THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 24, Number 3/2023, pp. 237–246
DOI: 10.59277/PRA-SER.A.24.3.05

ROBOTIC ARM TRAJECTORY TRACKING METHOD BASED ON IMPROVED PROXIMAL POLICY OPTIMIZATION

Qingchun ZHENG[1,2], Zhi PENG[3], Peihao ZHU[1,2], Yangyang ZHAO[4], Wenpeng MA[1,2]

[1] Tianjin University of Technology, School of Mechanical Engineering, Tianjin Key Laboratory for Advanced Mechatronic System Design and Intelligent Control, Tianjin 300384, China
[2] Tianjin University of Technology, National Demonstration Center for Experimental Mechanical and Electrical Engineering Education, Tianjin 300384, China
[3] Tianjin University of Technology, School of Mechanical Engineering, Tianjin 300384, China
[4] Tianjin University of Technology, School of Computer Science and Engineering, Tianjin 300384, China
Corresponding author: Peihao ZHU, E-mail: zhupeihao_gp@163.com

Abstract. To study the method of trajectory tracking for robotic arms, the traditional tracking method has low accuracy and cannot realize the complex tracking tasks. Compared with traditional methods, deep reinforcement learning is an effective scheme with the advantages of robustness and solving complex problems. This study aims to improve the tracking efficiency of robotic arms based on deep reinforcement learning. Thereby, we propose an approach to improve the proximal policy optimization (Improved-PPO) in this paper, which can be applied to multiple degrees of freedom robotic arms for trajectory tracking. In this study, proximal policy optimization (PPO) and model predictive control (MPC) are integrated to provide an effective algorithm for robotic arm applications. MPC is employed for trajectory prediction to design the controller. Further, the Improved-PPO algorithm is employed for trajectory tracking. The Improved-PPO algorithm is further compared with the asynchronous advantage actor-critic (A3C) and PPO algorithms. The simulation results show that the convergence speed of the Improved-PPO algorithm is increased by 84.3% and 15.4% compared with the A3C and PPO algorithms. This method provides a new research concept for robotic arm trajectory tracking.

***Key words**:* proximal policy optimization, model predictive control, trajectory tracking, robotic arm.

1. INTRODUCTION

With the continuous development of the global manufacturing industry, robotic arms are widely used in the fields of industry and transportation [1]. The robotic arm needs to plan the task path and track the trajectory to meet complex task requirements [2]. The complexity, strong coupling, and overfitting of robotic arms make trajectory tracking extremely challenging. Therefore, this paper further investigates an efficient method for robotic arm trajectory tracking.

Currently, the trajectory tracking algorithm is mainly based on traditional methods and deep reinforcement learning (DRL) algorithms. Carron et al. [3] proposed a model-based control method for robotic arms that uses collected offline data to realize offset-free tracking. Tang et al. [4] used a proximal policy optimization algorithm to train a neural network of robotic arms to solve trajectory planning problems in complex environments. However, this method uses a huge humanoid robot model for training, which requires a large training dataset and greatly increases the computational complexity.

Traditional tracking methods are not reliable and cannot achieve high performance. Therefore, deep reinforcement learning-based methods are effective approaches for trajectory tracking. We further investigate the work of robotic arms combining model predictive control (MPC) with deep reinforcement learning. Jiang et al. [5] proposed a multi-agent reinforcement learning-based trajectory planning for dual-arm robots, named multi-agent twin delayed deep deterministic policy gradient (MATD3), for a real-time inverse kinematics solution for continuum manipulators. Pang et al. [6] introduced two reinforcement

learning-based trajectory compensation methods. The proposed learning algorithms were evaluated on a 6-DoF industrial robot manipulator to follow different kinds of reference paths, such as square or circular paths, or to track trajectories on a three-dimensional surface. This method compares the performance of the learning-based methods with the MPC. Wen et al. [7] proposed a novel framework based on a fuzzy controller reinforcement learning trajectory planning strategy for robot manipulators with large errors and poor stability in the trajectory tracking process. This method plans an optimal motion trajectory through the reinforcement learning algorithm so that the end of the manipulator can track the optimal trajectory and realize effective obstacle avoidance. In recent literature, Shin et al. [8] proposed a new algorithm that combines meta-reinforcement learning with MPC, which is based on an offline strategy of meta-reinforcement learning and transformation samples generated by MPC to train the strategy. However, this method uses an offline strategy and heavily relies on MPC-generated samples for training.

