Article

A Research on Manipulator Path Tracking based on Deep Reinforcement Learning

Pengyu Zhang[1], Jie Zhang[1] and Jiangming Kan[1,*]

[1] Key Laboratory of State Forestry Administration on Forestry Equipment and Automation, College of Engineering, Beijing Forestry University, People's Republic of China

* Correspondence: kanjm@bjfu.edu.cn; Tel.: +86-10-62338155

Abstract: We propose a deep reinforcement learning based manipulator path tracking method to solve the computationally difficult and non-unique problem of manipulator path tracking methods based on inverse kinematics. By transforming the path tracking task into a sequence decision problem, our method adopts an end-to-end learning method for closed-loop control and avoids the process of finding the inverse solution. We first explored the feasibility of the deep reinforcement learning method in the path tracking of the manipulator. After verifying the feasibility, the path tracking of the multi-degree-of-freedom(multi-DOF) manipulator was realized by combining the maximum entropy deep reinforcement learning algorithm. The experimental results show that our method has a good effect on the path tracking of the manipulator, which not only avoids the process of finding the inverse kinematics solution, but also requires no dynamic model. Therefore, we believe that our method has great significance in the study of manipulator path tracking.

Keywords: path tracking; deep reinforcement learning; maximum entropy; inverse kinematics

1. Introduction

A manipulator is a highly integrated mechanical system combined with electromechanical control. It is a typical multi-input multi-output nonlinear system, and its dynamics are time-varying and strongly coupled. As a complex system, it has many uncertainties, so its control is very complicated. In recent decades, researchers have carried out a lot of research on the control methods of manipulator. The existing control methods of robots include the computational torque control, robust adaptive control[1], adaptive neural network control [2], output feedback control [3-6], dead zone nonlinear compensation control [7], virtual decomposition control [8], and so on.

Path tracking [9] is an important topic in manipulator control. After successfully planning an optimal path using the path planning algorithm, how to make the end of the manipulator follow this optimal path is a problem that needs to be solved. Cai ZX [10] and Patolia H [11] decompose the speed and acceleration of each joint of the manipulator respectively. They adjust the desired speed or desired acceleration of each joint through position or speed feedback, and use the error as the control input. In the actual control of the manipulator, the end effector often clamps different objects, so it is difficult to accurately obtain the dynamic parameters of each link of the manipulator. At the same time, the existence of external interference and dynamic modelling errors make the manipulator track and control the belt. come difficult. Ma BL [12] proposed an adaptive control method, which dynamically adjusts the controller by identifying the system parameters online. In order to improve the stability of tracking, Spong MW [13] added a robust term to the control input to compensate for the deviation between the estimated model and the real model of the manipulator and limit the uncertain factors to a certain range. Purwar S [14] used the Lyapunov stability criterion to optimize the parameters of the neural network controller to improve the robustness and stability of the tracking in order to compensate for the error caused by the external disturbance and the linearization of the dynamic model.

The above methods all involve the process of precise dynamic modelling and inverse kinematics solutions. However, since the increase of the degree of freedom of the manipulator, the difficulty of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

dynamic modelling and the calculation amount of inverse kinematics solution will increase, and the inverse solution will not be unique. They all limit the applicability of methods [15-18]. In recent years, with a series of successful applications of reinforcement learning in decision control problems, it does not require system dynamics modelling, but only through trial and error learning by interacting with the environment, which provides new ideas for path tracking problems. The reinforcement learning [19] method can model the manipulator path tracking problem as a Markov decision process (MDP), and interact with samples to optimize the control policy. Guo X [20] used the depth value function method (DQN) to complete the path tracking of the UR5 manipulator. However, it discretizes the action space, and it is difficult to finely control the manipulator. Liu YC [21] realized the tracking control of the manipulator in the continuous action space, but its learning process is unstable, and the small changes of hyperparameters have a great impact on the performance of the algorithm. In the path tracking problem, since the output of the system follows a time-independent geometric path, most articles use the inverse kinematics method for path tracking. However, the manipulators capable of performing complex tasks usually have a high degree of freedom. As the degrees of freedom increase, the solutions of inverse kinematics will not be unique and more difficult, also the computational load will be larger.

In this paper, a reinforcement learning method for multi-DOF manipulator path tracking is proposed, which converts the tracking accuracy requirements and energy constraints into cumulative rewards obtained by the control strategy to ensure the stability and control accuracy of the tracking trajectory. The entropy of policy is used as an auxiliary gain of the agent and introduced into the training process of the control strategy, thereby increasing the robustness of the path tracking. We first explore the feasibility of deep reinforcement learning methods to solve the path following problem on a planar dual-link manipulator. Then, we verified it on the 6-DOF manipulator and analysed the effects of different training parameters and different dynamic characteristics on the path tracking effect of the manipulator. The results show that our method has good path tracking performance.

2. Problem Statement

We model the path tracking problem of manipulator as a Markov Decision Process (MDP), which represented by $< S,\ A, R, T, \gamma >$, where $s_t \in S$ represents the observations of the agent. The policy $\Pi: S \to A$ maps the current environmental state s_t to the control input $a_t \in A$ of each joint of the manipulator, $T(s_{t+1}|s_t, a_t)$ represents the dynamic characteristics of the robotic arm, that is, the probability that the system transitions from state s_t to s_{t+1} under the control of a_t. The expected path $P^* \in \mathbb{R}^{N\times 3}$ of the manipulator can be generated by traditional path planning methods, where N is the number of points on the path. The instantaneous reward obtained by the agent at time t is represented by $r_t \in R$,which is related to the tracking accuracy of the robot arm on the desired path and the energy consumed. The policy continuously interacts with the manipulator system to obtain the sampling trajectory $\tau = \{s_0, a_0, \cdots, s_t, a_t, \cdots, s_T, a_T\}$. The goal of reinforcement learning is to optimize the policy so as to maximize the expected cumulative reward obtained by the agent:

$$\text{maximize}_{\pi\in\Pi}\ \mathrm{E}_{\tau\sim p(\tau)}\left[\sum_{t=0}^{T}\gamma^t r\left(s_t, a_t\right)\right] \tag{1}$$

$$p(\tau) = p(s_0)\prod_{t=0}^{T}\pi\left(a_t \mid s_t\right)T\left(s_{t+1} \mid s_t, a_t\right) \tag{2}$$

