \tag{3.13}$$

where N is the total time step, superscript is used to indicate the time step, S_g is the generalized compensated error.

Algorithm 1 PER-DDPG-GIC for quadrotor control

Require:

Randomly initialize the weight of the actor network A^{μ}, critic network Q^{w}, the replay buffer D

Ensure:

for episode from 1 to EpisodeMax **do**
  Observe the initial quadrotor state s_t.
  for time step from 1 to StepMax **do**
    Choose action $a_t = A^{\mu}(s_t) + n_t$.
    Run the dynamic model according to the control signal a_t.
    Observe reward r_t and reach the next state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D.
    Replay important transitions based on sampling probability from the replay buffer.
    Update the critic network following (3.7) (3.8).
    Update the actor network following (3.9) (3.10).
    Update the target network following (3.3) (3.4).
    if S_{t+1} exceeds the safe range **then**
      break
    end if
  end for
end for

Remark 3.2. Compared with the integral compensator in [28] and [14], the generalized integral compensator proposed in this paper gives weight to the error in the past moment according to the distance to the current moment. It means that weight is given according to the importance of the error in the proposed generalized integral compensator, while [28] and [14] treats errors in the past state equally.

Remark 3.3. In this paper, the integral gain β is 0.2. That is to say, the error coefficient is 0.2 at time step N, the error coefficient is 0.04 at time step $N-1$, and the error coefficient is β^{N+1-t} at time step t. It means that the closer the error is to the current moment, the error coefficient is larger. This method pays more attention to the error of the previous step at the current moment, and the weight of the error farther from the current moment decreases exponentially.

Based on the above theoretical support, we applied the generalized integral compensator mechanism in the experiment. Firstly, the GIC mechanism has been integrated into the reward function, the PER-DDPG with GIC reward algorithm is presented. Then, by incorporating the GIC mechanism into the actor-critics structure, the PER-DDPG-GIC algorithm has been proposed. The algorithm flow of the PER-DDPG-GIC algorithm for quadrotor control is shown in Algorithm 1:

4. Experiments and results

In this section, DDPG with different reward functions and three groups of comparative experiments are conducted respectively, the first group is the comparison of the training effect of DDPG and PER-DDPG algorithm, the second group is the comparison of the training effect of PER-DDPG and PER-DDPG with GIC reward

algorithm and the last group is the comparison of the training effect of PER-DDPG and the PER-DDPG-GIC algorithm. The three simulations are called experiment 1, experiment 2, and experiment 3.

4.1. Experimental description

In the simulation, the given algorithm is used for quadrotor control. The basic model parameters are shown in Table 1:

Table 1. Parameters of the quadrotor aircraft

Parameter	Description	Value	Unit
m	Mass	0.958	kg
L	Motor-to-center distance	0.2	m
I_x	Moments of inertia of frame on x axis	0.02517	$kg \cdot m^2$
I_y	Moments of inertia of frame on y axis	0.02517	$kg \cdot m^2$
I_z	Moments of inertia of frame on z axis	0.04153	$kg \cdot m^2$
g	Acceleration due to Gravity	9.81	m/s^2

The experiments run on Windows operating system powered by an Intel Core i7-9700CPU@ 3.00 GHz. The drone simulation is programmed with Python. For network training optimization, this paper uses the built-in Adam optimizer. The network training parameter settings of DDPG are shown in Table 2:

Table 2. Training parameters of DDPG algorithm

Parameter	Value
Discount factor γ	0.99
Learning rate of actor network α_μ	0.0001
Learning rate of critic network α_ω	0.001
Batch size N	128
Buffer size M	100000
Soft update rate τ	0.001
Noise σ	0.01
Time step/s	0.05
Maximum steps in an episode/s	10

Each experiment is conducted in 1000 episodes, and the maximum time in each episode is 10 seconds. A time step is 0.05 seconds and the maximum time steps in an episode are 200. Each episode can be divided into three steps. Firstly, the agent needs to select an action from the current state through the actor network. Secondly, the agent takes action based on (3.9)-(3.10) and ends up in a new state. Lastly, repeat this process until the end of the episode.

To ensure the safety of the quadrotor during flight, safe ranges are set. For the x-axis, y-axis and z-axis, safe ranges are set to [-150,150], [-150,150] and [0,300] respectively. As a gesture to security, two termination conditions are set, one is going beyond the safe range, and the other is exceeding the maximum time steps of the episode.

4.2. Target height control task

The task of the experiments is to use the given algorithms to drive the quadrotor from the initial position to hover at the desired position. The initial position is $[0, 0, 0]$, and the desired position is set to $[0, 0, 10]$. To be specific, the agent needs to use rewards and punishments information to learn by trial and error, and finally reach the desired height without any prior knowledge.

Experiment 1. In this experiment, DDPG with different reward functions are used in the height control of quadrotor. As a crucial element in RL, the reward function guides the expected behavior of the agent. The agent takes action after interacting with the environment, and the environment will give rewards according to the quality of the behavior. All goals in reinforcement learning can be described as maximizing the total reward received by the agent. Therefore, an effective reward function is easier to improve the training effects and convergence speed of the algorithm. In the light of the theory of reward shaping [19], a novel reward function is designed to guide the agent to move within safe ranges. The reward function designed in this experiment is defined as:

$$reward = \begin{cases} 1 - 0.01\sqrt{x_e^2 + y_e^2 + z_e^2}, m = 0, \\ 1 - 10 - 0.01\sqrt{x_e^2 + y_e^2 + z_e^2}, m = 1, \end{cases} \tag{4.1}$$

where x_e, y_e, z_e are the errors between the current coordinate position and the desired coordinate position respectively, m is the symbol that judges whether the quadrotor is going beyond the safe range, which is indicated by 0 and 1.

