Electronics and Electrical Engineering

AIMS Electronics and Electrical Engineering, 9(1): 99–117.
DOI: 10.3934/electreng.2025006
Received: 15 October 2024
Revised: 01 January 2025
Accepted: 23 January 2025
Published: 11 February 2025

https://www.aimspress.com/journal/ElectrEng

Research article

Research on tracking strategy of manipulator based on fusion reward mechanism

Ruyi Dong*, Kai Yang and Tong Wang

College of Information and Control Engineering, Jilin Institute of Chemical Technology, Jilin, China

*** Correspondence:** Email: dongruyi@163.com.

Abstract: Deep reinforcement learning algorithms are widely used in the field of robot control. Sparse reward signals lead to blind exploration, affecting the efficiency of the manipulator during path planning for multi-axis systems at any given end-effector start and target position. To address the problem of tracking randomly located targets in three-dimensional space, this paper proposes a PPO (proximal policy optimization) algorithm with a fused reward mechanism, which enhances the tracking and guidance capabilities of the manipulator in multiple dimensions and reduces the blind randomness of the manipulator during the detection and sampling process. The fusion reward mechanism consists of four dimensions: trajectory correction reward, core area acceleration guidance reward, ladder adaptability reward, and abnormal termination penalty. Finally, a 7-degree-of-freedom Kuka manipulator is built on the PyBullet platform for simulation experiments. Experimental results show that, compared with the sparse reward mechanism, the PPO algorithm with the fused reward mechanism has a higher average success rate as high as 94.88% in task tracking, which can effectively improve the tracking efficiency and accuracy of the spatial manipulator.

Keywords: deep reinforcement learning; tracking; fusion reward mechanism; intelligent control; manipulator; proximal policy optimization

1. Introduction

The manipulator is a widely used device in aviation, industry, medicine, and other fields, replacing human beings to complete high-precision, high-intensity repetitive tasks and greatly improving social productivity. Although traditional motion control strategies [1] can achieve high

accuracy, they are only applicable to the operation of target objects under specific conditions. When the task or execution environment changes, it is necessary to re-extract motion features and design the motion attitude or re-code a new task for implementation, which limits intelligence, generalization, and productivity development. To enable a manipulator to replace human beings in a complex and changeable unstructured environment [2], it must have enough intelligence and generalization ability to deal with complex and changeable environments. Future environments for manipulators to perform tasks will become more and more complex and unstructured. How to make a manipulator calmly cope with this complex environment and successfully perform tasks has become the focus of artificial intelligence research.

The continuous promotion of Industry 4.0 [3] has enabled humanity to enter the fourth industrial revolution, dominated by intelligent manufacturing, where reinforcement learning algorithms play a pivotal role. In the field of intelligent manufacturing [4], the emergence of reinforcement learning algorithms has provided a new solution for the manipulator to work in high-dimensional and complex environments. Although early reinforcement learning algorithms such as Q-learning [5] enable the manipulator the intelligence of active exploration and updating strategies to a certain extent, they rely too much on Q tables, and the time and space costs of searching and storage are too high; also, they cannot be applied to solve high-dimensional problems.

With the rapid development of deep learning, reinforcement learning researchers have applied neural network principles to reinforcement learning technology, bringing a new round of development opportunities for deep reinforcement learning [6] (e.g., DQN [7], Dueling DQN [8], Double DQN [9], Rainbow DQN [10]). While the DQN series algorithms have successfully addressed the time and space complexity issues associated with the Q table, having achieved notable progress in solving high-dimensional problems, their applicability is more suited to discrete motion spaces. Unfortunately, they fall short when it comes to addressing continuous problems, such as the movement environment of a spatial manipulator. Schulman et al. [11] put forward the trust region policy optimization algorithm (TRPO), which is very effective for optimizing large nonlinear strategies [12] and solving high-dimensional continuous problems. However, the constraint of the TRPO algorithm is that there is little difference between the old and new strategies, and it has the disadvantage of consuming too many computing resources. Schulman et al. [13] proposed a PPO algorithm to directly constrain the probability ratio of new and old strategies so that the computational resource consumption is reduced, and the overall performance of the algorithm is better than TRPO. For solving high-dimensional continuous problems, the balance performance of the PPO algorithm is better than other deep reinforcement learning algorithms. Therefore, this paper implements the tracking control task [14] of the manipulator based on the PPO algorithm.

Regarding the manipulator tracking control application scenario, to ensure it has enough intelligence and generalization ability to cope with complex and changing real-world environments, this paper designs a PPO algorithm based on a convolutional neural network (CNN) that incorporates a reward improvement mechanism, including trajectory correction reward, core area acceleration guidance reward, ladder adaptability reward, and abnormal termination penalty. On one hand, the issue of reward sparsity in tracking control tasks for the manipulator is addressed; on the other hand, it also solves to some extent the problem of a lack of exploration and blind ineffective exploration of the PPO algorithm itself. Finally, the results of simulation experiments in different ranges show that the mean value of tracking success rate of the manipulator under a fused reward mechanism is higher than that of a sparse reward mechanism by 5 to 7 percentage points, and the mean value of the

number of steps required for successful tracking with the fused reward mechanism is around 9.7, lower than that of the sparse reward mechanism by about 13. Thus, the manipulator of the PPO algorithm with the improved fused reward mechanism has higher accuracy and faster execution speed in tracking and control tasks.

The remaining paper is organized as follows: Section 2 introduces the basic principle of PPO algorithm. Section 3 displays the design of the manipulator tracking strategy based on the fusion reward mechanism. Section 4 presents the experimental results and analysis. Section 5 gives the conclusion.

1.1. Related work

The primary challenge in deep reinforcement learning lies in the issue of sparse rewards, which leads the manipulator to engage in blind random exploration [15]. Experts have proposed various solutions from different perspectives. Vecerik et al. [16] introduced a demonstration method that partially addressed the sparse reward problem. Colas et al. [17] proposed the GEP-PG algorithm, aiming to resolve the sparse reward problem by optimizing the performance of the exploration phase. Conti et al. [18] combined the novelty search algorithm (NS) [19] and the quality diversity (QD) [20] algorithm with evolutionary strategies, creating a new algorithm that effectively improved performance in a sparse reward environment. Riedmiller et al. [21] presented scheduling-assisted control (SAC-X), capable of learning complex behaviors from scratch in the presence of multiple sparse reward signals. Wang et al. [22] suggested a non-expert-assisted deep reinforcement learning (RL) algorithm, enhancing its suitability for addressing sparse reward problems. Qi et al. [23] proposed a novel hybrid-vehicle energy management strategy algorithm based on reinforcement learning, effectively addressing the sparse reward problem during vehicle power allocation task training. Li et al. [24] introduced a stochastic curiosity-driven model rooted in deep reinforcement learning to address reward sparsity through internal reward guidance. Liu et al. [25] proposed cooperative multi-agent exploration (CMAE), fostering exploration through cooperation between multi-agents to overcome the impact of sparse rewards. Xia et al. [26] introduced the SparKGR hybrid multi-hop reasoning model, optimizing the guidance strategy to improve the reasoning performance of sparse knowledge graphs. Cai et al. [27] suggested a reward shaping and discounting scheme based on LDGBA that relies solely on the EP-MDP state, overcoming the problem of reward sparseness. Roghair et al. [28] proposed a convergence-based method and a domain network-based guidance approach to address the sparse reward problem in UAV flight training. Eoh et al. [29] introduced an automatic process learning (ACL) method for object transfer based on deep reinforcement learning (DRL) to address the sparse reward problem. Christianos et al. [30] proposed the SEAC algorithm, facilitating experience sharing between agents for effective exploration and partially resolving the issue of reward sparsity. Liu et al. [36] proposed a deep reinforcement learning-based autonomous obstacle avoidance combined with SLAM for a wheeled snake robot, by constructing a 3D model of the snake robot in Gazebo, which is able to achieve effective path planning and environment mapping in an environment with obstacles. Wen et al. [37] proposed the maximum entropy algorithm LGE-SAC based on prioritized experience replay, which effectively avoids obstacles and successfully reaches the goal through simulation experiments in a Gazebo simulator environment and real experiments on a Turtlebot3 robot equipped with a LiDAR sensor. Azar et al. [38] used a deep reinforcement learning (DRL) approach to robot navigation and

