Hindawi
International Journal of Aerospace Engineering
Volume 2019, Article ID 7854173, 12 pages
https://doi.org/10.1155/2019/7854173

Research Article

Three-Dimensional Path-Following Control of a Robotic Airship with Reinforcement Learning

Chunyu Nie,[1] **Zewei Zheng**,[2,3] **and Ming Zhu**[1]

[1]*School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China*
[2]*School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*
[3]*The Seventh Research Division, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Zewei Zheng; zeweizheng@buaa.edu.cn

Received 25 November 2018; Accepted 20 February 2019; Published 25 March 2019

Academic Editor: Zhiguang Song

This paper proposed an adaptive three-dimensional (3D) path-following control design for a robotic airship based on reinforcement learning. The airship 3D path-following control is decomposed into the altitude control and the planar path-following control, and the Markov decision process (MDP) models of the control problems are established, in which the scale of the state space is reduced by parameter simplification and coordinate transformation. To ensure the control adaptability without dependence on an accurate airship dynamic model, a Q-Learning algorithm is directly adopted for learning the action policy of actuator commands, and the controller is trained online based on actual motion. A cerebellar model articulation controller (CMAC) neural network is employed for experience generalization to accelerate the training process. Simulation results demonstrate that the proposed controllers can achieve comparable performance to the well-tuned proportion integral differential (PID) controllers and have a more intelligent decision-making ability.

1. Introduction

The lift of an airship mainly comes from the buoyancy of the gas, and therefore, it does not have to do continuous movement to balance gravity, which makes it a promising platform for long-endurance and low-energy consumption. In the last two decades, airships such as heavy-duty airships and stratospheric airships [1] have gained increasing interest given their potential in communication relay, space observation, and military reconnaissance. Control strategy has always been an important subject in airship research. After years of study, airship motion control has gained significant achievements with regard to hovering control, trajectory tracking, and path-following control [2–4]. The objective of the hovering control is to maintain the airship in a certain area against the wind, and the primary purpose of the trajectory tracking is to track a time-parameterized desired path, while the path-following control is mainly concerned with tracking a desired path without a specified temporal constraint.

The control response of an airship is slow with a long time delay. The flexibility and elasticity of the hull and the stochastic atmosphere disturbance make the control of the airship a problem with uncertainties. At present, common airship control approaches include PID control, backstepping, dynamic inversion, sliding mode control, and robust control [5–8]. The PID controller is widely used for its simplicity and effectiveness, but its parameters such as proportion, integral, and differential should be tuned properly [9]. When the model of the system or environment changes, the performance of the PID controller may get worse before being tuned again. The modern model-based nonlinear control approaches can ensure the control robustness and global stability while being accurately controlled; however, these improvements incur the work of system modeling and parameter identification with ever increasing complexity [10, 11]. With the rapid development of information technology, machine learning algorithms have been successfully applied in various complex applications [12] and can make intelligent decisions similar to human beings. As an important unsupervised learning algorithm, reinforcement learning is able to realize adaptive control and decision in an unknown environment through the mechanism of "action

and reward" [13, 14], which does not need an accurate model that costs human effort.

Up to now, reinforcement learning has been extensively studied in robot control, traffic dispatch, communication control, and game decision-making [15–18]. In the field of aircraft navigation and control, reinforcement learning also witnessed many successful applications. However, due to the complexity of aircraft motion and the large scale of the state space, the reinforcement learning control can easily be trapped in the "curse of dimensionality," which confines the applications to outer-level decisions rather than inner actuator commands. In particular, Pearre and Brown [19] and Dunn et al. [20] took the flight path angle as the action of the reinforcement learning algorithm and constructed the reward function using energy consumption, time spent, and collision loss to flexibly plan and optimize the reference flight path. To further optimize the flight path, Zhang et al. [21] proposed a "geometric reinforcement learning" algorithm to provide more available actions by transferring between nonadjacent states. Palunko et al. [22] and Faust [23] introduce a reinforcement learning algorithm to the adaptive acceleration control of a quadrotor with suspended load. The primary purpose of the above studies is to generate the outer loop guidance law; however, the inner loop controllers still need to be designed. In [24], a reinforcement learning approach for the airship model parameter identification is suggested by Ko et al., which can reduce the control error caused by the model uncertainties. In a previous work in [25], Rottmann et al. designed an airship altitude controller based on reinforcement learning and completed the online training process in a few minutes. However, the controller training of [24, 25] is performed in the restricted low-dimensional state space, and therefore, the approach is difficult to be directly employed in the multidimensional motion control. Hwangbo et al. [26] proposed a neural network quadrotor control strategy based on a reinforcement learning algorithm, and the well-trained controller can achieve autonomous flight control even if the quadrotor is thrown upside down into the air, which is superior to human operators. But in [26], the quadrotor's model was used for training the neural network controller off-line to avoid the "curse of dimensionality" problem, which brought in the risk of control failure when the dynamic model is inaccurate or unavailable.

For the purpose of acquiring a feasible airship control strategy without dependence on an accurate dynamic model, this paper directly takes the inner actuator commands as the action of the reinforcement learning algorithm and implements the training by actual motion. The main contributions of this paper are as follows:

(1) The MDP models of the airship 3D path-following control are established based on motion analysis, which provide new models and methods for airship control. Furthermore, in contrast to the previous reinforcement learning airship control without dependence on an accurate dynamic model, for the first time, this study proposed an airship control strategy through autonomous online training in 3D space

(2) To deal with the "curse of dimensionality" problem in the airship planar path-following control, a novel coordinate frame is proposed to describe the relative state between the airship and the target. This coordinate transformation form can reduce the scale of the state space and generalize the experience by rotation and makes it possible to learn the control strategy online

(3) In the proposed reinforcement learning airship control strategy, a CMAC neural network is employed to generalize the experience in a local neighborhood, which effectively accelerates the training process

The rest of the paper is organized as follows. Section 2 presents the problem formulation and establishes the MDP models of the airship control. Section 3 introduces the structure, the principles, and the training process of the proposed airship reinforcement learning control strategy. Section 4 presents the simulation results, and Section 5 concludes the paper.

2. Problem Formulation and the MDP Model of the Airship Control

2.1. Problem Formulation. After takeoff and ascent, the airship stays steadily near its design altitude. Thereafter, the altitude of the airship is mainly controlled by an elevator, and the horizontal motion is mainly controlled by a propeller, rudder, and vector propeller. Compared with the fixed-wing aircraft, the airship has a smaller roll angle, and the rolling motion has little effect on the horizontal motion; hence, the coupling of the airship longitudinal motion and lateral motion is weak [27]. To simplify the control problem, we decompose the airship 3D path-following control into the altitude control and the planar path-following control.

The autonomous training process of reinforcement learning is similar to the animal learning process, in which an intuitive objective will contribute to the acceleration of learning. In this paper, the airship completes the path-following mission by reaching the target points on the desired path one by one. $O_g x_g y_g z_g$ represents the earth reference frame (ERF), where (x_g, y_g, z_g) denotes the position of the airship. Let h denote the altitude of the airship which satisfies $h = -z_g$; then, the position of the airship can be described as (x_g, y_g, h).