Remark 4.1. In view of the theory of reward shaping, the reward function is designed. Since it is difficult for the agent to reach the ultimate goal soon, the sparse reward may not work. We need to guide the agent to reach the goal through multiple reward settings gradually. In each time step, a basic reward of 1 is given to the agent, which can overcome conservative behaviors and encourage exploration. The distance between the current position and the desired position is used as a penalty item. If the agent goes beyond the safe range, a penalty of -10 will be given and the episode terminates.

The reward functions are designed based on an out-of-bounds plenty mechanism. The reward function without out-of-bounds plenty is:

$$reward = 1 - 0.01\sqrt{x_e^2 + y_e^2 + z_e^2}. \tag{4.2}$$

In [1], the soft actor-critic (SAC) algorithm is used for the position and attitude control of quadrotor. The reward function in [1] is used in the framework of DDPG. The reward function in [1] is:

$$reward = 1.5 - 1.0\sqrt{x_e^2 + y_e^2 + z_e^2}. \tag{4.3}$$

In this section, the novel reward function based on an out-of-bounds plenty mechanism (4.1) is compared with the reward function in (4.2) and (4.3). The accumulated reward and steady-state error pairs during training are shown in Figure 4.

Experiment 2. In this experiment, the control policy is trained based on the DDPG algorithm and the PER-DDPG algorithm.

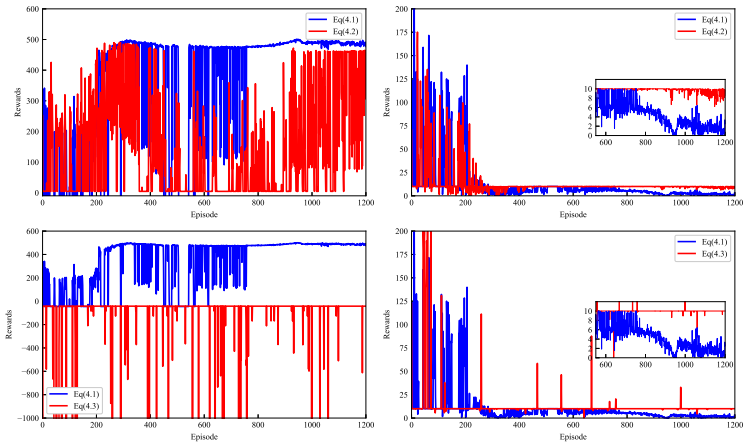

Figure 4. Comparison of accumulated reward and steady-state error of different reward functions

The training evaluation of the task is measured by two metrics, one is the accumulated reward, and the other is the steady-state error. In each time step, a larger accumulated reward represents a smaller error to the desired state, which means a more accurate control policy. Taking the DDPG algorithm as the baseline, we compare the PER-DDPG algorithm with it. The accumulated reward and steady-state error of each episode during the training process are shown in Figs. 4-5:

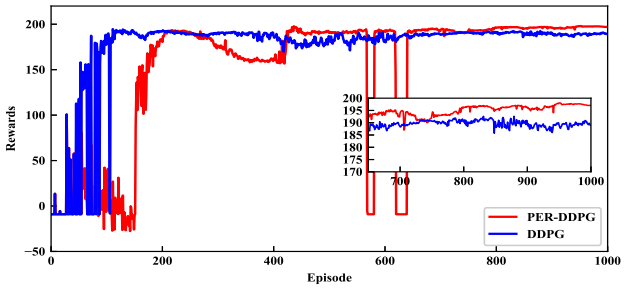

Figure 5. Accumulated reward comparison between DDPG and PER-DDPG.

Remark 4.2. Around 600 episodes, the reward of the PER-DDPG algorithm dropped twice. Our conjecture is that the prioritized experience replay technique is to replay important transitions, which reduces the diversity of transitions to a certain extent and may cause overfitting. However, the accumulated reward of the PER-DDPG algorithm remains at a higher level than the DDPG algorithm after 700 episodes.

As shown in Figs. 4-5, in the initial stage of training, two algorithms do not converge, therefore the reward is small and fluctuates greatly. In the later stage of training, especially after 700 episodes, the reward remains at a high level. In addition, the reward of the PER-DDPG algorithm is higher than the DDPG algorithm. This difference is particularly obvious in terms of steady-state errors. To

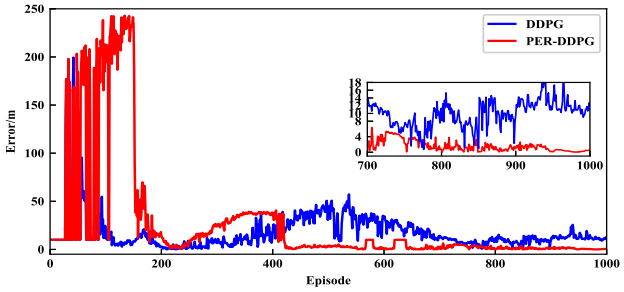

Figure 6. Steady-state error comparison between DDPG and PER-DDPG.

further highlight the convergence of the results, this paper displays the descriptive statistics of the accumulative reward and steady-state error of the last 100 episodes, as is shown in Table 3: From the descriptive statistics of accumulated

Table 3. Descriptive statistics of reward and error

	method	mean	std	min
reward	DDPG	188.8848	0.9870	185.9812
reward	PER-DDPG	196.8483	0.9141	194.0524
error/m	DDPG	12.6755	2.6963	9.0077
error/m	PER-DDPG	0.8037	0.6690	0.0010

rewards and steady-state errors in the last 100 episodes, the PER-DDPG algorithm has higher accumulated rewards, lower steady-state errors, and lower volatility than the DDPG algorithm. As a result, our simulation is conducted under the framework of the PER-DDPG algorithm.