exploration, integrating learned strategies and waypoint selection to develop a complete autonomous navigation and exploration system. Haider et al. [39] introduced the most important DRL-based navigation and control algorithms for mobile robots, describing the subcomponents of navigation perception, mapping, localization, and motion planning for mobile robots. Ahmed et al. [40] proposed an adaptive fixed-time fractional-order integral control for externally perturbed Euler-Lagrange systems, which has better tracking and convergence performance when compared to adaptive fractional-order sliding mode control schemes. Jahanshahi et al. [41] provided a comprehensive overview of the integration of machine learning techniques into robotic grasping, with a particular emphasis on the challenges and advances in spatial applications, providing valuable insights into the integration of deep reinforcement learning in robotic manipulation tasks. The aforementioned studies aim to overcome the sparse reward problem from several perspectives and approaches and demonstrate unique advantages and technical features in their specific application scenarios. However, when facing complex unstructured application scenarios, these approaches may exhibit certain limitations in terms of accuracy and generalization ability.

1.2. Motivation and contribution

The algorithms above provide solutions to the sparse reward problem in a variety of application fields but lack pertinence and applicability in the tracking control task of manipulators.

To enhance the tracking and guidance capabilities of the manipulator in a multi-dimensional space and reduce its blind randomness in the detection and sampling process, this study proposes an enhanced proximal policy optimization (PPO) reinforcement learning algorithm. The improvement is achieved by incorporating a convolutional neural network (CNN) for image feature extraction and introducing a fusion reward mechanism. This paper provides a detailed exposition of the CNN model and outlines the fusion reward mechanism. Experimental trials are conducted to validate the proposed algorithm, and corresponding results are presented for comparative analysis.

The primary innovation of this study is the implementation of an enhanced PPO algorithm that integrates the manipulator and the environment. The algorithm incorporates a synthesis of reward mechanisms, encompassing trajectory correction rewards, core area adaptive rewards, and stage adaptive rewards and punishments. These components collaborate to facilitate the manipulator's rapid and seamless learning of the target object's maneuvering strategy, thereby enhancing the overall efficiency of the algorithm.

2. Deep reinforcement learning algorithm

2.1. Principle of deep reinforcement learning algorithm

Reinforcement learning algorithms are essentially end-to-end intelligent control decision-making algorithms. Deep reinforcement learning algorithms integrate deep learning with reinforcement learning, mainly including five elements: environment, agent, action, reward, and state.

Figure 1 shows the basic interaction process between the agent and the environment. The agent first obtains the state and reward from the environment, then outputs the action, and acts on the environment by updating the reinforcement learning algorithm. Finally, the agent continues to

interact with the environment, iteratively updating state, reward, and action through the current state and current reward, obtaining as many cumulative rewards as possible, and ultimately obtaining the action strategy that best suits the current environment (that is, the strategy that can satisfy the maximum reward). Deep reinforcement learning combines the powerful feature extraction [31] capability of deep learning and the excellent decision-making capability of reinforcement learning.

Figure 1. Agent interaction with the environment.

2.2. Fundamentals of the improved PPO algorithm

The PPO algorithm represents an improvement on the strategy gradient algorithm, which mainly solves the problems of low sampling efficiency and unstable training. The sampling method of the strategy gradient algorithm is based on Monte Carlo sampling [32]. In this method, the agent takes the current policy to sample the current environment for an episode and updates the policy according to the sampling data; the sampling data is used only once. This sampling method causes insufficient data utilization and low sampling efficiency, but the PPO algorithm uses importance sampling [33] to skillfully solve this problem. The following section delineates the two pivotal steps of the PPO algorithm: cumulative reward [34] and importance sampling.

(1) Cumulative rewards:

If the step of a sampling episode is defined as $\tau(\tau = s_1, a_1, s_2, a_2, s_3, a_3, \ldots, s_t, a_t)$, the probability of τ can be expressed as:

$$\begin{aligned} p_\theta(\tau) &= p(s_1)p_\theta(a_1|s_1)p(s_2)p_\theta(s_2|s_1,a_1)p(s_1)p_\theta(a_2|s_2)\ldots \\ &= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t) \end{aligned} \tag{2.1}$$

Then, the cumulative rewards obtained in an episode are defined as:

$$R(s_1) = \sum_t^T \gamma^{t+1} r_t \tag{2.2}$$

A reward is the core of deep reinforcement learning algorithms and an important basis for guiding agents to make decisions. The difference in reward design greatly affects the result of the agent's task execution. Sparse rewards will make the agent's convergence speed very slow or even nonconvergent. Therefore, a fusion reward mechanism is designed to solve this problem.

(2) Importance sampling:

Importance sampling is essentially an approximate sampling method. Supposing there are two distributions $p(x)$ and $q(x)$, and knowing that $p(x)$ satisfies a certain distribution but $p(x)$ cannot be integrated and can only be sampled from $q(x)$, then for a distribution function $f(x)$ whose independent variable x satisfies $p(x)$, its expectation is:

$$E_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim p}\left[f(x)\frac{p(x)}{q(x)}\right] \tag{2.3}$$

The policy update gradient of the PG algorithm is as follows:

$$\nabla J(\theta) = E_{\tau - p\theta'(\tau)}\left[R(\tau)\nabla \log p\theta(\tau)\right] \tag{2.4}$$

Transform Equation 2.3 to get:

$$\nabla J(\theta) = E_{\tau - p\theta'(\tau)}\left[\frac{p\theta(\tau)}{p\theta'(\tau)}R(\tau)\nabla \log p\theta(\tau)\right] \tag{2.5}$$

This makes the algorithm perform the transition from on-policy to off-policy. This way, the collected data can be updated several times, reusing data and greatly improving data utilization.

3. Design of manipulator tracking strategy based on a fusion reward mechanism

3.1. Setting neural network parameters

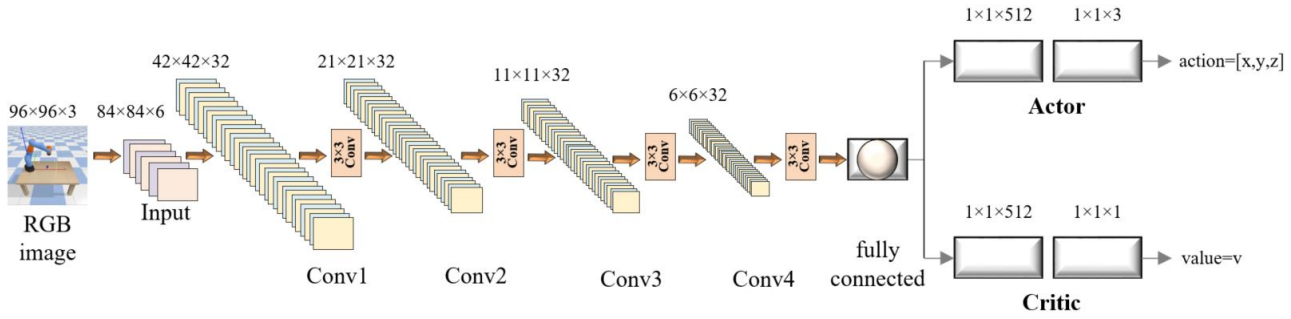


Figure 2. Convolutional neural network structure.