In the airship altitude control, suppose the desired altitude is h_{at} and the control objective is given by

$$|h - h_{at}| < \varepsilon_h, \tag{1}$$

where ε_h is defined as the valid range of the desired altitude.

In the airship planar path-following control, let (x_t, y_t) denote the position of the target point, and the control objective can be described as

$$\sqrt{(x_g - x_t)^2 + \left(y_g - y_t\right)^2} < \varepsilon_r, \tag{2}$$

where ε_r indicates the valid area of the target point.

Remark 1. During the training process, the desired altitude and the target point of the two controllers can be set randomly and independently. In the 3D path-following control, we choose the planar path-following control as the main control mission and the altitude control as the auxiliary. To ensure that the actual path is close to the desired 3D path, the horizontal position of the current target is taken as the target point in the planar path-following control and the altitude of the nearest point on the desired path to the airship is taken as the desired altitude in the altitude control. Let (x_1, y_1, h_1) denote the current target, (x_0, y_0, h_0) denote the last reached target, and (x_g, y_g, h) denote the position of the airship. Then, the target point (x_t, y_t) in the planar path-following control can be set as (x_1, y_1) and the desired altitude h_{at} in the altitude control can be calculated as

$$h_{at} = \frac{\mathbf{H}_1 \cdot \mathbf{H}_g}{|\mathbf{H}_1|^2} \cdot (h_1 - h_0) + h_0, \tag{3}$$

where $\mathbf{H}_1$ and $\mathbf{H}_g$ are given by

$$\begin{aligned} \mathbf{H}_1 &= (x_1 - x_0, y_1 - y_0, h_1 - h_0), \\ \mathbf{H}_g &= \left(x_g - x_0, y_g - y_0, h - h_0\right). \end{aligned} \tag{4}$$

2.2. MDP Model of the Airship Control. MDP is the basis for the implementation of reinforcement learning; thus, we establish the MDP models of the airship altitude control and airship planar path-following control first. MDP can be expressed by five elements $\{\mathbf{S}, \mathbf{A}, r, \mathbf{P}, J\}$, where $\mathbf{S}$ indicates the state space constructed by parameters such as the position, speed, and attitude of the airship; $\mathbf{A}$ is the action set composed of available control actuator commands, r is the reward of the state and action, $\mathbf{P}$ is the state transition probability, and J is the optimization objective function of the sequential decision. MDP satisfies the following property:

$$\begin{aligned} &p\left(s_{t+1} = s_j | s_t = s_i, a_t = a_k, s_{t-1}, a_{t-1}, \cdots, s_0, a_0\right) \\ &\quad = p\left(s_{t+1} = s_j | s_t = s_i, a_t = a_k\right) = p_{ij}(a_k) \\ &\qquad\qquad \forall s_i, s_j \in \mathbf{S}, a_k \in \mathbf{A}, \forall t \geq 0, \end{aligned} \tag{5}$$

where $p_{ij}(a_k)$ is defined as the transition probability from state s_i to state s_j after taking the action a_k, and t is the time parameter. It can be seen from the property (5) that parameters of the state space $\mathbf{S}$ are sufficient statistics of the airship motion. The primary work of airship control MDP modelling is to construct the state space. By reasonably selecting parameters to construct state space, the property (5) can be approximately satisfied.

2.2.1. State Space $\mathbf{S}$ *of the Altitude Control.* As shown in Figure 1, let $O_g x_g y_g z_g$ denote the earth reference frame (ERF) and $O_b x_b y_b z_b$ denote the body reference frame (BRF) attached to the airship. The attitude of the airship is described as the Euler angles $[\phi, \theta, \psi]^T$ in EBF. The speed and angular velocity of the airship in BRF are defined as

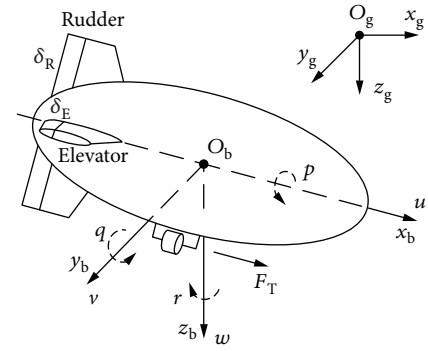


Figure 1: Coordinate frame of the airship.

$[u, v, w]^T$ and $[p, q, r]^T$, respectively. δ_R, δ_E, and F_T represent the rudder deflection, the elevator deflection and the propeller thrust of the airship, respectively.

The airship resultant velocity is defined as V in ERF, which satisfies the following equation:

$$V = \sqrt{u^2 + v^2 + w^2}. \tag{6}$$

According to the kinematics of the airship, the speed in ERF $[\dot{x}_g, \dot{y}_g, \dot{z}_g]^T$ can be calculated as

$$\left[\dot{x_g}, \dot{y_g}, \dot{z_g}\right]^T = \mathbf{K}[u, v, w]^T. \tag{7}$$

$\mathbf{K}$ is the rotation matrix from BRF to ERF, which is given by

$$\boldsymbol{K} = \begin{bmatrix} c\theta c\psi & s\theta c\psi s\phi - s\psi c\phi & s\theta c\psi c\phi + s\psi s\phi \\ c\theta c\psi & s\theta s\psi s\phi + c\psi c\phi & s\theta s\psi c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{8}$$

where sx is $\sin x$ and cx is $\cos x$.

According to (7) and $h = -z_g$, the altitude-changing rate $\dot{h}$ can be calculated as

$$\dot{h} = -\dot{z_g} = \sin\theta \cdot u - \cos\theta \sin\phi \cdot v - \cos\theta \cos\phi \cdot w. \tag{9}$$

In the actual motion of the airship, the speeds v and w are much smaller than u, and due to the strong restoring torque and large damping of the rolling motion, the roll angle ϕ is small. Therefore, from (6) and (9), approximate equations can be obtained as

$$\begin{aligned} V &= \sqrt{u^2 + v^2 + w^2} \approx u, \\ \dot{h} &\approx \sin\theta \cdot u \approx \sin\theta \cdot V. \end{aligned} \tag{10}$$

It is can be seen that the three parameters of the velocity V, the pitch angle θ, and the altitude-changing rate $\dot{h}$ have strong correlation.

The airship motion has 12 state parameters of position, velocity, angle, and angular velocity in 3 coordinate axes. If

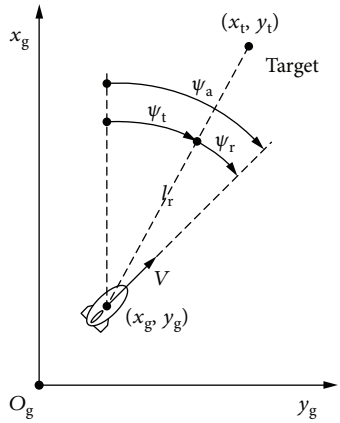


Figure 2: Coordinate frame of the relative state between the airship and the target.

all the parameters are taken into account in the airship altitude control, the scale of the state space and the amount of memory to store value function estimates will be too large for the controller to complete the training in a limited time. To avoid this "curse of dimensionality" problem, it is necessary to simplify the state parameters according to their relevance to motion control.