Experiment 3. In this experiment, the control policy is trained based on the PER-DDPG algorithm and the PER-DDPG with GIC reward algorithm. Under the framework of the PER-DDPG algorithm, this paper introduces a generalized integral compensator, which is reflected in the error calculation of the reward function. The integral compensator proposed in [28] and [29] treats errors in the past state equally. In fact, the closer the error is to the current moment, the more valuable it is. Based on these ideas, this experiment gives weight to the state error in the past moment according to the distance to the current moment, and forms a generalized integral compensator. As a result, a PER-DDPG with GIC reward control policy is presented. By utilizing the GIC mechanism, the novel reward function is defined as:

$$reward = \begin{cases} 1 - 0.01\sqrt{x_g^2 + y_g^2 + z_g^2}, m = 0, \\ 1 - 10 - 0.01\sqrt{x_g^2 + y_g^2 + z_g^2}, m = 1, \end{cases} \tag{4.4}$$

where x_g, y_g, z_g are the errors with generalized integral compensator respectively by (3.13), m is the symbol that judges whether the quadrotor is going beyond the safe range, which is indicated by 0 and 1. The accumulated reward and steady-state error of each episode during the training process are shown in Figs. 6-7:

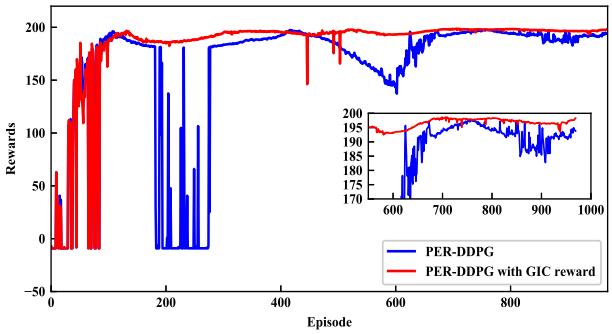

Figure 7. Accumulated reward comparison between PER-DDPG and PER-DDPG with GIC reward.

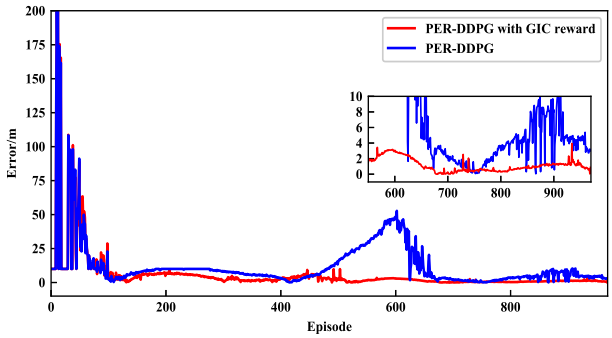

Figure 8. Steady-state error comparison between PER-DDPG and PER-DDPG with GIC reward.

As shown in Figs. 6-7, in the initial stage of training, PER-DDPG and PER-DDPG with GIC reward algorithms cannot converge owing to the fluctuating reward, and large steady-state error. The reward of the PER-DDPG algorithm gradually increases after 300 episodes, while the PER-DDPG with GIC reward control policy maintains at a higher level after 100 episodes, and the accumulated reward is larger than the PER-DDPG policy, indicating that the proposed PER-DDPG with GIC reward control policy has faster convergence and better training performance. Although both policies are stable at the end of the training, the PER-DDPG with GIC reward algorithm is still better. Using the PER-DDPG with GIC reward control policy for quadrotor control can keep the steady-state error at a very low level, which further illustrates the effectiveness of the algorithm. To further highlight the convergence of the results, this paper displays the descriptive statistics of the accumulated reward and steady-state error of the last 100 episodes, as shown in Table 4:

Remark 4.3. In the last 100 episodes, the volatility of PER-DDPG is higher than that of the PER-DDPG with GIC reward in terms of accumulated reward and steady-state error. For the accumulated reward and steady-state error, the standard deviations of PER-DDPG are 2.8340 and 2.3937 respectively, while the standard deviations of PER-DDPG with GIC reward are 0.6402 and 0.3920 respectively. It

Table 4. Descriptive statistics of reward and error

	method	mean	std	min
reward	PER-DDPG	190.4356	2.8340	182.7491
reward	PER-DDPG with GIC reward	196.6515	0.6402	193.5726
error/m	PER-DDPG	5.5910	2.3937	0.3204
error/m	PER-DDPG with GIC reward	1.2126	0.3920	0.0368

indicates that an effective reward function can better guide the behavior of the agent, and the reward function of the PER-DDPG with GIC reward algorithm is more effective.

In the last 100 episodes of training, the PER-DDPG with GIC reward control policy is superior to the PER-DDPG policy in terms of mean, standard deviation, and minimum value, indicating that the proposed policy has higher rewards and smaller fluctuations. In addition, the steady-state error is smaller and the volatility is smaller than the PER-DDPG algorithm as well.

The PER-DDPG algorithm and PER-DDPG with GIC reward algorithm are tested in the desired height control task, the initial position and desired position are set to $[0, 0, 0]$ and $[0, 0, 10]$ respectively. The simulation results show that the PER-DDPG with GIC reward control policy can drive the quadrotor to the specified height and hover with small steady-state errors. The simulation result of the two control policies is shown in Fig. 8:

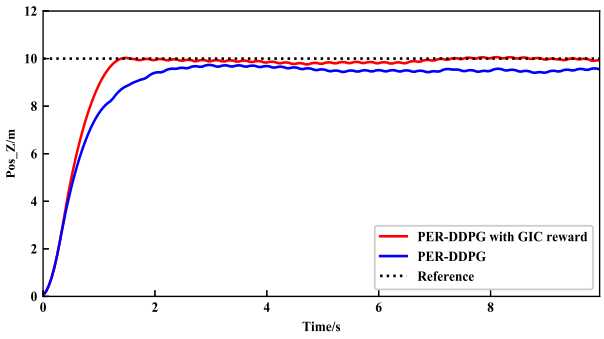

Figure 9. Height control performance of PER-DDPG and PER-DDPG with GIC reward.

