actuators

MDPI

Article

Encouraging Guidance: Floating Target Tracking Technology for Airborne Robotic Arm Based on Reinforcement Learning

Jiying Wu [1], Zhong Yang [1,*], Haoze Zhuo [1], Changliang Xu [2], Luwei Liao [1], Danguo Cheng [1] and Zhiyong Wang [1]

[1] College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China; wujiying@nuaa.edu.cn (J.W.); zhuohaoze@nuaa.edu.cn (H.Z.); llw@nuaa.edu.cn (L.L.); bx2403510@nuaa.edu.cn (D.C.); wangzhiyong@nuaa.edu.cn (Z.W.)

[2] College of Electronic Engineering, Nanjing Xiaozhuang University, Nanjing 211171, China; xuchangliang@nuaa.edu.cn

* Correspondence: yangzhong@nuaa.edu.cn

Abstract: Aerial robots equipped with operational robotic arms are a powerful means of achieving aerial contact operations, and their core competitiveness lies in target tracking control at the end of the airborne robotic arm (ARA). In order to improve the learning efficiency and flexibility of the ARA control algorithm, this paper proposes the encouraging guidance of an actor–critic (Eg-ac) algorithm based on the actor–critic (AC) algorithm and applies it to the floating target tracking control of ARA. It can quickly lock in the exploration direction and achieve stable tracking without increasing the learning cost. Firstly, this paper establishes approximate functions, policy functions, and encouragement functions for the state value of ARA. Secondly, an adoption rate controller (ARC) module was designed based on the concept of heavy rewards and light punishments (HRLP). Then, the kinematic and dynamic models of ARA were established. Finally, simulation was conducted using stable baselines3 (SB3). The experimental results show that, under the same computational cost, the convergence speed of the Eg-ac is improved by 21.4% compared to deep deterministic policy gradient (DDPG). Compared with soft actor–critic (SAC) and DDPG, Eg-ac has improved learning efficiency by at least 20% and has a more agile and stable floating target tracking effect.

Keywords: airborne robotic arm; floating target tracking; reinforcement learning; inverse kinematic solution

Received: 17 December 2024
Revised: 22 January 2025
Accepted: 29 January 2025
Published: 31 January 2025

Citation: Wu, J.; Yang, Z.; Zhuo, H.; Xu, C.; Liao, L.; Cheng, D.; Wang, Z. Encouraging Guidance: Floating Target Tracking Technology for Airborne Robotic Arm Based on Reinforcement Learning. *Actuators* **2025**, *14*, 66. https://doi.org/10.3390/act14020066

1. Introduction

With the continuous development of aerial robot technology, flying robots carrying different onboard equipment to reach designated positions and complete high-altitude tasks have gradually become a situation of interest in aerial operations [1–4]. Fast and safe working methods are gradually replacing traditional manual operations. At present, there is still a certain gap between the various applications of aerial robots and the widespread applications in the concept [5]. There are still many technical issues that need to be further addressed, such as designing specific robot configurations, operating tools, and operating modes for different application goals [6–9]. Moreover, in-depth research and the optimization of control strategies for aerial work robots have become urgent technical challenges that need to be addressed.

This article takes obstacle removal, line repair, bolt tightening, and other aerial contact operations of high-altitude transmission lines as the research background. At present, these tasks still mainly rely on manual operations with significant safety hazards [10,11], and the intelligent operation of airborne robotic arms is a powerful means to solve the above problems. As shown in Figure 1, the aerial robot system studied in this article can perform line hanging operations on high-altitude routes. This robot integrates a multi-rotor unmanned aerial vehicle, a hanging line walking system, and an airborne multidegree of freedom robotic arm that can accurately reach the target position in complex aerial environments and complete hanging line, walking, and line clearing tasks. Due to the complex aerial operation environment and the constantly changing external disturbances, the core competitiveness of this system lies in the precise control of the aerial operation robotic arm. Only by improving positioning accuracy and tracking performance can the accuracy and effectiveness of aerial operations be ensured. Based on this, this work aimed to study the control problem of an airborne operation robotic arm in an aerial robot system and conduct research on the target tracking task of the robotic arm tool end.

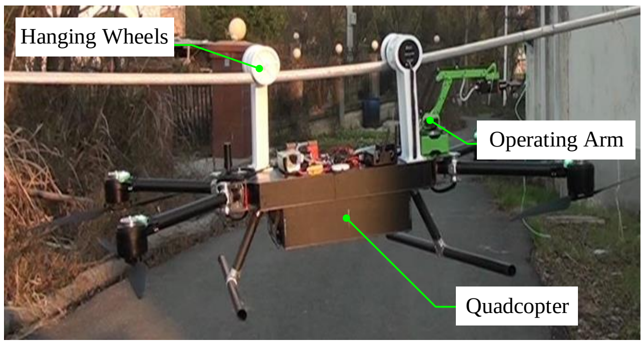


Figure 1. Aerial robot system.

Prior to this, scholars both domestically and internationally have conducted extensive research on the control issues of manipulator arms [12–17]. In recent years, control algorithms based on machine learning [18,19] have gradually emerged, and reinforcement learning frameworks have become a promising algorithmic tool for improving the control of aerial robotic arms [20–22]. The resulting deep reinforcement learning [23–28] combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning, achieving end-to-end control from input to output, attracting the attention of a large number of researchers. And research on robotic arm control based on deep reinforcement learning is also becoming increasingly popular. After the successful application of deep learning in object detection, H. Sekkat et al. [29] proposed a neural inverse kinematic solution based on deep reinforcement learning for object detection using deep learning models, which evolved grasping tasks by achieving expected goals. This method calculates the joint angle of the detected position through inverse kinematics, causing the robot arm to move toward the position of the target object. The simulation results showed that the accuracy of the end effector grip joint angle and posture of the robot is satisfactory. T. Lindner [30] studied six combinations of four reinforcement learning algorithms for robot localization tasks. These algorithms were used for the positioning control of robot arm models, taking into account the evaluation of positioning accuracy, motion trajectory, and the number of steps required to achieve the target. The simulation and experimental results indicated that the RL algorithm can be successfully applied to the learning of robot arm positioning control. K. M. Oikonomou et al. [31] argued that although deep neural networks (DNNs) have achieved significant results in many robot applications, energy consumption remains a major limitation. They proposed a hybrid variant based on the deep deterministic policy gradient (DDPG) learning method for training six-degree-of-freedom robotic arms for target arrival tasks. Among them, a peak neural network was introduced into the actor model, and a DNN was introduced into the critic model. Finally, the hybrid DDPG model was compared with the classical DDPG model, demonstrating the superiority of the hybrid method. Coincidentally, B. Y. Song [32] proposed a dual-delay space robotic arm trajectory planning technique based on deep reinforcement learning to solve complex dynamics and control problems in the process of space debris removal. This technique can achieve end-to-end control effects comparable to the human grasping of objects. This study utilized joint and end effector control strategies developed using trajectory planners, trajectory trackers, and seven different weighted reward functions to implement a trajectory planning method for a floating space robotic arm to capture space debris. The experimental results indicated that this capture policy can maintain a high capture success rate. P. Wu [33] believed that traditional robotic arm control algorithms often struggle to adapt to the challenges posed by dynamic obstacles. Therefore, a reinforcement learning-based dynamic obstacle avoidance method was proposed to solve the real-time processing problem of dynamic obstacles. This method introduces a feature extraction network with an integrated gating mechanism based on traditional reinforcement learning algorithms. In addition, an adaptive dynamic reward mechanism was designed to optimize obstacle avoidance strategies. Verification showed that this method can effectively avoid randomly moving obstacles and significantly improved in convergence speed compared to traditional algorithms.

The summary above indicates that deep reinforcement learning has been widely applied in the research of space robotic arm control algorithms for tasks such as arrival, grasping, and trajectory tracking. Most of the robotic arms studied are series-connected with fixed bases, and the algorithms currently being researched can also achieve the basic goal of completing tasks. However, most algorithms do not consider the training efficiency and external disturbances of the robotic arm while pursuing positioning accuracy. When it comes to the end positioning of unstable bases and tracking of dynamic target objects, some algorithms seem inadequate and often yield unsatisfactory results.

The aerial operation-type robotic arm studied in this paper is mounted on the fuselage of a rotary wing of an unmanned aerial vehicle. Due to the special working environment, its end tool will sway with the shaking of the base and fuselage. In the task of clearing obstacles on the route studied in this article, most of the objects were floating objects with irregular floating phenomena. Therefore, the positioning accuracy and flexibility requirements of the tool end were high. Conventional control struggles to quickly locate the target object due to lag, while current reinforcement learning algorithms for robotic arms can achieve ideal results through learning. Generally speaking, value-based methods output the values of actions and are typically used in environments with discrete action spaces [23]. The policy-based approach, which outputs the probability of a direct action or action, is usually more suitable for environments with high-dimensional or continuous action spaces [26]. The behavior space of the airborne robotic arm (ARA) end positioning control is continuous and large-scale, and basic algorithms such as Monte Carlo reinforcement learning based on complete sampling and temporal differential reinforcement learning based on incomplete sampling may have low efficiency and even fail to achieve good solutions [34]. In view of this, this study adopted a policy-based learning method that regards the policy as a parameterized policy function of the ARA state and joint power output. Through establishing an objective function and using the rewards generated by the interaction between the ARA and the environment, the parameters of the policy function are learned. Currently, the most commonly used policy learning methods have a high degree of randomness in the early stages of learning, which may result in useless exploration or even completely opposite exploration paths to the target task. This situation can lead to a slower iteration speed in the early stages of learning, thereby reducing the convergence speed of the objective function. Although it ensures the exploration function of the algorithm, it increases the computational cost as a result.

In summary, this article proposes an encouragement-guided policy learning algorithm, Eg-ac, that adds an encourager to the actor–critic algorithm. The main function of the encourager is to generate strategies consistent with the final target task direction based on the current and target states of the ARA. In addition, the algorithm also incorporates a heavy rewards and light punishments (HRLP) reward mechanism and an adoption rate controller (ARC) module. The basic idea of HRLP is to increase rewards for good behavior and reduce punishments for bad behavior, thereby accelerating learning and encouraging exploration. The ARC is used to randomly generate adoption rates for the encourager under certain constraints, and the final policy is for actors and the encourager to output the ARA's behavior strategies under the regulation of the ARC. The advantage of doing so is that the ARA executes the policy and obtains the corrected state before the next training begins. As the saying goes, 'a good start is half the battle', and correcting the ARA status at any time is beneficial for approaching the final target position more quickly in subsequent training, thereby shortening the number of training steps included in an experience and improving training efficiency. Figure 2 illustrates the control policy for ARA's floating target tracking control learning using the Eg-ac algorithm proposed in this paper. Firstly, this paper establishes approximate functions, policy functions, and encouragement functions for the state value of the ARA. The value function can evaluate and optimize strategies, and the optimized policy function, through the ARC joint encouragement function, will output more reasonable behavioral strategies for the ARA, which will in turn make the value function more accurate in reflecting the value of the state. The three function types mutually promote each other and ultimately obtain the optimal tracking policy for the ARA. Then, this paper establishes the kinematic and dynamic models of the ARA, which obtain the joint positions through inverse kinematic calculation and input them into the dynamic system to obtain the current state. Finally, this study conducted simulations using the open source reinforcement learning library stable baselines3 (SB3) built on the pytorch framework.