The parameters related to the airship altitude control mainly include the altitude h, the altitude-changing rate $\dot{h}$, the velocity V, and the pitch angle θ. From (10), $\dot{h}$, V, and θ have strong correlation; thus, two of them can contain the main information. Since the speed parameters are relatively easy to measure, we construct the state space of the airship altitude control MDP model by $(h, \dot{h}, V)$.

2.2.2. State Space **S** *of the Planar Path-Following Control.* According to the control objective (2), the state parameters related to the planar path-following control mainly include the position of the airship (x_g, y_g), the position of the target (x_t, y_t), the velocity V, and the course angle ψ_a. We use the parameters $(x_g, y_g, x_t, y_t, \psi_a, V)$ to describe the main information of the airship planar path-following control.

Remark 2. The yaw angle ψ of airship is also an important parameter for the airship path-following control. However, the course angle ψ_a has strong correlation with the yaw angle ψ due to the course stability of the airship. Under the influence of the airship vertical tail, the course angle ψ_a will soon follow the yaw angle ψ, and the difference between them is limited. Since the course angle ψ_a is directly related to the planar motion, we choose ψ_a to describe the state of the airship.

To avoid the "curse of dimensionality" problem during the training process in multidimensional space, the parameters $(x_g, y_g, x_t, y_t, \psi_a, V)$ can be simplified and merged based on airship motion analysis. A common method is to rebuild the coordinate frame with a relative position. Denote target position (x_t, y_t) as the origin, and the relative position can be calculated as $x_r = x_g - x_t$ and $y_r = y_g - y_t$; then, the state of the airship can be expressed as (x_r, y_r, ψ_a, V). To further accelerate the training process of the planar path-following controller, the experience can be generalized by rotating around the origin. Inspired by this idea, we present a novel coordinate frame (l_r, ψ_r, V) to describe the relative state between the airship and target, as shown in Figure 2.

In Figure 2, l_r represents the distance from the airship to the target, ψ_t is the angle between the $O_g x_g$ axis and the line from the airship to the target, and ψ_r is the angle between the line from the airship to the target and the direction of V. According to the geometrical relation, the distance l_r, the angle ψ_r, and the velocity V can describe the relative state between the airship and target, which are calculated by

$$l_r = \sqrt{x_r^2 + y_r^2} = \sqrt{\left(x_g - x_t\right)^2 + \left(y_g - y_t\right)^2},$$
$$\psi_r = \psi_a - \psi_d = \psi_a - \tan^{-1}\left(\frac{y_t - y_g}{x_t - x_g}\right). \tag{11}$$

By means of coordinate transformation, the state space of the planar path-following control can be constructed by (l_r, ψ_r, V), and the dimensions of the planar path-following control state space is reduced from 6 to 3, which effectively avoids the "curse of dimensionality" problem in multidimensional online training.

2.2.3. $\mathbf{A}, r, \mathbf{P}, J$ *of the Airship Control.* The action set $\mathbf{A}$ of MDP can be flexibly adjusted according to the actual control actuators. In this study, the action set $\mathbf{A}$ of the airship altitude control is constructed by elevator deflection and $\mathbf{A}$ of the planar path-following control is constructed by rudder deflection. The reward function r consists of the positive reward R_+ wherein the airship reaches the valid target area and the negative reward R_- wherein the airship flies outside the border. In this study, the controller is trained online and the state transition probability $\mathbf{P}$ is determined by the actual response of the airship to the control output. The optimization objective function J is set as the total cumulative reward. Let π denote the action policy and J^* denote the optimal cumulative reward, which is given by

$$J^* = \max_{\pi} J = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{12}$$

where r_t is the reward at time t and $\gamma \in (0, 1)$ is the reward discount factor.

3. Path-Following Control Design

The reinforcement learning altitude controller and the reinforcement learning planar path-following controller have similar structures as shown in Figure 3. The mapping between the "state input" and the "control output" is created autonomously through online training.

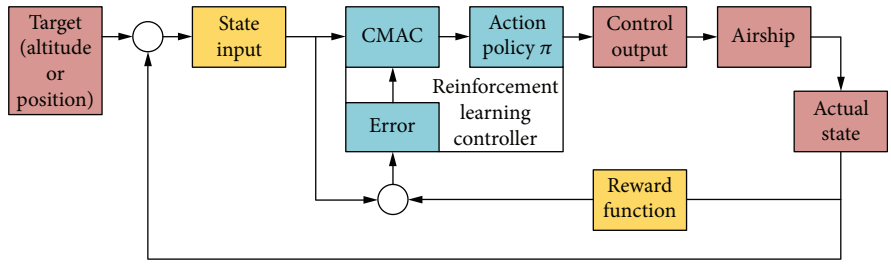


Figure 3: Structure of the airship reinforcement learning controller.

As can be seen from Figure 3, the airship reinforcement learning controller consists of three modules: the error calculation module, the CMAC neural network module, and the action policy module. The functions of each module are as follows.

3.1. Error Calculation. The error of the value function estimation is calculated based on the reinforcement learning algorithm according to the actual state transition of the airship. This error signal will be used for updating the weights of the CMAC neural network.

3.2. CMAC Neural Network. The value function of each action is estimated by the CMAC neural network, which provides the decision-making basis for the action policy module.

3.3. Action Policy. The main function of this module is to balance between exploration and the greedy policy according to the value function estimations of actions and then output the corresponding control command.

Remark 3. Different from the supervised learning, the error for modifying the neural network in the reinforcement learning controller is not from the sample's label but is calculated from the estimation of the value function and reward function. In fact, if there are enough training samples with labels provided by an expert controller, similar structures can be used for imitating the control strategy of the expert controller.

3.4. Q-Learning Algorithm. The reinforcement learning controller improves its action policy by interacting with the external environment. The controller selects the action based on the estimation of the value function and updates the estimation to maximize the cumulative reward which is given by the external environment with a random delay. Based on this concept, the Q-Learning algorithm estimates and stores the action value function in a different state and uses a "trial and error" approach to update the estimation of the action value function. Q-Learning is an appropriate reinforcement learning algorithm for the online training of the airship controller, and the properties of the airship also make it a suitable platform for the Q-Learning algorithm. Compared with an airplane and a helicopter, the attitude of the airship is relatively stable, and the "trial and error" operations of the airship rarely cause a catastrophic consequence. Furthermore, the airship is a platform with long endurance, which can provide enough time for the training and iterative function estimation of the algorithm. In this paper, the Q-Learning algorithm is adopted for calculating the modifying error of the neural network.

Let $Q(s,a)$ denote the value function estimation of the action a in state s, which can be calculated as

$$Q(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right], \tag{13}$$

where r_t is the reward at time t, γ is the reward discount factor, s_0 is the initial state, and a_0 is the initial action.