The framework of the robot arm path tracking model based on deep reinforcement learning is shown in Figure **1**. The framework consists of four parts: desired path, control strategy, feedback controller, and manipulator body. The policy calculates the expected position/velocity of each joint at the next moment according to the desired path and the current state of the manipulator as the reference signal of the feedback controller. The feedback controller combines the current position and speed information of each joint to output the required joint torque to change the position of the end-point of the manipulator.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

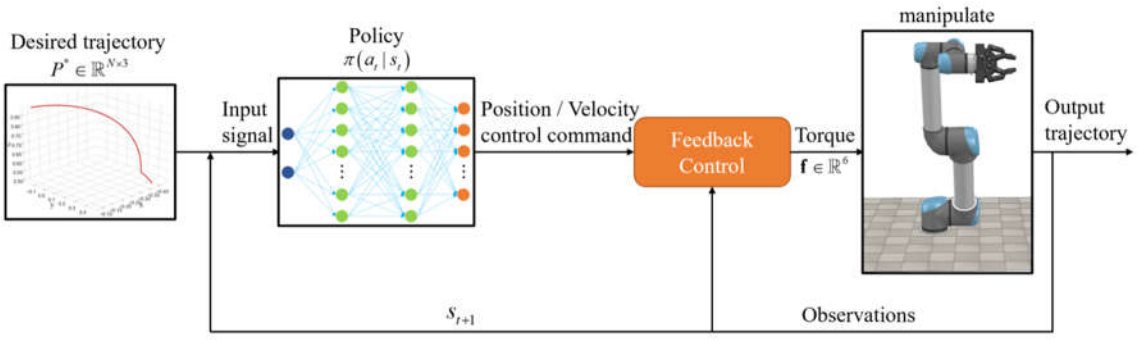

Figure 1. The framework of the robot arm path tracking model based on deep reinforcement learning

3. Method

3.1. Deep Q Network

The Deep Q-Network (DQN) algorithm [22] is a classic algorithm based on the value function method in deep reinforcement learning, which was originally derived from the Q-Learning algorithm in classical reinforcement learning. Q-Learning is an algorithm based on the Q value, which is defined as the expected future cumulative reward value of action a (action a must be finite and discrete) taken according to policy π in state S:

$$Q_\pi(s,a) = \mathrm{E}_\pi\left[\sum_{t=0}\gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a\right] \tag{3}$$

The optimal value Q^* is defined as the maximized Q value, and the strategy that can get the optimal value is defined as the optimal strategy π^*. DQN uses a deep neural network $Q_\pi(s,a;\theta)$ with parameters θ to replace the Q value $Q_\pi(s,a)$, which can make the input The algorithm is still valid in the case of high-dimensional and continuous state S. In addition, the experience playback pool Replay Buffer and a target Q network with parameters θ' are also added to DQN. The Replay buffer improves the utilization efficiency of samples, and the use of the target Q network solves the loss in the neural network function problem. The target Q network is defined as follows:

$$y = r + \gamma \max_{a'} Q(s',a';\theta') \tag{4}$$

Therefore, this problem can be transformed into a supervised learning problem to solve, that is $\min_\theta \sum(y - Q(s,a;\theta))$, where θ every τ steps copy their own parameters to θ'.

3.2. Soft Actor Critic

Although the DQN algorithm, which is a milestone in deep reinforcement learning, solves the problem of high-dimensional and continuous input states that cannot be solved by classical reinforcement learning, it still cannot solve the situation where the output actions are high-dimensional and continuous (such as multi-degree-of-freedom manipulator). Although the other deep reinforcement learning algorithms (such as DDPG [23], TD3 [24], and other algorithms) can handle the case where the output action is high dimensional and continuous, they usually have a high sample complexity and weak sample convergence, which lead to some additional hyperparameter tuning.

The Soft Actor-Critic (SAC) algorithm [25] is a reinforcement learning algorithm that introduces the maximum entropy theory. In the framework of the algorithm, the strategy not only needs to maximize the expected cumulative reward value, but also needs to maximize the expected entropy:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

$$\pi^* = \max \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\pi}[r(s_t,a_t) + \alpha H(\pi(\cdot \mid s_t))] \tag{5}$$

where α is the weight of the entropy term, which can determine the relative importance of the entropy term relative to the reward term, thereby controlling the randomness of the optimal strategy.

In order to maximize this goal, a method of alternating policy evaluation and policy improvement is used in the maximum entropy framework - Soft Policy Iteration. In an environment where the state space is discrete, the method can obtain the soft Q-value from the randomly initialized function $Q: S \times A \to R$ and repeatedly apply the modified Bellman backup operator T^π :

$$T^\pi Q(s_t,a_t) = r(s_t,a_t) + \gamma E_{s_{t+1}\square p}[V(s_{t+1})] \tag{6}$$

where

$$V(s_t) = E_{a_t\sim\pi}[Q(s_t,a_t) - \log \pi(a_t \mid s_t)] \tag{7}$$

is the soft state value function used to calculate the policy value in policy evaluation. While in the continuous state, a neural network with parameters is first used to replace the soft Q-function $Q_\theta(s_t,a_t)$, and then it is trained to minimize the Bellman residual:

$$J_Q(\theta) = E_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma E_{s_{t+1}\sim p}[V_{\bar{\theta}}(s_{t+1})]\right)\right)^2\right] \tag{8}$$

which can also be optimized with stochastic gradients:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t,a_t)\left(Q_\theta(s_t,a_t) - r(s_t,a_t) - \gamma V_{\bar{\theta}}(s_{t+1})\right) \tag{9}$$

where $V_{\bar{\theta}}(s_{t+1})$ is estimated by the target network of Q and the Monte Carlo estimation of the soft state value function sampled from the experience pool.