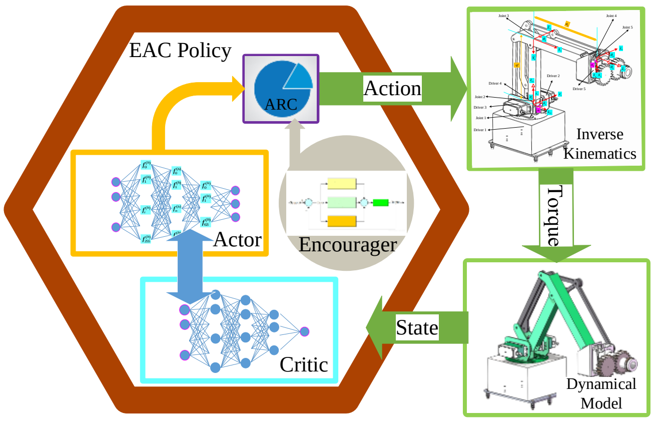


Figure 2. Control policy of Eg-ac algorithm.

The following sections of this paper are arranged as follows. The second part elaborates on the principle and process of the Eg-ac algorithm. The third part describes the establishment of the ARA model and the setting of simulation parameters. The fourth part presents the simulation details. The fifth part summarizes the results and the method.

2. Algorithm Principles and Processes

2.1. ARA Status Description

The ARA system is a continuous power system with state characteristic $s_t \in \mathbb{R}^s$ and system input $a_t \in \mathbb{R}^a$, denoted as Equation (1).

$$s_{t+1} = f\left(s_t, a_t\right) + g(t) \tag{1}$$

Among them, s_{t+1} is the prediction of the next state based on the current state feature s_t taking action a_t, $f(\cdot)$ is an unknown function, and $g(t)$ is a random noise function for exploring a small range around the control variable, where t represents all the time steps recorded in the previous step during the training process.

Assuming that each temporal process of ARA has Markovian properties, establish a Markov decision process tuple $\langle S, A, P, R, \gamma \rangle$ for the ARA:

$$\begin{cases} s_t = [p_{ax}, p_{ay}, p_{az}, \dot{p}_{ax}, \dot{p}_{ay}, \dot{p}_{az}, p_{tx}, p_{ty}, p_{tz}]^{\mathrm{T}}, s_t \in S \\ a = \left[\ddot{p}_x, \ddot{p}_y, \ddot{p}_z\right], a \in A \\ P^a_{s_t s_{t+1}} = P\left[S_{t+1} \mid S_t = s_t, A_t = a\right] \\ R^a_{s_t} = \mathrm{E}\left[R_{t+1} \mid S_t = s_t, A_t = a\right] \\ \gamma \in [0,1] \end{cases} \tag{2}$$

Among them, s_t is a set of state representations in state set S. p_{ax}, p_{ay}, and p_{az} are the position coordinates of the ARA in the three axis directions of the end position in the body coordinate system at the current time. $\dot{p}_{ax}$, $\dot{p}_{ay}$, and $\dot{p}_{az}$ are the velocities of the ARA in the three coordinate axes of the end position in the body coordinate system. p_{tx}, p_{ty}, and p_{tz} are the target position coordinates of the ARA in the three coordinate axis directions of the end position in the body coordinate system. $s_t = [p_{ax}, p_{ay}, p_{az}, \dot{p}_{ax}, \dot{p}_{ay}, \dot{p}_{az}, p_{tx}, p_{ty}, p_{tz}]^{\mathrm{T}}$ describes the state parameters of the ARA in the body coordinate system. Since the ARA operation object studied in this article is a floating object ahead, the pitch angle of the ARA operation end position in the body coordinate system was set to remain unchanged, and the state parameters do not need to consider the end effector attitude. a is a set of control variables in control set A, where $\ddot{p}_x$, $\ddot{p}_y$, and $\ddot{p}_z$ represent the acceleration of the ARA in the three coordinate axis directions of the end position in the body coordinate system. $P^a_{s_t s_{t+1}}$ represents a set of transition probabilities in the state transition probability matrix P, which represents the transition probability from the current state s_t to the next state s_{t+1} through control variable a. The reward obtained during this process is R_{t+1}. Subscripts t and $t+1$ in the above variables represent the current time step and the next time step, respectively. $R^a_{s_t}$ represents a set of reward values in the reward function R, which are the rewards obtained from the current state s_t after passing through control variable a. The γ in tuple $\langle S, A, P, R, \gamma \rangle$ is the decay factor, and S_t is the finite set of states for the current time step t. A_t is a finite set of control variables for the current time step t.

Establish the motion state update equation for the ARA as follows:

$$\begin{cases} s_{t+1} = f\left(S_t = s_t, A_t = a \mid \pi_\theta\left(s_t, a\right)\right) + g(t) \\ \pi_\theta\left(s_t, a\right) = P\left[A_t = a \mid S_t = s_t, \theta\right] \end{cases} \tag{3}$$

Among them, $\pi_\theta\left(s_t, a\right)$ is the ARA control strategy. θ is the strategy parameter. $P\left[A_t = a \mid S_t = s_t, \theta\right]$ is the probability distribution matrix of executing control variable a under the given state s_t and strategy parameter θ. $f\left(S_t = s_t, A_t = a \mid \pi_\theta\left(s_t, a\right)\right)$ is the dynamic equation obtained by continuously updating $\left\{\left(s_t, a\right), s_{t+1}\right\}$ of the strategy, and $g(t)$ is the noise function.

2.2. *Eg-ac Control Policy*

The Eg-ac algorithm proposed in this article is based on the AC algorithm, and the specific control policy is as follows:

$$\pi_{\text{Eg-ac}}\left(s_t, a\right) = P\left[A_t = a^\mu \oplus a^E \mid S_t = s_t, \theta^\mu,\ \theta^E\right] \tag{4}$$

Among them, θ^μ is the actor policy network parameter, and θ^E is the encourager policy parameter. The Eg-ac policy advocates that the predicted behavior given is the dynamic weighted behavior value of the actor's behavior and the encourager's output behavior, i.e., $A_t = a^\mu \oplus a^E$, where the dynamic weight comes from the ARC module. The symbol $\oplus$ represents weighted addition. The critic network estimates the behavior value based on the ARC output behavior and calculates the gradient of the objective function and the loss of the critic, thereby updating the parameters of each network. It is worth noting that the design of the Eg-ac control strategy references the HRLP reward mechanism and incorporates the ARC model. The specific design details are as follows.

(a) ARC Module

In order to improve learning efficiency while also considering algorithm exploration function, this paper proposes introducing an ARC before behavior output to randomly generate adoption rates η_{ARC} for the encourager under certain constraints (e.g., $\eta_{ARC} < 30\%$). N contains a set of data corresponding to the joint positions of the ARA that will be described later. And the actor will also generate a probability of adoption with a value of $1-\eta_{ARC}$. And after training to a certain extent, $\eta_{ARC} \to 0$ is set. It is known that the encourager produces controllable and directionally correct behaviors. In the initial training stage, the actor network is not yet mature, and the output values generated are irregular, even going against the expected values. In this situation, appropriately reducing the adoption rate of the actor and partially accepting encourager's suggestion output value will correct the final output action value appropriately. In the actual execution process, the output action value develops in a positive direction, that is, moving along the expected trajectory. Firstly, the punishment level will be reduced, and the reward level will be increased. Secondly, the corrected state will be obtained before the next training, which is beneficial for approaching the final target position more quickly in subsequent training, thereby shortening the number of training steps included in an experience and improving training efficiency. The reason for setting $\eta_{ARC} \to 0$ after training to a certain extent is that the encourager only plays a corrective guidance role in the initial stage, and it adopts an error control method. When it comes to target tracking tasks, there will be lag like in traditional control methods. Therefore, later participation will affect the exploratory ability of the actor itself, not only hindering the improvement of training effectiveness but also limiting the flexibility of the algorithm. Therefore, with the real-time correction of the ARA status, the next training step can approach the final target position more quickly, thus completing an experience as soon as possible, shortening the training steps, and improving training efficiency.

(b) HRLP Reward Mechanism

The presentation of the HRLP reward mechanism is based on the output behavior of the ARC module mentioned above. Due to the regulation of the ARC, the positive behavior output is amplified, which is then used as an execution strategy to obtain a positive ARA state and generate greater reward values. Bad behavior can be corrected, and executing the behavior will result in a less inappropriate state than before, thereby reducing punishment. In summary, HRLP is responsible for determining rewards based on the output values of policy functions. Its basic idea is to increase rewards for good behavior and reduce punishments for bad behavior, thereby accelerating learning and encouraging exploration.

Let the actor network function be $\mu\left(\theta^{\mu}\right)$, the critic network function be $Q\left(\theta^{Q}\right)$, the encourager function be $E\left(\theta^{E}\right)$, the target actor network function be $\mu'\left(\theta^{\mu'}\right)$, and the target critic network function be $Q'\left(\theta^{Q'}\right)$. The specific control process of the Eg-ac policy is shown in Figure 3, and $g'(t)$ is the random noise function. In the randomly selected state–action pairs $(s_i, a_i, r_{i+1}, s_{i+1})$ from the experience pool, state s_i generates a specific behavior a^{μ} through the actor network as one of the action input values for the ARC module, and the next state s_{i+1} generates a predicted behavior a' through the target actor network as the action input value for the target critic network to calculate the estimated value. The encourager network inputs the pre and post state values and error values from historical data into the PD controller, generating specific behavior a^{E} as another action input value for the ARC module. The ARC module uses a randomly generated set of adoption rates to integrate two sets of actions and outputs them to the critic network to calculate the behavior value $Q(s_i, a_i)$ corresponding to the state s_i and the integrated behavior $a_i = a^{\mu} \oplus a^{E}$. The target critic network generates the behavioral value $Q'(s_{i+1}, a')$ for calculating the target value $Q_{T\arg et}$ based on the subsequent states s_{i+1} and a'.