As can be seen from the above research, the combination of model predictive control with deep reinforcement learning achieves superior results, but there is less research in the field of robotic arm trajectory tracking. Therefore, the trajectory tracking of the robotic arm is further studied. This paper proposes a method for trajectory tracking with improved proximal policy optimization (Improved-PPO). The main contributions of this paper can be described as follows:

(1) We propose an improved PPO approach, which can be applied to multiple degrees of freedom robotic arms for trajectory tracking.

(2) In this paper, PPO and MPC are integrated to provide an effective algorithm for robotic arm applications.

(3) Compared with the A3C and PPO algorithms, the convergence speed of the Improved-PPO algorithm is increased by 84.3% and 15.4%.

The main content of the work is described as follows. Section 2 presents the kinematic model of the robotic arm. Section 3 introduces the trajectory tracking algorithm based on deep reinforcement learning. In Section 4, the simulation results of different deep reinforcement learning algorithms are presented. Finally, in Section 5, we summarize our research progress.

2. KINEMATICS MODEL OF THE ROBOTIC ARM

In this section, the kinematic analysis of the robotic arm is presented. The end-effector of the robotic arm is taken as the object of study, and its position and posture changes are used as a reference to perform the trajectory action. The kinematic equations of the end-effector of the robotic arm are derived as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \tag{1}$$

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix}, \tag{2}$$

where $X=[x,y,\theta]^{\mathrm{T}}$ is defined as the actual spatial posture of the end-effector of the robotic arm in Cartesian space, and (x,y) represents the actual position of the end-effector of the robotic arm in Cartesian space. θ represents the joint angle of the end-effector of the robotic arm, which is the angle between link i and link $i-1$. $X_r=[x_r,y_r,\theta_r]^{\mathrm{T}}$ represents the reference posture of the end-effector in Cartesian space, and (x_r,y_r) represents the reference position of the end-effector in Cartesian space. θ_r is the reference joint angle of the end-effector of the robotic arm, $u=[v,w]^{\mathrm{T}}$ represents the actual control input of the whole robot system, and $u_r=[v_r,w_r]^{\mathrm{T}}$ represents the reference control input of the robot system. v is the linear velocity of the end-effector of the robotic arm, and w is the angular velocity of the end-effector of the robotic arm. v_r

represents the reference linear velocity of the end-effector of the robotic arm, and w_r represents the reference angular velocity of the end-effector of the robotic arm. Finally, the tracking error model of the robotic arm end-effector is obtained as follows:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_r - X \end{bmatrix}. \tag{3}$$

where $[x_e, y_e, \theta_e]$ is the error vector, (x_e, y_e) is the deviation between the actual position and the reference position, and θ_e is the angle deviation.

3. TRAJECTORY TRACKING ALGORITHM

3.1. Deep reinforcement learning

Deep reinforcement learning is an intelligent algorithm that combines the perception ability of deep learning (DL) with the decision-making ability of reinforcement learning (RL). DL uses convolutional neural networks to train value networks and policy networks to obtain information from the environment, which provides information about the state of the current environment to an intelligent agent. RL maps the current state to action to obtain the maximum reward.

At present, the deep reinforcement learning continuous control [9] algorithm mainly includes asynchronous advantage actor-critic (A3C) and proximal policy optimization (PPO). The A3C algorithm is an asynchronous multi-threading algorithm that optimizes the actor-critic method [10], which has better convergence. However, the A3C algorithm needs to run the data first to calculate the gradient and then update the global network.

In addition to the A3C algorithm, the PPO algorithm [11] is essentially an on-policy algorithm that can utilize the sampled samples numerous times, solving the problem of low utilization of samples to a certain extent and thus making up for the shortcomings of the A3C algorithm. Based on the policy gradient (PG), the PPO algorithm performs better while updating the policy problem offline, transforming the on-policy into an off-policy [12], and adding constraints to the objective function to form the PPO.

The PPO algorithm effectively solves the problems of PG learning rate and step size. If the step size is too large, the result is jittery and does not converge. The PPO algorithm uses the ratio of new and old strategies, which can solve the problem that the learning rate is difficult to determine in the PG algorithm. Therefore, this paper makes improvements based on the PPO algorithm.

3.2. Improved-PPO algorithm

To improve the robustness of the tracking algorithm, the PPO algorithm is improved based on the stable policy gradient. The PG algorithm calculates an unbiased estimate of the policy gradient and uses stochastic gradient ascent (SGA) to update the parameters θ. The gradient is estimated as:

$$\hat{g} = \hat{E}_t \left[\nabla_\theta \log \pi_\theta(a_t | s_t) \hat{A}_t \right], \tag{4}$$

where π_θ is the random policy, $\hat{A}_t$ is the estimate of the advantage function for the time step t, $\hat{E}_t$ is the empirical mean of a finite sample, and ∇ is the sign of the gradient.