In network structure configuration, the *actor network* and the *critic network* of the PPO algorithm exhibit congruence in their internal composition. The network comprises four convolutional layers and two fully connected layers. Each convolutional layer is equipped with a 3 × 3 convolutional kernel, with a step size of 2, a boundary padding of 1, and a nonlinear activation function using the ReLU function. Both the actor and the critic networks use images as the input, and the original image needs to go through a preprocessing step. Specifically, the RGB image is first converted to a grayscale image to reduce channel dimensions, the pixel value range is normalized from [0–255] to [0–1], and the image size is cropped to 84 × 84 pixels; then, features are extracted from the image.

The primary distinction between the two networks lies in the input parameters; those of the actor network encompass the image and the end position determined by the camera, while the critic network receives the image captured by the camera. Subsequent to two fully connected layers, the actor network generates 3D action vectors, and the critic network yields scalars as a fitted estimate of

the value function. The comprehensive structural design of the actor and critic networks is illustrated in Figure 2.

3.2. Application of PPO algorithm in the manipulator tracking environment

The tracking application of the manipulator is a high-dimensional, complex, and continuous problem. The PPO algorithm provides an efficient and effective solution to continuous problems, as previously explained in the algorithm analysis. Consequently, the PPO algorithm is employed in the manipulator's tracking application.

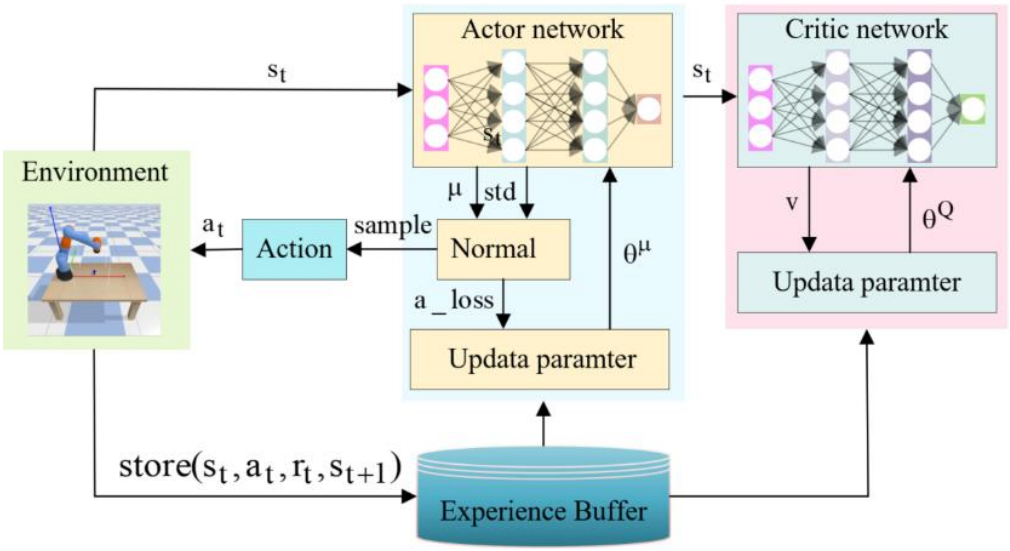


Figure 3. Basic architecture.

Figure 3 illustrates the fundamental procedure of the PPO algorithm implementation within the manipulator tracking environment. Throughout the entire basic procedure, the target object emerges randomly within the white cube frame within the simulation environment, as illustrated in Figure 4. Subsequently, the tracking environment image data is acquired by the built-in camera of the PyBullet simulation. The environment state $s = [x_e, y_e, z_e, x_b, y_b, z_b]([x_e, y_e, z_e]$ is the coordinate position of the end-effector of the manipulator, and $[x_b, y_b, z_b]$ is the coordinate of the target object. Data is transferred to the actor and critic networks of the CNN [35] architecture of the PPO algorithm through the Gym interface. The actor network outputs two parameter values, μ and std, according to the current state, uses these two values to sample actions from a Gaussian-distributed action policy, selects an action a_t and passes it to the intelligent body, and then computes a_loss to backpropagate to the actor network to update the parameter θ^μ. The agent makes an action a_t, the environment state becomes s_{t+1}, and at the same time it gets a reward r_t. The state, action, and reward are stored as a tuple (s_t, a_t, r_t, s_{t+1}) in the experience pool. The critic network calculates the state value estimate v based on the current state s_t and then back propagates to update the critic network parameters θ^μ. The range of motion of each joint is set to avoid exceeding the physical limits, and a smooth path planning algorithm is used to deal with the joint limits and singularities in the workspace to ensure the safety and efficiency of the manipulator's motion. This process iterates continuously to optimize the manipulator's path and control strategy.

3.3. Fusion reward mechanism

The guidance mechanism composed of rewards is the core of the deep reinforcement learning algorithm to control the agent to achieve the goal task. The highly complex environment and wide exploration space of the manipulator make it difficult for the manipulator to track the target object with sparse reward guidance. When the manipulator successfully tracks the target, it obtains a reward of 1, and when it fails to track the target, it does not obtain a reward. This makes the reward very sparse, resulting in slow convergence or even inability to converge. To solve this problem, a fusion reward guidance mechanism was developed from multiple design perspectives, such as accelerated access to rewards in core areas and step-length guidance rewards.

Sparse reward is defined as follows:

$$r_{sparse} = \begin{cases} 1, & The\ target\ was\ successfully\ tracked \\ 0, & The\ target\ was\ not\ successfully\ tracked \end{cases}$$

Fusion reward is defines as follows:

(1) Trajectory correction reward

$$r_1 = \begin{cases} 0.1, & d_{ao} < d_{bo} \\ -0.01, & d_{ao} = d_{bo} \\ -0.1, & d_{ao} > d_{bo} \end{cases} \tag{3.1}$$

where r_1 is the trajectory correction reward, d_{ao} is the distance from the end coordinate position of the manipulator to the target coordinate position at the current time calculated by Eq 3.2, and d_{bo} is the distance from the end coordinate position of the manipulator at the previous time to the target coordinate position calculated by Eq 3.3. It is well-established that the sparseness of the reward signal can lead to a slow or complete failure of the learning process. Therefore, this study finalized the configuration of the reward values in Eq 3.1 through a series of systematic experiments and parameter adjustments. These adjustments were made to significantly improve the stability and generalization ability of the algorithm.

$$d_{ao} = \sqrt{\left(x_a - x_o\right)^2 + \left(y_a - y_o\right)^2 + \left(z_a - z_o\right)^2} \tag{3.2}$$

where x_a, y_a, z_a represent the position coordinates of the end of the manipulator at the current time, and x_o, y_o, z_o represent the position coordinates of the target object.

$$d_{bo} = \sqrt{\left(x_b - x_o\right)^2 + \left(y_b - y_o\right)^2 + \left(z_b - z_o\right)^2} \tag{3.3}$$

where x_b, y_b, z_b represent the end position coordinates of the manipulator at the previous time, and x_o, y_o, z_o represent the position coordinates of the target object.