In the above process, in assuming that the reward obtained from interacting with the environment under strategy π_{θ} is $R_{s,a}$, the objective function is designed as follows:

$$J(\theta) = \mathrm{E}_{\pi_{\theta}}\left[R_{s,a}\right] \tag{5}$$

The ultimate goal of the algorithm is to find the optimal behavior $a_i = (1-\eta_{ARC})a^{\mu} + \eta_{ARC}a^{E}$ output by the ARC to maximize the value of the output behavior, where the reward obtained by executing a_i is r_{i+1}, and this transformation process is stored as $(s_i, a_i, r_{i+1}, s_{i+1})$. In assuming that each batch has Γ execution steps, the system carries out a certain amount of the conversion process and randomly selects X samples to calculate the target value $Q_{T\arg et}$ and the loss $Loss_c$ of the critic network:

$$Q_{T\arg et} = r_{i+1} + \gamma Q'\left(s_i, \mu'\left(s_i \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right) \tag{6}$$

$$Loss_c = \frac{1}{X}\sum_i \left(Q_{T\arg et} - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2 \tag{7}$$

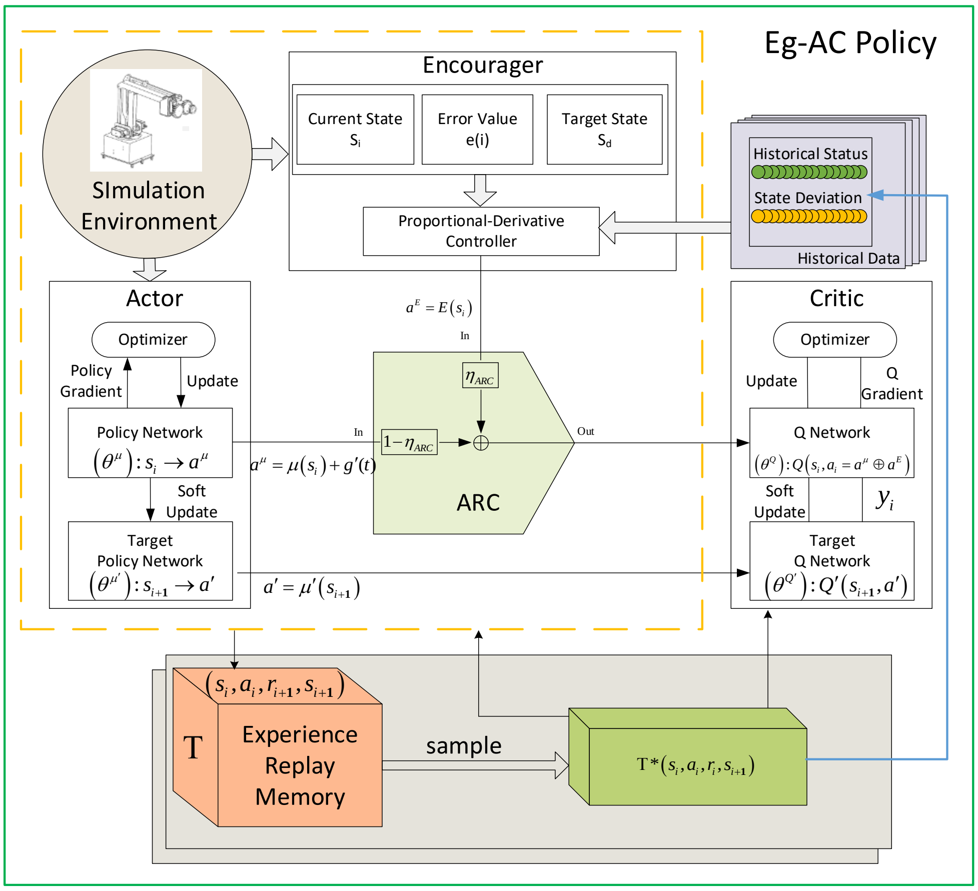


Figure 3. Eg-ac control policy.

The update method of the actor network adopts the gradient ascent method. Considering that most optimizers are designed for gradient descent, this study achieves the effect of gradient ascent by minimizing the negative Q value and uses the average negative Q value of all samples in the batch as the loss. The gradient $\nabla_{\theta^A} J$ of the objective function and the loss $Loss_a$ of the actor network are as follows:

$$\nabla_{\theta^A} J \approx \frac{1}{X} \sum \nabla_a Q\left(s, a \mid \theta^Q\right)|_{s=s_i, a=\mu(s_i)\oplus E(s_i)} \nabla_{\theta^{\mu}} \mu\left(s \mid \theta^{\mu}\right)|_{s_i} \tag{8}$$

$$Loss_a = -\frac{1}{X} \sum_i Q\left(s_i, a_i \mid \theta^Q\right) \tag{9}$$

Target network update:

$$\varsigma \theta^Q + (1-\varsigma)\theta^{Q'} \to \theta^{Q'} \tag{10}$$

$$\varsigma \theta^{\mu} + (1-\varsigma)\theta^{\mu'} \to \theta^{\mu'} \tag{11}$$

Among them, $g(t)$ is the noise function, γ is the update parameter, and ς is the learning rate.

The proportional proprietary controller corresponding to the encourager is defined as

$$\begin{cases} E(s_i) = \theta^E \left[e(k) \ \dot{e}(k) \right]^T \\ e(k) = s_i - s_d \end{cases} \tag{12}$$

Among them, θ^E is the controller adjustment parameter, $e(k)$ is the error of state $\dot{e}(k)$, which is the derivative of the state error, and s_i and s_d are the sampled state and its reference value, respectively.

Algorithm 1 provides an overview of the proposed algorithm.

Algorithm 1: Eg-ac Algorithm

```
Input: $\gamma, \varsigma, \theta^Q, \theta^\mu, \theta^E$
Output: optimized $\theta^Q, \theta^\mu$
initialize $E(s \mid \theta^E)$ with weights $\theta^E$ by Equation (12)
randomly initialize ARC with weights $\eta_{ARC}$
randomly initialize $Q(s,a \mid \theta^Q)$, $\mu(s \mid \theta^\mu)$, with weights $\theta^Q$, $\theta^\mu$
initialize target network $Q'(s,a \mid \theta^{Q'})$, $\mu'(s \mid \theta^{\mu'})$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{A'} \leftarrow \theta^A$
initialize the experience cache space $\mathrm{T}$
for episode from 1 to Limit do
    Initialize a noise function
    receive the initial state
    for $t = 1$ to $\Gamma$ do
        Perform action $a_t$ via ARC, get reward $r_{t+1}$, next state $s_{t+1}$, store transition $(s_{t,} a_t, r_{t+1}, s_{t+1})$ in $\mathrm{T}$
        sample a batch of random $X$ transitions $(s_{i,} a_i, r_{i+1}, s_{i+1})$ in $\mathrm{T}$, where $i = 0,1,2,\cdots X$
        set target value function via Equation (6)
        update critic by minimizing the loss via Equation (7)
        update actor policy by policy gradient via Equations (8) and (9)
        update networks via Equations (10) and (11)
end for
end for
```

This paper compares the performance of the SAC algorithm, DDPG algorithm, and Eg-ac algorithm in the following sections.

3. Preparation Work for Simulation

This section will elaborate on the simulation preparation work for the algorithm proposed in the paper. Section 3.1 first describes conducting a kinematic analysis of the ARA based on the Denavi–Hartenberg (D-H) method [35] to obtain the inverse kinematic solution of the ARA. Next, a dynamic analysis of the ARA was conducted to prepare for subsequent simulations. Section 3.2 introduces the simulation network structure and parameter settings.

3.1. Establishment of ARA Mathematical Model

In the Eg-ac algorithm, both the input and output of the ARC module are composed of the acceleration at the end of the ARA operation, while in the simulation process, the ARA power execution system requires inputs for each joint position. This requires the use of an inverse kinematic solution, and the ARA power system is required during the algorithm training process, so this study conducted kinematic and dynamic analyses of the ARA.

3.1.1. ARA Kinematic Analysis

The ARA operating mechanism of this paper was designed as a five-degree-of-freedom robotic arm, and Figure 4 is a three-dimensional schematic of the robotic arm. In order to enhance the structural stiffness of the main connecting rod and improve positioning accuracy, the motion chain of the robotic arm is not a simple open-loop system. Instead, a four-bar linkage mechanism is used to connect driving motor 3 to connecting rod 3, and two four-bar linkage mechanisms are used to connect driving motor 4 to connecting rod 4. The movements of the other three driving motors are directly connected to the joints. The layout of the driver and connecting rod is shown in the figure. Drive motor 2 controls the position of joint 2, while the posture of connecting rod 3 remains unchanged relative to the base; drive motor 3 can adjust the posture of connecting rod 3 relative to the base, while the posture of connecting rod 4 relative to the base remains unchanged. Similarly, driving motor 4 specifically adjusts the posture of connecting rod 4 relative to the base, regardless of the position of joints 2 and 3. The control of each joint is relatively independent, unlike serial joints where the movement of the latter joint is relative to the movement of the previous link.

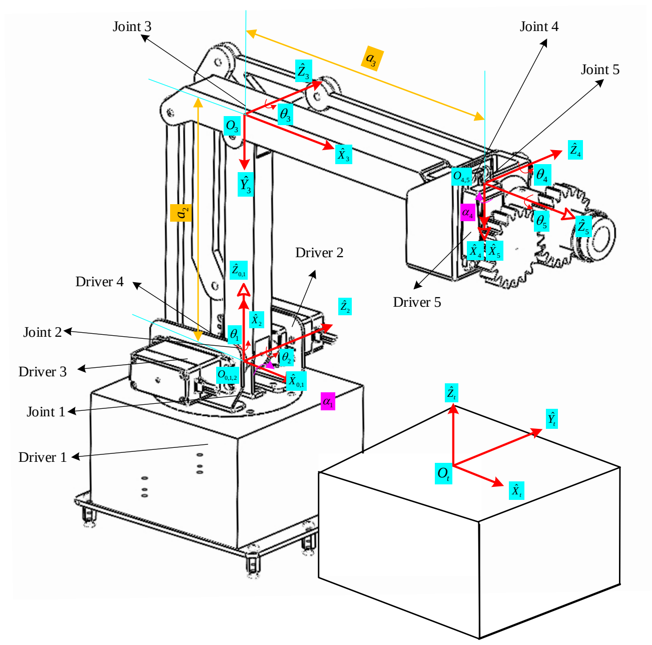


Figure 4. Schematic of ARA structure design and coordinate establishment.

The coordinate systems of each connecting rod are marked in Figure 4. Among them, the relative coordinate of the workbench coordinate system $\hat{O}_t\hat{X}_t\hat{Y}_t\hat{Z}_t$ with respect to the body coordinate system $\hat{O}_0\hat{X}_0\hat{Y}_0\hat{Z}_0$ is $(l_3,0,0)$. The plane where the origin of each coordinate system is located is designated as the operating arm plane. The position of the ARA shown in the figure corresponds to joint vector $\Theta=(0,-90^\circ,90^\circ,90^\circ,0)$. Table 1 describes symbols used in the text.

Table 1. Symbol explanation.

Symbol	Explanation
${}^{i-1}_{i}T$	Transformation matrix of coordinate system $\{i\}$ relative to coordinate system $\{i-1\}$
$c\theta_i$ or c_i	Abbreviation of $\cos(\theta_i)$

$s\theta_i$ or s_i	Abbreviation of $\sin(\theta_i)$
c_{12}	$c_{12} = \cos(\theta_1 + \theta_2)$
s_{12}	$s_{12} = \sin(\theta_1 + \theta_2)$
α_{i-1}	Angle of rotation from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ around axis $\hat{X}_{i-1}$
a_{i-1}	Distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ along axis $\hat{X}_{i-1}$
d_i	Distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ along axis $\hat{Z}_i$
θ_i	Angle of rotation from $\hat{X}_{i-1}$ to $\hat{X}_i$ around axis $\hat{Z}_i$

Calculate the linkage transformation matrix based on the D-H parameters in Table 2.