Policy improvement is updating the policy in the direction of maximizing its available reward, therefore, the policy needs to be updated to the exponential form of the new soft Q-function and restricted to some parameterized distribution (such as Gaussian distribution) and then project it back into the acceptable policy space using an information projection defined in terms of the Kullback-Leibler (KL) divergence.

$$\pi_{new} = \arg\min_{\pi'\in\Pi} D_{KL}\left(\pi'(\cdot \mid s_t) \square \frac{\exp(Q^{\pi_{old}}(s_t,\cdot))}{Z^{\pi_{old}}(s_t)}\right) \tag{10}$$

where $Z^{\pi_{old}}(s_t)$ can be ignored because it has no effect on the gradient. And parameterize the policy $\pi_\phi(a_t|s_t)$ with a neural network that can output mean and variance to define a Gaussian distribution, and then learn the parameters of the policy by minimizing the expected KL divergence:

$$J_\pi(\phi) = E_{s_t\sim D}\left[E_{a_t\sim\pi_\phi}\left[\log \pi_\phi(a_t \mid s_t) - Q_\theta(s_t,a_t)\right]\right] \tag{11}$$

However, since the Gaussian distribution $a\sim N(m,s)$ is difficult to find its gradient, it is converted into an easy-to-find gradient form $a = m + s\varepsilon,\ \varepsilon\sim N(0,1)$, i.e. $a_t = f_\phi(\varepsilon_t,s_t)$, the policy network can then be optimized by applying the policy gradient to the expected future reward:

$$J_\pi(\phi) = E_{s_t\sim D,\varepsilon_t\sim N}\left[\log \pi_\phi(f_\phi(\varepsilon_t;s_t) \mid s_t) - Q_\theta(s_t,f_\phi(\varepsilon_t;s_t))\right] \tag{12}$$

We can also approximate the gradient of Equation (12) with:

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t \mid s_t) + \left(\nabla_{a_t} \log \pi_\phi(a_t \mid s_t) - \nabla_{a_t} Q(a_t \mid s_t)\right)\nabla_\phi f_\phi(\varepsilon_t;s_t) \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

4. Experiments and Results

4.1. Planar two-link manipulator

The plane two-link manipulator simulation system is shown in Figure **2**, and the simulation platform adopts V-REP PRO EDU 3.6.0. The settings of the two-link manipulator in the simulation environment are as follows. The length of the rods are $l_1 = 1.0m, l_2 = 0.8m$, and the mass of the rods are $m_1 = 0.1kg, m_2 = 0.08kg$. Each joint adopts the incremental control method, that is, the joint rotates a fixed angle $|\Delta\theta|$ in the direction given by the control signal a_t at any time t; where $a_t := [\Delta\theta_t^1, \Delta\theta_t^2] \in \mathbb{R}^2$, $|\Delta\theta_t^i| = 0.05^\circ, i = 1,2$. The state of the entire simulation system $s_t := [\theta_t^1, \theta_t^2, \dot{\theta}_t^1, \dot{\theta}_t^2, x_t, y_t, x_t^*, y_t^*] \in \mathbb{R}^8$, where $\theta_t^i, \dot{\theta}_t^i$ are the angle and angular velocity of each joint at ith time. (x_t, y_t) is the position of the endpoint of the manipulator at time t, and the desired target point position (x_t^*, y_t^*) is set as follows:

$$\begin{cases} x_t^* = l_2 \cos(\omega_1 t + \omega_2 t) + l_1 \cos(\omega_1 t) \\ y_t^* = l_2 \sin(\omega_1 t + \omega_2 t) + l_1 \sin(\omega_1 t) \end{cases}, \quad \omega_1 = \omega_2 = 1 \text{ rad/s} \tag{14}$$

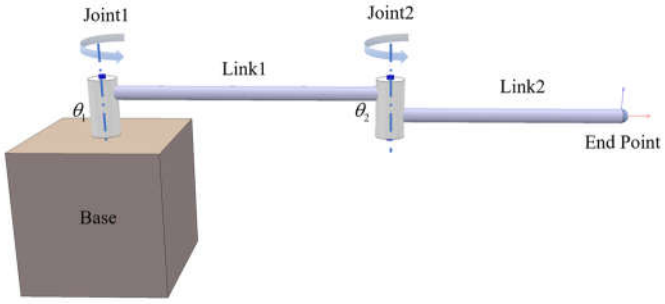

Figure 2. Simulation system of two-link manipulator

This experiment is mainly used to verify the feasibility of the reinforcement learning algorithm in the field of robotic arm path tracking. Because the experimental environment is simple and the output action dimension of the two-link robotic arm is low, it can be approximated as a discrete quantity, so the algorithm adopts the classic DQN algorithm. The strategy network structure in the DQN algorithm is: the input state is 8-dimensional, the output action is 2-dimensional, the hidden layer has two layers and the number of nodes in each layer is 50. The hyperparameters are set as: replaybuffer = 1e6, learning-rate = 3e-4, discount-factor = 0.99, batch-size = 64, the update between the Q network and the target Q network adopts the soft update method, and its soft parameter tau=0.001. In addition, the setting of the reward is: $r_t = exp(-|p_t^* - p_t|)$, where p_t^*, p_t are the target path points at time t respectively and the position of the end point of the robot arm. The tracking curve results of this experiment are shown in Figure **3**:

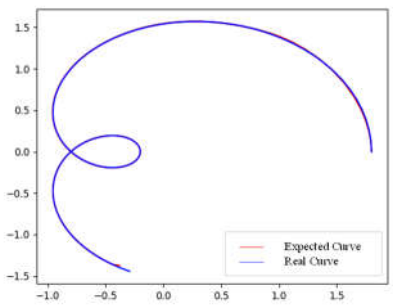

Figure 3. Path tracking curve of two-link manipulator based on DQN

The red line is the desired target path, and the blue line is the actual running end path. From the tracking results in Figure **3**, it can be seen that the method based on deep reinforcement learning

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

perfectly achieves the tracking of the target path. Figure **4** shows the experimental results in the simulation environment:

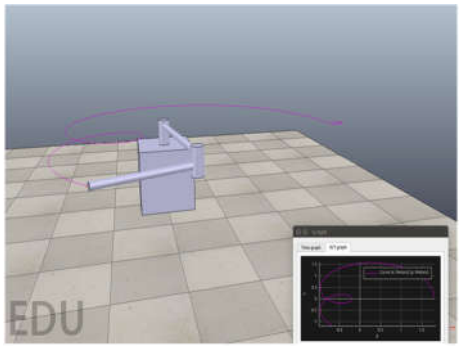

Figure 4. Simulation results of path tracking of two-link manipulator based on DQN

The experimental results show that it is completely feasible to use the deep reinforcement learning algorithm to achieve path tracking on a simple two-link manipulator.

4.2. UR5 manipulator

After exploring the application of reinforcement learning in path tracking, and successfully applied to the two-link manipulator to achieve the tracking target. We will further explore the application of the multi-degree-of-freedom manipulator-UR5 to realize path tracking under continuous control. The UR5 simulation system is shown in Figure **5**, and the simulation platform still uses V-REP PRO EDU 3.6.0. The system is used to realize the path tracking by using the deep reinforcement learning algorithm after the path is generated by the traditional path generation algorithm when there are obstacles. The system actions $a \coloneqq [a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,] \in \mathbb{R}^6$, and states set as $s \coloneqq [\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \dot{\theta}_1,\ \dot{\theta}_2,\ \dot{\theta}_3,\ \dot{\theta}_4,\ \dot{\theta}_5,\ \dot{\theta}_6,\ \Delta x,\ \Delta y,\ \Delta z] \in \mathbb{R}^{15}$, where $\theta_i,\ \dot{\theta}_i$ is the angle and angular velocity of the first joint, $\Delta x,\ \Delta y,\ \Delta z$ are the distance between the endpoints p and the corresponding desired target points p^*. The initial position of the endpoint: [-0.095, -0.160, 0.892], the initial position of the target point: [-0.386, 0.458, 0.495]. The desired path is generated by the traditional RRT [26] path generation algorithm with the stride set to 100.

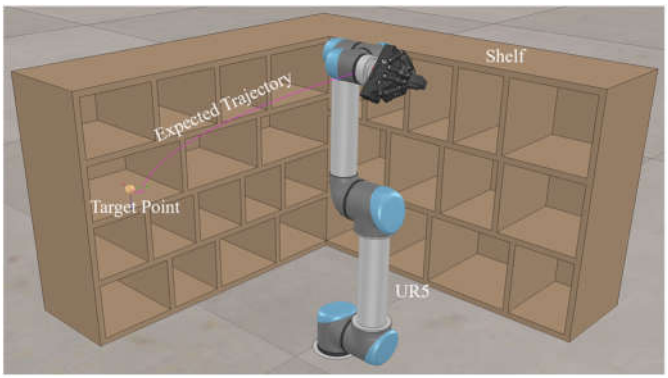

Figure 5. UR5 simulation system

In addition, this experiment set up 4 additional variables to explore the impact of these factors on tracking performance:

1. The upper control method of the manipulator adopts two control methods, position control or velocity control. The position control is the control of the joint angle, and the input action is the increment of the joint angle. The range of the increment at each moment is set as [- 0.05, 0.05] rad. The velocity control is the control of joint angular velocity. The input action is the increment of joint angular velocity. The increment range of each moment is set to [-0.8, 0.8] rad/s in the experiment. In addition, the underlying control of the manipulator adopts the traditional PID torque control algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

2. Adding the noise to the observations: We set up two groups of control experiments, one of which adds random noise to the observations, the noise is adopted from the standard normal distribution N(0,1), the size is 0.005 *N(0,1).
3. Setting the time interval distance n. The target path points given by the manipulator at every n time are the target path points at the N*n time, where N=1, 2, 3..., and study the effect of different interval points on the tracking results. In our experiments, we set the interval distance interval=0, 5, 10 respectively.
4. Terminal reward. Setting up a control experiment that during the training process, when the distance between the endpoint of the robotic arm and the target point is within 0.05m (the termination condition is met), an additional +5 reward is given to study its impact on the tracking results.

The continuous control reinforcement learning algorithm SAC is used in this experiment. All network structures are as follows: each network contains two hidden layers, the number of nodes in each layer is 200, and the activation function of the hidden layer is set to Relu. The hyperparameters are set as: replaybuffer = 1e6, discount-factor = 0.99, batch-size = 128, the update between the Q network and the target Q network adopts the soft update method, the soft parameter tau=0.01, and the learning rate of Actor and Critic network are both set to learning-rate = 1e-3, the weight coefficient of policy entropy during the entire training process α=1e-3. The reward settings for this experiment are:

$$r_t = -\left\| p^*_{\tau(t,n)} - p_t \right\| \quad , \quad \tau(t,n) = \begin{cases} n\left(\text{floor}\left(t/n\right)+1\right) & n>1 \\ t & n=1 \end{cases} \tag{15}$$

where n is the interval distance. In addition, an experiment is terminated when the robot arm runs for 100 steps or the distance between the end point of the robot arm and the target point is within 0.05m.