However, the PG algorithm has low sampling efficiency, so the PPO algorithm introduces an important sampling method [13] to improve the sampling efficiency. The probability density function (PDF) of the network is $p(x)$. The probability density function of the network with another parameter is $q(x)$. As is shown in Equation (5).

$$E_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)} \right]. \tag{5}$$

where $p(x)/q(x)$ represents the weight. When the weight is too large, $\mathrm{E}_{x\sim q}$ is negative. When the number of samples is large enough, $\mathrm{E}_{x\sim q}$ is positive. The policy and maximization constraint is updated by the PPO algorithm:

$$\max_{\theta}\hat{\mathrm{E}}_t\left[\frac{\pi_{\theta_{new}}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right]. \tag{6}$$

$$\hat{\mathrm{E}}_t\left[KL\left[\pi_{\theta_{old}}(\cdot|s_t),\pi_{\theta_{new}}(\cdot|s_t)\right]\right]\le\delta. \tag{7}$$

$$L(\theta)=\hat{\mathrm{E}}_t\left[\frac{\pi_{\theta new}(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\hat{A}_t-\beta KL\left[\pi_{\theta_{old}}(\cdot|s_t),\pi_{\theta_{new}}(\cdot|s_t)\right]\right]. \tag{8}$$

where $r_t(\theta)=\pi_{\theta_{new}}(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$. β represents the penalty coefficient. θ_{old} represents the policy parameter before the update, and θ_{new} represents the policy parameter after the update.

Table 1

Pseudocode of the Improved-PPO algorithm

Input: (s_t,a_t,r_t), θ_{new}, θ_{old}
Output: KL, β

Initialize replay buffer in R
For $i\in\{1,\ldots,N\}$ do
  for $\mathrm{t}=1,\mathrm{T}$ do
    Select a_t based on the MPC policy and $\mathcal{N}$
    Obtain the predictive reward r_t and s_t
    Store transition (s_t,a_t,r_t) in R
  end for
  Run policy π_θ for T timesteps, collecting (s_t,a_t,r_t)
  Estimate advantages $\hat{A}_t=-V(s_t)+r_t+\gamma r_{t+1}+\ldots+\gamma^{T-t+1}r_{T-1}+\gamma^{T-1}V(s_t)$
  $\pi_{\theta_{old}}\leftarrow\pi_{\theta_{new}}$
  for $j\in\{1,\ldots,M\}$ do
    $J(\theta)=\max\mathrm{E}\left[r_t(\theta)A_{\theta_{old}}(s_t,a_t)-\beta KL(\pi_{\theta_{old}},\pi_{\theta_{new}})\right]$
    $\theta_{new}=\theta_{old}+\alpha\nabla_\theta J$
  end for
  for $j\in\{1,\ldots,O\}$ do
    $L_{OL}(\phi)=-\sum_{t=1}^{T}(\hat{A}_t)^2$
    Update ϕ by a gradient method w.r.t $L_{OL}(\varnothing)$
  end for
  If $KL<KL_{\text{targe}}\times 2$ then
    $\beta\leftarrow\beta/2$
  Else $KL>KL_{\text{targe}}/2$ then
    $\beta\leftarrow\beta\times 2$
  end**If**
ENDFor

The pseudocode of the Improved-PPO algorithm is shown in Table 1. KL [14] represents the scatter value of the policy before and after the update. When KL is less than twice the target value, β decreased it by half. When KL is greater than half of the target value, β increased by a factor of two [15].

4. SIMULATION RESULTS AND DISCUSSIONS

4.1. MPC simulation results

Model predictive control is a special control method that solves an open-loop optimal control problem, which is essentially an online rolling optimization. MPC has three basic features [16], which are a predictive model, rolling optimization, and feedback correction. MPC has the advantages of simple modeling, strong robustness, and stable performance.

The MPC designer toolbox is based on MATLAB to design the MPC controller. Then the MPC controller outputs the control variables to the robot arm. Finally, the arm outputs the variables to the state estimator. The MPC controller is further designed to adjust the stability of the controller and ensure that it can converge to a steady state. Finally, the parameters of the MPC controller in steady state are defined as a model prediction horizon of 15, a control horizon of 2, a sampling period of 1 unit, a step size of 5 between each period, and a simulation time of 30s. Then the minimum and maximum constraints are adjusted, and the weight coefficients are optimized to 0.6.

