



OPEN

Deep reinforcement learning trajectory planning for robotic manipulator based on simulation-efficient training

Bin Zhao^{1,2,3,4}✉, Yao Wu⁴, Chengdong Wu^{1,4} & Ruohuai Sun^{1,3}

The paper proposes a new M2ACD (Multi-Actor-Critic Deep Deterministic Policy Gradient) algorithm to apply trajectory planning of the robotic manipulator in complex environments. First, the paper presents a general inverse kinematics algorithm that transforms the inverse kinematics problem into a general Newton-MP iterative method. The M2ACD algorithm based on multiple actors and critics is structured. The dual-actor network reduces the overestimation of action values, minimizes the correlation between the actor and value networks, and mitigates instability during the actor's selection process caused by excessively high Q-values. The dual-critic network reduces the estimation bias of Q-values, ensuring more reliable action selection and enhancing the stability of Q-value estimation. Secondly, The robotic manipulator's TSR (two-stage reward) strategy is designed and divided into the approach and close. Rewards in the approach phase focuses on safely and efficiently approaching the target, and rewards in the close phase involves final adjustments before contact is made with the target. Thirdly, to solve the position hopping jitter problem in traditional reinforcement learning trajectory planning, the NURBS (Non-Uniform Rational B-Splines) curve is used to smooth the hopping trajectory generated by M2ACD. Finally, the correctness of the M2ACD and the kinematics algorithm is verified by experiments. The M2ACD algorithm demonstrated superior curve smoothing, convergence stability and convergence speed compared to the TD3, DARC and DDPG algorithms. The M2ACD algorithm can be effectively applied to collaborative robots' trajectory planning, establishing a foundation for subsequent research.

Keywords Collaborative robots, Application of artificial intelligence, Deep reinforcement learning, Trajectory planning, Reward priority

DRL (Deep Reinforcement Learning) has become a research hotspot in the trajectory planning of robot arms, enabling robots to show behaviors close to humans in tasks such as grasping, opening doors, and walking^{1,2}. Traditional industrial robot arm motion and grasping tasks usually rely on kinematics and S-type trajectory planning to provide accurate corner path planning for each robot arm joint³. However, cooperative robotic arms have multiple degrees of freedom and redundant axes, increasing the complexity of joint space path solving. In complex working environments, it is difficult for cobots to obtain closed-form analytical solutions, thus limiting their ability to obtain effective action strategies from kinematics and planning algorithms. Models based on DRL training have been successfully applied in various fields of robotics to some extent, and scholars at home and abroad have carried out in-depth research on trajectory Planning and inverse kinematics of robotic arm technology. Li Xiangjian et al.⁴ propose a general motion planning framework integrating deep reinforcement learning (DRL), aiming at the key and challenging points of motion planning and optimization of redundant robots. Based on the exploration ability of DRL and the nonlinear fitting ability of the artificial neural network, the energy optimal solution of the inverse kinematics is derived, and the experimental results verify the performance of the proposed work. Su Hang et al.⁵ establish a kinematic model of the relationship between the humanoid robot arm and the human arm to solve the humanoid robot arm's humanoid behavior. This paper proposes a new incremental learning framework using kinematic mapping in robot redundancy, combining the

¹School of Information Science and Engineering, Northeastern University, Shenyang 110819, China. ²Liaoning University of Technology College of Interdisciplinary Sciences, Jinzhou 121001, China. ³SIASUN Robot & Automation Co., Ltd., Shenyang 110169, China. ⁴College of Robot Science and Engineering, Northeastern University, Shenyang 110169, China. ✉email: zhaobin@stumail.neu.edu.cn

incremental learning method with a deep convolutional neural network to realize fast and efficient learning. This structure improves the accuracy of regression and significantly reduces the processing time of learning human motion data. Guo Qiangqiang et al.⁶ present an innovative approach utilizing a two-step strategy that integrates data-driven imitation learning and model-based trajectory optimization to generate optimal trajectories for autonomous excavators. This research offers a new solution for autonomous trajectory planning in excavation robotics with significant theoretical and practical implications. Leng Shu et al.⁷ proposes a flexible two-arm strategy introducing a recursive neural network to handle the measurement error. Regarding the safety factor, the researcher considered the collision limit and the minimum joint moment constraint in designing the reward function. Fang, Zheng et al.⁸ propose a new snake-tongue algorithm based on a slope-type potential field is introduced and combined with a genetic algorithm (GA) and reinforcement learning (RL) to reduce the path length and the number of path nodes. Furthermore, the path search capability of the artificial potential field method is enhanced by integrating a genetic algorithm and reinforcement learning, thereby enabling more effective path searches in diverse and complex obstacle distributions. Marco Ewerton et al.⁹ addressed the problem of trajectory planning algorithms failing collaborative tasks due to unexpected obstacles and changes in the surrounding environment. They proposed a reinforcement learning algorithm based on trajectory distribution and validated it in the context of assisted teleoperation and motion planning. In the literature of inverse kinematics, although the inverse kinematics solution based on energy optimal solution and the inverse kinematics solution based on deep learning solve some problems, the deep learning model needs a large amount of training data to learn the inverse kinematic mapping effectively. The correlation inverse kinematics algorithm lacks a clear physical explanation. Their kinematics only consider the solution of the inverse kinematics of the robot and do not care about the correctness of the joint solution in the teaching reproduction task. In the relevant research of trajectory planning, the DDPG algorithm is prone to a significant overestimation bias in the continuous control domain. TD3 addresses this issue by employing dual critics for value correction, which may result in significant underestimation. Only a few scholars have tried to apply the method based on reinforcement learning to the motion planning of robotic arms in real scenes. Applying DRL in operations such as trajectory planning of robotic arms often relies on real robot platforms for training. However, this approach requires a lot of human intervention and long hours of training but also leads to high consumption of resources and a potential risk of collision. Although trajectory planning technology has made significant progress in continuous control, many algorithms still need to overcome the challenge of slow speed and unstable accuracy, especially after a period of training, which is prone to sudden changes^{10–12}.