The experimental results are shown in the following figures. Figures 6(a) and 6(b) are the path tracking results without observation noise and with observation noise in the position control mode, respectively. Figures 6(c) and 6(d) are the speed control results, respectively. The path tracking results with and without observation noise in the mode. Different time intervals are set in each picture, and the upper three curves of each picture are the results of not giving the terminal reward, and the lower three curves are the results of giving the terminal reward.

In addition, this experiment also quantitatively analyzed the tracking results, and calculated the average error between the obtained path and the target path under different experimental conditions and the average distance between the endpoint of the manipulator and the target point at the last moment. The results are shown in Table **1** and Table **2** shows:

Table 1. Results of position control mode path tracking

Position Control		w/o observation noise			observation noise		
		interval			interval		
		1	5	10	1	5	10
Average error between tracks (m)	w/o terminal reward	0.0374	0.0330	0.0592	0.0394	0.0427	0.0784
	terminal reward	0.0335	0.0796	0.0502	0.0335	0.0475	0.0596
Distance between end-point (m)	w/o terminal reward	0.0401	0.0633	0.0420	0.0443	0.0485	0.0292
	terminal reward	0.0316	0.0223	0.0231	0.0111	0.0148	0.0139

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

Table 2. Results of velocity control mode path tracking

Velocity Control		w/o observation noise interval			observation noise interval		
		1	5	10	1	5	10
Average error between tracks(m)	w/o terminal reward	0.0343	0.0359	0.0646	0.0348	0.0318	0.0811
	terminal reward	0.0283	0.0569	0.0616	0.0350	0.0645	0.0605
Distance between end-point (m)	w/o terminal reward	0.0233	0.0224	0.0521	0.0456	0.0365	0.0671
	terminal reward	0.0083	0.0030	0.0337	0.0275	0.0192	0.0197

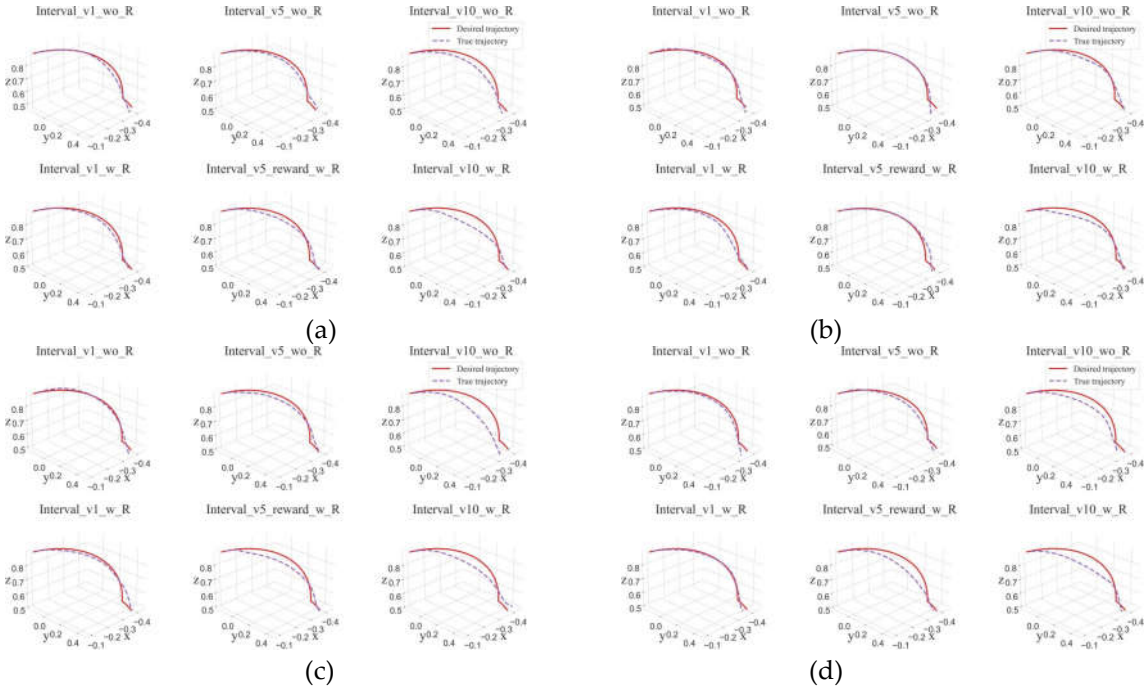

Figure 6. Path tracking results of UR5 manipulator based on maximum entropy reinforcement learning (a) Tracking results without observation noise in position control mode (b) Tracking results with observation noise in position control mode (c) Tracking results without observation noise in velocity control mode (d) Tracking results with observation noise in velocity control mode

The training process curve is shown in Figure 7:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

