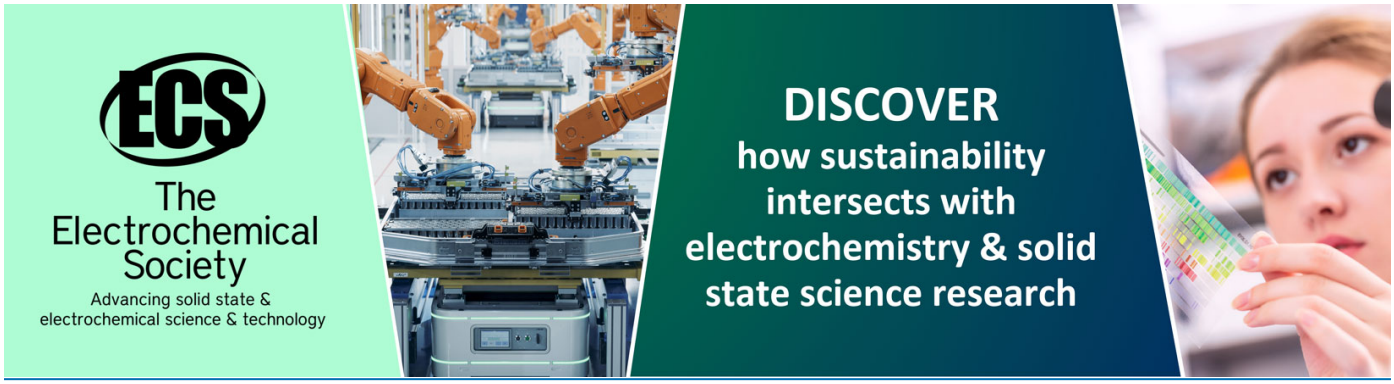
Application of Deep Reinforcement Learning in Control of Drawing Robots



Application of Deep Reinforcement Learning in Control of Drawing Robots

Cheng Liu[1], Liucun Zhu[1,*], Xinyu Ji[1] and Xiaodong Zheng[2]

[1]School of Information Engineering, Yangzhou University, Yangzhou, China
[2]School of Dept of Naval Architecture & Ocean Engineering , Beibu Gulf University, Guangxi, China

*Corresponding author e-mail: mz120180649@yzu.edu.cn

Abstract. In this paper, the drawing robot studied is after the system extracts the edge of the image, convert its pixels to actual physical coordinates, then control the robot to track these coordinate values and realize image drawing. In many cases, the edges extracted by the edge detection algorithm have a lot of unnecessary details and edge lines of many branch points. Here, some techniques are used to remove these redundant and branch points. When the robot is tracking the trajectory, the quality of its control algorithm directly determines its performance. In this paper, the control system is designed using the deep deterministic strategy gradient (DDPG) algorithm in reinforcement learning, focusing on the setting of its reward function. Here we propose a reward function. After simulation testing, the reward function setting meets the system requirements, the controller can achieve high-precision and high-speed trajectory tracking effect, and it can also suppress chattering caused by external interference and model errors.

1. Introduction

At present, there are a lot of research on humanoid robots. These robots not only increase the human-computer interaction, but also have a series of entertainment purpose, such as educational robots [1] and portrait drawing robots [2]. These robot application scenarios are inseparable from the precise control of the robot. More and more researchers are beginning to pay attention to the control of robots, especially their trajectory planning. In reference [3], the kinematic model of the two link manipulator is established by using the rotating coordinate system, and the trajectory planning and tracking are carried out. In reference [4], a hybrid fuzzy adaptive controller is designed to ensure the performance of robot trajectory tracking. However, in the design of actual robot controllers, it is difficult to design a good controller because the robot, as a dynamic system, has problems such as nonlinearity and parameter uncertainty, so there is a lot of literature about advanced control methods. In reference [5] for the limitation of traditional Jacobian Matrix Pseudo-Inverse (JMPI) method, a Robot Tracking Control Method based on Jacobian Matrix Adaptive (JMA) is proposed. Continuous control strategy based on nonlinear model proposed for flexible robots [6], response control strategy designed for human-machine interaction [7], in high-precision application scenarios, a synovial observer [8] and an ultra-high-order synovial controller [9] have been proposed. For the control scenario of uncertain kinematics and dynamics, an observer-based controller is proposed to achieve exponential tracking control [10].

In recent years, the wave of artificial intelligence has arrived. Artificial intelligence has been applied in various industries and has achieved good results. As a branch of artificial intelligence, reinforcement learning (RL) has also received great attention from researchers. Many researchers have applied reinforcement learning to the design of robot controllers, and have achieved good results. For example, a reinforcement learning control scheme based on neural network model is proposed in reference [11] for unknown parameters and nonlinear robot models，the RL algorithm based on interval fuzzy logic control proposed in reference [12] for rotary steering drilling overcomes the control problems caused by the complexity and uncertainty of control caused by the actual environment. A new policy iteration Q-learning algorithm is proposed in [13]. Literature [14] proposed a tracking algorithm for WMR system based on partial reinforcement learning neural network (PRLNN), in the nonlinear discrete dynamic robot system with slip, literature [15] proposed an adaptive neural tracking algorithm based on reinforcement learning.