Given this, this study proposes solutions to the problems of redundant kinematics and complex trajectory equation construction in traditional robotic arm trajectory planning algorithms. The inverse kinematics problem is transformed into a general Newton-MP iterative method to improve the efficiency and accuracy of the solution. Then, this paper proposes a DRL-based M2ACD algorithm for robotic manipulator trajectory planning. The algorithm has three key components: (1) the Double Actor and Double Critic networks, which improve the agent adaptability and overall estimation accuracy and stability. The dual-actor network typically comprises two distinct policy models, allowing the simultaneous learning of two strategies to address different situations and task requirements. The dual-critic network usually employs two independent value estimation models, thereby reducing potential biases or overfitting arising from using a single model during training. (2) The M2ACD algorithm divides the grasping reward strategy into two stages, approaching and closing, and this strategy prioritizes reward value to enhance the efficiency and overall quality of the training process. (3) In order to solve the problem of position hopping jitter in traditional reinforcement learning trajectory planning, the M2ACD trajectory uses the NURBS spline curve to generate the smooth trajectory curve. By conducting extensive training in a simulation environment, issues such as collisions and engine overspeed during training on real hardware are avoided, thus reducing the need for manual intervention and training time while conserving resources and reducing potential risks. The M2ACD trajectory planning algorithm exhibited superior stability and generalization abilities, effectively circumventing the computational and model complexity issues commonly encountered when dealing with high-dimensional, nonlinear, and complex environments.

Related program will be open source in the future: <https://github.com/SimonZhaoBin/DRLTrajectory-planning>.

Motion control for collaborative robots Planning strategies and simulation systems

Compared to traditional industrial robots, collaborative robots show greater flexibility and adaptability in environments where they interact with humans and can perform complex tasks safely. Figure 1 shows the M2ACD algorithm under the DRL framework, and the relevant symbols and formulas in the figure are explained in detail later in the article.

In order to solve the trajectory planning problem in DRL, the kinematics problem must be solved first so that the DRL algorithm can explore trajectory planning independently in Cartesian space. Kinematics provides the fundamental mathematical model for trajectory planning by describing the relationship between the robot's joints and the end-effector, thereby assisting in generating feasible paths. Trajectory planning, under kinematic constraints, considers the smooth movement of the robot from the starting point to the target while ensuring the feasibility of the path. Kinematics ensures the feasibility of executing the planned trajectory, while trajectory planning, within the kinematic constraints, generates suitable paths and trajectories. DRL trajectory planning is designed to deal with the trajectory problem of the robot's end in a rectangular coordinate system, which needs to be transformed into a joint coordinate system through kinematics.

As illustrated in Fig. 2, the seven-axis collaborative robotic arm of the SIASUN SCR3 is selected for the simulation experiment of target object trajectory planning based on reinforcement learning. In the initial state, the target object's position is located within the working range of the end-effector of the robotic arm, and the

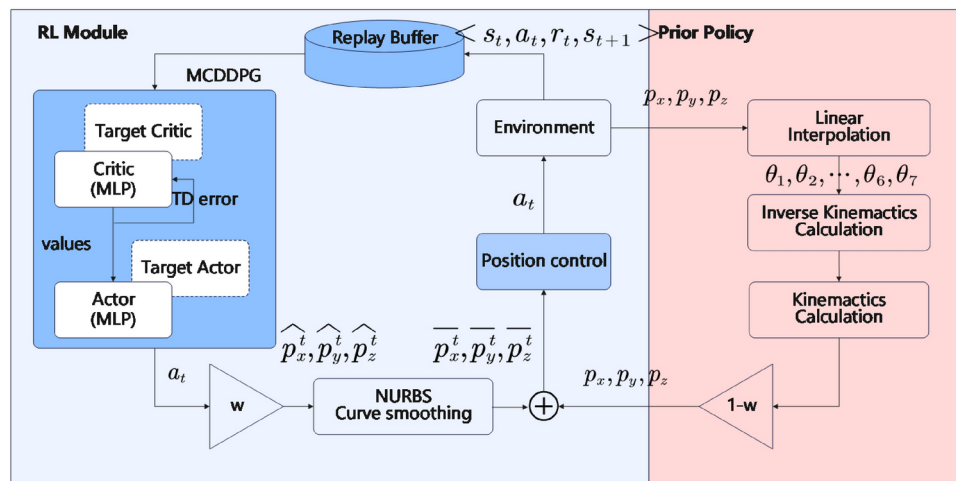


Figure 1. DRL framework of robot M2ACD algorithm flow.

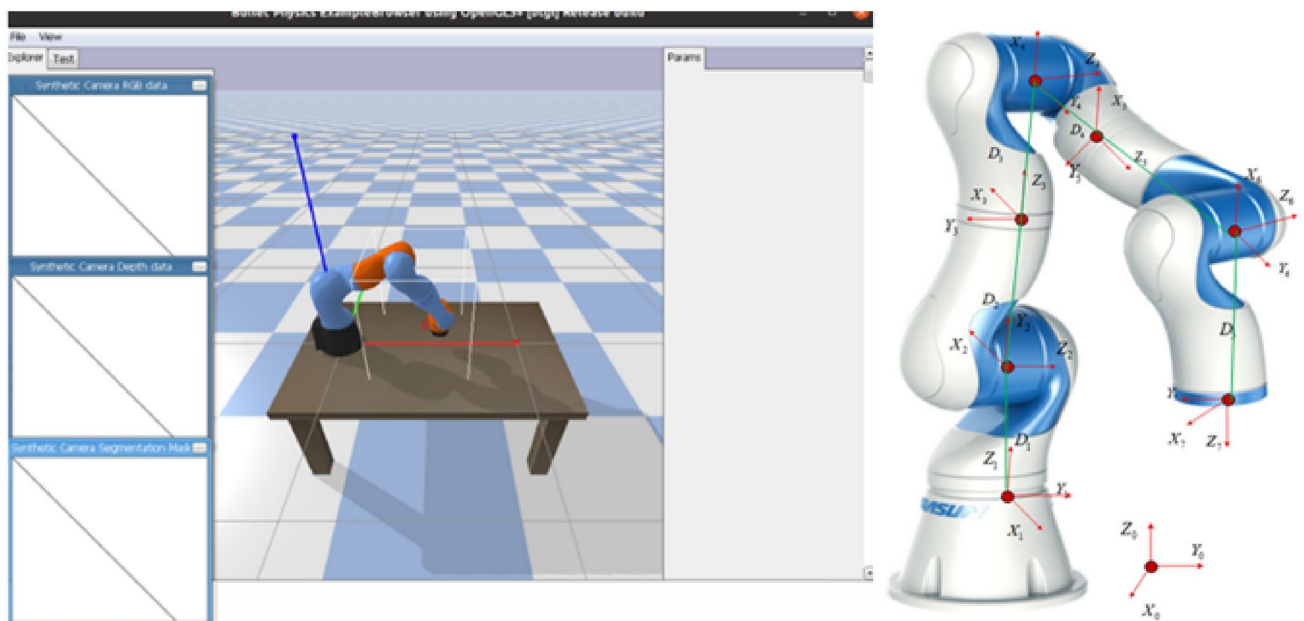


Figure 2. Coordinate system and experiment platform for SCR3 collaborative robots.

randomly generated target position should be in the robot operating space. In the simulation process, trajectory planning moves the suction cup at the end of the manipulator to a certain distance from the center point of the target object's surface, and the planning is considered successful.