Each time the manipulator interacts with the environment and moves one step, it will receive a reward or punishment according to the distance relationship between d_{ao} and d_{bo}. When $d_{ao} < d_{bo}$, it means that the manipulator moves toward the target, and a reward is given to the manipulator; when $d_{ao} = d_{bo}$, it indicates that the manipulator is in a stagnant state, and a little punishment will be given; when $d_{ao} > d_{bo}$, it indicates that the manipulator moves in the wrong direction, and a big punishment will be given at this time.

(2) Core area acceleration guidance reward

The core area acceleration guidance reward is designed to distribute rewards according to the importance of the area where the manipulator is located.

$$r_2 = \begin{cases} -10d_{ao} + 10, & 0.04 \le d_{ao} < 0.1 \\ -10d_{ao} - 4e^{-d_{ao}} + 10, & 0.01 < d_{ao} < 0.04 \\ -10d_{ao} - 4e^{-d_{ao}} + 100, & d_{ao} \le 0.01 \end{cases} \tag{3.4}$$

where r_2 is the core area acceleration guidance reward, provided that the end-effector of the manipulator has not exceeded the motion execution space and the maximum number of steps is not exceeded. Based on the 7-degree-of-freedom KUKA manipulator with an arm length of 0.8 m, a workspace size of [0.5, 0.6, 0.55], and a target object size of [0.025, 0.1, 0.025], three different distance thresholds are designed in Eq. 3.4 to allow the control system enough time and space to make adjustments for finer control over the process of approaching the target object. The core area acceleration guidance reward is divided into three different regions, and different functional forms are used to incrementally increase or decrease the reward. The outermost region indicates that when the coordinate distance between the manipulator end-effector and the target object ranges from 0.04 to 0.1, the rewards are distributed through a linear function. The middle region indicates that when the coordinate distance between the manipulator end-effector and the target object ranges from 0.01 to 0.04, the rewards are distributed through the composite function composed of linear and exponential functions, which accelerates rewards. The core region indicates that when the coordinate distance range between the manipulator end-effector and the target object is less than or equal to 0.01 (belonging to the range of tracking success determination), a larger reward is added to the composite function to enhance guidance.

(3) Ladder adaptability reward

$$r_3 = -step + r\_step, \ d_{ao} \le 0.01 \tag{3.5}$$

where r_3 denotes the ladder adaptability reward, which is designed to make the manipulator track the target object faster. When the manipulator enters into the reward range of the core region of Eq 3.4, the reward is given at this moment according to the cumulative length step from the initial to the tracking success. Equation 3.5 adds a variable r_step set to a fixed value of 20, mainly to introduce a baseline value so that the manipulator will be rewarded positively for completing the task within the first 20 steps and negatively otherwise. Therefore, the smaller the step, the larger the reward; the larger the step, the smaller the reward.

(4) Abnormal termination penalty

$$r_4 = \begin{cases} R\_p, & Out\ of\ range\ penalty \\ R\_p, & Penalty\ for\ exceeding\ the\ maximum\ step\ size \end{cases} \tag{3.6}$$

where r_4 is the abnormal termination penalty. When the manipulator exceeds the tracking range and the maximum tracking step, penalty reward R_p is given, and the round is terminated. In order to accelerate the convergence of the agent, when the manipulator exceeds the tracking range and the maximum tracking step, the parameter R_p in Eq 3.6 is set to -13. The value of this parameter has been experimentally derived to enable the manipulator to complete the task quickly, preventing wasted time. Then, the final fusion reward is calculated as follows:

$$r = r_1 + r_2 + r_3 + r_4 \tag{3.7}$$

where r is the fusion reward. The pseudocode of the improved PPO algorithm is formulated in Algorithm 1.

Algorithm 1 Improved PPO algorithm

1 Initialize the policy network $\pi_{\theta(a|s)}$ and the value network $\mu(s)$, and synchronize network parameters into multiple processes

2 Initialize experience buffer $B(s_t, a_t, \gamma, \lambda)$

3 Accept environmental observation state s_t

4 for episode = 1 to N do

5 Initialize the number of steps T that each process needs to perform

6 for step = 1 to T do

7 Sample an action a_t from the policy distribution π

8 Calculate the sum $p = \sum_t^T \log \pi(a_t)$ of the logarithmic probabilities of the action

9 Get an estimate $v(s_t)$ of the current state value through the value network

10 Perform actions a_t, get the next status s_{t+1}, reward r_t, terminate status d

11 Store $(s_t, a_t, r, s_{t+1},)$ to experience buffer B

12 The probability ratio of the strategy optimization is calculated, $\hat{A}$ based on the value function method v^{μ}

13 Update the policy network parameters

$$L\left(s_t, a_t, \theta_{old}, \theta\right) = min\left(exp^{\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)}} A^{\pi_{\theta_{old}}}\left(s_t, a_t\right), clip\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)}, 1-\in, 1+\in\right) A^{\pi_{\theta_{old}}}\left(s_t, a_t\right)\right)$$

14 Update value network parameters

$$\phi\left(s_t, r_t\right) = \left(V_\mu\left(s_t\right) - r_t\right)^2$$

4. Experimental test and analysis

Table 1. Parameter settings of PPO algorithm.

Parameter	Value
Maximum step size of a single round	4200
Strategy network learning rate	0.0003
Value network learning rate	0.001
Generalization advantage estimate (λ)	0.97
Reward discount factor (γ)	0.99
Clipping threshold (ϵ)	0.2

In order to evaluate the performance of the improved PPO algorithm, a 7-degree-of-freedom Kuka manipulator simulation experiment was set up on the PyBullet platform. The experimental environment was based on an Ubuntu operating system with an NVIDIA GeForce RTX 3090 and an Intel Gold 5218 processor, and the algorithm was written based on the PyTorch framework. In the

whole simulation experiment setup, the size of the blue target block was [0.025, 0.1, 0.025], the field of view of the camera was 60 ° and the position was [0.59, 0, 0.8], and the white cube box on the desktop on the right side of the simulation environment (Figure 4) was designed by Gym to indicate the range of action of the manipulator, with a size of [0.5, 0.6, 0.55]. The target object randomly appears on different positions of the desktop within the action range of the manipulator. The manipulator uses the PPO algorithm to make a strategic decision based on the current environment state, selects the action with the highest expected reward value, and plans the motion trajectory accordingly. The main parameters of the PPO algorithm were set as in Table 1. A comparison of model training and testing under the two different reward guiding mechanisms (i.e., sparse and fusion reward) was performed. Experimental analysis was based on the tracking success rate and the average number of steps of the manipulator.