Reinforcement learning has a very adaptable nature. After acquiring the system observations, it learns through rewards and punishments to make the robot learn how to perform the designed tasks correctly. When using reinforcement learning to track the robot's trajectory, use an appropriate strategy to map the state to the action, and when training the strategy, consider its convergence and convergence speed. Improper state representation may lead to the decrease of learning speed, and the system may not converge [16]. Another issue worth noting is the "dimensional disaster" problem, which exists in continuous control tasks. In [17], offline training is used to avoid this problem.

The main work done in this paper is as follows. First, extract the edge of the image, perform the preprocessing of branch removal, and then design the controller so that the robot can accurately draw the given image with minimum control force. In order to achieve the above goals, this paper proposes a reward function to optimize the strategy. On this basis, actor network and critic network are designed to train the DDPG agent, and this agent and reward function are used as the robot controller. The simulation results show that the method has good convergence speed and tracking performance, which indicates that the reward function can cooperate with the DDPG agent effectively, and the design of the controller is effective and feasible.

2. Background

2.1. Reinforcement learning

The process of reinforcement learning [18] is that the agent obtains observations from the environment, performs an action according to the strategy, and then gets a reward.

The goal is to design a strategy that maximizes the rewards it receives when it executes an action. The initial state s_0 is randomly sampled from the initial state distribution, and the action a_0 represents the initial action obtained by executing the strategy. The trajectory is described by the state-action pair, which is $\tau = \{s_0, a_0, s_1, a_1 \cdots\}$, in state s_0Go to state $s_1$can be deterministic，

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

or it can be random，

$$s_{t+1} \sim P(\cdot\,|s_t, a_t) \tag{2}$$

Where f(·) is the transfer function and P(·) represents the sampling probability. The agent performs actions according to its strategy and exerts influence on the environment.

In addition, the setting of the reward function is particularly important in reinforcement learning. The goal of the agent is to maximize the cumulative reward on a trajectory τ.

One kind of return is finite-horizon undiscounted return, and its reward accumulation value R (τ) is calculated as follows:

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{3}$$

Where r_t represents the reward value obtained at time t.

Another reward is infinite-horizon discounted return,

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{4}$$

Where $\gamma \in (0,1)$ represents the discount factor. As far as common sense is concerned, the current gain must be better than the past and the future, so it will not directly add up all the rewards.

Whatever the desired measurement and strategy, there is only one purpose to maximize the reward received. When discussing the expected return of the trajectory, first consider the probability distribution on the action trajectory. In the case where both the state transition and the strategy are random, the probability of a trajectory of length T is:

$$P(\tau|\pi) = \rho_0(s_0)\prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(s_t|a_t) \tag{5}$$

then the expected return is:

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = \underset{\tau\cdot\pi}{E}[R(\tau)] \tag{6}$$

The best strategy is the one that maximizes the expected return.

2.2. Deep Deterministic Policy Gradient (DDPG)

Deep deterministic Policy gradient [19] is an algorithm for learning Q function and strategy at the same time. Assuming the optimal Q-value function is $Q^*(s,a)$ and the optimal action function is $a^*(s)$, you need to learn $Q^*(s,a)$ and $a^*(s)$ approximators in DDPG. We construct a main neural network with $\emptyset$ parameter in DDPG to approximate the $Q^*(s,a)$ value, and then update the parameters of the secondary network with the trained parameters of this network to approximate $a^*(s)$.

The Bellman equation of the optimal action value function is as follows:

$$Q^*(s,a) = \underset{s'\cdot P}{E}[r(s,a) + \gamma \max_{a'} Q^*(s', a')] \tag{7}$$

Where $S' \sim P$ is shorthand for saying that the next state, s', is sampled by the environment from a distribution P (• | s, a). This Bellman equation is the starting point of the approximator, in other words, it is the initial state of the neural network we constructed. Now suppose that a set D of state transitions is obtained, and each state transition is represented as (s, a, r, d, s'), where d is the end flag. Then the loss function of the neural network is as follows:

$$L(\phi, D) = \underset{(s,a,r,s',d)\cdot D}{E}\left[\left(Q_\phi(s,a) - (r + \gamma(1-d)\max_{a'} Q_\phi(s', a'))\right)^2\right] \tag{8}$$

The parameter update formula for the target network is as follows:

$$\phi_{t\arg} \leftarrow \rho\phi_{t\arg} + (1-\rho)\phi \tag{9}$$

The loss function is：

$$L(\phi, D) = \underset{(s,a,r,s',d)\cdot D}{E}\left[\left(Q_\phi(s,a) - (r + \gamma(1-d)Q_{\phi_{t\arg}}(s', \mu\theta_{t\arg}(s')))\right)^2\right] \tag{10}$$

Since the action space of the drawing robot is continuous, and we assume the Q-function is differentiable with respect to action, we can just perform gradient ascent (with respect to policy parameters only) to solve. Another optimization technique of DDPG is to use a circular experience buffer to store past experience, and then the agent randomly obtains a small batch of experience from the buffer to update the critic and actor network.