Table 2. Connecting rod parameters of ARA.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	−90°	0	0	θ_2
3	0	l_2	0	θ_3
4	0	l_3	0	θ_4
5	90°	0	0	θ_5

$$
{}_1^0T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}_2^1T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_2 & -c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}_3^2T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & l_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$
$$
{}_4^3T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & l_3 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}_5^4T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}
$$

The linkage transformation matrix ${}_5^0T$ is multiplied by the matrices of Equation (13):

$$
{}_5^0T = {}_1^0T\, {}_2^1T\, {}_3^2T\, {}_4^3T\, {}_5^4T \tag{14}
$$

$$
{}_5^0T == \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}
$$

$$\begin{aligned}
r_{11} &= c_1 c_{234} c_5 - s_1 s_5 \\
r_{21} &= s_1 c_{234} c_5 + c_1 s_5 \\
r_{31} &= -s_{234} c_5 \\
r_{12} &= -c_1 c_{234} s_5 - s_1 c_5 \\
r_{22} &= -s_1 c_{234} s_5 + c_1 c_5 \\
r_{32} &= s_{234} s_5 \\
r_{13} &= c_1 s_{234} \\
r_{23} &= s_1 s_{234} \\
r_{33} &= c_{234} \\
p_x &= c_1 \left(l_2 c_2 + l_3 c_{23} \right) \\
p_y &= s_1 \left(l_2 c_2 + l_3 c_{23} \right) \\
p_z &= -l_2 s_2 - l_3 s_{23}
\end{aligned} \tag{16}$$

By the kinematic equations of the ARA, the position and orientation of the end effector coordinate system can be calculated from the joint vectors. In order to simplify the solution and avoid the occurrence of multiple solutions, this study chose to use partial algebraic and partial geometric solutions for the inverse kinematic solution of the ARA.

$$\begin{cases}
\theta_1 = A\tan 2\left(p_y, p_x \right) \\
\theta_2 = -A\tan 2\left(p_z, \sqrt{{p_x}^2 + {p_y}^2} \right) - A\tan 2\left(l_3 \sin\theta_3, l_2 + l_3 \cos\theta_3 \right) \\
\theta_3 = A\tan 2\left(\sqrt{1 - \cos^2\theta_3}, \cos\theta_3 \right) \\
\theta_4 = \theta_{234} - \theta_2 - \theta_3 \\
\theta_5 = A\tan 2\left(r_{21} c_1 - r_{11} s_1, r_{22} c_1 - r_{12} s_1 \right)
\end{cases} \tag{17}$$

Among them, function $A\tan 2(y, x)$ is a bivariate arctangent function, and when calculating $\tan^{-1}\left(\dfrac{y}{x}\right)$, the quadrant where the angle is located can be determined based on the signs of x and y. θ_{234} and $\cos\left(\theta_3\right)$ are, respectively,

$$\theta_{234} = A\tan 2\left(r_{13} c_1 + r_{23} s_1, r_{33} \right) \tag{18}$$

$$\cos\left(\theta_3\right) = \frac{{p_x}^2 + {p_y}^2 + {p_z}^2 - {l_2}^2 - {l_3}^2}{2 l_2 l_3} \tag{19}$$

3.1.2. ARA Dynamic Analysis

Use Lagrange method to dynamically model the designed ARA. The joint position variable set $q_i (i = 1, 2, \cdots, 5)$ of ARA is defined as a generalized coordinate system, and its Lagrange function is defined as

$$\begin{cases}
L = T - U \\
T = \sum_{i=1}^{5} T_i \\
U = \sum_{i=1}^{5} U_i
\end{cases} \tag{20}$$

Among them, L is the Lagrange function; T and U are the total kinetic energy and total potential energy of the obstacle-clearing operation mechanism, respectively; and T_i and U_i are the kinetic energy and potential energy of the first link of the robotic arm, respectively. Due to the fact that link 4 and link 5 are located at the end of the robotic arm, only the kinetic and potential energies of the first three links are considered. The total kinetic energy is

$$\begin{aligned} T = &\frac{1}{2}I_1\dot{q}_1^2 + \frac{1}{2}m_2r_2^2\dot{q}_1^2\cos^2(q_2) + \frac{1}{2}m_2r_2{}^2\dot{q}_2^2 \\ &+\frac{1}{2}m_3\dot{q}_1^2\left(a_2\sin(q_2)+r_3\sin(q_2+q_3)\right)^2 + \frac{1}{2}m_3a_2^2\dot{q}_2^2 + \frac{1}{2}m_3r_3^2\left(\dot{q}_2+\dot{q}_3\right)^2 + m_3a_2r_3\left(\dot{q}_2^2+\dot{q}_2\dot{q}_3\right) \end{aligned} \tag{21}$$

The total potential energy is

$$U = m_2gr_2 - m_2gr_2\cos(q_2) + m_3ga_2 - m_3ga_2\cos(q_2) + m_3gr_3 - m_3gr_3\cos(q_2+q_3) \tag{22}$$

Among them, I_i is the inertia tensor of joint i; $\dot{q}_i$ is the generalized velocity of joint i; m_i is the mass of joint i; a_i is the length of connecting rod i; r_i is the distance from the centroid of joint i to the joint axis; and g is the acceleration due to gravity.

The Lagrange equation can be expressed as Equations (23) and (24):

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau \tag{23}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} + \frac{\partial U}{\partial q} = \tau \tag{24}$$

Among them, τ is the joint torque, and τ_i is the joint torque at joint i.

Calculate the relevant partial derivatives as follows:

$$\frac{\partial T}{\partial \dot{q}} = \begin{bmatrix} \left[I_1 + m_2r_2^2\cos^2(q_2) + m_3\left(a_2\sin(q_2)+r_3\sin(q_2+q_3)\right)^2\right]\dot{q}_1 \\ \left(m_2r_2^2 + m_3a_2^2 + m_3r_3^2 + 2m_3a_2r_3\right)\dot{q}_2 + \left(m_3r_3^2 + m_3a_2r_3\right)\dot{q}_3 \\ \left(m_3r_3^2 + m_3a_2r_3\right)\dot{q}_2 + m_3r_3^2\dot{q}_3 \end{bmatrix} \tag{25}$$

$$\frac{\partial T}{\partial q} = \begin{bmatrix} 0 \\ \frac{1}{2}\left(m_3\dot{q}_1^2a_2^2 - m_2r_2^2\dot{q}_1^2\right)\sin(2q_2) + m_3\dot{q}_1^2a_2r_3\sin(2q_2+q_3) + \frac{1}{2}m_3\dot{q}_1^2r_3^2\sin(2q_2+2q_3) \\ m_3\dot{q}_1^2a_2r_3\sin(q_2)\cos(q_2+q_3) + m_3\dot{q}_1^2r_3^2\sin(q_2+q_3)\cos(q_2+q_3) \end{bmatrix} \tag{26}$$

$$\frac{\partial U}{\partial q} = \begin{bmatrix} 0 \\ m_2gr_2\sin(q_2) + m_3ga_2\sin(q_2) + m_3gr_3\sin(q_2+q_3) \\ m_3gr_3\sin(q_2+q_3) \end{bmatrix} \tag{27}$$

Obtain the dynamic equation of the ARA:

$$\begin{aligned} \tau_1 = &\left[I_1 + m_2r_2^2\cos^2(q_2) + m_3\left(a_2\sin(q_2)+r_3\sin(q_2+q_3)\right)^2\right]\ddot{q}_1 \\ &+\left[\left(2m_3a_2\sin(q_2) + 2m_3r_3\sin(q_2+q_3)\right)\left(a_2\cos(q_2)+r_3\cos(q_2+q_3)\right) - m_2r_2^2\sin(2q_2)\right]\dot{q}_1\dot{q}_2 \\ &+\left[2m_3a_2r_3\cos(q_2+q_3)\sin(q_2) + 2m_3r_3^2\cos(q_2+q_3)\sin(q_2+q_3)\right]\dot{q}_1\dot{q}_3 \end{aligned} \tag{28}$$

$$\begin{aligned}\tau_2 &= \left(m_2 r_2^2 + m_3 a_2^2 + m_3 r_3^2 + 2m_3 a_2 r_3\right)\ddot{q}_2 + \left(m_3 r_3^2 + m_3 a_2 r_3\right)\ddot{q}_3 \\ &\quad -\frac{1}{2}\left(m_3 \dot{q}_1^2 a_2^2 - m_2 r_2^2 \dot{q}_1^2\right)\sin\left(2q_2\right) - m_3 \dot{q}_1^2 a_2 r_3 \sin\left(2q_2 + q_3\right) - \frac{1}{2} m_3 \dot{q}_1^2 r_3^2 \sin\left(2q_2 + 2q_3\right) \\ &\quad + \left(m_2 r_2 + m_3 a_2\right) g \sin\left(q_2\right) + m_3 r_3 g \sin\left(q_2 + q_3\right)\end{aligned} \tag{29}$$

$$\begin{aligned}\tau_3 &= \left(m_3 r_3^2 + m_3 a_2 r_3\right)\ddot{q}_2 + m_3 r_3^2 \ddot{q}_3 \\ &\quad - m_3 \dot{q}_1^2 a_2 r_3 \sin\left(q_2\right)\cos\left(q_2 + q_3\right) - \frac{1}{2} m_3 \dot{q}_1^2 r_3^2 \sin\left(2q_2 + 2q_3\right) + m_3 r_3 g \sin\left(q_2 + q_3\right)\end{aligned} \tag{30}$$

Simplify the dynamic equation and obtain the matrix form of the dynamic equation for the ARA:

$$\tau = D(q)\ddot{q} + B(q,\dot{q}) + G(q) \tag{31}$$

Among them,

$$D(q) = \begin{bmatrix} I_1 + m_2 r_2^2 \cos^2\left(q_2\right) + m_3\left(a_2 \sin\left(q_2\right) + r_3 \sin\left(q_2 + q_3\right)\right)^2 & 0 & 0 \\ 0 & m_2 r_2^2 + m_3 a_2^2 + m_3 r_3^2 + 2m_3 a_2 r_3 & m_3 r_3^2 + m_3 a_2 r_3 \\ 0 & m_3 r_3^2 + m_3 a_2 r_3 & m_3 r_3^2 \end{bmatrix} \tag{32}$$

$$B(q,\dot{q}) = \begin{bmatrix} \left[\left(2m_3 a_2 \sin\left(q_2\right) + 2m_3 r_3 \sin\left(q_2 + q_3\right)\right)\left(a_2 \cos\left(q_2\right) + r_3 \cos\left(q_2 + q_3\right)\right) - m_2 r_2^2 \sin(2q_2)\right]\dot{q}_1 \dot{q}_2 \\ + \left[2m_3 a_2 r_3 \cos\left(q_2 + q_3\right)\sin\left(q_2\right) + 2m_3 r_3^2 \cos\left(q_2 + q_3\right)\sin\left(q_2 + q_3\right)\right]\dot{q}_1 \dot{q}_3 \\ -\frac{1}{2}\left(m_3 \dot{q}_1^2 a_2^2 - m_2 r_2^2 \dot{q}_1^2\right)\sin\left(2q_2\right) - m_3 \dot{q}_1^2 a_2 r_3 \sin\left(2q_2 + q_3\right) - \frac{1}{2} m_3 \dot{q}_1^2 r_3^2 \sin\left(2q_2 + 2q_3\right) \\ -m_3 \dot{q}_1^2 a_2 r_3 \sin\left(q_2\right)\cos\left(q_2 + q_3\right) - \frac{1}{2} m_3 \dot{q}_1^2 r_3^2 \sin\left(2q_2 + 2q_3\right) \end{bmatrix} \tag{33}$$

$$G(q) = \begin{bmatrix} 0 \\ \left(m_2 r_2 + m_3 a_2\right) g \sin\left(q_2\right) + m_3 r_3 g \sin\left(q_2 + q_3\right) \\ m_3 r_3 g \sin\left(q_2 + q_3\right) \end{bmatrix} \tag{34}$$

3.2. Network Architecture Design

Based on the preparation work analyzed by the above institutions, this study used the open source reinforcement learning library SB3 built on the pytorch framework for simulation. The real-time position $\left(p_{ax}, p_{ay}, p_{az}\right)$ and velocity $\left(\dot{p}_{ax}, \dot{p}_{ay}, \dot{p}_{az}\right)$ of the ARA were obtained in the workbench coordinate system through the ARA power system. Then, the ARA state is fed into the input of the Eg-ac algorithm. The output behavior $a = \left[\ddot{p}_{ax}, \ddot{p}_{ay}, \ddot{p}_{az}\right]$ describes the position control quantity under system control, and its learning process is the iterative strategy function solving process. The ARA power system will provide current status data and determine reward values at each time step.