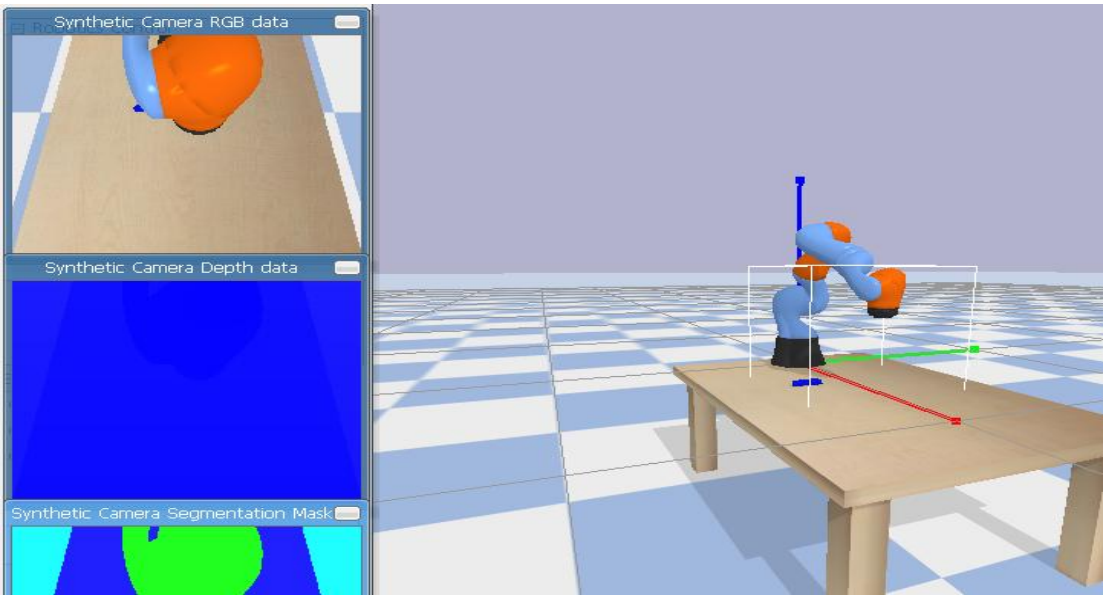


Figure 4. Robot tracking simulation environment.

4.1. Comparison of experimental results

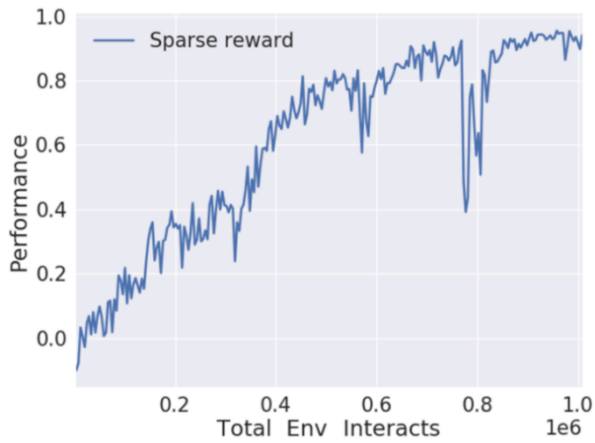


Figure 5. Sparse reward.

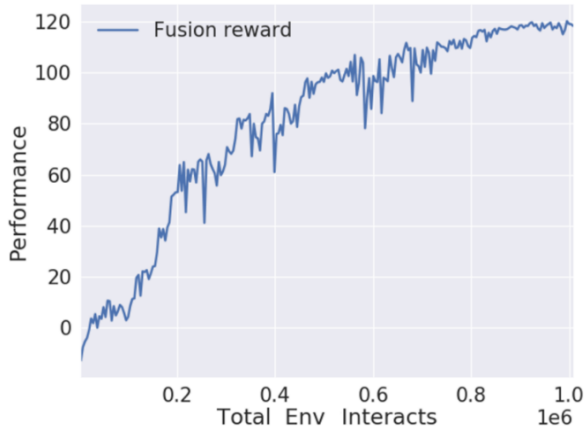
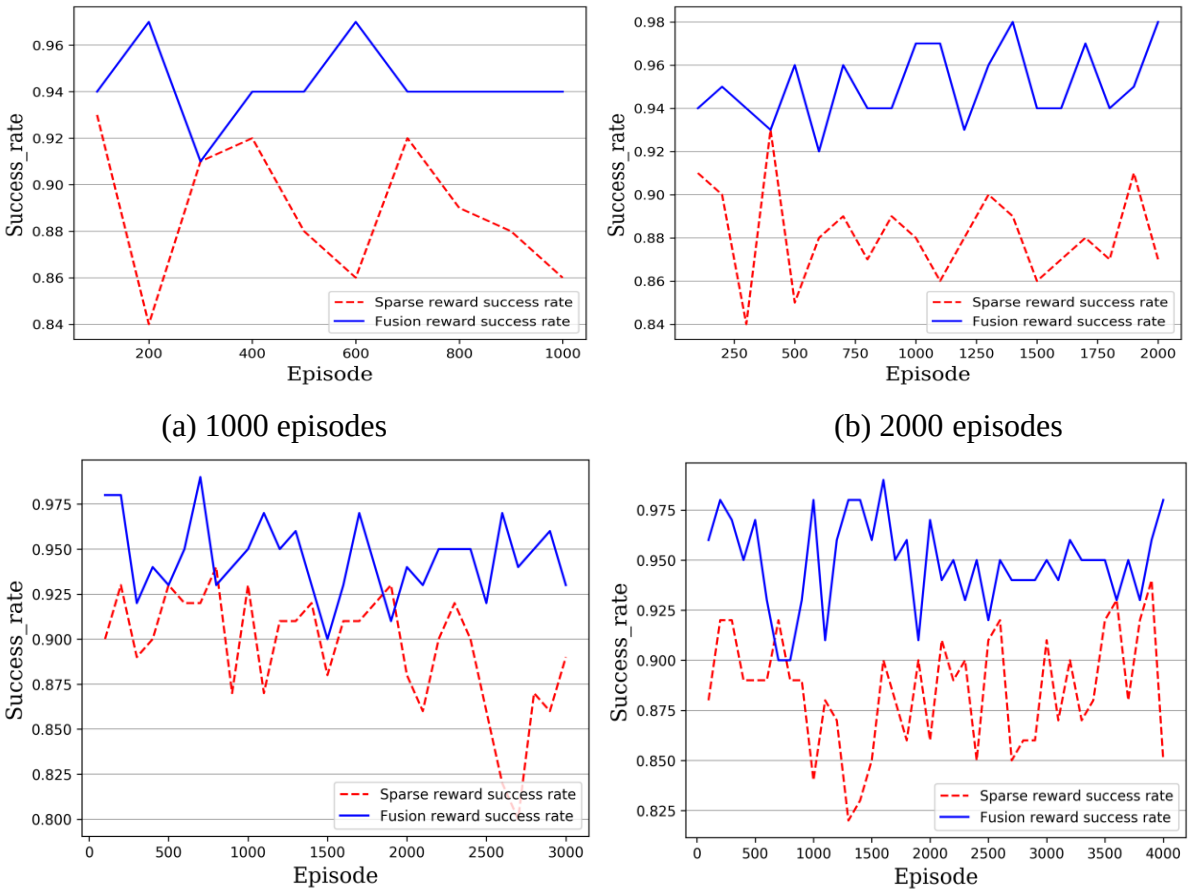


Figure 6. Fusion reward.

Figures 5 and 6 show the iteration curves of the average episode reward of sparse reward and fusion reward in the training stage. The vertical axis (performance) refers to the average episode reward, and the horizontal axis refers to the complete number of interactions in the whole training stage. The sparse reward curve shows large fluctuations, especially in the late stage, which is due to the lack of effective guidance under the influence of the sparse reward; the manipulator can only search for the optimal strategy to track the target object through large-scale exploration, which is overall less stable. Compared with the sparse reward, the fluctuation amplitude of the fusion reward curve is significantly smaller than that of the sparse reward, and the overall convergence speed is relatively faster, indicating that the improved fusion reward can improve the convergence speed in the training phase.



(a) 1000 episodes (b) 2000 episodes

(c) 3000 episodes (d) 4000 episodes

Figure 7. Curve of success rate.