3. Related Work

3.1. Image processing

When we try to make the robot draw a picture, we must first find the important feature points or edges of the image. Edge detection technology is an important image processing technology, and many edge detection algorithms [20] are available. We tried various algorithms, and finally found that the image edges extracted by the canny algorithm [21] meet the design requirements.

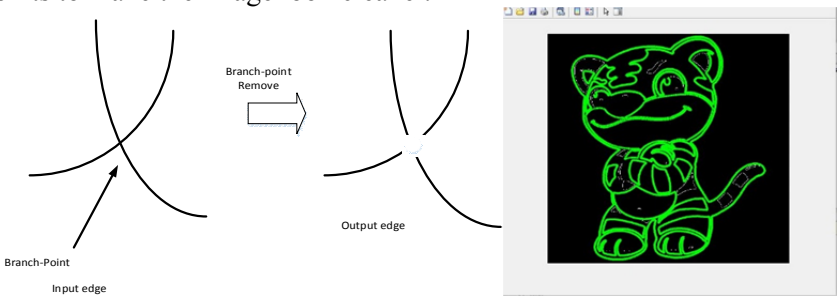
Figure 1. Output of Canny's Edge Detector.

3.2. Location generation

After acquiring the image edge points, the branch points need to be processed. In order for the arm to move smoothly from one end point to another, it is required that there should be no branch points on the edge line, so the branch points need to be removed as follows:

（1） After setting the threshold, use the corner detection algorithm [22] to detect the intersection between the image contours，change the pixel of the detected intersection from 1 to 0.

（2） Delete the edge lines that cover fewer areas and have fewer pixel points, which reduces the noise impact, prevents the robot from entering the singularity, and removes some unimportant feature points to make the image look cleaner.



（a） (b)

Figure 2. (a)Branch point processing, (b), the thick green line is the reserved edge line, and the thin white line is the removed edge line.

The next step is to determine the starting pose and the scale factor from which the pixels are converted to actual physical points. In addition, to simplify the control difficulty, the posture is fixed to $(0, \pi, 0)$ when the posture points are generated, and then the pixel points processed by the scale factor are combined with the posture points to form a posture matrix, so the posture points are generated.

3.3. Controller design based on reinforcement learning

In the design of the drawing robot controller, the parts outside the controller are treated as a black box. The controller obtains the desired joint position, the actual joint position and the torque from the black box, and then the controller outputs a torque to the black box.

The main purpose of this paper is to train a six-axis robot to use DDPG agent to complete image drawing. It is necessary to construct a critical network and an actor network here. The input of the critic network is observation and action, and the output is a gradient parameter. In this network, it mainly provides an internal estimate of the network performance. In an actor network, the input is observation and the output is action, its update method does not use weighted gradient update, but uses gradient ascent. The relationship between these two networks is like this: First, the environment gives an observation, the agent generates an action value (with noisy) according to the decision made by the actor network, and the environment will give a reward R after receiving this action. Then update the observations, use this reward R to update the critic network, and then update the actor network according to the direction suggested by the critic.

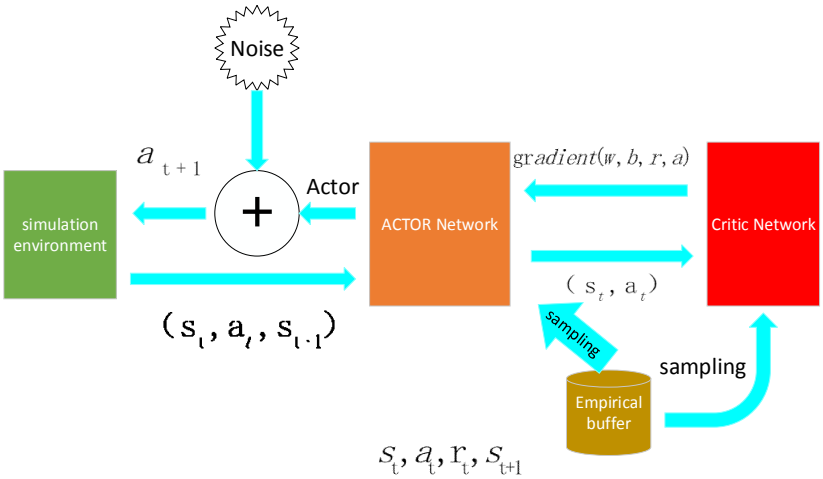


Figure 3. Controller frame.