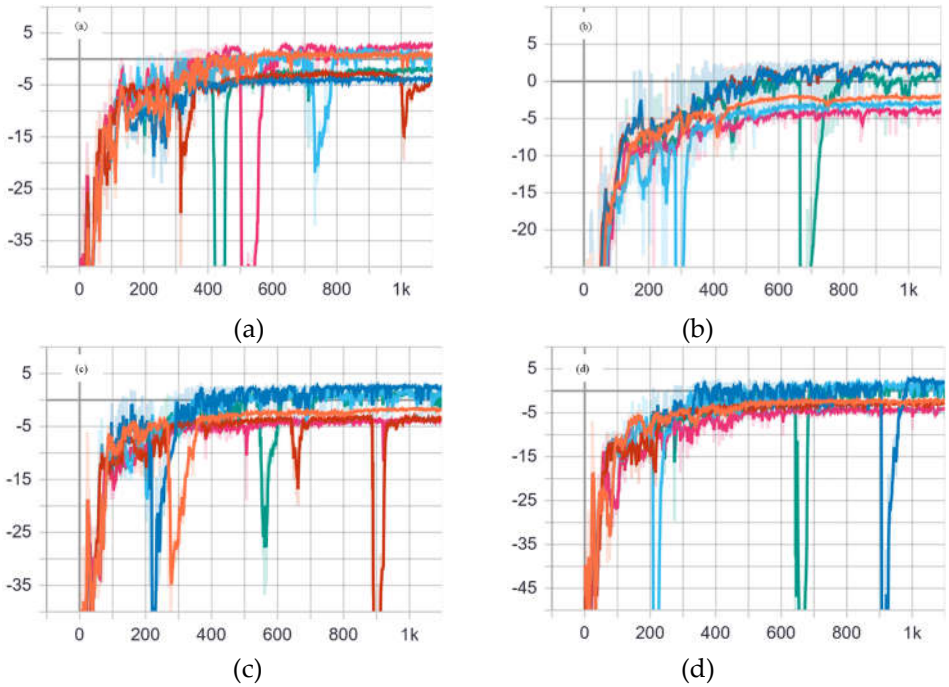

Figure 7. Training process curve (a)Training process curve without observed noise in position control mode (b) Training process curve with observation noise in position control mode (c)Training process curve without observed noise in speed control mode (d) Training process curve with observation noise in speed control mode

Through the path tracing results, it can be found that:

1. In this experimental scenario, the target path generated by the RRT algorithm has obvious non-smoothness; while the tracking path generated by the reinforcement learning algorithm SAC based on the target path is also very smooth under the condition that the tracking accuracy is satisfied.
2. By analyze the influence of the n value on the generated path, it can be found that when the n value is too large (n=10), its approaching effect on the target point is better, but its tracking effect on the target path is poor. But when n=1, the situation is opposite. Therefore, when selecting the value of n, it is necessary to balance the path tracking effect and the final position of the endpoint.
3. Adding noise to the observations of the system during the simulation training process helps to improve the robustness of the control strategy and the anti-interference to noise, so that the strategy has better performance.
4. During the simulation training process, when the endpoint of the manipulator reaches the allowable error range of the target point, adding a larger reward to the current strategy can make the approaching result of the robotic arm to the target point better, but it will lose some precision of path tracking;
5. Experiments show that the algorithm achieves good results in both position control and velocity control, and it can be seen from the curve of the training process that the curve can converge at an earlier time.

In addition, since the system dynamics model is not considered in the path tracking experiment based on deep reinforcement learning. In order to verify the advantage of the method based on deep reinforcement learning that does not need the dynamic model, we further explore the influence of the change of dynamic characteristics on the experimental results. So, we change the quality of the end effector, the trained model is tested, and the experimental results are shown in Table **3** and Table **4**.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

Table 3: Analysis of Dynamic Characteristics of Position Control

Position Control			0.5kg	1kg	2kg	3kg	5kg
Average error between tracks(m)	w/o observation noise	w/o terminal reward	0.03742	0.03743	0.03744	0.03745	0.03746
		terminal reward	0.03354	0.03354	0.03359	0.03355	0.03355
	observation noise	w/o terminal reward	0.03943	0.03943	0.03943	0.03942	0.03941
		terminal reward	0.03346	0.03346	0.03346	0.03346	0.03345
Distance between end-point(m)	w/o observation noise	w/o terminal reward	0.04047	0.04047	0.04048	0.04049	0.04050
		terminal reward	0.03165	0.03166	0.03157	0.03159	0.03161
	observation noise	w/o terminal reward	0.04430	0.04441	0.04438	0.04430	0.04436
		terminal reward	0.01110	0.01109	0.01109	0.01108	0.01105

Table 4: Analysis of Dynamic Characteristics of Velocity Control

Velocity Control			0.5kg	1kg	2kg	3kg	5kg
Average error between tracks(m)	w/o observation noise	w/o terminal reward	0.03426	0.03427	0.03425	0.03426	0.03425
		terminal reward	0.02826	0.02825	0.02866	0.02873	0.02882
	observation noise	w/o terminal reward	0.03478	0.03479	0.03483	0.03486	0.03497
		terminal reward	0.03503	0.03503	0.03503	0.03502	0.03501
Distance between end-point(m)	w/o observation noise	w/o terminal reward	0.02326	0.02444	0.02436	0.02430	0.02422
		terminal reward	0.00831	0.01201	0.01395	0.01463	0.01513
	observation noise	w/o terminal reward	0.04560	0.04562	0.04565	0.04569	0.04578
		terminal reward	0.02748	0.02746	0.02743	0.02741	0.02733

The experimental results show that in the model trained under the condition of fixed load quality, the path tracking based on deep reinforcement learning can still ensure sufficient stability when the load changes. That is, the change of dynamic characteristics will not affect the algorithm effect.

Furthermore, we also compare the results of the proposed algorithm and the traditional inverse kinematics method in terms of energy consumption and trajectory smoothness. The experimental results are shown in Tables **5** and **6**. Among them, the calculation method of the energy consumption of the manipulator during the entire path tracking process is:

$$E = \int_{t_0}^{t_M} P(t)dt \cong \sum_{k=0}^{M} P(k) \cdot dt \tag{16}$$

$$P(k) = \sum_{i=1}^{n} P_i(k) \tag{17}$$

$$P_i(k) = \tau_i(k) \cdot \dot{\theta}_i(k) \tag{18}$$

where k is the kth path point in the entire path, i is the ith joint of the manipulator, $\tau, \dot{\theta}$ is the joint torque and joint speed, M is the number of path points, dt is the distance between the path-points. The smoothness of the trajectory is measured by the angle between the tangent vectors of adjacent points of the curve, and the degree of smooth movement of the robotic arm is measured by analyzing the mean value of the turning angles in the entire trajectory.