The MPC simulation results are shown in Fig. 1. The reference velocity and the reference angular velocity are shown as state inputs in Fig. 1a. The specific actual values of output are shown in Fig. 1b. To verify the superiority of MPC, the trajectory tracking experiment for a circular trajectory is shown in Fig. 1c. The variation curves of the lateral position x, longitudinal position y, and joint angle θ of the end-effector of the robotic arm tracking output are shown in Fig. 1d.

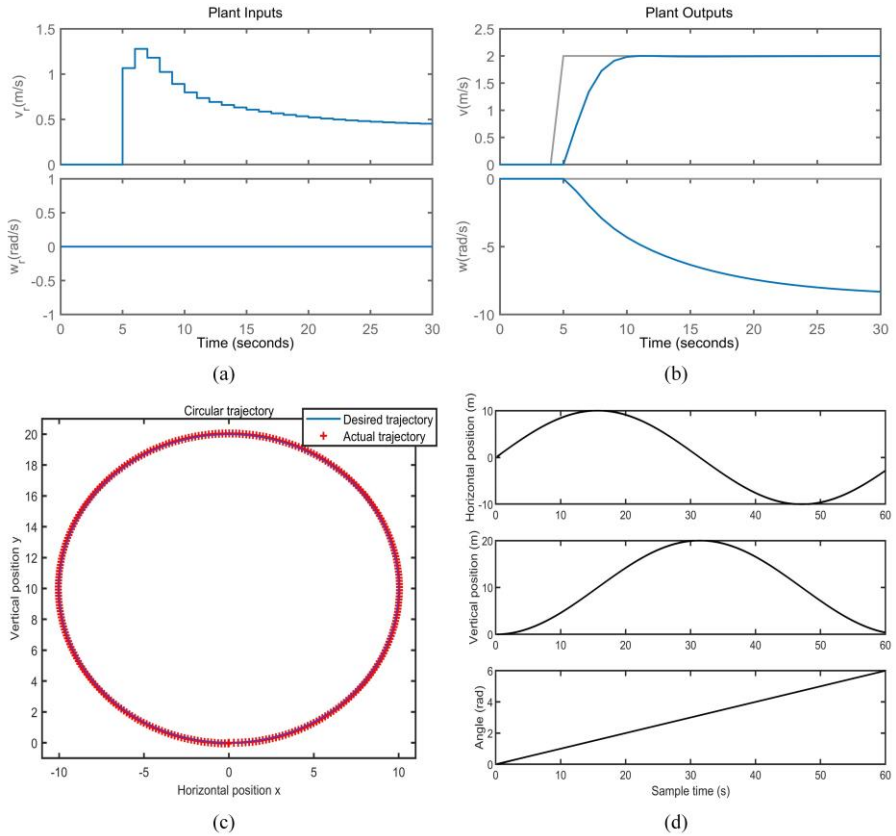

Fig. 1 – MPC simulation results: a) MPC reference value; b) MPC actual value; c) circular trajectory tracking diagram, d) position variation curve.

4.2. Improved PPO algorithm results

This simulation experiment is based on Python and MATLAB 2022a to build the simulation environment. The SIASUN SRC robot arm is trained based on the OpenAI Gym [17] development kit. In addition, we built a server running the Improved-PPO algorithm with an Intel(R) Core i7-11800H processor, 64G of RAM, and an NVIDIA GeForce RTX3090 GPU. The operating system is Windows 10. TensorFlow [18] is used to build the neural network model of the robotic arm. Numpy is used to process the data for learning.

After several simulations, the maximum number of training episodes for the Improved-PPO algorithm is 10 000, and the maximum number of steps is 100. The actor learning rate is 0.0001, and the critic learning rate is 0.0002. The reward discount factor is 0.99, and the difference between the old and new policies is set to 0.2.

The average reward curve of the Improved-PPO algorithm is shown in Fig. 2. The reward curve for the training set of 10000 with the same parameters. The reward curve drops to the lowest point at the beginning and then slowly rises to the highest average reward value of –102. The overall reward of the Improved-PPO algorithm is high.

To further verify the reliability of the Improved-PPO algorithm, we compare the Improved-PPO algorithm with the A3C and PPO algorithms. The average reward curve of the A3C and PPO algorithms is shown in Fig. 3. Figure 3a shows the average reward curve of the A3C algorithm. The A3C algorithm has the lowest average reward value among the three algorithms, reaching –820 at episode 167. The A3C algorithm does not train well and fluctuates a lot at the beginning of training. It converges around 5000 sets, and the overall reward is low. Fig. 3b shows the average reward curve of the PPO algorithm. The reward of the PPO algorithm fluctuates greatly, and the reward increases linearly from the lowest at the beginning of the training. Then it stabilizes around a reward of –600.