3.3.1. Actor Network. In this network model, a 9-layer perceptual network is constructed. In the environment of a drawing robot, 27observations are provided, which are:

- Coordinate points of end effector.
- X (forward), Y (lateral), and Z (vertical) translation velocities.
- The angular position and velocity of six axes.
- Action values (torque for each joint) from the previous time step。

The output is the torque value of six joints.

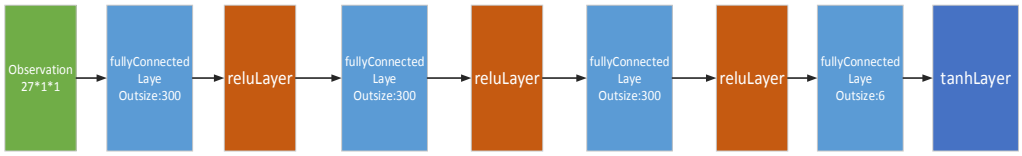


Figure 4. Actor Network.

3.3.2. Critic Network. In this network model, we construct an 11-layer neural network with inputs as observations and actions, and outputs as parameters of the Q (s, a) (Bellman equation of action-value function) approximator.

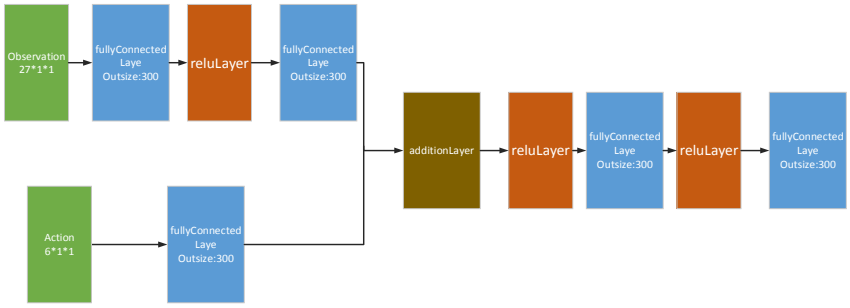


Figure 5. Critic Network.

3.3.3. episode terminates and return. The episode terminates if either of the following conditions occur.

- The error between the expected position and the actual position of each joint of the robot is between $\pm$ 0.5.

- The robot's end effector's expected and actual coordinates have an error of ±0.1 in the x, y, and z directions.

The reward function is the core of the control device. The setting of reward function directly affects the performance of the controller. Set e_1^i and $\dot{e}_1^i$ is the error of the i axis of the robot arm in the x direction and its derivative, e_2^i and $\dot{e}_2^i$ is the error of the i axis of the robot arm in the y direction and its derivative, v_x^i is the i axis of the robot arm in the x direction speed, v_y^i is the i axis of the robot arm in the y direction speed, T_s and T_f is the sampling time and the simulation deadline, u_{t-1}^i is the action value of the joint i at the previous time step. Considering that the main trajectory space of the drawing robot is two-dimensional, the end executor of the drawing robot can get a reward as soon as it reaches the specified position and posture point. Because we fixed the pose angle earlier, only the x-axis and y-axis values are considered when designing the reward function. According to continuous testing and parameter adjustment, the following reward function formula is given.

$$r_{\mathrm{t}} = \sum_{\mathrm{i}=1}^{6} v_x^i + \sum_{\mathrm{i}=1}^{6} v_y^i + 50\frac{T_s}{T_f} - 100\left(\sum_{\mathrm{i}=1}^{6}\left|e_1^i\right| + \sum_{\mathrm{i}=1}^{6}\left|e_2^i\right|\right) - 20\left(\sum_{\mathrm{i}=1}^{6}\left|\dot{e}_1^{\mathrm{i}}\right| + \sum_{\mathrm{i}=1}^{6}\left|\dot{e}_2^{\mathrm{i}}\right|\right) - 0.05\sum_{\mathrm{i}}(u_{\mathrm{t}-1}^i)^2 \tag{11}$$

This reward function enables the end effector to reach the desired position by providing a positive incentive for the x- and Y-direction speeds of the arm. In addition, to avoid premature termination of training, a fixed profit value（$50T_s/T_f$）is given in the reward function. Why add a decay rate to the cumulative value of past actions here? Intuitively speaking, past actions certainly have less influence on the current state than current actions. Mathematically, the sum of an infinite number of rewards will most likely not converge to a finite value. With decay rates and appropriate constraints, the value of an infinite sum will converge. Eliminate the negative effects of past actions on the current state by subtracting the accumulated amount of past actions. The rest of the formula is undesirable states, such as high errors and trends toward high errors.