According to the theory of operation research, $Q(s,a)$ satisfies the following Bellman equation:

$$\begin{aligned} Q(s_t,a_t) = &\sum_{s_{t+1}} [p(s_t,a_t,s_{t+1}) \cdot r(s_t,a_t,s_{t+1})] \\ &+ \gamma \sum_{s_{t+1},a_{t+1}} [p(s_t,a_t,s_{t+1}) \cdot Q(s_{t+1},a_{t+1})], \end{aligned} \tag{14}$$

where $p(s_t,a_t,s_{t+1})$ denotes the transition probability from state s_t to state s_{t+1} after taking the action a_t and $r(s_t,a_t,s_{t+1})$ indicates the reward of the action a_t and the transition from s_t to s_{t+1}. Let π denote the action policy; then, the optimal action policy $\pi^*(s)$ and the optimal action estimation $Q^{\pi^*}(s,a)$ corresponding to equation (14) can be described as follows:

$$\begin{aligned} Q^{\pi^*}(s,a) &= \max_{\pi} Q(s,a), \\ \pi^*(s) &= \arg \max_{\pi} Q^{\pi^*}(s,a). \end{aligned} \tag{15}$$

From (14) and (15), the one-step update formula of $Q(s,a)$ under the optimal action policy $\pi^*(s)$ can be obtained as

$$\begin{aligned} Q(s_t,a_t) = Q(s_t,a_t) + \alpha_Q \Bigg(\Big(&r(s_t,a_t,s_{t+1}) \\ &+ \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) \Big) - Q(s_t,a_t) \Bigg), \end{aligned} \tag{16}$$

where α_Q is the learning rate of the Q-Learning algorithm.

3.5. CMAC Neural Network. In actual control applications, the multidegrees of system freedom always result in the large scale of the state space. In order to accelerate the training

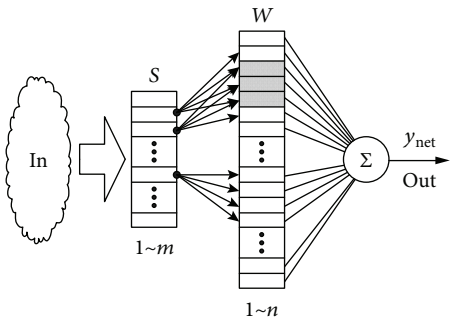


Figure 4: Structure of the CMAC neural network.

process of the reinforcement learning controller, an effective approach is to fit the value function based on the theory of stochastic process and the neural network. Neural networks can be divided into global approximation neural networks and local approximation neural networks. For global approximation neural networks like the Back Propagation (BP) neural network, each weight may affect the output and each training sample will update all the weights. As a result, the convergence speed is slow and can hardly meet the requirement of online training. For local approximation neural networks like the CMAC neural network, only a few weights affect the output and each training sample only updates the local weights, which is able to quickly approximate time-varying nonlinear functions.

During the training of the reinforcement learning controller, there are not enough training samples but only limited data from online motion; hence, a rapid convergence speed is necessary. For an airship, the control strategies are alike in similar states, and the control strategies may be completely different in very dissimilar states. The advantages of the local approximation neural network suit these properties of the airship. The CMAC neural network fits the nonlinear function by table querying and updates the weights in the local neighborhood, and it has less computation, which is convenient to be applied in the embedded system of a flight control computer. For the above advantages, in this study, the CMAC neural network is employed to generalize the training experience of the airship reinforcement learning controller.

The structure of a typical CMAC neural network is shown in Figure 4, where $\mathbf{S}$ is the state space formed by m states, $\mathbf{W}$ is the weights stored in n memory addresses, and y_{net} is the output of the network. The actual state input is mapped to state s in the state space $\mathbf{S}$, and each state s_i corresponds to an activation vector $\mathbf{F}(s_i) = \mathbf{F}_i$. The $\mathbf{W}$ and $\mathbf{F}_i$ are given by

$$\begin{aligned} \mathbf{W} &= \left[w_1, w_2, \cdots, w_j \cdots, w_n\right]^T, \\ \mathbf{F}_i &= \left[f_{i1}, f_{i2}, \cdots, f_{ij} \cdots, f_{\text{in}}\right]^T \quad i = 1, 2 \cdots, m, \\ & \quad j = 1, 2 \cdots, n, \end{aligned} \tag{17}$$

where $f_{ij} = 1$ or 0 represents the activation state of s_i to w_j. For any two states, the closer the states are, the more overlaps their activation vectors have. And if two states are far away from each other, their activation vectors have no overlap.

The output $y_{\text{net}}(s_i)$ of the CMAC neural network can be obtained as

$$y_{\text{net}}(s_i) = \mathbf{W}^T \mathbf{F}(s_i) = \mathbf{W}^T \mathbf{F}_i = \sum_{j=1}^{n} w_j f_{ij}. \tag{18}$$

We establish one CMAC neural network for each action in the action set $\mathbf{A}$, and the network weights of $a \in \mathbf{A}$ can be defined as $\mathbf{W}_a$. Letting $\tilde{Q}(s, a)$ denote the value function estimation of the state-action pair (s, a), we have

$$\tilde{Q}(s, a) = \mathbf{W}_a^T \mathbf{F}(s). \tag{19}$$

Then, the partial derivative of $\tilde{Q}(s, a)$ to $\mathbf{W}_a$ is

$$\frac{\partial \tilde{Q}(s, a)}{\partial \mathbf{W}_a} = \mathbf{F}(s). \tag{20}$$

Suppose a_t as the action at time t, then in state s_t, the value function estimation error δ_t of $Q(s_t, a_t)$ is given by

$$\delta_t = Q(s_t, a_t) - \tilde{Q}(s_t, a_t). \tag{21}$$

The network weights $\mathbf{W}_{a_t}$ of action a_t can be updated by a gradient algorithm

$$\text{“}_{a_t t+1} = \mathbf{W}_{a_t t} + \alpha_{\mathbf{W}} \delta_t \frac{\partial \tilde{Q}(s_t, a_t)}{\partial \mathbf{W}_{a_t}}, \tag{22}$$

where $\alpha_{\mathbf{W}}$ is the learning rate of the weights. By substituting (20) and (21) into (22), we have

$$\mathbf{W}_{a_t t+1} = \mathbf{W}_{a_t t} + \alpha_{\mathbf{W}} \left(Q(s_t, a_t) - \tilde{Q}(s_t, a_t)\right) \mathbf{F}(s_t), \tag{23}$$

but $Q(s_t, a_t)$ is unknown. Based on the concept (16) of the Q-Learning algorithm, $Q(s_t, a_t)$ can be replaced by $r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} \tilde{Q}(s_{t+1}, a_{t+1})$ which is calculated from the reward and value function estimation at time $t+1$. Taking into account the impact of the state visit frequency, the updated formula of $\mathbf{W}_{a_t}$ with a fitness coefficient can be written as

$$\begin{aligned} \mathbf{W}_{a_t t+1} &= \mathbf{W}_{a_t t} - \alpha_{\mathbf{W}} e_t(s_t, a_t) \Bigg(\Big(r(s_t, a_t, s_{t+1}) \\ &\quad + \gamma \max_{a_{t+1}} \tilde{Q}(s_{t+1}, a_{t+1}) \Big) - \tilde{Q}(s_t, a_t) \Bigg) \mathbf{F}(s_t) \\ &= \mathbf{W}_{a_t t} - \alpha_{\mathbf{W}} e_t(s_t, a_t) \Bigg(\Big(r(s_t, a_t, s_{t+1}) \\ &\quad + \gamma \max_{a_{t+1}} \left(\mathbf{W}_{a_{t+1}}^T \mathbf{F}(s_{t+1})\right) \Big) - \mathbf{W}_{a_t}^T \mathbf{F}(s_t) \Bigg) \mathbf{F}(s_t), \end{aligned} \tag{24}$$