Kinematic solution

In order to resolve the DRL issue, it is first necessary to determine the kinematics of the redundant collaborative robot. Figure 2 depicts the mechanical configuration and joint coordinate system of the redundant collaborative robot. Although the increase in redundancy enhances the performance of obstacle avoidance, singularity avoidance, and dexterity, it also increases the complexity of the problem. Due to its distinctive structural characteristics, there are an infinite number of potential solutions, which are challenging to resolve effectively through traditional geometric relations and analytical methods. This paper presents a generalized inverse kinematics algorithm for addressing the kinematics problem of redundant collaborative robots. The algorithm transforms the inverse kinematics solution problem into a generalized Newton-MP iterative method solution, thereby enhancing the efficiency and accuracy of the solution. In the simulation process, trajectory planning moves the suction cup at the end of the manipulator to a certain distance from the center point of the target object's surface, and the planning is considered successful.

1) Forward kinematics solving:

The forward kinematics of the redundant collaborative robot are divided into two parts: the four joints of the large arm ($\theta_1, \theta_2, \theta_3, \theta_4$) and the three joints of the end wrist ($\theta_5, \theta_6, \theta_7$). Calculate the ${}^0_{el}T$, and then right multiply ${}^{el}_{tcp}T$. The overall solution is ${}^0_{tcp}T$, where the wrist coordinate system is $\{el\}$ and the end coordinate system is $\{tcp\}$. The length of the connecting rod is $D_i (i = 1 \sim 5)$. As the forward kinematics of the collaborative robot are relatively straightforward to solve, only the computational method is presented here.

$${}^0_{tcp}T = {}^0_1T {}^1_2T {}^2_3T {}^3_4T {}^4_5T {}^5_6T {}^6_7T \quad (1)$$

2) Inverse kinematics Newton-MP method for optimal solution:

The Newton-MP algorithm is employed to solve this problem, and its algorithmic steps can be divided into five parts as follows:

Step 1: Collect the current values of each joint of the collaborative robot, and calculate the initial position vector in the world coordinate system by forward kinematics P_x, P_y, P_z . The inverse kinematics of the $F(\Theta^{(k)})$ are established by solving the nonlinear equations. The set of nonlinear inverse kinematics equations for the first k iteration value is given by $\Theta^{(k)} = (\theta_0^k, \theta_1^k, \dots, \theta_{n-1}^k)^T$, where k is the number of iterations performed by the Newton-MP algorithm. The iteration formula for calculating the value of the $K + 1$ iteration of the Newton descent method is as follows (Eq. 2).

$$\Theta^{(k+1)} = \Theta^{(k)} - \omega (F'(\Theta^{(k)}))^{-1} F(\Theta^{(k)}) \quad (2)$$

where ω is the iteration factor and J is the ratio of the 3×4 Jacobi matrix of order.

$$J = F'(\Theta^{(k)}) = \begin{bmatrix} \frac{\partial f_0^{(k)}(\Theta)}{\partial \theta_0} & \frac{\partial f_0^{(k)}(\Theta)}{\partial \theta_1} & \dots & \frac{\partial f_0^{(k)}(\Theta)}{\partial \theta_{n-1}} \\ \frac{\partial f_1^{(k)}(\Theta)}{\partial \theta_0} & \frac{\partial f_1^{(k)}(\Theta)}{\partial \theta_1} & \dots & \frac{\partial f_1^{(k)}(\Theta)}{\partial \theta_{n-1}} \\ \dots & \dots & \ddots & \dots \\ \frac{\partial f_{n-1}^{(k)}(\Theta)}{\partial \theta_0} & \frac{\partial f_{n-1}^{(k)}(\Theta)}{\partial \theta_1} & \dots & \frac{\partial f_{n-1}^{(k)}(\Theta)}{\partial \theta_{n-1}} \end{bmatrix} \quad (3)$$

Step 2: Verify that the selected initial joint vector Θ^k is optimal for the complete constrained equation ϕ . The complete stroke power constraint equation ϕ is as follows (Eq. 4).

$$\begin{cases} \text{NORM}(F(\Theta^{(k+1)}), 3) < \text{NORM}(F(\Theta^{(k)}), 3) < 1 \times 10^{-5}, \\ \text{NORM}(\tau^{(k+1)}, 4) < \text{NORM}(\tau^{(k)}, 4) < 1 \times 10^{-5}, \\ \text{NORM}(\Theta^{(k)}, 4) < 1 \times 10^{-5} \end{cases} \quad (4)$$

In the formula, $\text{NORM}(\text{variable}, \text{number})$ represents the norm of the variable. F is a nonlinear equation, and τ is the torque. When the vector $\Theta^{(k)}$ is not optimal, we first take $\omega = 1/2$ and obtain $\Theta^{(k+1)}$ according to Eq. (4).

Step 3: Determine whether the $F(x^n)$ matrix of order $m \times n$ is full rank. Since it is impossible to invert the Jacobi matrix of order 3×4 established in this paper, the MP generalized inverse matrix is used to solve the pseudo-inverse matrix. When the row J is full rank, $J^{(k)-} = J^T(JJ^T)^{-1}$, the following can be derived:

$$\begin{cases} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}^{(k+1)} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}^{(k)} - \omega (J^{(k)})^{-1} \begin{bmatrix} F \\ F_2 \\ F_3 \end{bmatrix}^{(k)} J_{\text{full}}^{(k)}, \\ \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}^{(k+1)} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}^{(k)} - \omega J^T (JJ^T)^{-1} \begin{bmatrix} F \\ F_2 \\ F_3 \end{bmatrix}^{(k)} J_{\text{not full}}^{(k)} \end{cases} \quad (5)$$

Step 4: Determine whether the fully constrained equations ϕ is satisfied. If this is not satisfied, the ω should be updated to $1/2 \times \omega$.

Step 5: Verify whether the conditions $|\Theta^{(k+1)} - \Theta^{(k)}| < \delta$ are met. If so, set $\lim_{n \rightarrow \infty} F(\Theta^{(k)}) = 0$, thereby terminating the Newton-MP convergence of the iteration process. Otherwise, proceed to step 2 and continue iterating until the above condition is met.