3.4. Trajectory generation

We use standardized uniform B-spline [23] to fit the trajectory. Here is a simple definition of B-spline, there are n+1 vertices $p_0, p_1 \cdots p_n$, then the k-order (k-1 degree) B-spline curve is:

$$\mathrm{p}(t) = \sum_{i=0}^{n} p_i N_{i,k}(t) \tag{12}$$

It's here

$$N_{i,0}(u) = \begin{cases} 1 & u_i \le u \le u_{i+1} \\ 0 & otherwise \end{cases} \tag{13}$$

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u) \tag{14}$$

Set three 0 position points in front of the track point, and three repeated position points behind, indicating initialization and end. It takes the form of [0,0,0,0,1,2,3,……m-1,m,m,m,m]. The first three points and the last three points are denoted as $N_{0,3}(u)$, $N_{1,3}(u)$, $N_{2,3}(u)$, $N_{M+3,3}(u)$, $N_{M+4,3}(u)$, $N_{M+5,3}(u)$, here

$$N_{M+3,3}(u) = N_{2,3}(-u + \mathrm{M} - 1) \tag{15}$$

$$N_{M+4,3}(u) = N_{1,3}(-u + \mathrm{M} - 1) \tag{16}$$

$$N_{M+5,3}(u) = N_{0,3}(-u + \mathrm{M} - 1) \tag{17}$$

In this drawing system we use a third-order B-spline [24] to fit the trajectory. Total line curve representation:

$$\mathrm{s}_3(t) = \sum_{i=-2}^{0} c_i N_{i+2,3}(t) + \sum_{i=1}^{M} c_i B_3(t-i+1) + \sum_{i=M+1}^{M+3} c_i N_{i+2,3}(t) \tag{18}$$

c_i is a three-dimensional spatial coordinate point,$B_3(\cdot)$ is

$$B_3(u)\begin{cases}\frac{1}{6}u^3 & if \quad u \in [0,1)\\ \frac{1}{6}-\frac{1}{2}(u-1)^3+\frac{1}{2}(u-1)^2+\frac{1}{2}(u-1) & if \quad u \in [1,2)\\ \frac{1}{6}+\frac{1}{2}(u-3)^3+\frac{1}{2}(u-3)^2-\frac{1}{2}(u-3) & if \quad u \in [2,3)\\ -\frac{1}{6}(u-4)^3 & if \quad u \in [3,4)\end{cases} \tag{19}$$

4. Simulation

In the actual simulation test, the built neural network does not know the dynamics of the robot and the corresponding parameters. In this system, the controller treats the robot as a black box, and the agent interacts with the black box to find an optimal strategy to maximize the profits obtained.

This paper uses the PROBOT Anno robot model for simulation test. It is a 6-axis arm product based on ROS development. The physical parameters are as follows：

Table 1. Physical parameters of robotic manipulator.

Axis	Axis Length	Range of motion	Maximum uniaxial speed
First Axis	143.7mm	-180~180	200mm/s
Two Axis	152mm	-115~115	200mm/s
Third Axis	100mm	-130~130	200mm/s
Fourth Axis	125mm	-180~180	200mm/s
Fifth Axis	228.9mm	-165~165	200mm/s
Six Axis	55mm	-180~180	200mm/s
Total weight	15kg	Maximum working radius	482.3mm

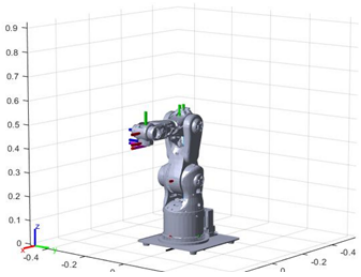


Figure 6. 6-axis robotic manipulator.

Before conducting agent training, some necessary settings need to be made, such as setting the maximum training episode to 6000, 10 iterations per episode, setting the initial weight randomly between (0,1), and setting the initial and end positions of the end executor randomly. Stop training if the average reward in consecutive 100 episodes exceeds 19, and save a copy of the agent when the average cumulative reward in the training set exceeds 20. After every 50 steps, each worker sends experience to the host. DDPG agents require workers to send 'Experiences' to the host, and perform a verification.

We give the loss function graphs of the two networks after the training episode number is 100 and the number of iterations is 1000. It can be seen that actor and critic networks converge rapidly after 100 iterations and stabilize after 400 iterations. In addition, the reward graph after 12,000 iterations after training 1200 sets is shown. It can be seen from the graph that as the number of training episode increases, the reward value is also increasing, and its Q value is also increasing, indicating that we set the reward function It meets the control requirements.