Figure 7 shows the experimental comparison curve of the tracking success rate of sparse reward and fusion reward in the test phase, describing the relationship between the number of episodes and the success rate. Success rate refers to the proportion of success in 100 episodes, which is counted once every 100 episodes. The red dotted line is the test success rate of sparse rewards, and the blue one is the test success rate of fusion rewards. To satisfy the persuasiveness of the experimental analysis, the experimental data of 1000, 2000, 3000, and 4000 episodes were counted. Because the position of the tracked target object is random, both the sparse reward curve and the fusion reward curve have a certain volatility. A comparative analysis of the overall curve, as well as the specific

values presented in Tables 2.1, 2.2, 2.3, and 2.4, reveals that, with the exception of a few discrete points, the success rate of fusion reward consistently surpasses that of sparse reward for the predominant part of the data. In addition, given that the manipulator needs to perform diverse trajectory tracking tasks based on the positions of different target objects, the fusion reward mechanism exhibits higher robustness and adaptability, which can guide the manipulator to complete the tracking task more accurately and thus significantly improve the tracking success rate.

Table 2.1. Success rates of sparse rewards and fusion rewards over 1000 episodes.

Episode	200	400	600	800	1000
Fusion reward	0.97	0.94	0.97	0.94	0.94
Sparse reward	0.84	0.92	0.86	0.89	0.86

Table 2.2. Success rates of sparse rewards and fusion rewards over 2000 episodes.

Episode	250	500	750	1000	1250	1500	1750	2000
Fusion reward	0.95	0.96	0.95	0.97	0.94	0.95	0.94	0.98
Sparse reward	0.86	0.85	0.88	0.88	0.89	0.86	0.875	0.87

Table 2.3. Success rates of sparse rewards and fusion rewards over 3000 episodes.

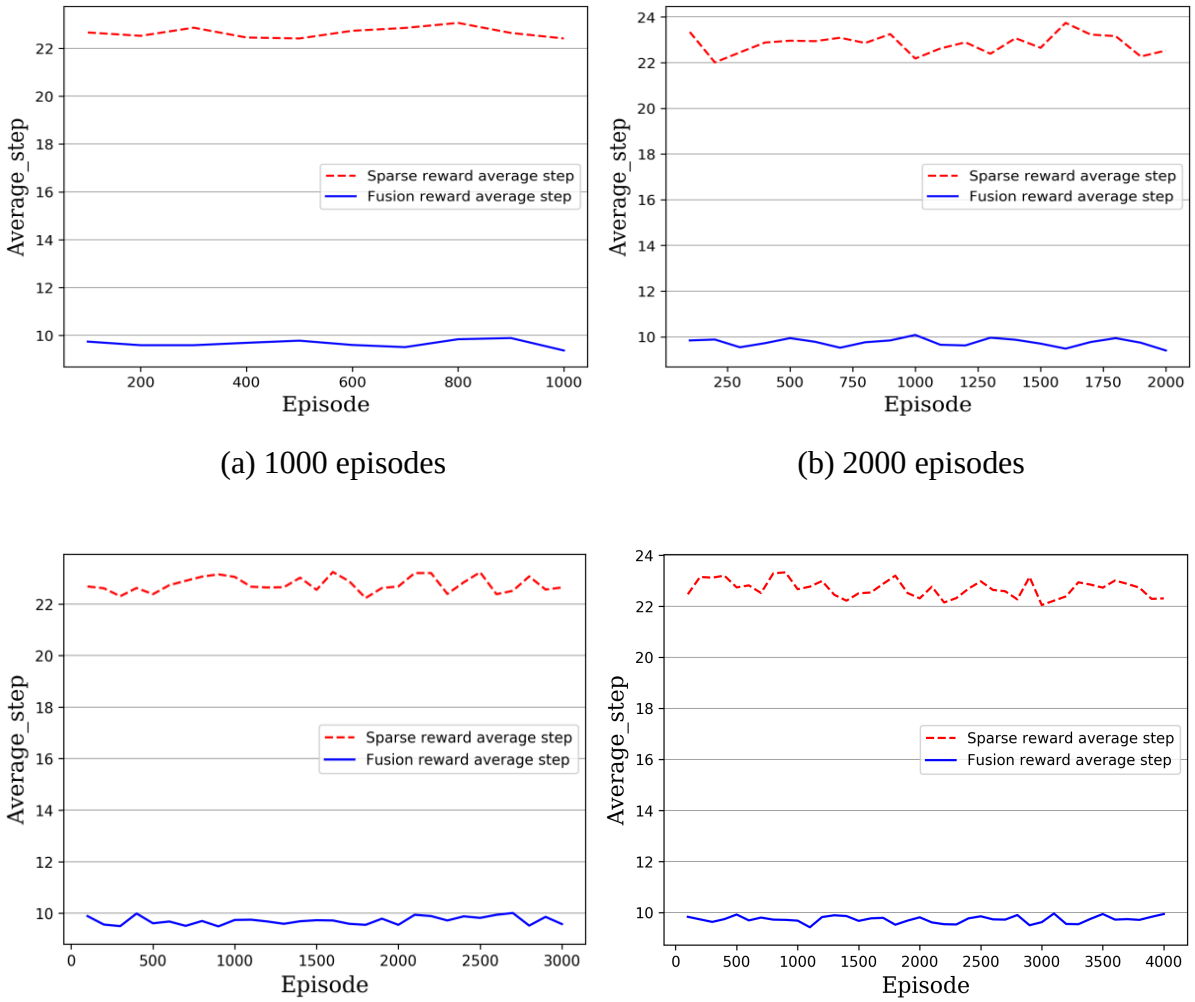
Episode	500	1000	1500	2000	2500	3000
Fusion reward	0.93	0.95	0.90	0.94	0.92	0.93
Sparse reward	0.93	0.87	0.88	0.89	0.875	0.89

Table 2.4. Success rates of sparse rewards and fusion rewards over 4000 episodes.

Episode	500	1000	1500	2000	2500	3000	3500	4000
Fusion reward	0.97	0.98	0.96	0.97	0.92	0.95	0.95	0.98
Sparse reward	0.88	0.84	0.85	0.86	0.875	0.91	0.92	0.85

Figure 8 shows the comparison curve of the average steps when the sparse reward and fusion reward track successfully in the test phase, describing the relationship between the average steps and episodes when the track succeeds. The *average_step* in the vertical axis refers to the cumulative steps for successful tracking in 100 episodes divided by 100. The average steps can effectively show the tracking efficiency of the manipulator. The smaller the average steps, the higher the tracking efficiency. The red dotted line describes the average steps of sparse reward, while the blue line describes the average steps of fusion reward. To satisfy the persuasiveness of the experimental analysis, the experimental data of 1000, 2000, 3000, and 4000 episodes were analyzed. It can be clearly observed through the dotted line that the tracking steps of the fusion reward are always located below the fusion reward, and the difference between the two is very large. The results show that the average number of steps of the sparse reward is above 22, while the average number of steps

of the fusion reward is significantly lower, fluctuating between 8 and 9. A smaller average number of steps indicates that the shorter the time needed to track the target object, the more efficient the tracking is. Compared to the sparse reward, the tracking efficiency of the fusion reward is two times that of the sparse reward.



(a) 1000 episodes (b) 2000 episodes

(c) 3000 episodes (d) 4000 episodes

Figure 8. Curve of average steps.