The above has already been solved for $\theta_1 \sim \theta_4$. Since the robot wrist joint orientation is determined by the joints $\theta_5, \theta_6, \theta_7$. It can be directly adopted ${}^0_7T = {}^0_4T {}^4_7T \Rightarrow ({}^0_4T)^{-1} {}^0_7T = {}^4_7T$ analytic method for solving.

M2ACD trajectory planning

The combined kinematic modeling and DRL approach circumvents the need to construct intricate trajectory equations by processing intricate relational mappings through interacting information with the environment^{13–18}. The robotic arm kinematic model furnishes the requisite coordinate information for trajectory planning and serves as the foundation for the trajectory planning task^{19–23}. The paper proposes a new M2ACD method for trajectory planning of a collaborative robotic arm. As illustrated in Fig. 3, the structure of the M2ACD algorithm proposed in this paper is described. The multi-strategy and multi-value network structure, action space, and reward function are designed to meet the robotic arm grasping task requirements.

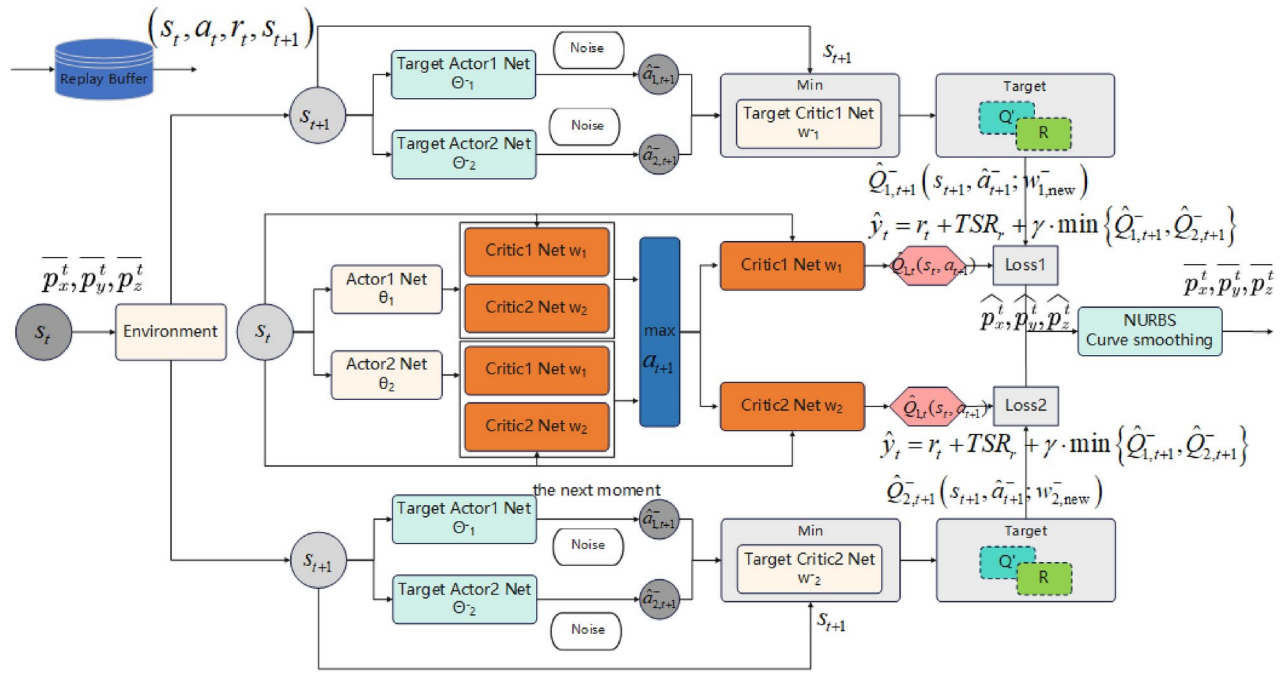


Figure 3. M2ACD network structure.

M2ACD algorithm

This paper proposes a reinforcement learning planning algorithm for M2ACD, which is based on a multi-strategy network and a multi-value network. As illustrated in Table 1, the flow of the M2ACD algorithm is depicted in detail, providing a clear depiction of the software execution process of the proposed algorithm.

- 1: Initialize networks and replay buffer.
Initialize actor networks \$(\theta_1, \theta_2, \theta_1^-, \theta_2^-)\$ and critic networks \$(w_1, w_2, w_1^-, w_2^-)\$ with random parameters \$(\theta_{rand}^-, w_{rand}^-)\$. Initialize replay buffer \$B = \{\}\$.
 - 2: **for** \$t = 1\$ to \$T\$ **do**
 - 3: Double Actor networks take action to be expressed as Equation 7
 - 4: Interact with the environment, store data and update:
Double Critic networks predict Q value, which in turn determines \$a_{t+1}\$. It expressed as Equation 8
Execute action \$a_{t+1}\$ and observe reward \$r\$, new states \$s_{t+1}\$ and done flag \$d\$. It expressed as Equation 9
Store transitions in the replay buffer: \$B = \{(s_t, a_t, r_t, s_{t+1})\}\$
 - 5: Target Actor network values target action can be expressed as Equation 10
 - 6: Target Critic networks generate target value of action, which can be expressed as Equation 11
 - 7: Calculate the total reward strategy \$TSR_r\$ of approaching and approaching the goal, which can be expressed as Equation 12
 - 8: Calculate the TD target, which can be expressed as Equation 13
 - 9: Calculate TD error can be expressed as Equation 14
 - 10: Update Critic network can be expressed as Equation 16
 - 11: Update Actor network can be expressed as Equation 17
 - 12: **if** update==True **then**
 - 13: Update Target Critic and Target Actor networks, which can be expressed as Equation 18 and 19
 - 14: **end if**
 - 15: **end for**
 - 16: Curve smoothing NURBS Curve smoothing

Algorithm 1. Multi Critic Deep Deterministic Policy Gradient

The fundamental steps of the M2ACD algorithm are as follows:

Step 1 Construct four networks:

Initialize replay buffer $B = \{\}$, the current double Actor and current double Critic are then initialized randomly. The current double Actor and current double Critic are initialized first. Subsequently, the network parameters of the current double Actor and current double Critic are copied to the double Target Actor and double Target Critic networks, respectively. For the sake of clarity, let the parameters of the double Actor network and double Target Actor network be denoted as θ and θ^- respectively, and the parameters of the Critic network and Target Critic network be denoted as w and w^- respectively.

$$\begin{cases} w_1^- = w_2^- = w_{\text{rand}}^-, w_1 = w_2 = w_{\text{rand}} \\ \theta_1^- = \theta_2^- = \theta_{\text{rand}}^-, \theta_1 = \theta_2 = \theta_{\text{rand}} \end{cases} \quad (6)$$

Figure 4 illustrates the Policy network (Actor and Target Actor) and the Q-value network (and Target Critic). The Policy network is responsible for generating actions based on the current state. In contrast, the Q-value network evaluates the value of these actions by computing the Q-value of the current state-action pair to assess their quality. To mitigate fluctuations during training, the Target Actor and Target Critic networks adopt a soft update strategy, ensuring consistency with the original networks. Specifically, these target networks act as delayed copies of the original networks and are updated incrementally, thereby reducing instability during training.