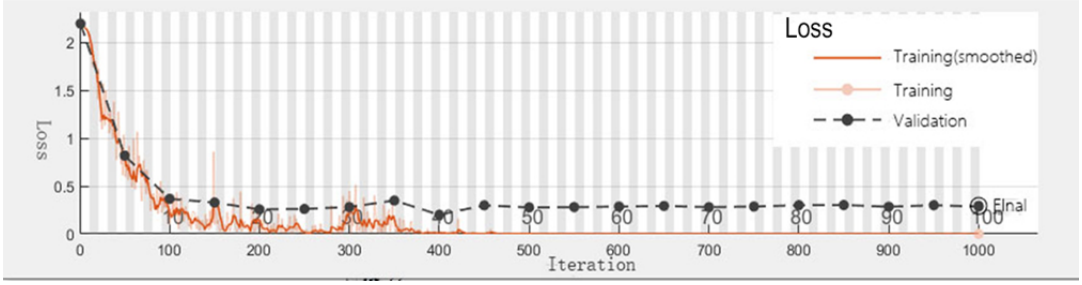


Figure 7. loss function of action network.

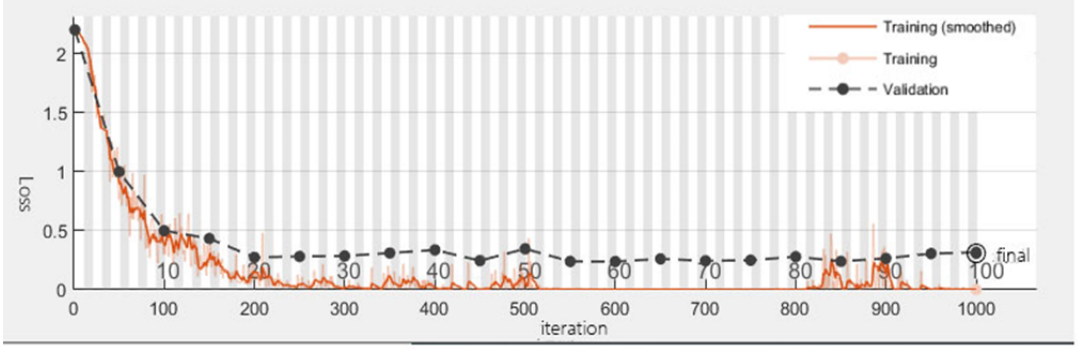


Figure 8. loss function of critic network.

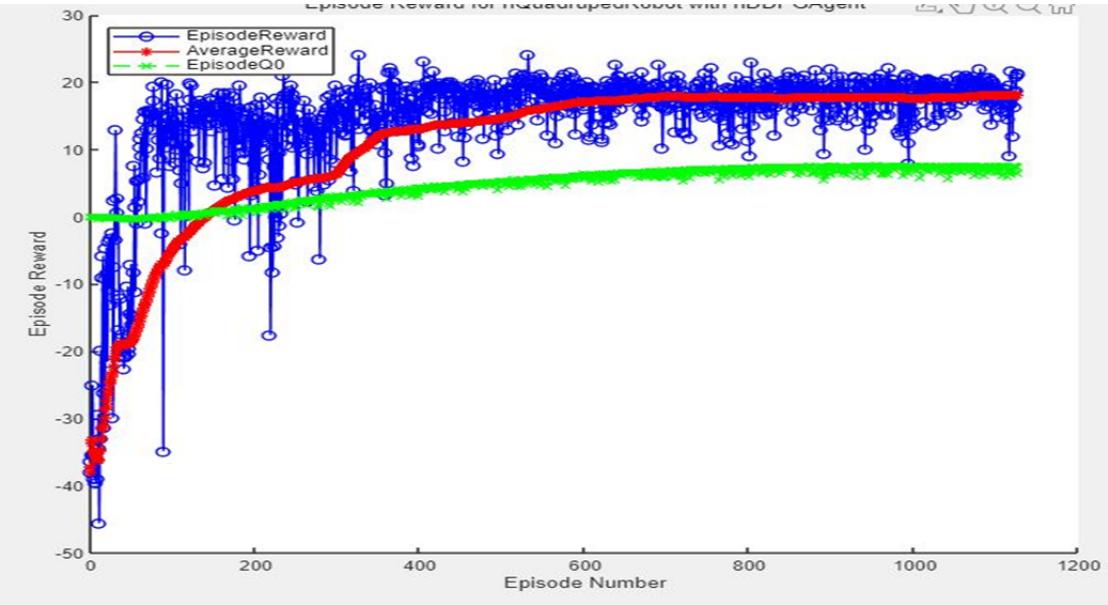


Figure 9. Episode Reward for Drawing robot with DDPG.

After successful learning, the agent can quickly control the robot arm to reach the desired position. Figure 10 shows the variation of the error between the actual position of the six axes and the expected position. As can be seen from the diagram, the agent can quickly reduce the error, adjust time is less, rise time is fast, and its control performance is stable. Finally, the working effect diagram of the drawing robot is given in Figure 11.