To verify the effectiveness of the control policy, four different desired height control tasks are additionally set up in this paper. The rest of the mission settings remain unchanged, the desired heights are $20m$, $30m$, $40m$, and $50m$. The simulation results show that the aircraft can quickly reach the different specified altitudes and remain to hover. The response curve of the height channel is shown in Fig. 9: During the quadrotor training, the initial position is set as a fixed point. In order to show the effectiveness of the algorithm, now randomly select the initial state. We conduct three groups of comparative experiments. Firstly, randomly select any initial height from $0m$ to $20m$, and the tracking height is $10m$. Secondly, randomly select any initial height between $0m$ and $50m$, and the tracking height

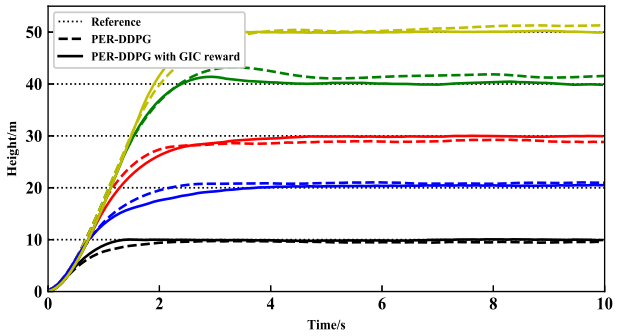

Figure 10. Response curves at different desired heights.

is $25m$. Lastly, randomly select any initial height between $0m$ and $80m$, and the tracking height is $40m$. The simulation results show that the quadrotor can adapt to various initial states. The response curves of different tracking heights are shown in Figs. 10-12: To quantify the performances of PER-DDPG and PER-DDPG

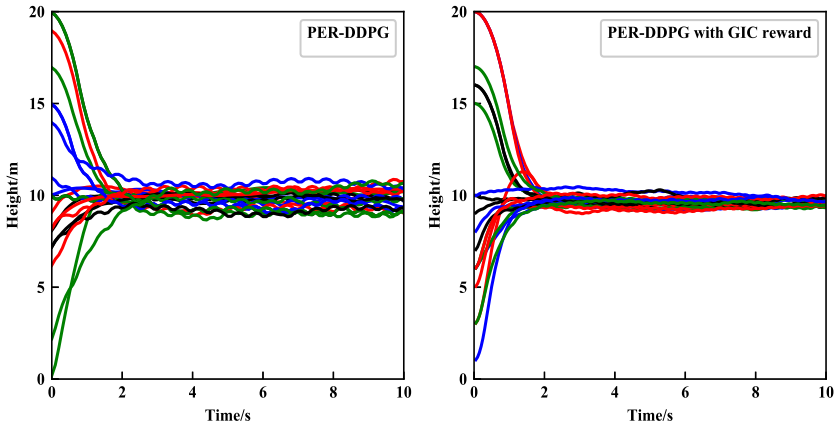

Figure 11. Tracking height $10m$.

with GIC reward control policies, this paper calculates the sum of the steady-state errors of the 20 episodes. The comparison chart of the sum of steady-state errors of different tracking heights is shown in Fig. 13: In addition, the steady-state error is not enough to measure the intermediate process of the two control policies. Based on the above considerations, to compare the two control policies in-depth, it can be seen from the above pictures that the response height of each episode reached a relatively stable stage after $4s$. Therefore, except for the steady-state error, this paper calculates the error between the current height and the desired height at each time step between $4s$ and $10s$. The sum of errors between $4s$ and $10s$ at different tracking heights is shown in Fig. 14:

Experiment 4. In this experiment, the control policy is trained based on the PER-DDPG algorithm and the PER-DDPG-GIC algorithm. In Experiment 2, the

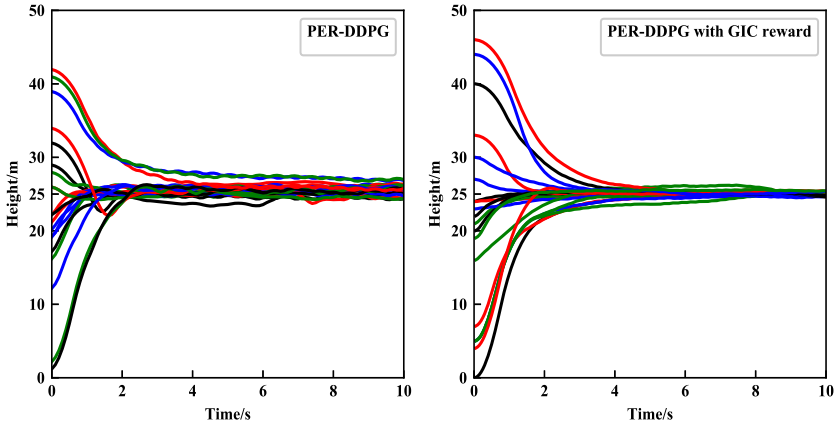

Figure 12. Tracking height $25m$.

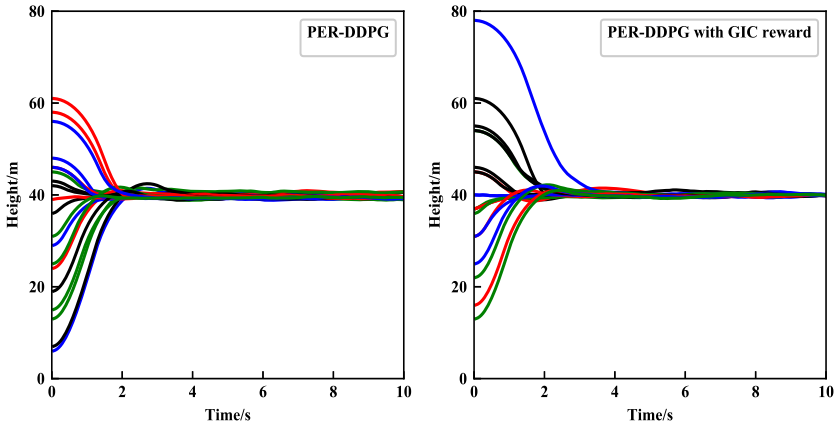

Figure 13. Tracking height $40m$.