4.2. Experimental data analysis

Table 3 shows data related to the tracking success rate of sparse rewards and fusion rewards in different ranges. The statistical indicators include mean, variance, maximum, minimum, and t-statistics analyzed by t-statistics and p-values. The statistical range covers 1000, 2000, 3000, and 4000 episodes. To facilitate the statistical analysis of data, mean data are kept at two decimal places (rounded), and variance data are kept with three significant figures (expressed in scientific notation). According to Table 3, compared with sparse rewards, the mean, maximum, and minimum values of tracking accuracy of fusion rewards in different ranges are increased, and the variance is greatly reduced. It can be concluded that fusion rewards not only have a higher tracking success rate than sparse rewards but also a higher tracking stability.

Table 3. Statistics of tracking the success rate of sparse reward and fusion reward.

Type	1000 episodes		2000 episodes		3000 episodes		4000 episodes	
	Sparse reward	Fusion reward	Sparse reward	Fusion reward	Sparse reward	Fusion reward	Sparse reward	Fusion reward
Variance	9.21E-04	2.90E-04	4.77E-04	3.00E-04	**1.09E-03**	4.46E-04	8.41E-04	4.98E-04
Maximum	93.00%	97.00%	93.00%	98.00%	94.00%	**99.00%**	94.00%	**99.00%**
Minimum	84.00%	91.00%	84.00%	**92.00%**	80.00%	90.00%	82.00%	90.00%
Mean	88.90%	94.30%	88.15%	**95.05%**	89.50%	94.53%	88.50%	94.88%
t_statistic	-4.66E+02		-7.43E+02		-3.85E+02		-5.23E+02	
p_value	3.49E-38		7.79E-42		1.06E-36		4.28E-39	

Table 4 shows the data related to the average number of steps for sparse and fusion rewards over different ranges. The statistical indicators include mean, variance, maximum, minimum, and t-statistics analyzed by t-statistics and p-values. The statistical ranges include 1000, 2000, 3000, and 4000 sets. According to the results in Table 4, the mean, maximum, and minimum values of the average set lengths of the fusion rewards in different ranges are significantly smaller, and the variance is greatly reduced, compared to the sparse rewards. Under the same conditions, the time complexity of trajectory planning with fused rewards over sparse rewards, in the worst case, is O(T) and O(T^2), respectively, where T denotes the number of planning steps, indicating that fused rewards are faster and more stable than sparse rewards for tracking. Data analysis of variance, mean, maximum, and minimum of tracking success rate and average steps shows that compared with sparse rewards, fusion rewards have a higher and more stable tracking success rate, and when tracking is successful, fusion rewards have smaller and more stable tracking steps.

Table 4. Statistics of average steps of sparse reward and fusion reward.

Type	1000 episodes		2000 episodes		3000 episodes		4000 episodes	
	Sparse reward	Fusion reward	Sparse reward	Fusion reward	Sparse reward	Fusion reward	Sparse reward	Fusion reward
Variance	4.78E-02	2.52E-02	1.90E-01	3.14E-02	8.82E-02	2.45E-02	1.18E-01	1.78E-02
Maximum	2.31E+01	9.89E+00	2.37E+01	1.01E+01	2.32E+01	1.00E+01	2.33E+01	9.96E+00
Minimum	2.24E+01	9.37E+00	2.20E+01	9.40E+00	2.22E+01	9.49E+00	2.21E+01	9.42E+00
Mean	2.27E+01	9.66E+00	2.28E+01	9.75E+00	2.28E+01	9.72E+00	2.27E+01	9.73E+00
t_statistic	1.38E+02		8.35E+01		1.18E+02		1.04E+02	
p_value	1.06E-28		9.31E-25		1.76E-27		1.79E-26	

4.3. Comparison of control performance

In Figure 9, the success rate curves of the various control algorithms are depicted for each 100 rounds of testing. The purple, red, blue, and green colors represent the improved PPO algorithm, the original PPO algorithm, the DDPG (deep deterministic policy gradient) algorithm, and the TD3 (trust region policy optimization with deep deterministic policy) algorithm, respectively. It is evident that the success rates of the original PPO algorithm and TD3 algorithm exhibit significant fluctuations,

while the DDPG algorithm demonstrates a higher success rate and reduced variability compared to the previous two algorithms. In contrast, the improved PPO algorithm demonstrates a 94.88% success rate, which significantly surpasses the performance of the original PPO algorithm, the DDPG algorithm, and the TD3 algorithm. This enhancement signifies the enhanced stability of the manipulator trajectory planning control. Figure 10 illustrates the time required for successful planning for each algorithm in the test round. As shown, the improved PPO algorithm takes only 1.71 s to complete successful planning, a performance that outperforms the other three compared algorithms, reflecting a shorter planning time. Therefore, the improved PPO algorithm not only accelerates the planning process but also significantly improves the control performance of the robot arm through higher accuracy.

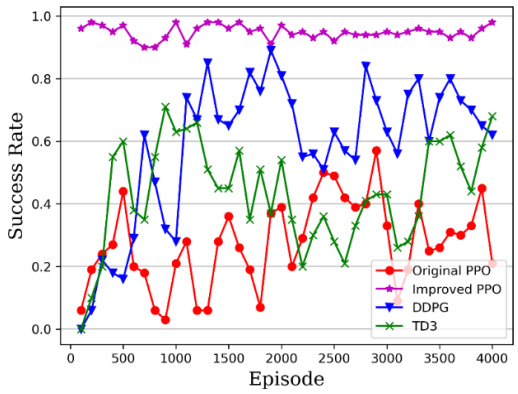


Figure 9. The curve of trajectory success rate.

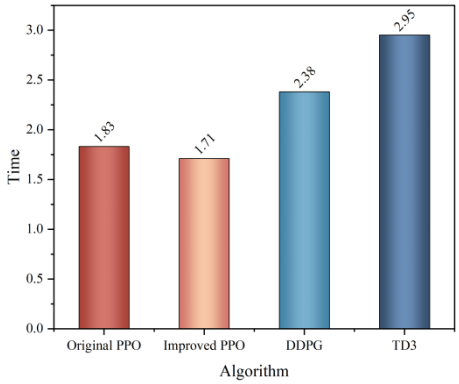


Figure 10. Test planning time.

5. Conclusions

The sparse reward mechanism in the PPO algorithm often leads the agent to collect many ineffective and expected samples during the sampling stage, which causes its policy optimization to update and converge slowly or even not converge. To overcome the influence of sparse rewards, this paper designs a fusion reward mechanism PPO algorithm and applies it to the manipulator target tracking. The experimental results show that the average tracking success rate of the PPO algorithm with fused reward mechanism is as high as 94.88%, and the average number of steps is about 9.7. The algorithm significantly improves the tracking accuracy and efficiency of the spatial manipulator and further enhances the stability of the manipulator in target-tracking tasks. However, the applicability of the algorithm on other types of manipulators remains to be verified, especially in terms of considering the effects of hardware differences, environmental adaptability, data requirements, and real application environments. The improved method proposed in this paper not only enables the manipulator to be better adapted to the traditional industrial automation field but also has the potential to be widely applied to the fields of healthcare, service robotics, and agriculture, thus enhancing the automation level and work efficiency in various industries.

Author contributions

Ruyi Dong: Conceptualization, Software, Investigation, Methodology, Validation, Writing − original draft, Writing − review & editing; Kai Yang: Conceptualization, Writing − original draft,

Writing − review & editing; Tong Wang: Conceptualization, Writing − original draft, Writing − review & editing. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was supported by the Science & Technology Development Project of Jilin Province, China (YDZJ202201ZYTS555).

Use of AI tools declaration

The authors declared they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare that there is no conflict of interest in this paper.