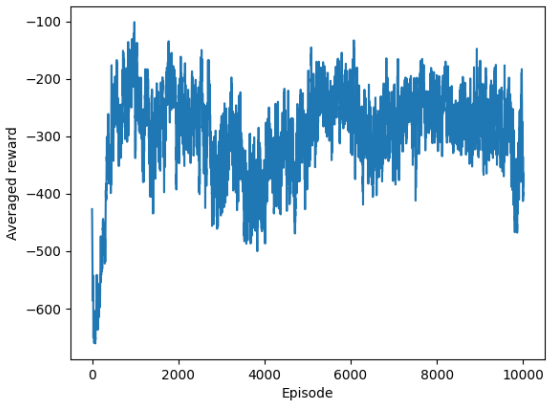

Fig. 2 – The average reward curve of the Improved-PPO algorithm.

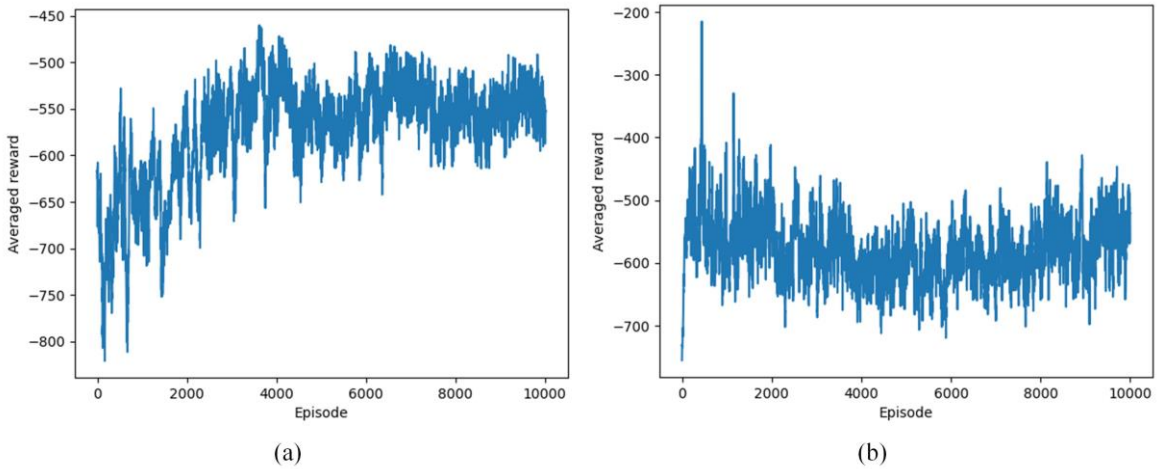

Fig. 3 – The average reward curve of the A3C and PPO algorithms: a) A3C algorithm; b) PPO algorithm.

The average reward values of the three different deep reinforcement learning algorithms are compared as shown in Table 2. The Improved-PPO algorithm has the highest reward of −102 at episode 570. It converges around episode 3500, and the convergence speed is superior to that of the A3C and PPO algorithms. Compared with the A3C algorithm, the Improved-PPO algorithm has improved the convergence speed by 84.3% and the reward value by 77.8%. Compared with the PPO algorithm, the Improved-PPO algorithm has improved the convergence speed by 15.4% and the reward value by 54.2%.

Table 2

The average reward values of the three algorithms

Algorithm	Minimum reward value	Episode	Maximum reward value	Episode
A3C	–820	167	-460	3624
PPO	–776	24	-223	674
Improved-PPO	**–708**	**30**	**-102**	**570**

4.3. Analysis of simulation results

In this paper, the simulation environment is built based on the virtual robot experimentation platform (V-REP) software. A remote application programming interface (API) [19] is used by V-REP for interactive simulation with Python. The Improved-PPO algorithm is applied to the simulation environment, and the robot arm end-effector is used as a mass tracking trajectory. The simulation is stopped when the last episode is finished.

The robotic arm repeatedly tracks the trajectory ten times, and the time to complete each track is recorded as shown in Table 3. From the results in Table 3, it can be seen that the shortest time among the ten experiments is the fourth experiment, with 46.3 s to complete the trajectory tracking. The results of the fourth V-REP simulation are shown in Fig. 4. As can be seen from Fig. 4a, the robot arm completes the trajectory tracking until the trajectory tracking is closed. From Fig. 4b, it can be seen that the robotic arm starts to track the trajectory after 12 s. The robotic arm further tracks the trajectory counterclockwise along the negative direction of the X-axis and the positive direction of the Y-axis. Finally, the trajectory tracking is finished with the robot arms at 46.3 s and reaches a steady state. Therefore, the simulation experiment verifies the effectiveness of the Improved-PPO trajectory tracking algorithm.