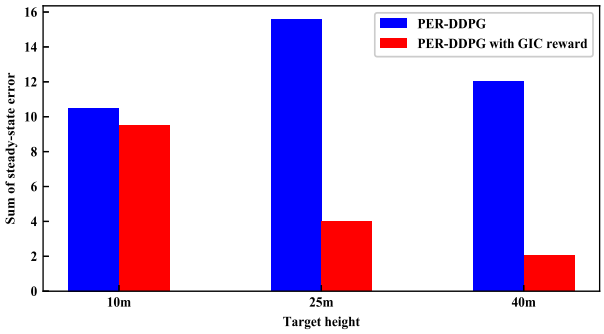

Figure 14. Error comparison.

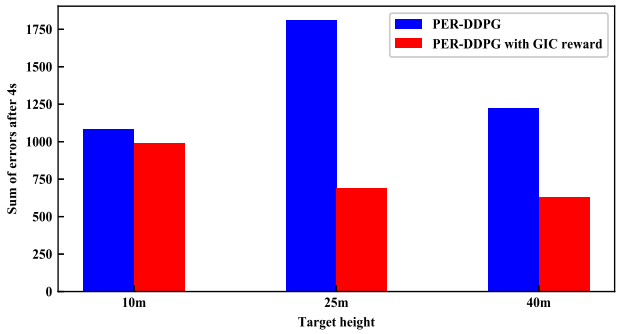

Figure 15. Total error comparison between 4s-10s.

generalized integral compensator is introduced into the reward function of PER-DDPG algorithm. Furthermore, this experiment draws lessons from the integral term of proportion-integration-differentiation (PID) controller. We introduce the integral term into PER-DDPG controller, and add the generalized integral compensated error state into the input vector of the neural network. The integral term is the integral of error with respect to time. With the increase of time, the integral term will increase. Therefore, when the generalized integral compensated error is input into the system through the neural network, the output of the controller will increase and the steady-state error will decrease until the steady-state error is eliminated. In the input vector, adding the generalized integral compensated error makes the controller have the function of integral adjustment. As long as the error exists, the integral adjustment will work until it is eliminated.

In the PER-DDPG-GIC policy designed in this paper, the control input is increased from the original 12-dimensional control vector s_t=$[x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ to 15-dimensional control vector $s_t^{'} = [x, y, z, \phi, \theta, \psi, x_g, y_g, z_g, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$, where x_g, y_g, z_g are the generalized integral compensated error defined by (3.13). Therefore, the neural network structure of PER-DDPG-GIC control strategy proposed in this paper is shown in Fig. 15:

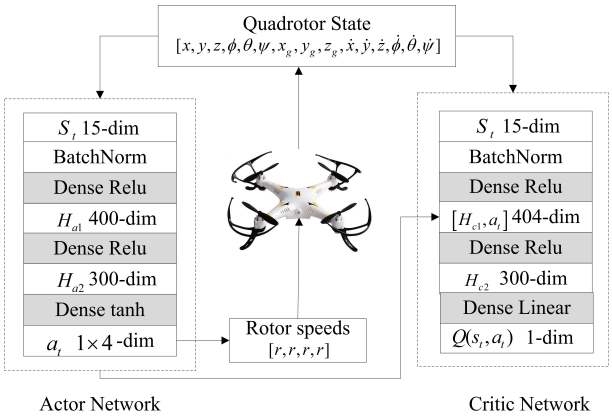

Figure 16. Actor-critic network of PER-DDPG-GIC for quadrotor control.

In order to show the superiority of PER-DDPG-GIC algorithm, the PER-DDPG is compared with PER-DDPG-GIC. Similarly, the control effect is measured by accumulated reward and steady-state error. A higher accumulated reward means a lower steady-state error, which is equivalent to a smaller difference between the current altitude and the desired. The comparison chart of accumulated reward and steady-state error of the two control strategies is shown in Figs.16-17:

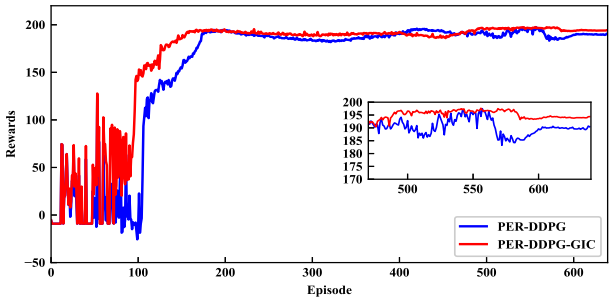

Figure 17. Accumulated reward comparison between PER-DDPG and PER-DDPG-GIC.

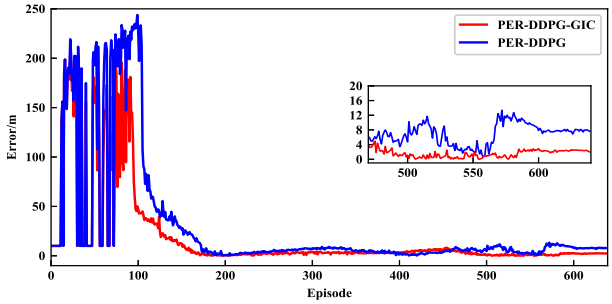

Figure 18. Steady-state error comparison between PER-DDPG and PER-DDPG-GIC.