The Eg-ac algorithm is based on the actor–critic algorithm, so this article refers to the processing method in DDPG. Target actor and target critic networks were added with the same structure as the actor and critic networks in the algorithm. The architectures of the actor and critic networks are shown in Figure 5. This study designed a critic with three hidden layers. The hidden layers for the processing states and behaviors are first operated separately, and the value of the state behavior is output by fully connecting them together through the last hidden layer. The input received by the actor network is the number of ARA features, and the specific value of each behavioral feature is output to the ARC module. The actor network is designed with three hidden layers, fully connected between layers. In order to achieve exploration, the algorithm adds a random noise with a mean of 0 to the generated behavior, allowing it to explore a certain range around the exact behavior. To prevent overfitting during the optimization process, the dropout algorithm is introduced to regularize the network. The dropout algorithm was proposed by Hinton [36] in 2012. In each training batch, through ignoring half of the feature detectors (setting half of the hidden layer nodes to 0), it can effectively alleviate the occurrence of overfitting and achieve regularization to a certain extent.

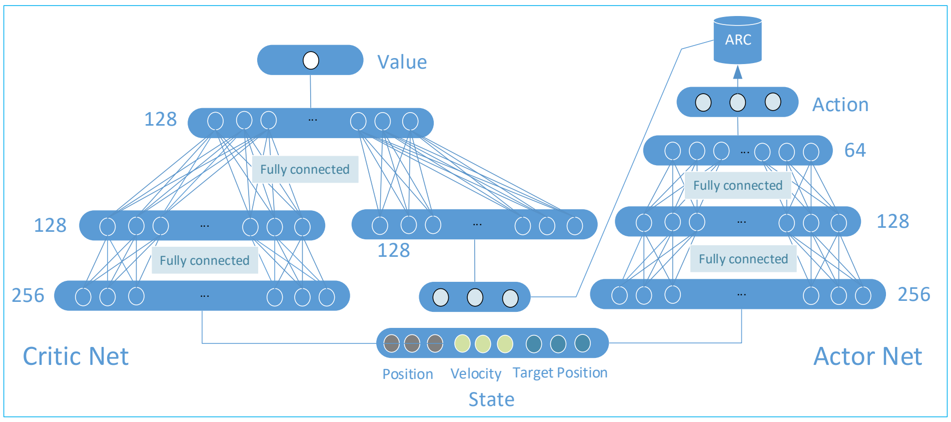


Figure 5. Network architecture settings in simulation.

Before simulation, the following parameters were set as follows:

(1) The definition of the reward value is the negative value of the distance between the end of the ARA and the target position, which means that the larger the error, the lower the reward value.

$$r_i = -\sqrt{(p_{tx} - p_{ax})^2 + (p_{ty} - p_{ay})^2 + (p_{tz} - p_{az})^2} \tag{35}$$

(2) Each training batch is executed $\Gamma = 256$ times. When there is an error $e \le 0.05$ between the end of the ARA task and the target position during an experience, the task is considered completed, and the current experience is stopped and reset to enter the next experience.

(3) Let the simulation time step be t and the total training steps be $\mathrm{N} = 20000$.

(4) Set learning rates $\varsigma = 0.001$, $\gamma = 0.99$, and batch sampling quantity $X = 100$.

(5) Controller parameters in the encourager: $\theta^E = [0.5\ 0.05]$.

(6) Output range setting: $Q(s, a \mid \theta^Q) \in (-10, 10)$, $\mu(s \mid \theta^\mu) \in (-0.05, 0.05)$, $E(s \mid \theta^E) \in (-0.05, 0.05)$.

(7) Set the random noise range to $g(t) \in (-0.002, 0.002)$.

(8) The size of the ARA connecting rod is $l_2 = 0.2(m), l_3 = 0.15(m)$.

(9) To ensure that the pitch and roll angles of the ARA end remain unchanged, and to make the end face of joint 5 perpendicular to the plane of the operating arm, the following settings are used: $\theta_4 = -(\theta_2 + \theta_3)$, $\theta_5 = 0$.

(10) The adoption rate parameter of the ARC is composed of a set of random numbers $\eta_1,\eta_2\cdots\eta_5\left(\eta_i \le 30\%, i=1,2,\cdots 5\right)$ with five elements generated by the computer. The process diagram is shown in Figure 6. In this process, the input variables are first subjected to inverse kinematics (IK) to obtain the joint positions $\theta_i\left(i=1,2,\cdots 5\right)$ of the ARA and then integrated into $\theta_i^{E\oplus\mu}\left(i=1,2,\cdots 5\right)$ through adoption rate calculation. Finally, the final behavior is output through kinematic calculations (KCs). Among them, $\theta_i^{E\oplus\mu}=\eta_i\theta_i^E+\left(1-\eta_i\right)\theta_i^\mu \quad \left(i=1,2,\cdots 5\right)$. Set $\eta_{ARC}\to 0$ when the training steps are $t \ge 3000$.

(11) The simulated computer configuration used an Intel(R) Core (TM) i5-7300HQ (ASUS, Taipei, China).

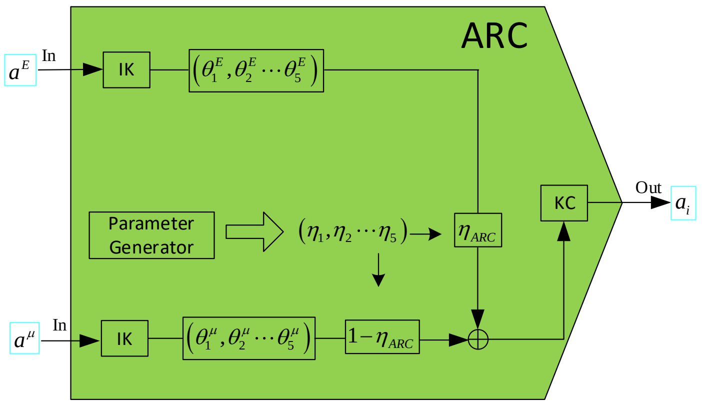

ARC
a^E
In
IK
$\left(\theta_1^E,\theta_2^E\cdots\theta_5^E\right)$
Parameter Generator
$\left(\eta_1,\eta_2\cdots\eta_5\right)$
η_{ARC}
KC
Out
a_i
a^μ
In
IK
$\left(\theta_1^\mu,\theta_2^\mu\cdots\theta_5^\mu\right)$
$1-\eta_{ARC}$

Figure 6. Schematic of ARC processing process.

4. Simulation Results and Analysis

The ARA was trained using the SAC, DDPG, and Eg-ac algorithms. During the process, every four episodes were grouped together, and the reward values obtained for each episode were recorded and averaged as an average reward value, as shown in Figure 7. For clarity and intuitiveness, the green boxes in the images represent enlarged areas. The trend of the reward episode lines for the three algorithms in the figure is basically the same, but compared with the three phases, Eg-ac has a faster learning efficiency and the fastest and most stable reward rise slope, followed by DDPG, and SAC converges the slowest and fluctuates slightly near convergence. When the reward values of the three algorithms reach their highest, the horizontal axis position shows that the learning speed of Eg-ac is at least 20% faster than that of the other two algorithms. The figure shows that for the same number of training steps, the numbers of episodes for the three algorithms are approximately as follows: SAC (4100), DDPG (4800), and Eg-ac (5800). The reason why Eg-ac has the most episodes is because it can quickly find the direction, reach the target position, and move on to the next experience with fewer steps in each episode. The same conclusion can also be drawn from the more intuitive table in Figure 8. In Figure 8, as mentioned above, every four episodes form a group, and the average number of time steps contained in each episode is taken as the average step count. It can be seen that Eg-ac has shown significant advantages in ARA training, as it can shorten learning time and converge quickly.

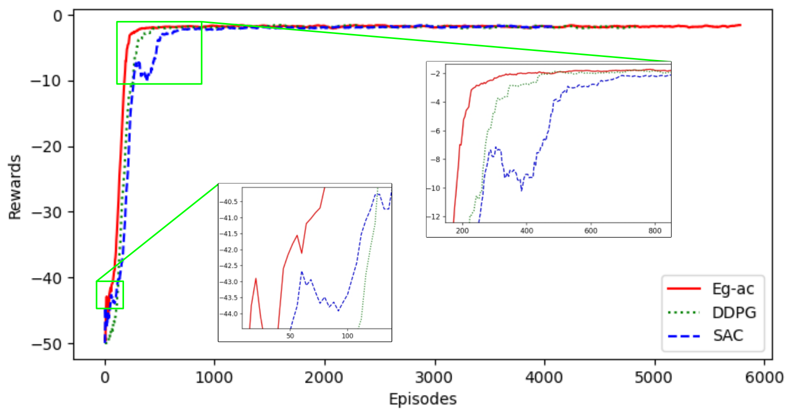


Figure 7. Comparison of rewards.

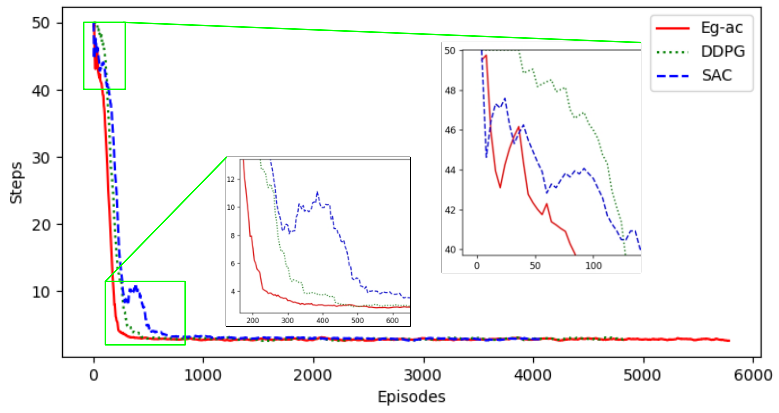


Figure 8. Comparison of episode steps.