Table 3

The trajectory tracking completion time of the robotic arm

Group	Time	Group	Time
1	46.80 s	6	48.10 s
2	47.25 s	7	47.90 s
3	47.45 s	8	47.50 s
4	**46.30 s**	9	48.00 s
5	48.00 s	10	47.70 s

From the simulation experimental results in Fig. 4, it can be seen that there is no displacement change of the end-effector of the robotic arm when running the trajectory tracking action. Therefore, the joint angle of the end-effector is constant in the spatial coordinate system, and we further consider the position trajectory tracking variation of the end-effector of the robotic arm.

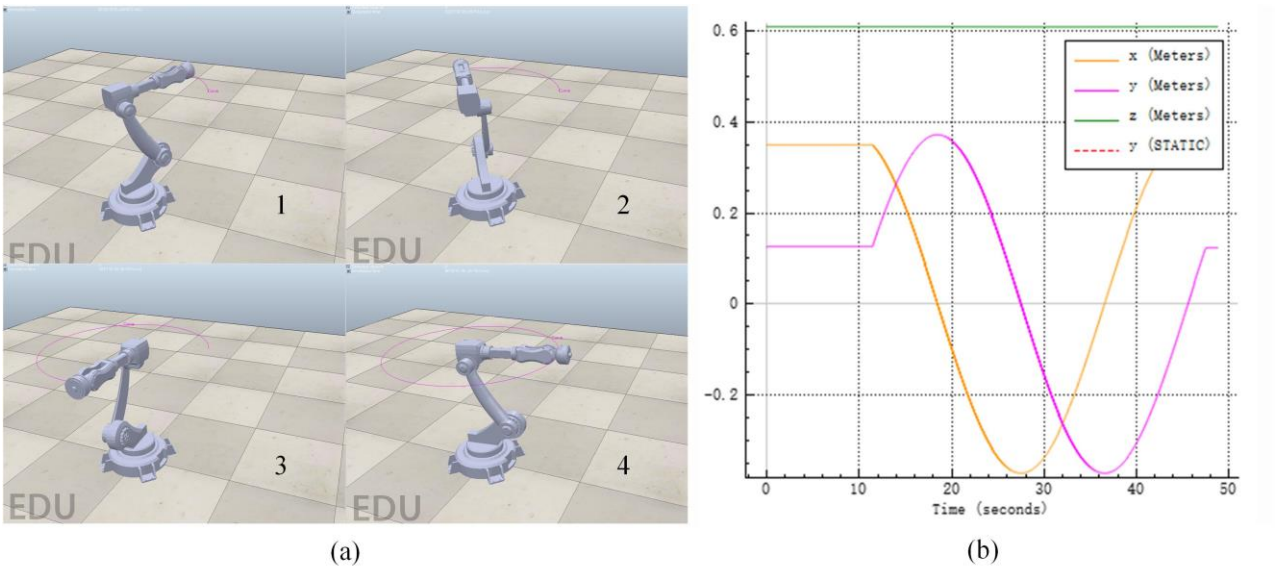

Fig. 4 – The results of the fourth V-REP simulation: a) robotic arm simulation of V-REP; b) simulation results.

The trajectory tracking curve of the end-effector of the robotic arm is shown in Fig. 5. The solid blue line in Fig. 5a is the expected trajectory of the robotic arm. The red solid line in Fig. 5b shows the actual trajectory of the robotic arm. The simulation results show that the Improved-PPO algorithm for robotic arm trajectory tracking is superior to the A3C and PPO algorithms. The Improved-PPO algorithm converges faster and has the shortest trajectory tracking time. The training process is stable, and the fluctuation is small.