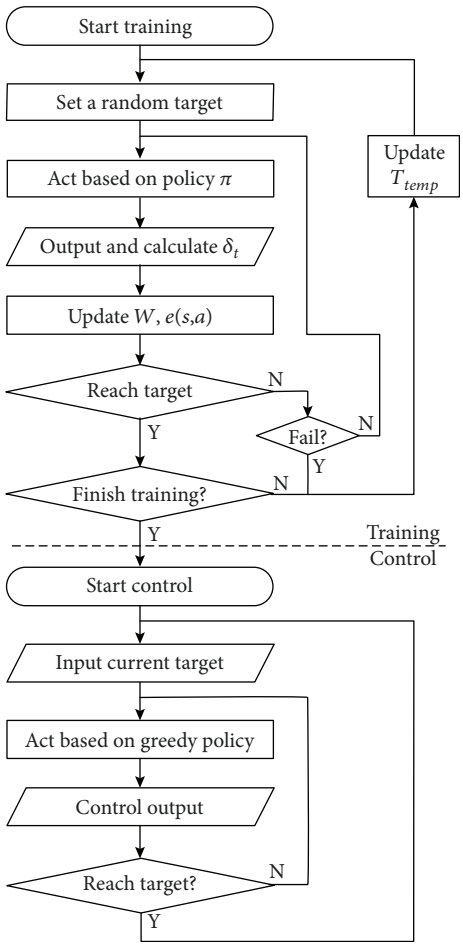


Figure 5: Training and control processes.

where $e(s, a)$ is the fitness coefficient with an initial value of 1, which reflects the state visit frequency:

$$e_{t+1}(s,a) = \begin{cases} \gamma e_t(s,a), & (s,a) \neq (s_t,a_t), \\ 1, & (s,a) = (s_t,a_t). \end{cases} \tag{25}$$

3.6. Process of Training and Control. To avoid being trapped in a local optimum and ensure the convergence speed of the training, a proper action policy π is necessary for the reinforcement learning controller. In this paper, a random action policy based on the Boltzmann distribution is adopted for balancing between the exploration and greedy policy. Let $\mathbf{A}$ denote the available action set and $p(a|s)$ denote the probability of selecting action a in state s; then, the action policy π is given by

$$p(a_t|s_t) = \frac{e^{Q(s_t,a_t)/T_{\text{temp}}}}{\sum_{a\in\mathbf{A}} e^{Q(s_t,a)/T_{\text{temp}}}}, \tag{26}$$

where T_{temp} is the exploration coefficient which is similar to the temperature coefficient of a simulated annealing algorithm. By progressively reducing T_{temp}, this policy ensures a high exploration probability in the initial states and gradually trends to the greedy policy.

The training and control process of the proposed airship reinforcement learning controller is described as shown in Figure 5. The flowchart before the dotted line is the online training process, and the rest is the actual control process.

As can be seen from Figure 5, during the training process, the airship starts one round of learning after another by randomly setting the position of the target. In each round of learning, the controller selects the action based on policy π periodically before reaching the target or boundary, and the fitness coefficient and the weights of the CMAC neural network are updated at the same time. When the algorithm and the neural network converge, the training process is completed and the controller can be applied in actual control missions. During the control process, the parameters of the algorithm are fixed, and the action policy is turned into the greedy policy. The controller maximizes the cumulative reward by selecting the action corresponding to the maximum value function estimation.

4. Simulation

To test the performance of the proposed controllers, simulations are implemented in MATLAB. We build the model of a LS-S1200 robotic airship [28] based on the approach in [9, 27]. The aerodynamic forces and torque can be calculated as

$$\begin{aligned}
F_{\text{aX}} &= Q_\infty \left[C_{\text{X1}} \cos^2\alpha \cos^2\beta + C_{\text{X2}} \sin(2\alpha) \sin\left(\frac{\alpha}{2}\right) \right], \\
F_{\text{aY}} &= Q_\infty \left[C_{\text{Y1}} \cos\left(\frac{\beta}{2}\right) \sin(2\beta) + C_{\text{Y2}} \sin(2\beta) \right. \\
&\qquad \left. + C_{\text{Y3}} \sin\beta \sin(|\beta|) + C_{\text{Y4}}\delta_{\text{R}} \right], \\
F_{\text{aZ}} &= Q_\infty \left[C_{\text{Z1}} \cos\left(\frac{\alpha}{2}\right) \sin(2\alpha) + C_{\text{Z2}} \sin(2\alpha) \right. \\
&\qquad \left. + C_{\text{Z3}} \sin\alpha \sin(|\alpha|) + C_{\text{Z4}}\delta_{\text{E}} \right], \\
L_{\text{a}} &= Q_\infty [C_{\text{L2}} \sin\beta \sin(|\beta|)], \\
M_{\text{a}} &= Q_\infty \left[C_{\text{M1}} \cos\left(\frac{\alpha}{2}\right) \sin(2\alpha) + C_{\text{M2}} \sin(2\alpha) \right. \\
&\qquad \left. + C_{\text{M3}} \sin\alpha \sin(|\alpha|) + C_{\text{M4}}\delta_{\text{E}} \right], \\
N_{\text{a}} &= Q_\infty \left[C_{\text{N1}} \cos\left(\frac{\beta}{2}\right) \sin(2\beta) + C_{\text{N2}} \sin(2\beta) \right. \\
&\qquad \left. + C_{\text{N3}} \sin\beta \sin(|\beta|) + C_{\text{N4}}\delta_{\text{R}} \right],
\end{aligned} \tag{27}$$

where $Q_\infty = \rho V^2/2$ is the dynamic pressure of the inflow, ρ is the atmospheric density, α is the attack angle, β is the sideslip angle, $F_{\text{aX}} \sim N_{\text{a}}$ are the aerodynamic forces and torque of the airship, and $C_{\text{X1}} \sim C_{\text{N4}}$ are the hull aerodynamic coefficients. The aerodynamic coefficients, the

Table 1: Parameters of the airship model and controller.

m_a (kg)	100	C_{Y2}	-10.798	C_{N1}	-36.364
$\nabla(m^3)$	79.0	C_{Y3}	-22.852	C_{N2}	56.948
ρ (kg/m^3)	1.225	C_{Y4}	2.336	C_{N3}	42.655
I_x (kg·m^2)	324	C_{Z1}	-5.629	C_{N4}	-12.324
I_y (kg·m^2)	474	C_{Z2}	-10.798	ε_h(m)	1.5
I_z (kg·m^2)	202	C_{Z3}	-21.622	ε_r(m)	15
k_1	0.084	C_{Z4}	-2.336	γ	0.9
k_2	0.856	C_{L2}	2.337	$\alpha_{\mathbf{W}}$	0.003
k_3 (m^2)	5.543	C_{M1}	36.364	T_{temp}	40
C_{X1}	-0.518	C_{M2}	-56.948	R_+	10
C_{X2}	-5.629	C_{M3}	-42.655	R_-	-10
C_{Y1}	-5.629	C_{M4}	-12.324		

structure parameters, and the reinforcement learning controller parameters used in the simulation are listed in Table 1, where m_a is the mass of the airship, ∇ is the volume of the airship, $I_x \sim I_z$ are the moments of inertia, and $k_1 \sim k_3$ are the inertia factors.