Step 2 Double Actor network take action to predict:

$$\begin{cases} a_{1,t+1} = \mu_1(s_t; \theta_{1,\text{now}}^-) + \xi_1 \\ a_{2,t+1} = \mu_2(s_t; \theta_{2,\text{now}}^-) + \xi_2 \end{cases} \quad (7)$$

Step 3 Interact with the environment, store data and update parameter:

Interact with the environment, inputting s_t and a_t into the double Critic networks for predictive evaluation, which in turn determines a_{t+1} .

$$\begin{cases} Q_{1,t} = Q_1(s_t, a_{1,t}; w_{1,\text{now}}) + Q_2(s_t, a_{1,t}; w_{1,\text{now}}) \\ Q_{2,t} = Q_1(s_t, a_{2,t}; w_{1,\text{now}}) + Q_2(s_t, a_{2,t}; w_{1,\text{now}}) \\ a_{t+1} = \text{if}(Q_{1,t} > Q_{2,t}) a_{1,t}, \text{ else } a_{2,t} \end{cases} \quad (8)$$

The action a_{t+1} is applied to the environment, which returns the next state s_{t+1} and the reward r_t . Use the double Critic networks evaluate separately the Q value.

$$\begin{cases} \hat{Q}_{1,t} = Q_1(s_t, a_{t+1}; w_{1,\text{now}}) \\ \hat{Q}_{2,t} = Q_2(s_t, a_{t+1}; w_{2,\text{now}}) \end{cases} \quad (9)$$

The tuple (s_t, a_t, r_t, s_{t+1}) is stored within the experience replay buffer. The subscripts “now” and “new” indicate the current and updated parameters of the neural networks, respectively.

Step 4 Target Actor network evaluates action:

Double Target Actor networks make prediction: state s_{t+1} and noise ξ_t are executed by selecting the action \hat{a}_{t+1}^- with the deterministic strategy μT . The Target Actor network is shown as follows:

$$\begin{cases} \hat{a}_{1,t+1}^- = \mu T_1(s_{t+1}; \theta_{1,\text{new}}^-) + \xi_{1,t} \\ \hat{a}_{2,t+1}^- = \mu T_2(s_{t+1}; \theta_{2,\text{new}}^-) + \xi_{2,t} \end{cases} \quad (10)$$

Step 5 Target Critic networks generate target value of action:

The double Target Critic network serves as a delayed copy of the Critic network to stabilize the target update of the \hat{Q}_{t+1}^- value. Specifically, calculate the minimum values for Target Critic1 and Target Critic2 by taking s_{t+1}

, $\hat{a}_{1,t+1}^-$, and $\hat{a}_{2,t+1}^-$ respectively.

$$\begin{cases} \hat{Q}_{1,t+1}^- = \min(QT_1(s_{t+1}, \hat{a}_{1,t+1}^-; w_{1,\text{new}}^-), QT_1(s_{t+1}, \hat{a}_{2,t+1}^-; w_{1,\text{new}}^-)) \\ \hat{Q}_{2,t+1}^- = \min(QT_2(s_{t+1}, \hat{a}_{1,t+1}^-; w_{2,\text{new}}^-), QT_2(s_{t+1}, \hat{a}_{2,t+1}^-; w_{2,\text{new}}^-)) \end{cases} \quad (11)$$

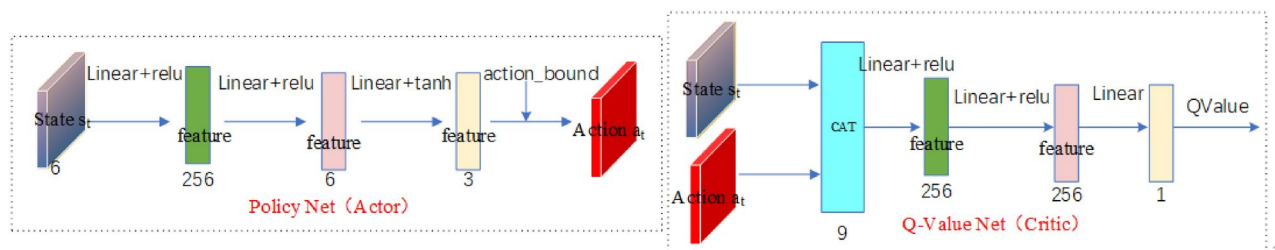


Figure 4. Actor network and critic network.

Step 6 Calculate the total reward of TSR Strategy:

$$TSR_r = \rho_{\min} R_{\text{near}} + \rho_{\max} R_{\text{approaching}} \quad (12)$$

Step 7 Calculate the TD target:

$$\hat{y}_t = r_t + R + \gamma \cdot \min \{ \hat{Q}_{1,t+1}^-, \hat{Q}_{2,t+1}^- \} \quad (13)$$

where r_t is the current reward. γ is the discount factor.

Step 8 Calculate the TD error:

$$\begin{cases} \delta_{1,t} = \hat{Q}_{1,t} - \hat{y}_t \\ \delta_{2,t} = \hat{Q}_{2,t} - \hat{y}_t \end{cases} \quad (14)$$

The \hat{y}_t represents the value based on actual observations, while \hat{Q}_t is the predicted result obtained through inference. The objective is to make the current Critic network's output \hat{Q}_t as close as possible to the target value \hat{y}_t , thereby transforming the problem into a supervised learning task. In this framework, \hat{Q}_{t+1}^- is treated as the target label, and the goal of optimization is to train the network so that its output \hat{Q}_t closely approximates the label value \hat{Q}_{t+1}^- .