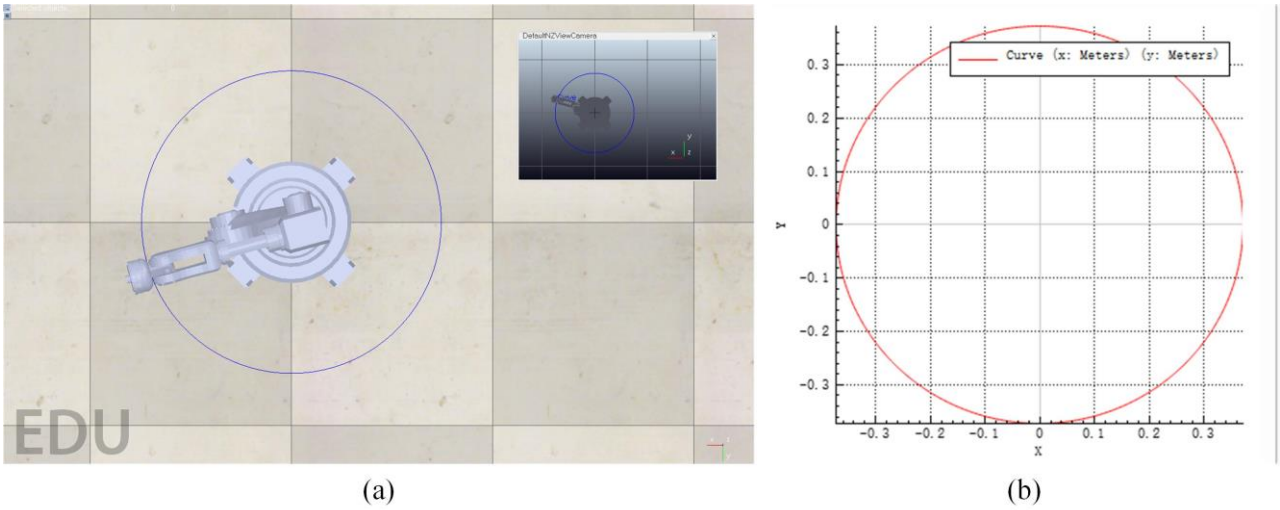

Fig. 5 – The end trajectory tracking curve of the robotic arm; a) the expected trajectory of the robotic arm; b) the actual trajectory of the robotic arm.

Finally, we overlap and compare the expected trajectory of the robotic arm in Fig. 5a with the actual trajectory of the robotic arm in Fig. 5b, and the comparison results are shown in Fig. 6. As can be seen in Fig. 6, the blue circle shape indicates the expected trajectory of the robotic arm corresponding to the trajectory in Fig. 5a, and the red cross shape indicates the actual trajectory of the robotic arm corresponding to the trajectory in Fig. 5b. The expected trajectory and the actual trajectory of the robotic arm perfectly coincide with each other, and the actual trajectory can track the expected trajectory very well, which shows a satisfactory performance. The comparison results demonstrate that the Improved-PPO algorithm proposed in this paper is superior to other traditional classical algorithms and can realize the trajectory tracking of the robotic arm with a certain degree of feasibility.

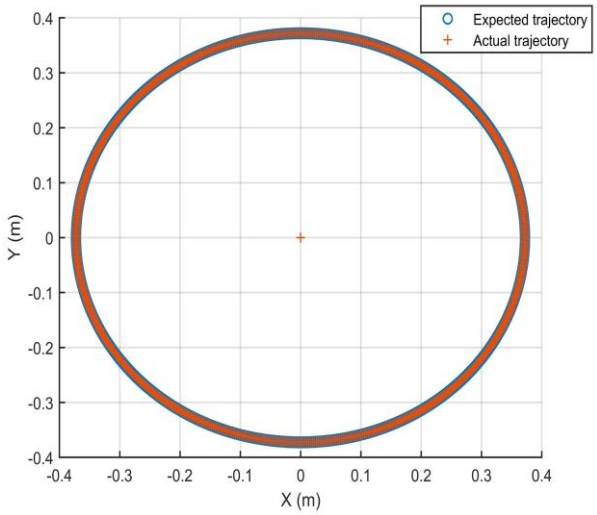

Fig. 6 – Comparison of trajectory tracking for the robotic arm.

5. CONCLUSIONS

In this paper, we propose an approach to improve proximal policy optimization that can be applied to multiple degrees of freedom robotic arms to track trajectory. MPC is employed for trajectory prediction to design the controller. Further, the improved PPO algorithm is employed for trajectory tracking. The Improved-PPO algorithm is further compared with the A3C and PPO algorithms. Compared with the A3C algorithm, the improved PPO algorithm increased the convergence speed by 84.3% and the reward value by 77.8%. Compared with the PPO algorithm, the Improved-PPO algorithm improved the convergence speed by 15.4% and increased the reward value by 54.2%. The simulation results show that the Improved-PPO algorithm outperforms the A3C and PPO algorithms for robotic arm trajectory tracking. The improved PPO algorithm converges faster and has the shortest trajectory tracking time. This method provides a new research idea for robotic arm trajectory tracking.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 62073239).