References

1. Chen Z, Wang S, Wang J, Xu K, Lei T, Zhang H, et al. (2021) Control strategy of stable walking for a hexapod wheel-legged robot. *ISA transactions* 108: 367–380. https://doi.org/10.1016/j.isatra.2020.08.033.
2. Liu J, Jayakumar P, Stein JL, Ersal T (2018) A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments. *Vehicle system dynamics* 56: 853–882. https://doi.org/10.1080/00423114.2017.1399209.
3. Lasi H, Fettke P, Kemper HG, Feld T, Hoffmann M (2014) Industry 4.0. *Business & information systems engineering* 6: 239–242. https://doi.org/10.1007/s12599-014-0334-4.
4. Zhong RY, Xu X, Klotz E, Newman ST (2017) Intelligent manufacturing in the context of industry 4.0: a review. *Engineering* 3: 616–630, https://doi.org/10.1016/J.ENG.2017.05.015.
5. Watkins CJCH, Dayan P (1992) Q-learning. *Machine learning* 8: 279–292. https://doi.org/10.1007/BF00992698.
6. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521: 436–444, https://doi.org/10.1038/nature14539.
7. Mnih V (2013) Playing atari with deep reinforcement learning. *arxiv preprint arxiv:1312.5602*.
8. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N (2016) Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.
9. Van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, 30. https://doi.org/10.1609/aaai.v30i1.10295.
10. Hessel M, Modayil J, Van Hasselt H, Schaul T, Ostrovski G, Dabney W, et al. (2018) Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, 32. https://doi.org/10.1609/aaai.v32i1.11796.
11. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust Region Policy Optimization. *arxiv preprint arxiv:1502.05477*.

12. Xian B, Dawson DM, de Queiroz MS, Chen J (2004) A continuous asymptotic tracking control strategy for uncertain nonlinear systems. *IEEE T Automat Contr* 49: 1206–1211. https://doi.org/10.1109/TAC.2004.831148.
13. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. *arxiv preprint arxiv:1707.06347*.
14. Galicki M (2016) Finite-time trajectory tracking control in a task space of robotic manipulators. *Automatica* 67: 165–170. https://doi.org/10.1016/j.automatica.2016.01.025.
15. Memarian F, Goo W, Lioutikov R, Niekum S, Topcu U (2021) Self-Supervised Online Reward Shaping in Sparse-Reward Environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2369–2375. https://doi.org/10.1109/IROS51168.2021.9636020.
16. Vecerik M, Hester T, Scholz J, Wang F, Pietquin O, Piot B, et al. (2017) Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*. https://doi.org/10.48550/arXiv.1707.08817.
17. Colas C, Sigaud O, Oudeyer PY (2018) Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *International conference on machine learning*, 1039–1048. PMLR.
18. Conti E, Madhavan V, Petroski Such F, Lehman J, Stanley K, Clune J (2018) Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in neural information processing systems*, 31.
19. Lehman J, Stanley K O (2008) Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, 329–336.
20. Pugh JK, Soros LB, Stanley KO (2016) Quality diversity: A new frontier for evolutionary computation. *Front Robot AI* 3: 202845. https://doi.org/10.3389/frobt.2016.00040.
21. Riedmiller M, Hafner R, Lampe T, Neunert M, Degrave J, Wiele T, et al. (2018) Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, 4344–4353.
22. Wang C, Wang J, Wang J, Zhang X (2020) Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards. *IEEE Internet of Things Journal* 7: 6180–6190. https://doi.org/10.1109/JIOT.2020.2973193.
23. Qi C, Zhu Y, Song C, Yan G, Xiao F, Zhang X, et al. (2022) Hierarchical reinforcement learning based energy management strategy for hybrid electric vehicle. *Energy* 238: 121703. https://doi.org/10.1016/j.energy.2021.121703.
24. Li J, Shi X, Li J, Zhang X, Wang J (2020) Random curiosity-driven exploration in deep reinforcement learning. *Neurocomputing* 418: 139–147. https://doi.org/10.1016/j.neucom.2020.08.024.
25. Liu IJ, Jain U, Yeh RA, Schwing A (2021) Cooperative exploration for multi-agent deep reinforcement learning. In *International conference on machine learning*, 6826–6836. PMLR.
26. Xia Y, Lan M, Luo J, Chen X, Zhou G (2022) Iterative rule-guided reasoning over sparse knowledge graphs with deep reinforcement learning. *Informa Process Manag* 59: 103040. https://doi.org/10.1016/j.ipm.2022.103040.
27. Cai M, Hasanbeig M, Xiao S, Abate A, Kan Z (2021) Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robot Autom Lett* 6: 7973–7980. https://doi.org/10.1109/LRA.2021.3101544.

28. Roghair J, Niaraki A, Ko K, Jannesari A (2022) A vision based deep reinforcement learning algorithm for UAV obstacle avoidance. *In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 1*, 115–128. Springer International Publishing. https://doi.org/10.1007/978-3-030-82193-7_8.
29. Eoh G, Park TH (2021) Automatic curriculum design for object transportation based on deep reinforcement learning. *IEEE Access* 9: 137281–137294. https://doi.org/10.1109/ACCESS.2021.3118109.
30. Christianos F, Schäfer L, Albrecht S (2020) Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems* 33: 10707–10717.
31. Hakak S, Alazab M, Khan S, Gadekallu TR, Maddikunta PKR, Khan WZ (2021) An ensemble machine learning approach through effective feature extraction to classify fake news. *Future Generation Computer Systems* 117: 47–58. https://doi.org/10.1016/j.future.2020.11.022.
32. Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications.
33. Neal RM (2001) Annealed importance sampling. Statistics and Computing.
34. Silver D, Singh S, Precup D, Sutton RS (2021) Reward is enough. *Artificial Intelligence* 299: 103535. https://doi.org/10.1016/j.artint.2021.103535.
35. Chen C, Hua Z, Zhang R, Liu G, Wen W (2020) Automated arrhythmia classification based on a combination network of CNN and LSTM. *Biomed Signal Process Contr* 57: 101819. https://doi.org/10.1016/j.bspc.2019.101819.
36. Liu X, Wen S, Hu Y, Han F, Zhang H, Karimi HR (2024). An active SLAM with multi-sensor fusion for snake robots based on deep reinforcement learning. *Mechatronics* 103: 103248. https://doi.org/10.1016/j.mechatronics.2024.103248.
37. Wen S, Shu Y, Rad A, Wen Z, Guo Z, Gong S (2025). A deep residual reinforcement learning algorithm based on Soft Actor-Critic for autonomous navigation. *Expert Syst Appl* 259: 125238. https://doi.org/10.1016/j.eswa.2024.125238.
38. Azar AT, Sardar MZ, Ahmed S, Hassanien AE, Kamal NA (2023) Autonomous robot navigation and exploration using deep reinforcement learning with Gazebo and ROS. *International Conference on Advanced Intelligent Systems and Informatics*, 287–299. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-43247-7_26.
39. Haider Z, Sardar MZ, Azar AT, Ahmed S, Kamal NA (2024) Exploring reinforcement learning techniques in the realm of mobile robotics. *Int J Autom Control* 18: 655–697. https://doi.org/10.1504/IJAAC.2024.142043.
40. Ahmed S, Azar AT, Tounsi M, Ibraheem IK (2023) Adaptive control design for Euler–Lagrange systems using fixed-time fractional integral sliding mode scheme. *Fractal and Fractional* 7: 712. https://doi.org/10.3390/fractalfract7100712.
41. Jahanshahi H, Zhu ZH (2024) Review of Machine Learning in Robotic Grasping Control in Space Application. *Acta Astronaut* 15. https://doi.org/10.1016/j.actaastro.2024.04.012.