During the training of the controller, the propeller thrust changes randomly every 15 seconds to simulate the variation of the airship velocity. When the training is completed and the control mission begins, the airship velocity is maintained by a simple proportion integral (PI) propeller thrust controller, in which the cruising velocity is set to 10 m/s.

For convenience in analysing the control performance, the well-tuned PID controller is chosen as a comparison. In the outer loop of the PID controller, the desired attitude angle is calculated based on the line-of-sight (LOS) guidance law [29]. In the inner loop, we establish the dynamic equations according to the airship's structure and aerodynamic parameters and then linearize the equations with a small perturbation method. After working out the transfer function from the rudder deflection to the attitude angle, the gain parameters of the PID controller can be tuned well using the root locus tool RLTOOL in MATLAB.

4.1. Training of Reinforcement Learning Controllers. The training of the proposed airship altitude control and planar path-following control is independent and can be implemented simultaneously. The controller is trained online by randomly setting the target near the current position of the airship. The setting range of the desired altitude is 10 m, and the setting range of the target point is 100 m. In this paper, the weights of the CMAC neural network are saved every 100 s for test and evaluation. Figure 6 shows the trend of the success rate and the average time spent on the training missions during the training process.

In Figure 6(a), after training for about 1000 s, the mission success rate of the airship altitude controller is stable at 100%; then, the altitude controller can be applied in actual control missions. And as shown in Figures 6(c) and 6(d), the training of the airship planar path-following controller is completed in about 3 hours. In Figures 6(b) and 6(d), the average time spent on the training missions increased first, then decreased gradually, and eventually tended to be stable. This process corresponds to three stages of learning: at the initial stage, the controller could not control the airship and failed soon; then, the controller learned to lead the airship to reach the target in an indirect and suboptimal path; finally, the action policy was continuously optimized and gradually stabilized.

4.2. Altitude Control. We use the trained reinforcement learning controller and a well-tuned PID controller to implement the airship altitude control mission. When the propeller thrust F_T remains constant and the airship keeps a straight flight ($\delta_R = 0$), the simulation result is shown in Figure 7(a). When the propeller thrust F_T and rudder deflection δ_R change randomly every 15 seconds, the simulation result is shown in Figure 7(b).

As shown in Figure 7(a), both the reinforcement learning controller and the PID controller perform well when the airship is in the steady flight state. The altitude control errors of the two controllers are within 5 m, and the PID controller performs slightly better.

In Figure 7(b), it can be seen that when the airship is in the maneuver flight state, the performance of the reinforcement learning controller is slightly affected, and its altitude control error is still within 5 m. But the PID controller is significantly affected and its maximum altitude control error reaches about 10 m, which is consistent with the actual experiment data of the airship [28]. The reinforcement learning controller performs better than the PID controller when the airship is in the maneuver flight state. This is because during the training process, the airship has been in a random maneuver flight state, and the reinforcement learning controller gradually learned the action policy in that state by autonomous adjustment.

4.3. Planar Path-Following Control. A desired path consisting of continuous turns, a straight line, and arcs is used for testing the performance of the planar path-following controller. Figure 8 shows the planar path-following control simulation results. Figure 9 shows the deviation distance of the airship to the target points and the desired path.

In Figures 8 and 9, it can be seen that the planar path-following control error of the reinforcement learning controller is close to that of the PID controller near the 20th target point. At this point, the airship is in the steady flight state on the desired straight path. During the rest of the flight, the airship is in the maneuver flight state, and the control error of the reinforcement learning controller is smaller. This is because the action policy of the reinforcement learning controller is not determined by the current deviation but based on the total cumulative reward. When the current deviation is small but a significant control error is about to occur, the reinforcement learning controller can react in time.

The deviation distance of the reinforcement learning controller and the PID controller is compared in Table 2. To complete the planar path-following control mission, the

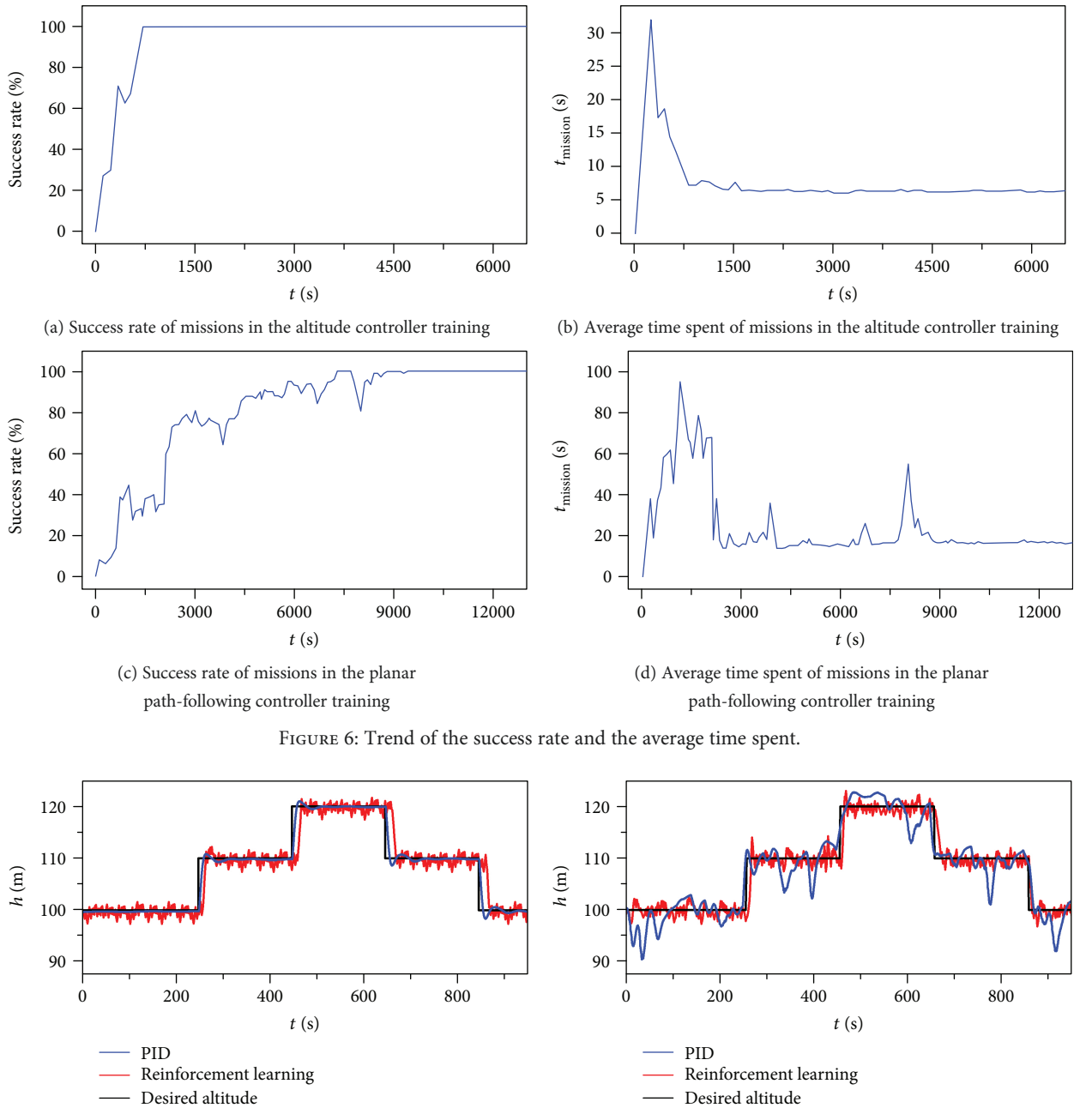


Figure 6: Trend of the success rate and the average time spent.