To evaluate the training cost of the proposed algorithm, the training time and execution frequency of the three algorithms were recorded. Figure 9 shows the comparison of the training time. The training time consumed by the Eg-ac algorithm is between that of the other two algorithms, and the training time of the three algorithms is almost the same. The average number of steps taken per unit time was calculated, i.e., the execution frequency, in groups of four episodes. As shown in Figure 10, the large initial values are due to factors such as network and parameter initialization time. In terms of execution frequency, the DDPG algorithm has the highest number of training steps per unit time. SAC is the lowest, while Eg-ac values are between the two. The three sets of values are quite close, so it can be concluded that Eg-ac does not have significant differences in computational cost compared to the other algorithms, and the training time difference between the three is not significant, which is within an acceptable range.

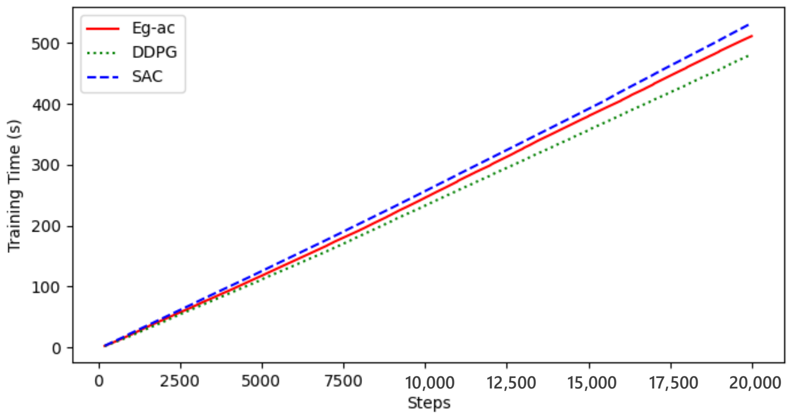


Figure 9. Comparison of training time.

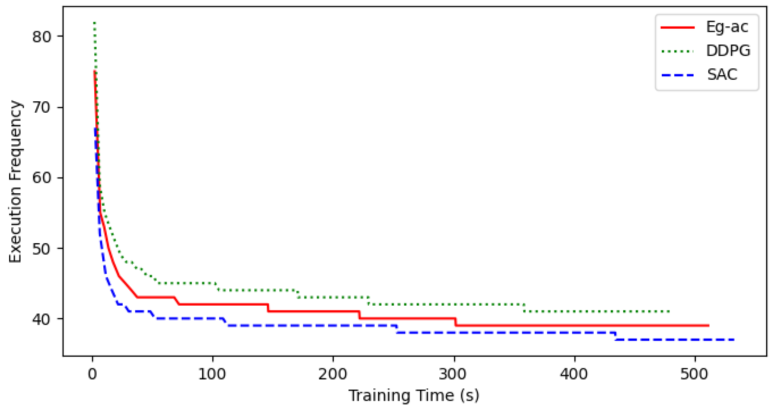


Figure 10. Execution frequency.

Due to the fact that both the Eg-ac and DDPG algorithms have two target networks in their network structures, while the critic loss function uses mean squared error (MSE) loss, the MSE loss will cause the prediction of the critic network to approach the target value, so its convergence speed is related to the training efficiency and learning effect of the algorithm. Figure 11 shows the loss comparison curves between the Eg-ac algorithm and the DDPG algorithm. The loss trend of both algorithms first increases and then decreases, with MSE loss decreasing and gradually lowering the loss value. The difference between the two curves is that the peak of Eg-ac is relatively earlier, indicating a faster reaction and a consistently lower loss than DDPG, ultimately remaining at around 0.1, while DDPG is around 0.2. It can be seen that the designs of the ARC module and HRLP reward mechanism in the Eg-ac algorithm effectively increase the training efficiency of the algorithm and improve the convergence speed by about 21.4%.

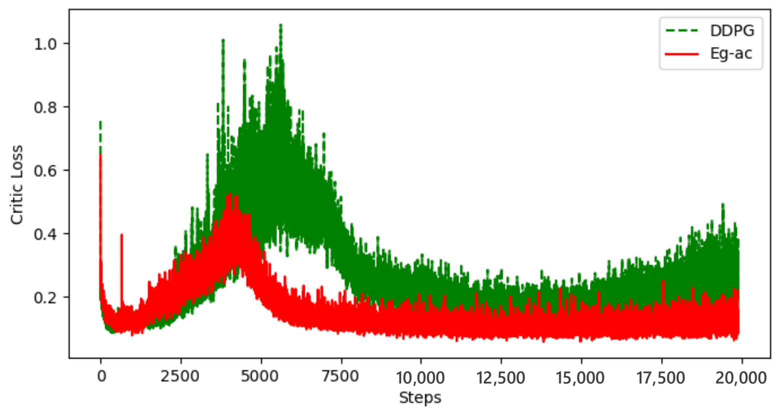


Figure 11. Loss curve of critic network.

Based on the above analysis, this study selected two sets of floating target trajectories for algorithm model validation experiments. Experiment 1 used a spatial spiral curve as the floating trajectory to simulate ARA tracking when the aircraft body shakes due to wind disturbance. Experiment 2 used a planar curve as the floating trajectory to simulate ARA tracking when the aircraft is stationary.

In Experiment 1, it is assumed that the floating trajectory of the obstacle to be cleared by the ARA under the dual effects of the base and wind disturbance is a spatial arc. The models were trained using three different algorithms, and the motion parameters of the ARA were recorded. Figure 12 shows the tracking trajectories after the execution of the three algorithms. It can be intuitively analyzed that the trajectory of SAC is the most unstable, with position jitter occurring during the tracking process, while the stability of the Eg-ac and DDPG algorithms was better. Figure 13 shows the tracking errors of three algorithms during execution. Analysis shows that Eg-ac has a smaller tracking error compared to DDPG, which is basically stable at around 0.01 m, while DDP is about 0.03 m. Figures 14 and 15, respectively, show the displacement and velocity curves of the ARA in the three coordinate axes of the workbench during Experiment 1. From the analysis of the simulation curve above, it can be seen that under the same number of training steps and training time, the algorithm proposed in this paper exhibits more accurate and stable tracking performance in the process of floating target tracking.

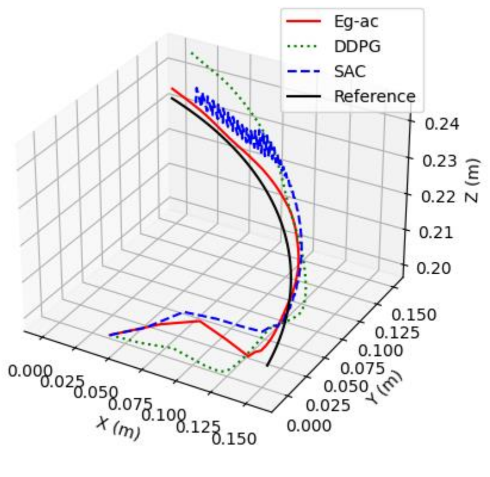


Figure 12. Three-dimensional curve of ARA tracking trajectory in Experiment 1.

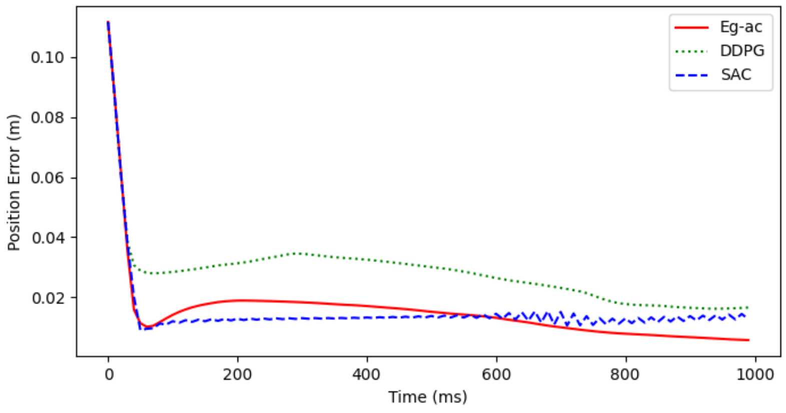


Figure 13. Tracking error curve in Experiment 1.