Step 9 Update the Critic network:

The gradient descent method is employed to minimize. Convergence of the value function is achieved using Bellman's equation, which relies on the loss function to update the Critic network:

$$\begin{cases} \text{Loss}_1 = \frac{1}{N} \sum_i (\hat{y}_t - Q_1(s_{t+1}, a_{t+1}; w_{1, \text{now}}^-))^2 \\ \text{Loss}_2 = \frac{1}{N} \sum_i (\hat{y}_t - Q_2(s_{t+1}, a_{t+1}; w_{2, \text{now}}^-))^2 \end{cases} \quad (15)$$

The Adam optimizer is employed to modify the parameters of an online policy network with the objective of enhancing its performance. The objective is to bring the \hat{q}_t value as close to the \hat{y}_t value as possible, thereby minimizing the TD error.

$$\begin{cases} w_{1, \text{new}} \leftarrow w_{1, \text{now}} - \alpha \cdot \delta_{1,t} \cdot \nabla_{w} q(s_t, \hat{a}_t; w_{1, \text{now}}) \\ w_{2, \text{new}} \leftarrow w_{2, \text{now}} - \alpha \cdot \delta_{2,t} \cdot \nabla_{w} q(s_t, \hat{a}_t; w_{2, \text{now}}) \end{cases} \quad (16)$$

During the updating process, only the weights w of the $Q(s, a; w)$ network are updated, while the weights w^- of the $Q(s, a; w^-)$ network remain unchanged. Subsequently, the weights of the updated evaluation network are transferred to the target network for the subsequent batch of updates, so that the target network can also be updated.

Step 10 Update the Actor networks:

The Actor network is employed to compute the action a_{new} at the state of the s , thereby providing the Q value $Q(s, a_{new})$ in the current Critic network.

Update the Actor in order to achieve the maximum Q value, which is defined as $Q(s, a_{new})$. This can be achieved by maximizing $Q(s, a_{new})$ or minimizing $-Q(s, a_{new})$. This process essentially informs the actor network that the optimal direction of action adjustment is towards a higher Q . Update all parameters θ of the Actor network with gradient ascent:

$$\begin{cases} \theta_{1, \text{new}} \leftarrow \theta_{1, \text{now}} + \beta \cdot \nabla_{\theta} \mu(s_t; \theta_{1, \text{now}}) \cdot \nabla_{\theta} Q(s_t, \hat{a}_t; w_{1, \text{now}}) \\ \theta_{2, \text{new}} \leftarrow \theta_{2, \text{now}} + \beta \cdot \nabla_{\theta} \mu(s_t; \theta_{2, \text{now}}) \cdot \nabla_{\theta} Q(s_t, \hat{a}_t; w_{2, \text{now}}) \end{cases} \quad (17)$$

Among them, $0 \leq \beta \leq 1$.

Step 11 Update Parameter of Target Critic and Target Actor networks:

The M2ACD algorithm is using a soft update approach. That is, a learning rate is introduced τ that makes a weighted average of the old target network parameters and the new corresponding network parameters, and then assigns a value to the target network.

Target Actor network update process:

$$\begin{cases} \theta_{1, \text{new}}^- \leftarrow \tau \theta_{1, \text{new}}^- + (1 - \tau) \theta_{1, \text{now}}^- \\ \theta_{2, \text{new}}^- \leftarrow \tau \theta_{2, \text{new}}^- + (1 - \tau) \theta_{2, \text{now}}^- \end{cases} \quad (18)$$

Target Critic Network update process:

$$\begin{cases} w_{1, \text{new}}^- \leftarrow \tau w_{1, \text{new}}^- + (1 - \tau) w_{1, \text{now}}^- \\ w_{2, \text{new}}^- \leftarrow \tau w_{2, \text{new}}^- + (1 - \tau) w_{2, \text{now}}^- \end{cases} \quad (19)$$

At each fixed time interval, the parameters of the Target Actor and Target Critic networks are updated using the parameters of the Actor and Critic networks.

Step 12 Return to the initial step:

The process will continue to iterate until the predefined termination condition is met, at which point NURBS curve smoothing will be performed. The training strategy for the network is to have the update frequency of the target network lower than that of the current network. This design allows for the prioritization of error minimization before introducing policy updates, thereby enhancing the overall effectiveness of training.

TSR strategy

According to the position of the target object and task requirements, the grasping task of the robot arm is divided into two stages: approach to the target and close to the target, and the TSR strategy function of the M2ACD algorithm is designed according to the two stages. In the approach to the target phase, the robot's primary task is to move from its current position to a region near the target object. During this phase, trajectory planning focuses on safely and efficiently approaching the target. Once the robot reaches the vicinity of the target object, it transitions into the close to the target phase, which involves final adjustments before contact is made with the target. Trajectory planning in this phase emphasizes precise end-effector control to ensure proper alignment and grasping posture when making contact with the target.

In the training process, the TSR strategy takes the distance between the end pos and the target pos as the primary basis of the reward function, and its reward rules are as follows: (1) If the end of the robot exceeds the working space, the maximum penalty will be given; (2) If the robot has been idle and can not touch the object in the maximum number of steps, it needs to give a certain punishment; (3) When the motion of the cooperative robot meets the requirements, neither punishment nor reward will be given;

As illustrated in Fig. 5, the deterministic goal-based trajectory planning reward strategy is divided into two regions, the r_{min} and r_{max} . If the distance of the planning goal from the robot is greater than r_{max} , the probability of capture is considered to be 0. Conversely, if the distance of the planning target from the robot is less than r_{min} , it is assumed that the probability of success of target trajectory planning is close to 90%. In the region between r_{min} and r_{max} , the probability of successful target trajectory planning is calculated using linear interpolation.

The grasp of the mechanical arm is progressive, and the judgment of whether the mechanical arm has reached the grasp position is as follows:

$$\rho_d = \frac{(r_{max} - r)^2}{(r_{max} - r_{min})^2} \quad (20)$$

The safe distance between the obstacle and the robot, designated as r_{max} , represents the distance within which the probability of collision with the robot is considered to be zero. Conversely, the progressive grasping distance, designated as r_{min} , represents the distance within which a significant reward value is given to the collaborative robot. The criterion for determining whether the robot arm has reached the grasping position is as follows:

$$S_{\{x,y,z\}}^{B1} - G_{\{x,y,z\}}^P < \Delta_{\{x,y,z\}}^{near} \quad (21)$$

The coordinates $S_{(x,y,z)}^{B1}$ represent the position of the end-effector of the collaborative robot, while $G_{\{x,y,z\}}^P$ denotes the target position of the robot. $\Delta_{\{x,y,z\}}^{near}$ represents the incremental distance between the current position and the target position. Suppose the collaborative robot meets the criteria defined by the above equations during the grasping process. In that case, it indicates that the robot has reached the grasping position and can execute the grasping action, resulting in the assignment of reward values R_{near} and $R_{approaching}$. Finally, the sum of the obtained reward values is calculated to yield the total reward function, as shown in Equation 22