It can be seen from the figure that the convergence speed of the PER-DDPG-GIC control strategy is faster and reaches a higher level faster than the PER-DDPG algorithm. At the same time, the reward value is slightly higher than PER-DDPG control strategy. Although, PER-DDPG-GIC has a slight decline during the 400-450 episodes. However, after 450 episode, it is always higher than PER-DDPG control strategy, which shows the superiority of PER-DDPG-GIC control strategy proposed in this paper. Similarly, from the perspective of the steady-state error, the error of PER-DDPG-GIC control strategy proposed in this paper is lower than that of the PER-DDPG control strategy. This shows that the generalized integral compensator has an obvious effect, which can effectively improve the reward and reduce the steady-state error, and improve the training accuracy of the control strategy. In order to further verify the effectiveness of PER-DDPG-GIC control strategy from the numerical aspect, the descriptive statistical comparison of reward value and steady-state error in the last 100 episodes are carried out, as shown in Table 5:

Table 5. Descriptive statistics of reward and error

	method	mean	std	min
reward	PER-DDPG	190.0747	3.5492	183.2128
reward	PER-DDPG-GIC	195.1881	1.4194	193.1947
error/m	PER-DDPG	7.3359	3.1851	0.6736
error/m	PER-DDPG-GIC	1.6603	0.8562	0.0234

From the descriptive statistics of the two control strategies, the PER-DDPG-GIC control strategy has more obvious advantages. In the last 100 episodes, the average reward value of the PER-DDPG-GIC control strategy is higher, and the standard deviation is lower, which indicates that the reward value of the PER-DDPG-GIC control strategy has been maintained at a higher level than that of the PER-DDPG control strategy, and the volatility is smaller. Similarly, from the perspective of steady-state error, the steady-state error of the PER-DDPG-GIC control strategy has been kept at a low level, and the fluctuation is also lower than that of the PER-DDPG control strategy. These two indicators, both show that the adjustment effect of the generalized integral compensator is effective, which can significantly improve the accumulative reward and reduce the steady-state error.

To verify the effectiveness of PER-DDPG-GIC control policy, we use this strategy to carry out the same simulation experiment as experiment 2.

1) Four different desired altitude control tasks are additionally set up in this paper, 20m, 30m, 40m, 50m respectively.

2) Randomly select the initial state and conduct three groups of comparative experiments.Firstly, randomly select any initial height from 0m to 20m, and the tracking height is 10m. Secondly, randomly select any initial height between 0m and 50m, and the tracking height is 25m. Lastly, randomly select any initial height between 0m and 80m, and the tracking height is 40m.

The comparison of experimental results is shown in Figs. 18-19: Similar to ex-

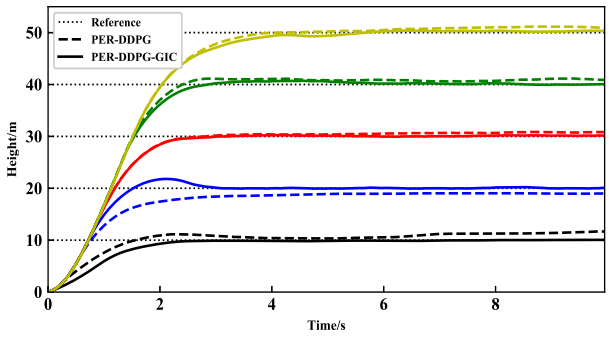

Figure 19. Steady-state error comparison between PER-DDPG and PER-DDPG-GIC.

periment 2, to quantify the accuracy of the two control policies, steady-state error, and sum of errors in each time step between $4s$ and $10s$ are calculated. The error comparisons are shown in Fig. 20:

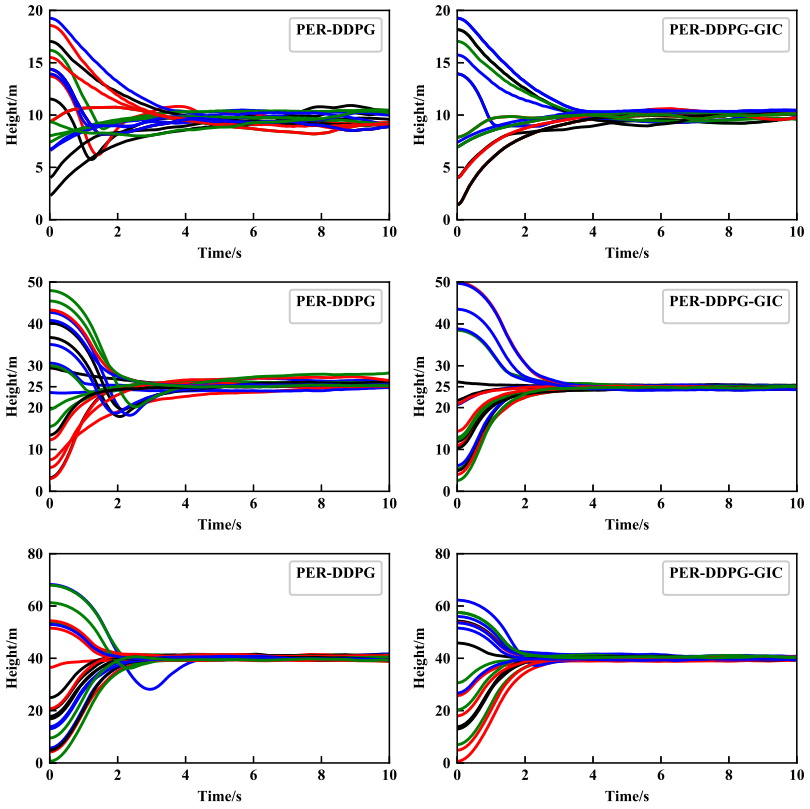

Figure 20. Steady-state error comparison between PER-DDPG and PER-DDPG-GIC.

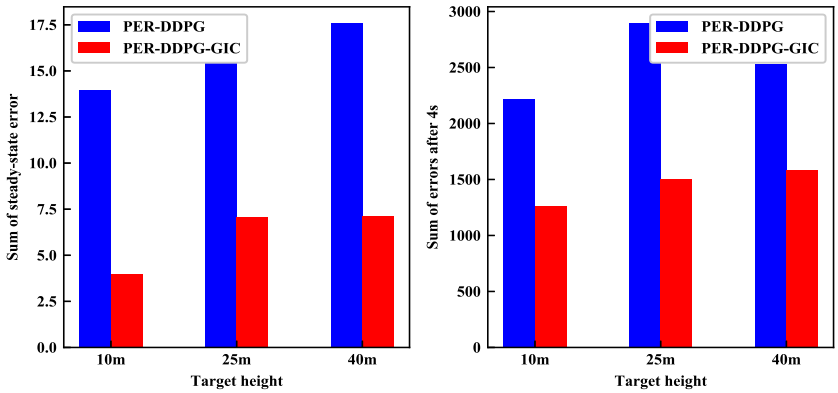

Figure 21. Steady-state error comparison between PER-DDPG and PER-DDPG-GIC.