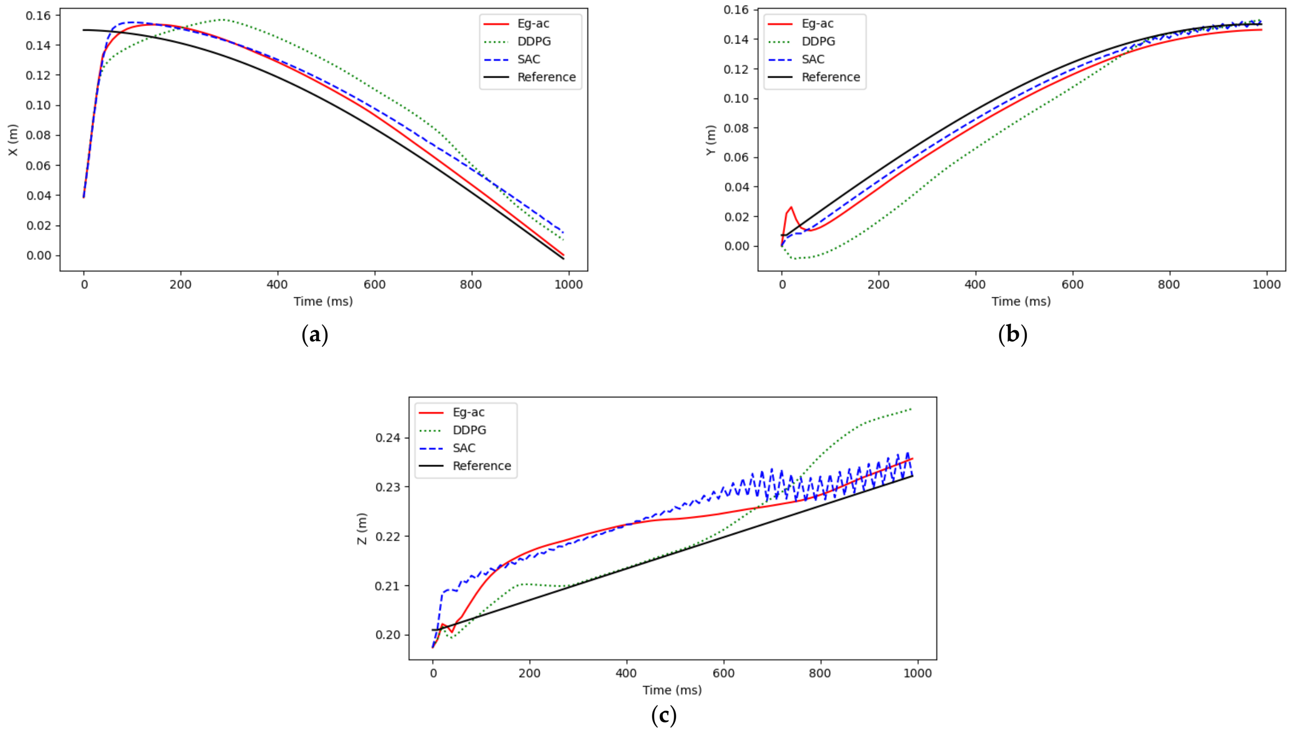
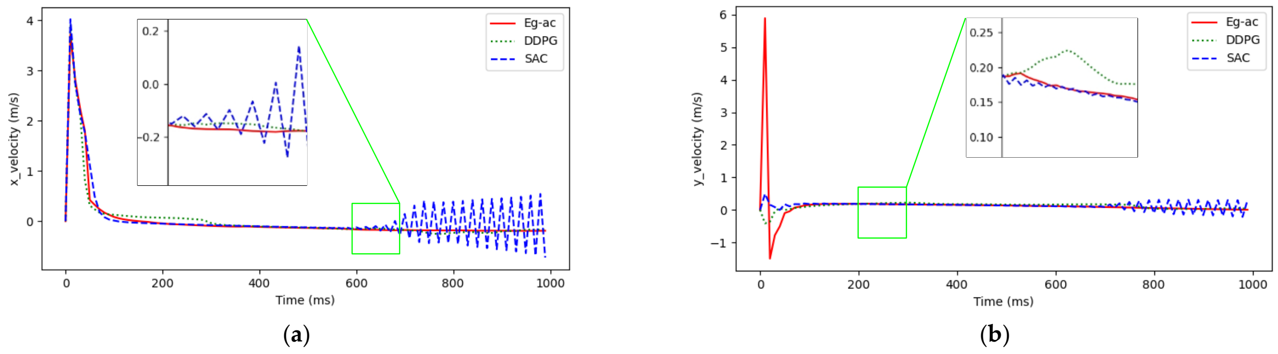
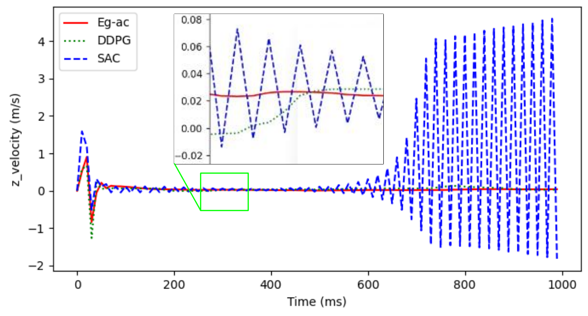
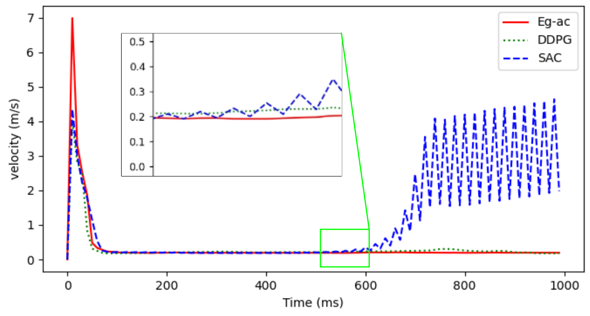


Figure 14. Displacement curves in each coordinate axis direction in Experiment 1: (**a**) X-axis displacement; (**b**) Y-axis displacement; (**c**) Z-axis displacement.





(**c**)



(**d**)

Figure 15. Tracking speed curve in Experiment 1: (**a**) X-axis velocity; (**b**) Y-axis velocity; (**c**) Z-axis velocity; (**d**) tracking linear velocity.

In Experiment 2, it is assumed that the obstacle to be cleared by the ARA will generate a floating trajectory as a planar arc without external disturbance. The models were trained using three different algorithms, and the motion parameters of the ARA were recorded. Figure 16 shows the tracking trajectories after the execution of the three algorithms. Figure 17 shows the tracking errors of three algorithms during execution. As shown in the figure, during the planar tracking process, the tracking error of Eg-ac is basically stable at around 0.01 m, while DDPG is about 0.02 m. Figures 18 and 19, respectively, show the displacement and velocity curves of the ARA in the three coordinate axes of the workbench during Experiment 2. Based on the analysis of the simulation curve above, it can be concluded that, similar to the conclusion of Experiment 1, the algorithm proposed in this paper exhibits more accurate and stable tracking performance in floating target tracking under the same number of training steps and training time.

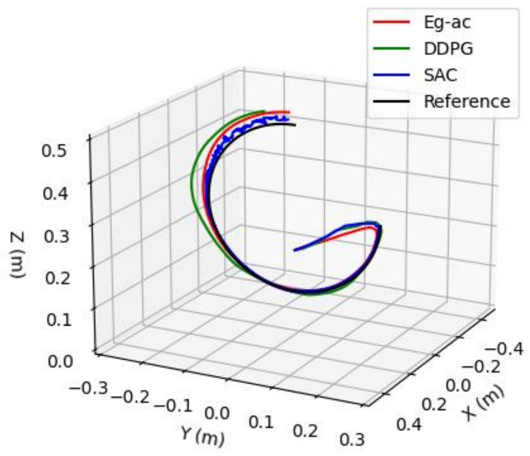


Figure 16. ARA tracking trajectory in Experiment 2.

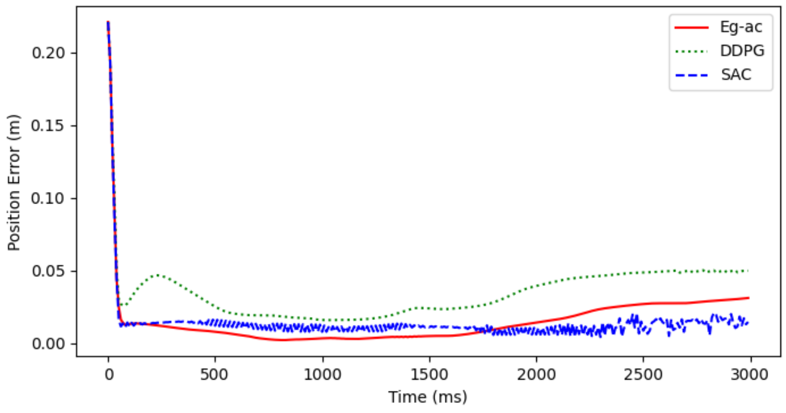
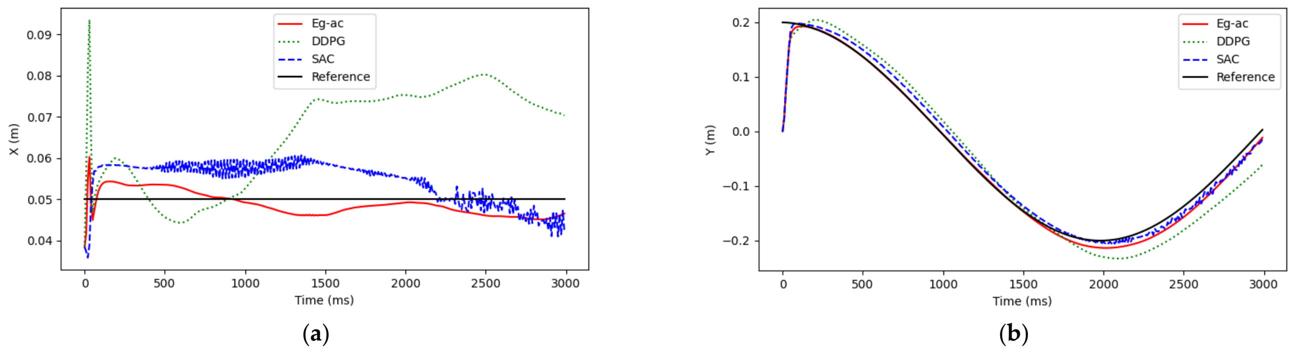


Figure 17. Tracking error curve in Experiment 2.



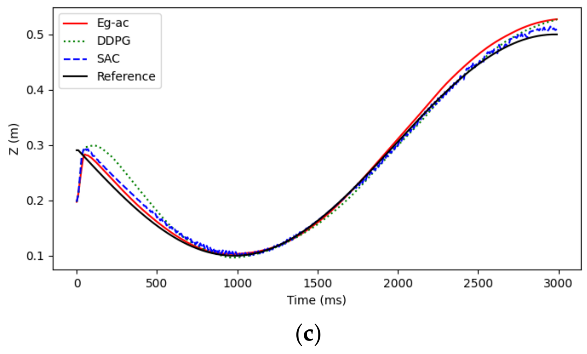
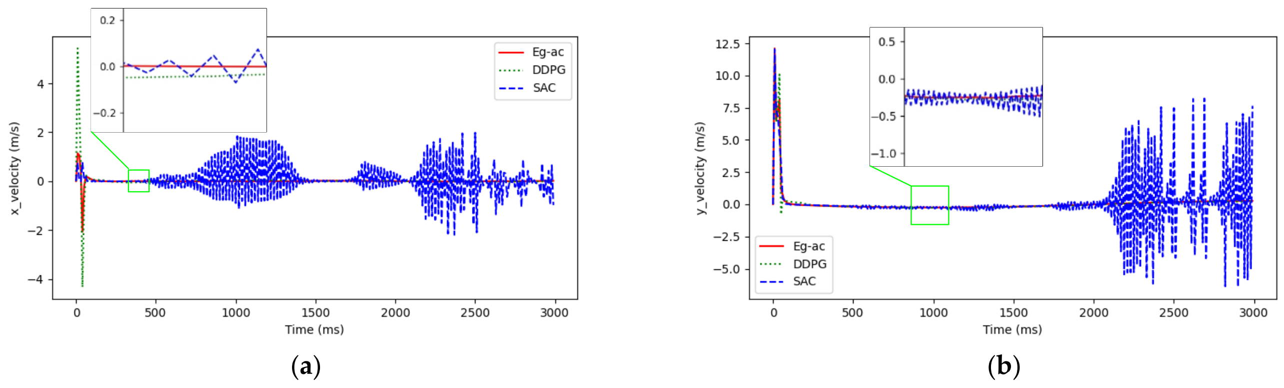
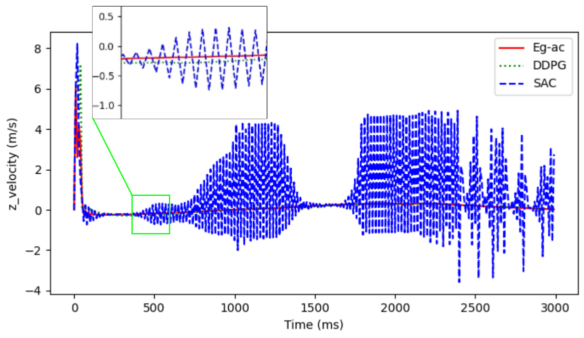
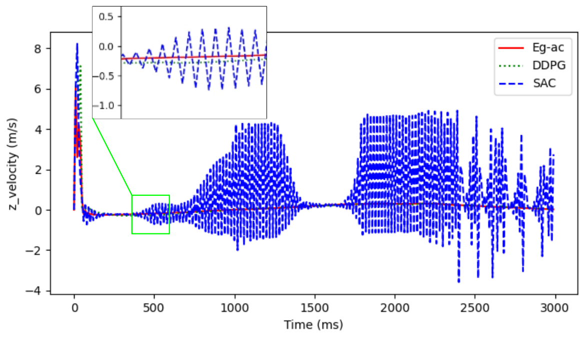


Figure 18. Displacement curves in each coordinate axis direction in Experiment 2: (**a**) X-axis displacement; (**b**) Y-axis displacement; (**c**) Z-axis displacement.