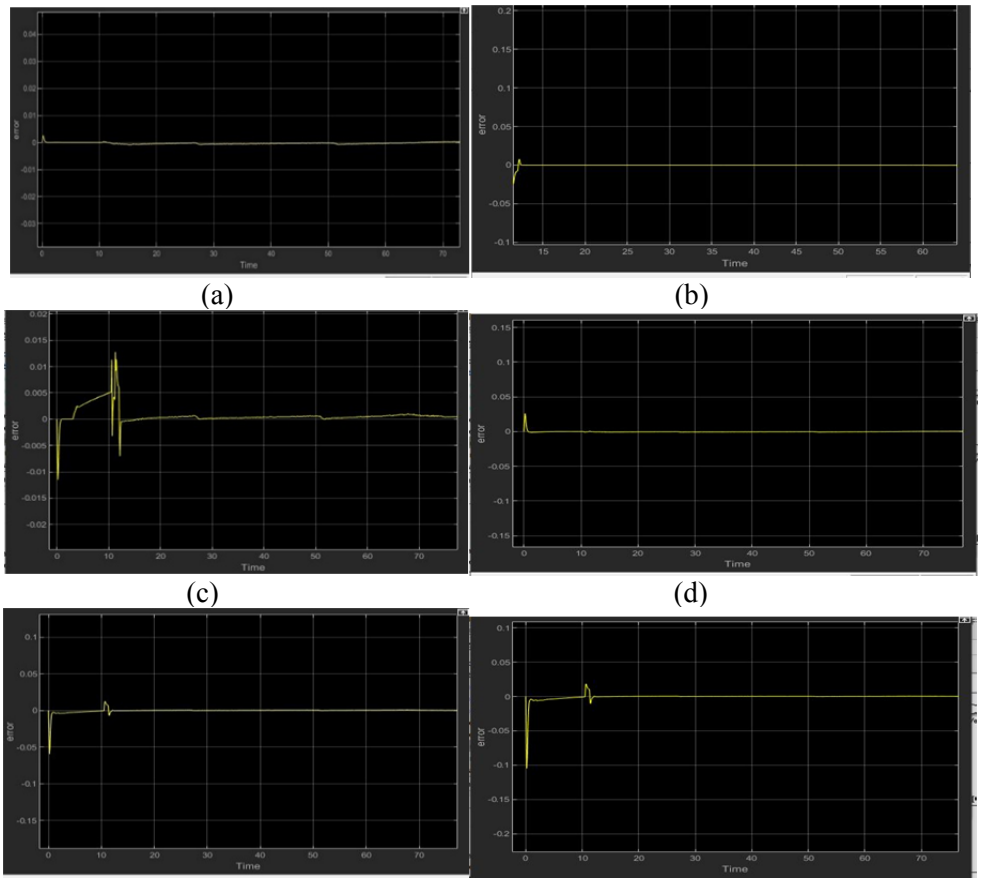


Figure 10. (a) is First Angle position error，(b) is Two Angle position error,(c) is Three Angle position error,(d) is Four Angle position error,(e) is Five Angle position error,(f) is six Angle position error.

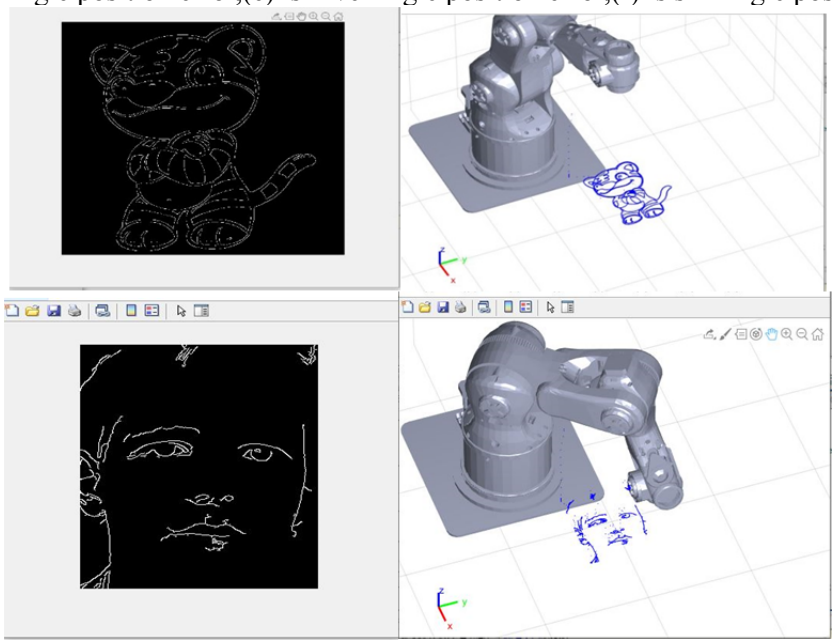


Figure 11. Images drawn by robots.