Table 5. Analysis of Track Smoothness

Velocity control	w/o terminal reward			terminal reward			Jacobian matrix
Interval	1	5	10	1	5	10	
Smoothness	0.5751	0.3351	0.5925	0.0816	0.5561	0.4442	0.7159

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

Table 6. Analysis of energy consumption

Energy consumption		0.5kg	1kg	2kg	3kg	5kg
Position Control	w/o terminal reward	4.44438	4.71427	5.27507	5.79426	6.92146
	terminal reward	5.01889	5.34258	5.95310	6.55227	7.76305
Velocity Control	w/o terminal reward	4.97465	5.38062	6.23886	6.95099	8.33596
	terminal reward	6.03735	6.37981	7.05696	7.75185	9.15828
Traditional	Jacobian matrix	8.95234	9.81593	10.8907	10.9133	13.3241

The experimental results show that the algorithm proposed in this paper is superior to the traditional inverse kinematics method in terms of energy consumption and trajectory smoothness.

4.3. Redundant manipulator

The algorithm proposed in this paper is experimentally verified and analyzed on the UR5 manipulator. The experimental results show that the algorithm proposed in this paper can effectively solve the path tracking problem of the manipulator. In order to further verify the effectiveness and generalization of the algorithm in this paper, we also conduct the verification on a redundant manipulator. The 7-DOF redundant manipulator simulation system is shown in Figure **8**. The simulation platform still uses V-REP PRO EDU 3.6.0, and the simulated manipulator uses the KUKA LBR ii wa 7 R800 redundant manipulator. The setting of actions is $a \coloneqq [a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7] \in \mathbb{R}^7$, and states set as $s \coloneqq [\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5,\ \theta_6,\ \theta_7,\ \dot{\theta}_1,\ \dot{\theta}_2,\ \dot{\theta}_3,\ \dot{\theta}_4,\ \dot{\theta}_5,\ \dot{\theta}_6,\ \dot{\theta}_7,\ \Delta x,\ \Delta y,\ \Delta z] \in \mathbb{R}^{17}$,where $\theta_i,\ \dot{\theta}_i$ is the angle and angular velocity of the first joint and $\Delta x,\ \Delta y,\ \Delta z$ is the distance between the end-point p of the manipulator and the corresponding desired target point p^*. The initial position of the end-point is [0.0044, 0.0001, 1.1743], and the initial position of the target point is [0.0193, 0.4008, 0.6715]. The expected path is an arc trajectory generated by the path generation algorithm, and the step size is set to 50.

Figure 8. The redundant manipulator simulation system

The setup of the redundant manipulator path tracking experiment is exactly the same as that of UR5. The experiment still uses the continuous control reinforcement learning algorithm SAC, and all network structures are also the same as the UR5 setup. That is, each network contains two hidden layers, the number of nodes in each layer is 200, and the activation function of the hidden layer is set to Relu. The hyperparameter settings are also: replaybuffer = 1e6, discount-factor = 0.99, batch-size = 128, the update between the Q network and the target Q network adopts the soft update method, the soft parameter tau=0.01, and the learning rate of Actor and Critic network are both set to learning-rate = 1e-3, the weight coefficient α=1e-3 of policy entropy during the whole training process.

The reward settings are also the same as before. The only difference is that the redundant manipulator path tracking experiment is an experiment to verify the generalization of the algorithm, so it does not involve the results when other hyperparameter conditions change. Therefore, the default separation distance in the reward setting is n=1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2023 doi:10.20944/preprints202305.1862.v1

Since this experiment is a verification experiment, this experiment only explores the path tracking results in the speed control mode. The path tracking results of the redundant manipulator are shown in Figure **9**.

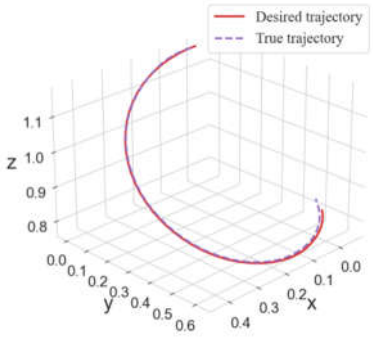

Figure 9. Verification results of redundant manipulator path tracking

In addition, we consider the randomness of the sampling of the deep reinforcement learning algorithm. And in order to reflect the stability of our method, we also carried out multiple experiments under the setting of multiple random seeds. The training process curve of the experimental results is shown in Figure **10**.

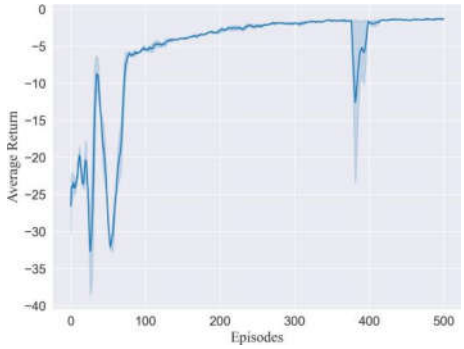

Figure 10. Redundant manipulator path tracking training process curve

The experimental results show that our method still has a good tracking effect on the redundant manipulator, and the training results under different random seed settings show that our method can be guaranteed in terms of generalization and stability

5. Conclusion

In this paper, we introduce a method of using a deep reinforcement learning algorithm to realize the path tracking of the manipulator. The traditional path planning algorithm is used to generate the target path, and the deep reinforcement learning algorithm is used to generate the control signal to control the manipulator and realize the tracking of the target path. The experimental results show that the method has a good effect on the path tracking of the manipulator, which not only avoids the process of seeking the inverse kinematics solution, but also maintains good performance when the dynamic characteristics change. In addition, through further verification experiments on the path tracking of the redundant manipulator, the generalization and stability of our method are reflected. So, we think that our method has great significance for the research of the path tracking of the manipulator.