4.3. Generalization ability of the control policy

The desired height control of the quadrotor is ultimately to apply the control policy to the actual flight. Therefore, the generalization ability and robustness of the

policy are particularly important. This section mainly studies the generalization ability and robustness of the control policy on quadrotor models of different weights and sizes. The policy is tested on a desired height control task: starting from the position of $[0, 0, 0]$, then fly to the position of $[0, 0, 10]$ and hovering.

The impact of weight on the quadrotor is particularly significant because, in actual flight, the quadrotor is often loaded with payloads, such as adding cameras and so on. This paper explores the generalization ability of the control policy for weight by gradually adding payloads to the quadrotor. The weights of payloads vary from 5% to 15% of the weight of the quadrotor. The simulation results of a quadrotor with different payloads are shown in Fig. 21:

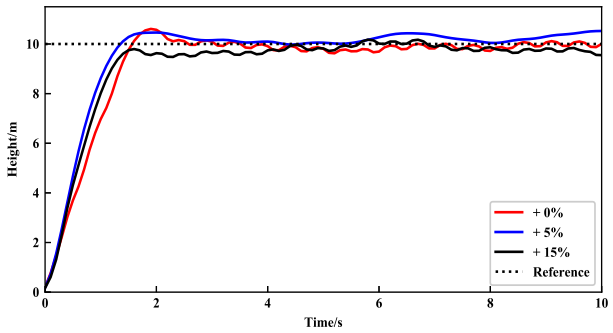

Figure 22. Generalization capability test of quadrotor with different payloads.

The generalization ability and robustness tests are performed on different sizes of the quadrotor. The radius of the quadrotor to be tested starts from $0.15m$ (25% smaller) to $0.25m$ (25% larger). The change of radius affects the moments of inertia which can be obtained from the website https://www.flyeval.com. The desired height control response curves are shown in Fig. 22: The above results demonstrate

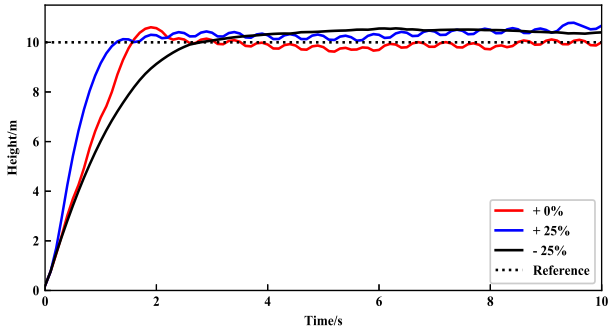

Figure 23. Generalization capability test of quadrotor with different sizes.

that the response curves of the altitude channel change little with different payloads and sizes of the quadrotor. Besides, the control policy can always complete the desired altitude control task of the quadrotor. Through the generalization ability

and robustness test of different weights and sizes, it indicates that the PER-DDPG-GIC algorithm proposed in this paper has robustness for quadrotor control.

4.4. Visualization platform

It is difficult to visually display the flight process of the quadrotor simulation only by state data $[x, y, z, \phi, \theta, \psi]$, so this paper designs a visual simulation platform. The platform is developed with python and uses a simple three-dimensional dynamic drawing toolkit to display the motion state of each time step. It can clearly show the flight trajectory and attitude changes of the quadcopter. Each time step is 0.05 seconds, there are 200-time steps in each episode, and the reward of each time step is displayed below. The task is to fly from the initial position $[0, 0, 0]$ to the desired position $[0, 0, 10]$ and hover. This paper intercepts the last frame of the gif animation. The gif animation of the whole episode during the simulation is uploaded as an attachment. The visual interface of the quadrotor is shown in Fig. 23:

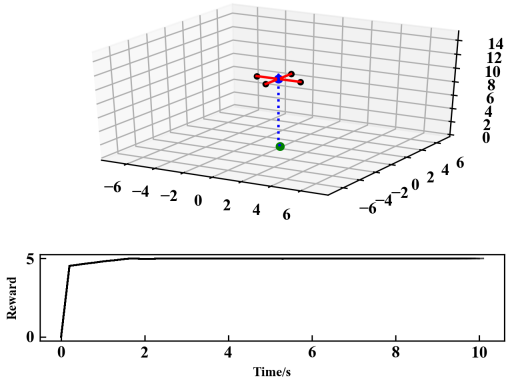

Figure 24. Quadrotor visual interface.

In Fig. 23, the green dot represents the initial position, the blue dot represents the desired position, and the blue dotted line is the flight trajectory. The reward represents a single-step reward.

5. Conclusion

In this paper, we have investigated the desired height control of the quadrotor under the framework of the PER-DDPG algorithm. Firstly, some novel reward functions that embody out-of-bounds plenty are designed by virtual of the reward shaping technology, which ensures the successful application of the PER-DDPG algorithm. Secondly, the performance of the PER-DDPG algorithm has been improved by further incorporating the generalized integral compensator into the reward function. Thirdly, the PER-DDPG-GIC algorithm has been proposed by taking the generalized integral compensated error as part of the input to a neural network. Compared with the PER-DDPG algorithm, the proposed algorithms have advantages in increasing reward and decreasing steady-state error. Finally, the results of

the experiment have demonstrated the generalization ability and robustness of the proposed algorithm in terms of weights and sizes of the quadrotor. Further work will focus on applying generalized integral compensator to both reward function and actor-critic structure.

Acknowledgment

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IFPIP-59-611-1442), the National Science Foundation of China (Grant No. 61773152) and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

References

[1] G. M. Barros and E. L. Colombini, *Using soft actor-critic for low-level uav control*, 2020.