5. Conclusion

For the design of high-precision controller in drawing robot, this paper uses reinforcement learning to design the controller, which is different from traditional control methods. It can also design a good controller to complete the specified tasks without the control designer knowing the control algorithm and detailed parameters of the robot. In image processing, this paper uses some small tricks to process the input edges, such as removing branch points by transforming the pixel values, and automatically ignoring some edge lines with low coverage. On the controller, all modules except the controller are treated as black boxes, and the DDPG agent is trained to find the optimal strategy by getting the relevant observations from the black boxes. The action produced by this strategy is the optimal action. In addition, the setting of the reward function is one of the keys, which directly affects the performance of the controller. In this paper, with the goal of reducing the speed and position errors of six axes, a reward function is proposed. After simulation testing, it is found that it meets the control requirements.

References

[1] El-Barkouky A , Mahmoud A , Graham J , et al. An interactive educational drawing system using a humanoid robot and light polarization. 2013 20th IEEE International Conference on Image Processing (ICIP). IEEE, 2013.
[2] Jean-Pierre Gazeau, Said Zeghoul, The artist robot: a robot drawing like a human artist," IEEE International Conference on Industrial Technology, pp.486,491, 19-21 March 2012.
[3] Sehoon Oh and Kyoungchul Kong. Two-degree-of-freedom control of a two-link manipulator in the rotating coordinate system. IEEE Transactions on Industrial Electronics, 62(9):5598–5607, 2015.
[4] Ho H F , Wong Y K , Rad A B . Robust fuzzy tracking control for robotic manipulators. Simulation Modelling Practice and Theory, 2007, 15(7):801-816.
[5] Apoorva D. Kapadia, Ian D. Walker, Darren M. Dawson. A model-based sliding mode controller for extensible continuum robots. Wseas International Conference on Signal Processing. World Scientific and Engineering Academy and Society (WSEAS), 2010.
[6] Sato M , Toda M . Robust Motion Control of an Oscillatory-Base Manipulator in a Global Coordinate System. Industrial Electronics, IEEE Transactions on, 2015, 62(2):1163-1174.
[7] Haddadin, Sami, AlbuSchäffer, Alin, Alessandro De Luca. Collision detection and reaction: A contribution to safe physical Human-Robot Interaction. IEEE/RSJ International Conference on Intelligent Robots & Systems. IEEE, 2008.
[8] Xiao B , Yin S , Kaynak O . Tracking Control of Robotic Manipulators With Uncertain Kinematics and Dynamics. IEEE Transactions on Industrial Electronics, 2016, 63(10):6439-6449.
[9] Goel A., Swarup A. Chattering Free Trajectory Tracking Control of a Robotic Manipulator Using High Order Sliding Mode. In: Bhatia S., Mishra K., Tiwari S., Singh V. (eds) Advances in Computer and Computational Sciences. Advances in Intelligent Systems and Computing, 2017,vol 553. Springer, Singapore.
[10] Bing, Xiao, Shen, et al. Exponential Tracking Control of Robotic Manipulators With Uncertain Dynamics and Kinematics. IEEE Transactions on Industrial Informatics, 2018.
[11] Hu Y , Si B . A Reinforcement Learning Neural Network for Robotic Manipulator Control. Neural Computation, 2018, 30(7):1983-2004.
[12] Zhang C , Zou W , Cheng N , et al. Trajectory tracking control for rotary steerable systems using interval type-2 fuzzy logic and reinforcement learning. Journal of the Franklin Institute, 2018, 355(2):803-826.
[13] Wei Q , Song R , Sun Q . Nonlinear neuro-optimal tracking control via stable iterative Q-learning algorithm. Neurocomputing, 2015, 168(nov.30):520-528.
[14] Jinhua Y E , Haibin W U . Reinforcement learning adaptive neural network control of WMR with unknown skidding and slipping. Journal of Fuzhou University(Natural Science Edition), 2016.
[15] Li S , Ding L , Gao H , et al. Adaptive neural network tracking control-based reinforcement learning for wheeled mobile robots with skidding and slipping. Neurocomputing, 2018, 283(mar.29):20-30.

[16] Messuri D , Klein C . Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. Robotics and Automation, IEEE Journal of, 1985, 1(3):132-141.
[17] E G Papadopoulos, D A Ray. A New Measure of Tipper Stability Margin for Mobile Manipulators. Proc. of IEEE Int. Conf.on Robotics and Automation. Minnesota, USA,1996::3111－3116.
[18] Susan Reed. Reinforcement Learning:An Introduction. MIT Press, 1995.
[19] Lillicrap T P , Hunt J J , Pritzel A , et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6):A187.
[20] Jan J. Koenderink. Theory of "Edge-Detection". Analysis for Science, Engineering and Beyond vol 6. Springer,2012.
[21] Canny, John. "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986, pp. 679-698.
[22] Harris C , Stephens M . A combined corner and edge detector. Alvey vision conference. 1988.
[23] Chen, Ben, M. Systems. Design and implementation with jerk-optimized trajectory generation for UAV calligraphy. Mechatronics the Science of Intelligent Machines, 2015.
[24] Chen X D , Ma W , Paul J C . Cubic B-spline curve approximation by curve unclamping. Computer Aided Design, 2010, 42(6):523-534.