$$R = \rho_{min} R_{near} + \rho_{max} R_{approaching} \quad (22)$$

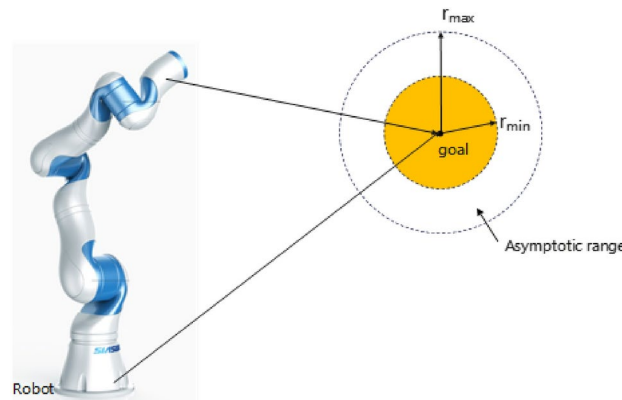


Figure 5. Capture probability of trajectory planning based on target distance.

Through different stages of reward design, TSR can effectively reduce the sparse reward problem and promote the rapid convergence of learning agents. Rewards in the approach phase can be based on proximity to the target. In contrast, rewards in the close phase can be based on alignment accuracy or task completion, enabling more targeted learning.

NURBS curve smoothing

The application of NURBS in trajectory smoothing offers superior shape and local adjustment control compared to polynomial splines and Bézier curves, particularly in trajectory planning involving complex geometries and constraints. Using rational functions, NURBS avoids the numerical instability caused by high-order polynomial curves. NURBS can precisely represent standard geometric shapes, such as straight lines and arcs while providing local control and global smoothness. In order to solve the problem of position hopping jitter in traditional reinforcement learning trajectory planning, the M2ACD uses the NURBS spline curve to smooth the hopping trajectory. This method is the optimal choice for the smoothing trajectory. The fitting process of the NURBS spline curve is shown in Fig. 6.

(1) The process of calculating node parameter u_i (cumulative chord length method), the data point passed by the k polynomial NURBS curve is p_i ($i = 0, 1, \dots, n$), according to the joint displacements - time node sequence $\{p_i, t_i\}$, $i = 0, 1, 2, \dots, n$ (there are $n+1$ position points), the parameter is the value:

$$\begin{cases} U = \{u_0, u_1, \dots, u_{n+2k}\}, u_0 = u_1 = \dots = u_k = 0 \\ u_{i+k} = u_{i+k-1} + \Delta p_i, i = 1, 2, \dots, n-1 \\ \Delta p_i = (p_i - p_{i-1})/s \\ u_{n+k} = u_{n+k+1} = \dots = u_{n+2k} = 1 \end{cases} \quad (23)$$

where Δp_i is the forward difference vector, and s is the distance between sequential adjacent two points. Each parameter value can be calculated accurately. In order to continue the following operation, when $k=3$, $k+1=4$ (repetition).

u_i formula calculated by cumulative chord length method:

$$u_{i+3} = u_{i+2} + \frac{|p_i - p_{i-1}|}{\sum |p_i - p_{i-1}|} i = 1, 2, \dots, n-1 \quad (24)$$

(2) Matrix R element calculation process:

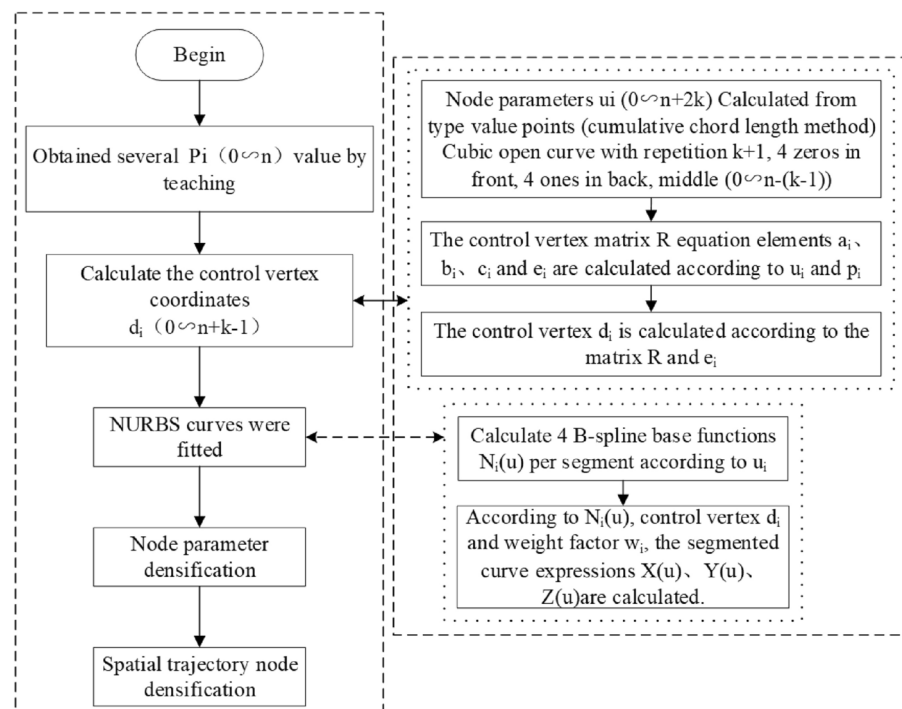


Figure 6. NURBS spline curve fitting process.

$$\left\{ \begin{array}{l} a_0 = 1, b_1 = c_1 = 0, a_{n+1} = b_{n+1} = 1, c_{n+1} = 1 \\ \Delta u_i = u_{i+1} - u_i \\ a_i = \frac{(\Delta u_{i+2})^2}{l_2} = \frac{(u_{i+3} - u_{i+2})^2}{u_{i+3} - u_i}, i = 2, 3, \dots, n \\ b_i = \frac{\Delta u_{i+2} l_0}{l_2} + \frac{\Delta u_{i+1} l_1}{l_3} = \frac{(u_{i+3} - u_{i+2})(u_{i+2} - u_i)}{u_{i+3} - u_i} + \frac{(u_{i+2} - u_{i+1})(u_{i+4} - u_{i+2})}{u_{i+4} - u_{i+1}} \\ c_i = \frac{(\Delta u_{i+1})^2}{l_3} = \frac{(u_{i+2} - u_{i+1})^2}{u_{i+4} - u_{i+1}} \\ e_1 = p_0 + \frac{\Delta u_3}{3} p'_0 = p_0 + \frac{u_4 - u_3}{3} p'_0 \\ e_{n+1} = p_n - \frac{\Delta u_{n+2}}{3} p'_n = p_n - \frac{u_{n+3} - u_{n+2}}{3} p'_n \\ e_i = (\Delta u_{i+1} + \Delta u_{i+2}) p_{i-1} = (u_{i+3} - u_{i+1}) p_{i-1} i = 0, 1, \dots, n \end{array} \right. \quad (25)$$

In the formula, $l_0 = \Delta u_i + \Delta u_{i+1}$, $l_1 = \Delta u_{i+2} + \Delta u_{i+3}$, $l_2 = \Delta u_i + \Delta u_{i+1} + \Delta u_{i+2}$, $l_3 = \Delta u_i + \Delta u_{i+1} + \Delta u_{i+2} + \Delta u_{i+3}$. p'_0 is the first end tangent vector, p'_n is the end tangent vector.