[2] L. Cao, X. Hu, S. Zhang and Y. Liu, *Robust flight control design using sensor-based backstepping control for unmanned aerial vehicles*, Journal of Aerospace Engineering, 2017, 30(6), 04017068.

[3] X. Cao, H. Wan, Y. Lin and S. Han, *High-value prioritized experience replay for off-policy reinforcement learning*, in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019.

[4] A. Das, K. Subbarao and F. Lewis, *Dynamic inversion with zero-dynamics stabilisation for quadrotor control*, IET Control Theory & Applications, 2009, 3(3), 303–314.

[5] T. De Bruin, J. Kober, K. Tuyls and R. Babuska, *Experience selection in deep reinforcement learning for control*, 2018.

[6] T. Dierks and S. Jagannathan, *Output feedback control of a quadrotor uav using neural networks.*, IEEE Transactions on Neural Networks, 2009, 21(1), 50–66.

[7] A. R. Dooraki and D. J. Lee, *An innovative bio-inspired flight controller for quad-rotor drones: Quad-rotor drone learning to fly using reinforcement learning*, Robotics and Autonomous Systems, 2021, 135, 103671.

[8] N. T. Duc, Q. Hai, D. N. Van et al., *An approach for UAV indoor obstacle avoidance based on AI technique with ensemble of ResNet8 and Res-DQN*, in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, 2019.

[9] J. Ghommam, M. Saad, S. Wright and M. Quan, *Relay manoeuvre based fixed-time synchronized tracking control for UAV transport system*, Aerospace Science and Technology, 2020, 103, 105887.

[10] U. H. Ghouri, M. U. Zafar, S. Bari et al., *Attitude control of quad-copter using deterministic policy gradient algorithms (DPGA)*, in *2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE)*, IEEE, 2019, 149–153.

[11] J. Han, *From PID to active disturbance rejection control*, IEEE Transactions on Industrial Electronics, 2009, 56(3), 900–906.

[12] H. Hasselt, *Double q-learning*, Advances in neural information processing systems, 2010, 23, 2613–2621.

[13] Y. Hou, L. Liu, Q. Wei et al., *A novel ddpg method with prioritized experience replay*, in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, IEEE, 2017, 316–321.

[14] H. Hu and Q. Wang, *Proximal policy optimization with an integral compensator for quadrotor control*, Frontiers of Information Technology & Electronic Engineering, 2020, 21, 777–795.

[15] Y. Jiang, Z. Mi and H. Wang, *An improved OLSR protocol based on task driven used for military UAV swarm network*, Intelligent Robotics and Applications, 2019.

[16] S. Kapturowski, G. Ostrovski, J. Quan et al., *Recurrent experience replay in distributed reinforcement learning*, in *International conference on learning representations*, 2018.

[17] W. Koch, R. Mancuso, R. West and A. Bestavros, *Reinforcement learning for UAV attitude control*, ACM Transactions on Cyber-Physical Systems, 2019, 3(2), 1–21.

[18] L. Liu, B. Tian, X. Zhao and Q. Zong, *UAV autonomous trajectory planning in target tracking tasks via a dqn approach*, in *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2019.

[19] A. Y. Ng, D. Harada and S. Russell, *Policy invariance under reward transformations: Theory and application to reward shaping*, in *Icml*, 99, 1999, 278–287.

[20] C. Peng, Y. Bai, X. Gong et al., *Modeling and robust backstepping sliding mode control with Adaptive RBFNN for a novel coaxial eight-rotor UAV*, IEEE/CAA Journal of Automatica Sinica, 2015, 2(1), 56–64.

[21] S. Santos, C. L. Nascimento and S. N. Givigi, *Design of attitude and path tracking controllers for quad-rotor robots using reinforcement learning*, in *Aerospace Conference*, 2012.

[22] T. Schaul, J. Quan, I. Antonoglou and D. Silver, *Prioritized experience replay*, arXiv preprint arXiv:1511.05952, 2015.

[23] M. Z. Shah, R. Samar and A. I. Bhatti, *Guidance of air vehicles: A sliding mode approach*, IEEE Transactions on Control Systems Technology, 2015, 23(1), 231–244.

[24] F. Shang, H. Chou, S. Liu and X. Wang, *A framework of power pylon detection for UAV-based power line inspection*, in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, IEEE, 2020, 350–357.

[25] D. Silver, G. Lever, N. Heess et al., *Deterministic policy gradient algorithms*, in *International conference on machine learning*, PMLR, 2014, 387–395.

[26] A. K. Tiwari and S. V. Nadimpalli, *Augmented random search for quadcopter control: An alternative to reinforcement learning*, in *International Journal of Information Technology and Computer Science(IJITCS)*, 2019.

[27] H. Van Hasselt, A. Guez and D. Silver, *Deep reinforcement learning with double q-learning*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 30, 2016.

[28] Y. Wang, J. Sun, H. He and C. Sun, *Deterministic policy gradient with integral compensator for robust quadrotor control*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 50(10), 3713–3725.

[29] Z. Wang, V. Bapst, N. Heess et al., *Sample efficient actor-critic with experience replay*, arXiv preprint arXiv:1611.01224, 2016.

[30] S. L. Waslander, G. M. Hoffmann, J. Jang and C. J. Tomlin, *Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning*, in *IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2005.

[31] H. Yin and S. Pan, *Knowledge transfer for deep reinforcement learning with hierarchical experience replay*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 31, 2017.

[32] B. Zhao, B. Xian, Y. Zhang and X. Zhang, *Nonlinear robust adaptive tracking control of a quadrotor UAV via immersion and invariance methodology*, IEEE Transactions on Industrial Electronics, 2015, 62(5), 2891–2902.

[33] J. Zhao, Y. Li, D. Hu and Z. Pei, *Design on altitude control system of quad rotor based on laser radar*, in *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, IEEE, 2016, 105–109.

[34] N. Zhen, N. Malla and X. Zhong, *Prioritizing useful experience replay for heuristic dynamic programming-based learning systems*, IEEE Transactions on Cybernetics, 2018, 49(11), 3911–3922.