REFERENCES

1. D. RODRIGUEZ-GUERRA, G. SORROSAL, I. CABANES, C. CALLEJA, *Human-robot interaction review: Challenges and solutions for modern industrial environments*, IEEE Access, **9**, pp. 108557-108578, 2021.
2. K. XU, Z. WANG, The design of a neural network-based adaptive control method for robotic arm trajectory tracking, Neural Computing and Applications, **35**, pp. 8785–8795, 2023.
3. A. CARRON, E. ARCARI, M. WERMENLINGER, L. HEWING, M. HUTTER, M.N. ZEILINGER, *Data-driven model predictive control for trajectory tracking with a robotic arm*, IEEE Robotics and Automation Letters, **4**, *4*, pp. 3758–3765, 2019.
4. W. TANG, C. CHENG, H. AI, L. CHEN, Dual-arm robot trajectory planning based on deep reinforcement learning under complex environment, Micromachines, **13**, 4, art. 564, 2022.
5. D. JIANG, Z. CAI, H. PENG, Z. WU, Coordinated control based on reinforcement learning for dual-arm continuum manipulators in space capture missions, Journal of Aerospace Engineering, **34**, 6, 2021.
6. Y.P. PANE, S.P. NAGESHRAO, J. KOBER, R. BABUSKA, *Reinforcement learning based compensation methods for robot manipulators*, Engineering Applications of Artificial Intelligence, **78**, pp. 236–247, 2019.
7. S. WEN, X. HU, X. LV, Z. WANG, Y. PENG, Q-learning trajectory planning based on Takagi–Sugeno fuzzy parallel distributed compensation structure of humanoid manipulator, International Journal of Advanced Robotic Systems, **16**, 1, 2019.
8. J. SHIN, A. HAKOBYAN, M. PARK, Y. KIM, G. KIM, I. YANG, *Infusing model predictive control into meta-reinforcement learning for mobile robots in dynamic environments*, IEEE Robotics and Automation Letters, **7**, *4*, pp. 10065–10072, 2022.

9. Y. MA, W. ZHU, M.G. BENTON, J. ROMAGNOLI, *Continuous control of a polymerization system with deep reinforcement learning*, Journal of Process Control, **75**, pp. 40–47, 2019.
10. V. MNIH, A.P. BADIA, M. MIRZA, A. GRAVES, T. LILLICRAP, T. HARLEY, K. KAVUKCUOGLU, *Asynchronous methods for deep reinforcement learning*, Proceedings of the 33rd International Conference on Machine Learning, **48**, pp. 1928–1937, 2016.
11. J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, O. KLIMOV, *Proximal policy optimization algorithms*, arXiv preprint arXiv: 1707.06347, 2017.
12. A.A SHAHID, D. PIGA, F. BRAGHIN, L. ROVEDA, Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning, Autonomous Robots, **46**, 3, pp. 483–498, 2022.
13. J.P. HANNA, S. NIEKUM, P. STONE, *Importance sampling in reinforcement learning with an estimated behavior policy*, Machine Learning, **110**, *6*, pp. 1267–1317, 2021.
14. E. UCHIBE, K. DOYA, Forward and inverse reinforcement learning sharing network weights and hyperparameters, Neural Networks, **144**, pp. 138–153, 2021.
15. N. HEESS, T.B. DHRUVA, S. SRIRAM, J. LEMMON, J. MEREL, G. WAYNE, Y. TASSA, T. EREZ, Z. WANG, S.M. ALI ESLAMI, M. RIEDMILLER, D. SILVER, *Emergence of locomotion behaviours in rich environments*, arXiv preprint arXiv:1707.02286, 2017.
16. T. DING, Y. ZHANG, G. MA, Z. CAO, X. ZHAO, B. TAO, Trajectory tracking of redundantly actuated mobile robot by MPC velocity control under steering strategy constraint, Mechatronics, **84**, art. 102779, 2022.
17. N. SAJID, P.J. BALL, T. PARR, K.J. FRISTON, *Active inference: Demystified and compared*, Neural Computation, **33**, *3*, pp. 674–712, 2021.
18. B.S. KRONHEIM, M.P. KUCHERA, H.B. PROSPER, *TensorBNN: Bayesian inference for neural networks using TensorFlow*, Computer Physics Communications, **270**, art. 108168, 2022.
19. D. ZHOU, M. XIE, P. XUAN, R. JIA, *A teaching method for the theory and application of robot kinematics based on MATLAB and V-REP*, Computer Applications in Engineering Education, **28**, *2*, pp. 239–253, 2020.

Received February 1, 2023