Figure 7: The airship altitude control.

reinforcement learning controller takes 419.2 s and the PID controller takes 443.7 s. By comparing the above data, it is clear that the reinforcement learning controller not only is slightly better in terms of control accuracy but also takes less time in the whole control mission.

To test the adaptability of the reinforcement learning controller, we can assume that asymmetric aerodynamic shape changes occur in the airship dynamic model. In this example, an additional yaw torque coefficient ($C_{\text{Nadd}} = -4.9$) is added to the dynamic equation, which will lead to the asymmetric changes in the airship's turning radius and turning ability. As a result, the performance of the original controllers may get worse, but the reinforcement learning controller can autonomously adjust its action policy by training online. Figure 10 shows the simulation result of the controllers after the airship's dynamic model changed.

As shown in Figure 10, the airship's right turning ability is weaker than the left due to the asymmetric aerodynamic shape. Controlled by the PID controller, the airship kept circling around the 2nd target point but cannot get closer. Through expanding the valid target area, the PID controller can complete the control mission, but its control error becomes larger because of the relaxation of the control requirements. After online training, the reinforcement

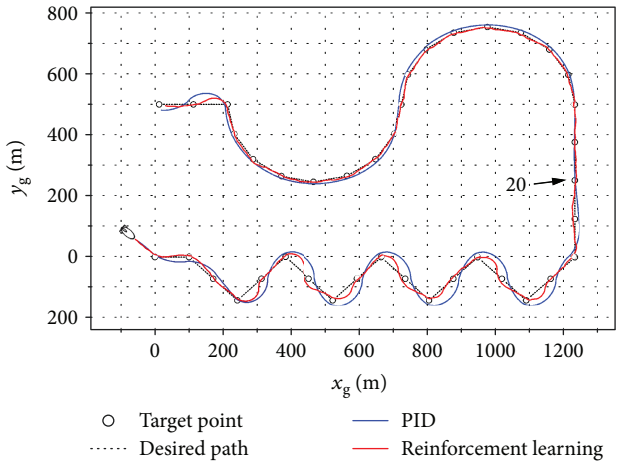
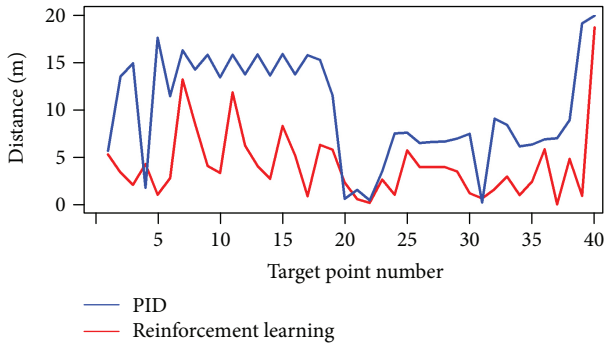
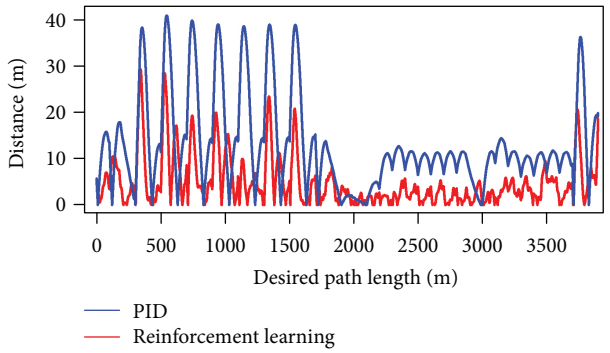


Figure 8: The airship planar path-following control.



(a) Distance from the airship to the target points



(b) Distance from the airship to the desired path

Figure 9: The deviation distance in the planar path-following control.

learning controller autonomously adapts to the model changes and is able to complete the control mission without expanding the valid target area. Furthermore, it is remarkable that near the 11th target point, the reinforcement learning controller shows an intelligent decision-making ability similar to human operators. When the airship could not reach the 11th target point directly because of the weak right turning ability, the reinforcement learning controller implemented the action policy of flying further at first and then turning left, which successfully led the airship to reach the target. This is owing to the experience of reaching the target in an indirect path that the reinforcement learning controller gained during the training process.

Table 2: The maximum and the average deviation distance.

Deviation distance	PID	Reinforcement learning
Airship to target maximum (m)	17.6	13.2
Airship to target average (m)	10.1	4.2
Airship to path maximum (m)	40.9	29.2
Airship to path average (m)	19.9	4.8

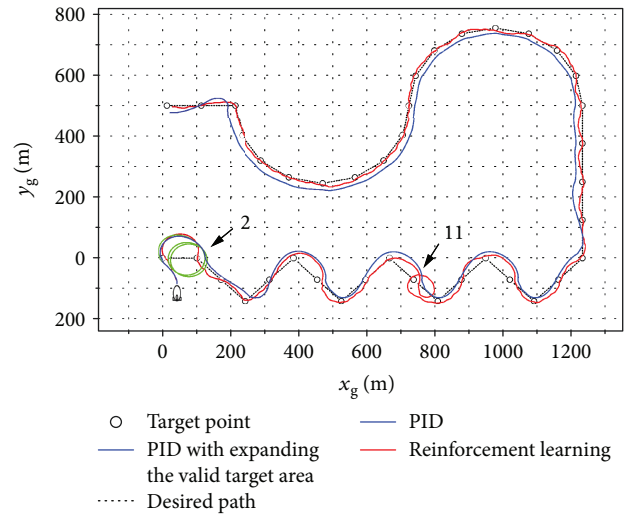


Figure 10: The planar path-following control after the airship's dynamic model changed.

4.4. 3D Path-Following Control. The 3D path-following control mission can be completed through the cooperation of the proposed controllers. In this paper, the target point of the planar path-following controller and the desired altitude of the altitude controller are calculated according to the following method.

Planar path-following control: in the 3D path-following control, the planar path-following is chosen as the main control mission; therefore, we project the 3D desired path onto the horizontal plane and create a sequence of target points.

Altitude control: while the airship is flying to the target point, the altitude controller works as an auxiliary and updates its desired altitude periodically. As shown in Remark 1, the altitude of the nearest point on the 3D desired path to the airship is taken as the desired altitude.