(**c**)



(**d**)

Figure 19. Tracking speed curve in Experiment 2: (**a**) X-axis velocity; (**b**) Y-axis velocity; (**c**) Z-axis velocity; (**d**) tracking linear velocity.

In order to more intuitively demonstrate the computational costs during the execution of the three algorithms, this study recorded the computation time during the execution of each algorithm. Figure 20 shows the partial single-step computation time of three algorithms. The average computation time of the three algorithms is $997(\mu s)$, which further proves that the algorithm proposed in this paper does not waste too much computational cost while improving learning efficiency and tracking performance.

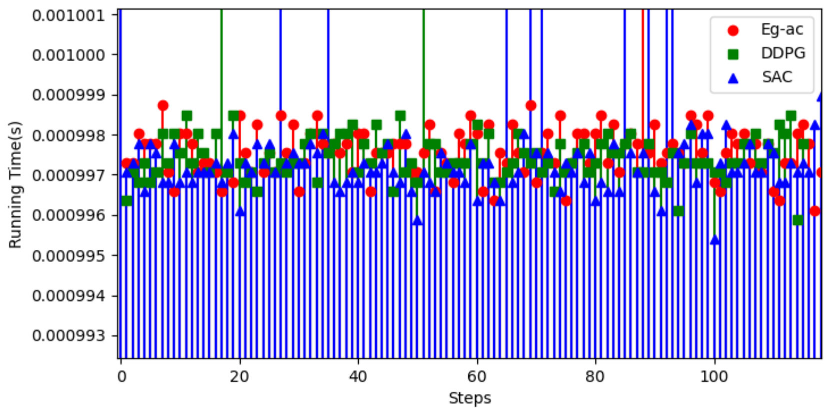


Figure 20. Comparison of running time.

5. Conclusions

This article proposes an Eg-ac algorithm based on the AC algorithm and applies it to the floating target tracking control of the ARA. The research objectives of the algorithm proposed in this paper were to quickly lock the exploration direction during the process of the ARA reaching the floating target position, improve learning efficiency, and obtain stable tracking results without increasing learning costs. Based on the above objectives, this study established approximate functions, strategy functions, and incentive functions for ARA state values in algorithm construction and designed an ARC module. Among them, the ARC generates the adoption rate for the encourager and outputs the ARA behavior strategy under the regulation of the ARC. Given that the inverse kinematic settlement and dynamic system execution of the ARA are required during the algorithm model training process, this paper establishes the kinematic and dynamic models of the ARA based on the D-H method. The target positions of each joint are obtained through inverse kinematic calculation, and the current state is obtained through the dynamic system. Finally, simulation was conducted using the open source reinforcement learning library SB3 built on the pytorch framework. The experimental results show that under the same computational cost, the loss function convergence speed of the Eg-ac algorithm designed in this study was improved by 21.4% compared to that of DDPG. Compared with SAC and DDPG, Eg-ac improved learning efficiency by at least 20% and has a more agile and stable floating target tracking performance.

While proposing a better algorithm in this article, there are some inevitable aspects that need improvement or can make the proposed algorithm better, such as the following: (1) Significant oscillation phenomena when approaching the target need to be improved upon. (2) The simulation did not consider the end effector attitude of the ARA. If it is necessary to consider the end effector attitude, research on attitude angles needs to be conducted. (3) There are various types of airborne disturbances, and in future work, it is necessary to further refine the disturbance effects and improve system stability. The authors will focus on addressing these areas in future research.

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

Funding: This research was funded in part by the Guangxi Power Grid Company's 2023 Science and Technology Innovation Project under Grant GXKJXM20230169 and in part by the Guizhou Provincial Science and Technology Projects, grant number Guizhou-Sci-Co-Supp [2020]2Y044.

Data Availability Statement: The data supporting the findings of this study are available within the article.

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

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Liao, L.; Yang, Z.; Wang, C.; Xu, C.; Xu, H.; Wang, Z.; Zhang, Q. Flight Control Method of Aerial Robot for Tree Obstacle Clearing with Hanging Telescopic Cutter. *Control Theory Appl*. **2023**, *40*, 343–352.
2. Wang, M.; Chen, Z.; Guo, K.; Yu, X.; Zhang, Y.; Guo, L.; Wang, W. Millimeter-Level Pick and Peg-in-Hole Task Achieved by Aerial Manipulator. *IEEE Trans. Robot.* **2024**, *40*, 1242–1260.
3. Kang, K.; Prasad, J.V.R.; Johnson, E. Active Control of a UAV Helicopter with a Slung Load for Precision Airborne Cargo Delivery. *Unmanned Syst.* **2016**, *4*, 213–226.
4. Amri bin Suhaimi, M.S.; Matsushita, K.; Kitamura, T.; Laksono, P.W.; Sasaki, M. Object Grasp Control of a 3D Robot Arm by Combining EOG Gaze Estimation and Camera-Based Object Recognition. *Biomimetic* **2023**, *8*, 208.
5. Villa, D.K.D.; Brandão, A.S.; Sarcinelli-Filho, M. A Survey on Load Transportation Using Multirotor UAVs. *J. Intell. Robot. Syst.* **2020**, *98*, 267–296.
6. Tagliabue, A.; Kamel, M.; Verling, S.; Siegwart, R.; Nieto, J. Collaborative transportation using MAVs via passive force control. In Proceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017.
7. Darivianakis, G.; Alexis, K.; Burri, M.; Siegwart, R. Hybrid Predictive Control for Aerial Robotic Physical Interaction towards Inspection Operations. In Proceedings of the IEEE International Conference on Robotics & Automation, Hong Kong, China, 31 May–7 June 2014.
8. Molina, J.; Hirai, S. Aerial pruning mechanism, initial real environment test. *Robot. Biomim.* **2018**, *7*, 127–132.
9. Roderick, W.R.T.; Cutkosky, M.R.; Lentink, D. Bird-inspired dynamic grasping and perching in arboreal environments. *Sci. Robot.* **2021**, *6*, eabj7562.

10. Sun, X. Application of Intelligent Operation and Maintenance Technology in Power System. *Integr. Circuit Appl.* **2023**, *40*, 398–399.
11. Zhang, Q.; Liao, L.; Xiao, S.; Yang, Z.; Chen, K.; Wang, Z.; Xu, H. Research on the aerial robot flight control technology for transmission line obstacle clearance. *Appl. Sci. Technol.* **2023**, *50*, 57–63.
12. Suarez, A.; Heredia, G.; Ollero, A. Physical-Virtual impedance control in ultralightweight and compliant Dual-Arm aerial manipulators. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2553–2560.
13. Zhang, G.; He, Y.; Dai, B.; Gu, F.; Yang, L.; Han, J.; Liu, G. Aerial Grasping of an Object in the Strong Wind: Robust Control of an Aerial Manipulator. *Appl. Sci.* **2019**, *9*, 2230.
14. Nguyen, H.; Lee, D. Hybrid force/motion control and internal dynamics of quadrotors for tool operation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013.
15. Zhong, H.; Miao, Z.; Wang, Y.; Mao, J.; Li, L.; Zhang, H.; Chen, Y.; Fierro, R. A Practical Visual Servo Control for Aerial Manipulation Using a Spherical Projection Model. *IEEE Trans. Ind. Electron*. **2020**, *67*, 10564–10574.
16. Alexis, K.; Huerzeler, C.; Siegwart, R. Hybrid predictive control of a coaxial aerial robot for physical interaction through contact. *Control. Eng. Pract.* **2014**, *32*, 96–112.
17. Zhuo, H.; Yang, Z.; You, Y.; Xu, N.; Liao, L.; Wu, J.; He, J. A Hierarchical Control Method for Trajectory Tracking of Aerial Manipulators Arms. *Actuators* **2024**, *13*, 333.
18. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529.
19. Hasselt, H.V.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. *arXiv* **2015**, arXiv.1509.06461.
20. Qiu, Z.; Liu, Y.; Zhang, X. Reinforcement Learning Vibration Control and Trajectory Planning Optimization of Translational Flexible Hinged Plate System. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108630.
21. Yang, A.; Chen, Y.; Naeem, W.; Fei, M.; Chen, L. Humanoid motion planning of robotic arm based on human arm action feature and reinforcement learning. *Mechatronics* **2024**, *7*, 102630.
22. Zhang, S.; Xia, Q.; Chen, M.; Cheng, S. Multi-Objective Optimal Trajectory Planning for Robotic Arms Using Deep Reinforcement Learning. *Sensors* **2023**, *23*, 5974.
23. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
24. Peters, J.; Schaal, S. Policy Gradient Methods for Robotics. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 2219–2225.
25. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 21–26 June 2014; pp. 387–395.
26. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; Levine, S. Soft Actor-Critic Algorithms and Applications. *arXiv* **2019**, arXiv:1812.05905.
27. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
28. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
29. Sekkat, H.; Tigani, S.; Saadane, R.; Chehri, A. Vision-based robotic arm control algorithm using deep reinforcement learning for autonomous objects grasping. *Appl. Sci.* **2021**, *11*, 7917.
30. Lindner, T.; Milecki, A.; Wyrwał, D. Positioning of the Robotic Arm Using Different Reinforcement Learning Algorithms. *Int. J. Control. Autom. Syst.* **2021**, *19*, 1661–1676.
31. Oikonomou, K.M.; Kansizoglou, I.; Gasteratos, A. A Hybrid Reinforcement Learning Approach with a Spiking Actor Network for Efficient Robotic Arm Target Reaching. *IEEE Robot. Autom. Lett.* **2023**, *8*, 3007–3014.
32. Song, B.Y.; Wang, G.L. A Trajectory Planning Method for Capture Operation of Space Robotic Arm Based on Deep Reinforcement Learning. *J. Comput. Inf. Sci. Eng.* **2024**, *24*, 091003-1.
33. Wu, P.; Su, H.; Dong, H.; Liu, T.; Li, M.; Chen, Z. An obstacle avoidance method for robotic arm based on reinforcement learning. *Ind. Robot* **2024**, *52*, 9–17.
34. Wu, J.; Yang, Z.; Liao, L.; He, N.; Wang, Z.; Wang, C. A State-Compensated Deep Deterministic Policy Gradient Algorithm for UAV Trajectory Tracking. *Machines* **2022**, *10*, 496.

35. Denavit, J.; Hartenberg, R.S. A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech.* **1955**, *22*, 215–221.
36. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing coadaptation of feature detectors. *Comput. Sci.* **2012**, *3*, 212–223.