Author Contributions: Conceptualization, P.Z. and J.Z.; methodology, P.Z. and J.K. ; software, P.Z. and J.Z.; resources, J.K.; data curation, J.Z.; writing—original draft preparation, P.Z.; writing—review and editing, P.Z.; funding acquisition, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key-Area Research and Development Program of Guangdong Province, grant number No.2019B020223003; and Guangdong Basic and Applied Basic Research Foundation, grant number No.2022A1515140013.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author only for reasonable requests.

Acknowledgments: We are very grateful to the anonymous reviewers for their constructive comments on improving this paper.

Conflicts of Interest: The authors report there are no competing interests to declare.

References

1. Parlaktuna, O.; Ozkan, M. Adaptive control of free-floating space robots in Cartesian coordinates. *Adv Robotics* **2004**, *18(9)*, 943–959.
2. Guo, Y.; Chen, L. Adaptive neural network control for coordinated motion of a dual-arm space robot system with uncertain parameters. *Appl Math Mech* **2008**, *29(9)*, 1131–1140.
3. Canudas, W.C.; Fixot, N. Robot control via robust estimated state feedback. *IEEE T Automat Contr* **1991**, *36(12)*, 1497–1501.
4. Kim, E.; Output feedback tracking control of robot manipulators with model uncertainty via adaptive fuzzy logic. *IEEE T Fuzzy Syst* **2004**, *12(3)*, 368–378.
5. Abdollahi, F.; Talebi, H.A, Patel RV. A stable neural network-based observer with application to flexible-joint manipulators. *IEEE Trans Neural Netw* **2006**, *17(1)*, 118-129.
6. Kim, Y.H.; Lewis, F.L. Neural network output feedback control of robot manipulators. *IEEE Trans Rob Autom* **1999**, *15(2)*, 301–309.
7. Selmic, R.R.; Lewis, F.L. Dead zone compensation in motion control systems using neural networks. *IEEE Trans Automat Contr*. **2000**, *45(4)*, 602–613.
8. Zhu, W.H.; Lamarche, T. Dupuis, E. et al. Networked embedded control of modular robot manipulators using VDC. *IFAC Proc* **2014**, *47(3)*, 8481–8486.
9. Cao, S.; Jin, Y.; Trautmann, T.; Liu, K. Design and Experiments of Autonomous Path Tracking Based on Dead Reckoning. *Appl. Sci.* **2023**, 13, 317. https://doi.org/10.3390/app13010317
10. Cai, Z.X.; *Robotics*. Tsinghua university press: Beijing (BJ), China, 2000.
11. Patolia, H.; Pathak, P.M.; Jain, S.C. Force control in single DOF dual arm cooperative space robot. *P 2010 Spr Simul Multicon* **2010**, 1–8.
12. Ma, B.L.; Huo, W. Adaptive Control of Space Robot System. *Iet Control Theory A* **1996**, *13(2)*, 191–197.
13. Spong, M.W.;On the robust control of robot manipulators. *Ieee T Automat Contr* **1992**, *37(11)*, 1782–1786.
14. Purwar, S.; Kar, I.N.; Jha, A.N. Adaptive output feedback tracking control of robot manipulators using position measurements only. *Expert Syst Appl* **2008**, *34(4)*, 2789–2798.
15. Annusewicz-Mistal, A.; Pietrala, D.S.; Laski, P.A.; Zwierzchowski, J.; Borkowski, K.; Bracha, G.; Borycki, K.; Kostecki, S.; Wlodarczyk, D. Autonomous Manipulator of a Mobile Robot Based on a Vision System. *Appl. Sci.* **2023**, 13, 439.
16. Zhang, T.; Song, Y.; Kong, Z.; Guo, T.; Lopez-Benitez, M.; Lim, E.; Ma, F.; Yu, L. Mobile Robot Tracking with Deep Learning Models under the Specific Environments. *Appl. Sci.* **2023**, 13, 273.
17. Tappe, S.; Pohlmann, J.; Kotlarski, J. et al. Towards a follow-the-leader control for a binary actuated hyper-redundant manipulator. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Hamburg, Germany, Sept 28 - Oct 2 **2015**, 3195–3201.
18. Palmer, D.; Cobos-Guzman, S.; Axinte, D. Real-time method for tip following navigation of continuum snake arm robots. *Robot Auton Syst* **2014**, *62(10)*, 1478–1485.
19. Sutton, R.S.; Barto, A.G. *Reinforcement learning: An introduction*. MIT press: Cambridge (MA), Britain, **2018**.
20. Guo, X. Research on the control strategy of manipulator based on DQN. master's thesis. Beijing Jiaotong University, Beijing(BJ), China, **2018**.
21. Liu, Y.C.; Huang, C.Y. DDPG-Based Adaptive Robust Tracking Control for Aerial Manipulators With Decoupling Approach. *IEEE Trans Cybern* **2022**, 52(8), 8258-8271.
22. Mnih, V.; Kavukcuoglu, K.; Silver, D. et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518(7540)*, 529-533.
23. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A. et al. Continuous control with deep reinforcement learning. ArXiv [Preprint]. **2015**; arXiv:1509.02971.
24. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. *International conference on machine learning* **2018**, *PMLR*, 1587–1596.
25. Haarnoja, T.; Zhou, A.; Abbeel, P. et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*. **2018**; *PMLR*, 1861–1870.
26. LaValle SM. Rapidly-exploring random trees: A new tool for path planning. **1999**, Research Report.