Taking a spiral path as an example, the simulation result of the proposed airship reinforcement learning controller is shown in Figure 11, where the horizontal distance between two adjacent target points is 100 m and the vertical distance is 10 m. It can be seen from Figure 11 that the performance of the planar path-following controller in the 3D path-following control is similar to that in the independent planar control, while the altitude of the airship slightly shows step changes. This is because the desired altitude of the altitude controller is updated periodically. The overall simulation result shows that the reinforcement learning planar

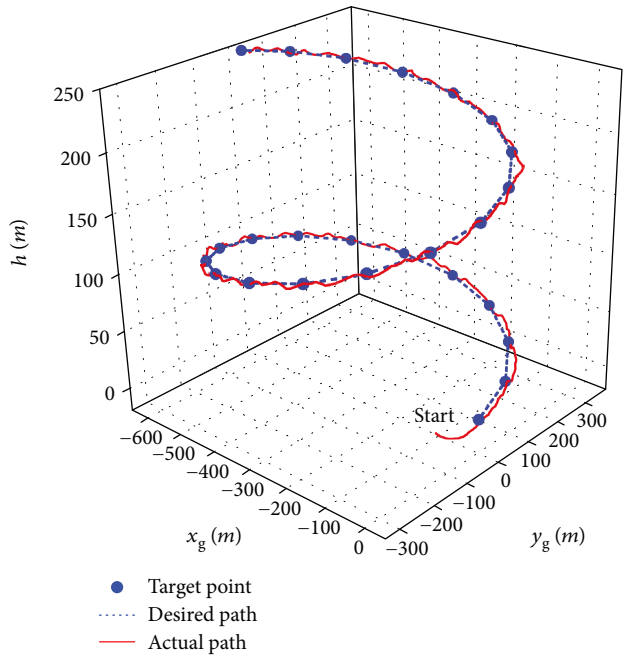


Figure 11: The airship 3D path-following control.

path-following controller and altitude controller do not interfere with each other and cooperate well.

5. Conclusions

In this paper, a reinforcement learning airship 3D path-following control is proposed, which is robust against dynamic model changes. To avoid the "curse of dimensionality" problem, the MDP models of the airship altitude control and planar path-following control are established, and the scale of the state space is reduced by parameter simplification and coordinate transformation. The airship reinforcement learning controllers are designed based on the Q-Learning algorithm and CMAC neural network, and the training process can converge in 3 hours. The reinforcement learning altitude controller and planar path-following controller can achieve comparable performance to well-tuned PID controllers, and the 3D path-following control mission can be completed through cooperation of the proposed controllers. Compared with existing approaches, the main advantage of this control strategy is that when the airship model dynamic parameters are uncertain or wrong, the controller is able to adjust its action policy autonomously and make intelligent decisions similar to human operators.

Data Availability

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

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61503010), the Aeronautical Science Foundation of China (No. 2016ZA51001), and the Fundamental Research Funds for the Central Universities (No. YWF-19-BJ-J-118).

References

[1] L. Chen, D. P. Duan, and D. S. Sun, "Design of a multi-vectored thrust aerostat with a reconfigurable control system," *Aerospace Science and Technology*, vol. 53, pp. 95–102, 2016.

[2] Y. Yang, J. Wu, and W. Zheng, "Positioning control for an autonomous airship," *Journal of Aircraft*, vol. 53, no. 6, pp. 1638–1646, 2016.

[3] Y. Yang, "A time-specified nonsingular terminal sliding mode control approach for trajectory tracking of robotic airships," *Nonlinear Dynamics*, vol. 92, no. 3, pp. 1359–1367, 2018.

[4] Z. Zheng and L. Xie, "Finite-time path following control for a stratospheric airship with input saturation and error constraint," *International Journal of Control*, pp. 1–26, 2017.

[5] Y. Yang and Y. Ye, "Backstepping sliding mode control for uncertain strict-feedback nonlinear systems using neural-network-based adaptive gain scheduling," *Journal of Systems Engineering and Electronics*, vol. 29, no. 3, pp. 140–146, 2018.

[6] Z. Zheng, L. Liu, and M. Zhu, "Integrated guidance and control path following and dynamic control allocation for a stratospheric airship with redundant control systems," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 230, no. 10, pp. 1813–1826, 2016.

[7] Y. Yang and Y. Yan, "Attitude regulation for unmanned quadrotors using adaptive fuzzy gain-scheduling sliding mode control," *Aerospace Science and Technology*, vol. 54, pp. 208–217, 2016.

[8] S. Liu, Y. Sang, and H. Jin, "Robust model predictive control for stratospheric airships using LPV design," *Control Engineering Practice*, vol. 81, pp. 231–243, 2018.

[9] G. A. Khoury, *Airship Technology*, Cambridge university press, 2012.

[10] T. Du, L. Guo, and J. Yang, "A fast initial alignment for SINS based on disturbance observer and Kalman filter," *Transactions of the Institute of Measurement and Control*, vol. 38, no. 10, pp. 1261–1269, 2016.

[11] T. Du and L. Guo, "Unbiased information filtering for systems with missing measurement based on disturbance estimation," *Journal of the Franklin Institute*, vol. 353, no. 4, pp. 936–954, 2016.

[12] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

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

[14] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: a survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[15] D. L. Leottau, J. Ruiz-del-Solar, and R. Babuška, "Decentralized reinforcement learning of robot behaviors," *Artificial Intelligence*, vol. 256, pp. 130–159, 2018.

[16] E. Walraven, M. T. J. Spaan, and B. Bakker, "Traffic flow optimization: a reinforcement learning approach," *Engineering*

Applications of Artificial Intelligence, vol. 52, pp. 203–212, 2016.

[17] S. Kosunalp, Y. Chu, P. D. Mitchell, D. Grace, and T. Clarke, "Use of *Q*-learning approaches for practical medium access control in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 146–154, 2016.

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

[19] B. Pearre and T. X. Brown, "Model-free trajectory optimisation for unmanned aircraft serving as data ferries for widespread sensors," *Remote Sensing*, vol. 4, no. 10, pp. 2971–3005, 2012.

[20] C. Duun, J. Valasek, and K. Kirkpatrik, "Unmanned air system search and localization guidance using reinforcement learning," in *Infotech@ Aerospace*, pp. 1–8, Garden Grove, CA, USA, June 2012.

[21] B. Zhang, Z. Mao, W. Liu, and J. Liu, "Geometric reinforcement learning for path planning of UAVs," *Journal of Intelligent and Robotic Systems*, vol. 77, no. 2, pp. 391–409, 2015.

[22] I. Palunko, A. Faust, P. J. Cruz, L. Tapia, and R. Feirro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *IEEE International Conference on Robotics and Automation*, pp. 1–6, Karlsruhe, Germany, May 2013.

[23] A. Faust, *Reinforcement Learning and Planning for Preference Balancing Tasks*, University of New Mexico, 2014.

[24] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *2007 IEEE International Conference on Robotics and Automation*, pp. 742–747, Roma, Italy, April 2007.

[25] A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard, "Autonomous blimp control using model-free reinforcement learning in a continuous state and action space," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1895–1900, San Diego, CA, USA, October-November 2007.

[26] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[27] Z. Zheng, Y. Huang, L. Xie, and B. Zhu, "Adaptive trajectory tracking control of a fully actuated surface vessel with asymmetrically constrained input and output," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1851–1859, 2018.

[28] Z. Zheng, W. Huo, and Z. Wu, "Autonomous airship path following control: theory and experiments," *Control Engineering Practice*, vol. 21, no. 6, pp. 769–788, 2013.

[29] Z. Zheng, L. Sun, and L. Xie, "Error-constrained LOS path following of a surface vessel with actuator saturation and faults," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, vol. 48, no. 10, pp. 1794–1805, 2018.