(3) The process of calculating the control vertex d_i : the piecewise connection points of the curve, which correspond to the nodes in the curve domain. The number of control vertices is two more than the data points; there are $n+3$ unknown vertices. The calculation formula for control vertices is as follows:

$$\left\{ \begin{array}{l} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n+1} \end{bmatrix} = \text{inv} \begin{bmatrix} 1 & \dots & \dots & \vdots \\ a_2 & b_2 & c_2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \dots & a_n & b_n & c_n \\ \vdots & \vdots & \dots & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n+1} \end{bmatrix} \\ \begin{bmatrix} dl_1 \\ dl_2 \\ \vdots \\ dl_{n+2} \\ dl_{n+3} \end{bmatrix} = \begin{bmatrix} p_0 \\ d_1 \\ \vdots \\ d_{n+1} \\ p_n \end{bmatrix} \end{array} \right. \quad (26)$$

(4) The process of calculating B-spline basis function $N_i(u)$ and the expression of four B-spline basis functions $N_{(j,3)}(u)$ for each segment is calculated as follows:

$$N_{j,3}(u) = \begin{cases} \frac{(u_{i+1}-u)^3}{h_0 h_1 h_2}, j = i - 3 \\ \frac{(u_{i+1}-u)^2(u-u_{i-2})}{h_0 h_1 h_2} + \frac{(u-u_{i-1})(u_{i+1}-u)(u_{i+2}-u)}{h_1 h_2 h_1} + \frac{(u_{i+2}-u)^2(u-u_i)}{g_0 g_1 h_2}, j = i - 2 \\ \frac{(u_{i-1}-u)^2(u_{i+1}-u)}{g_0 h_1 h_2} + \frac{(u-u_{i-1})(u_{i+2}-u)(u-u_i)}{g_0 g_1 h_2} + \frac{(u_i-u)^2(u_{i+3}-u)}{g_2 g_1 h_2}, j = i - 1 \\ \frac{(u-u_i)^3}{g_2 g_1 h_2}, j = i \end{cases} \quad (27)$$

w h e r e ,
 $h_0 = u_{(i+1)} - u_{(i-2)}$, $h_1 = u_{(i+1)} - u_{(i-1)}$, $h_2 = u_{(i+1)} - u_i$, $g_0 = (u_{(i+2)} - u_{(i-1)})$, $g_1 = u_{(i+2)} - u_i$, $g_2 = u_{(i+3)} - u_i$, $j = (i - 3, i - 2, i - 1, i)$, each paragraph needs to be calculated periodically.

(5) The expression of each NURBS spline curve is calculated, and segmentation curve expression calculation process:

$$\left\{ \begin{array}{l} P_x(u) = \frac{\sum_{j=i-3}^i \omega_j p_{xj} N_{j,3}(u)}{\sum_{j=i-3}^i \omega_j N_{j,3}(u)} \\ P_y(u) = \frac{\sum_{j=i-3}^i \omega_j p_{yj} N_{j,3}(u)}{\sum_{j=i-3}^i \omega_j N_{j,3}(u)} \\ P_z(u) = \frac{\sum_{j=i-3}^i \omega_j z_j N_{j,3}(u)}{\sum_{j=i-3}^i \omega_j N_{j,3}(u)} \end{array} \right. i = k - 1, k, \dots, n + k - 1 \quad (28)$$

(6) in the section of the spline connection, such as node x_i must satisfy $s_i(x_i) = s_{(i+1)}(x_i)$, $s'_i(x_i) = s'_{(i+1)}(x_i)$, $s''_i(x_i) = s''_{(i+1)}(x_i)$. At this point, the NURBS spline curve has been generated.

Through NURBS's smooth transitions, local control, acceleration smoothing, and weight control features, NURBS curves effectively eliminate the issue of "positional jumps and jitters" in trajectories, ensuring smoothness and stability in complex trajectory planning. They not only allow for precise control of trajectory shapes but also guarantee the trajectory's continuity, stability, and adaptability. These features make NURBS suitable for trajectory generation tasks requiring high precision and stability.

Parameter	Value
Experience replay buffer	1000
Batch	16
Learning rate	1×10^{-2}
Network update rate	100
Exploration rate	1×10^{-2}
Epsilon decay	10

Table 1. Hyperparameters configuration of the M2ACD.

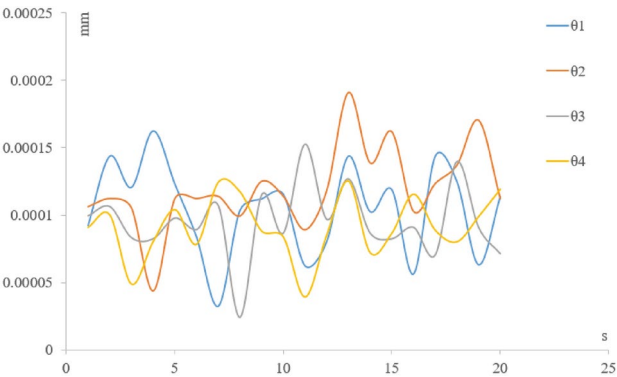


Figure 7. Kinematic absolute error.

Experiments

Experiment settings

Using the SIASUN robot verifies the M2ACD algorithm in the real-world scenarios. The SIASUN robot is equipped with advanced features such as rapid configuration, drag-and-teach functionality, visual guidance, and collision detection. These robots have a payload capacity of 5 kg, a maximum speed of 90°/s, a maximum reach diameter of 1600 mm, and a minimum reach diameter of 750 mm. They are well-suited for various industrial operations, including precision assembly, product packaging, polishing, inspection, and machine tending. The hardware environment of the experimental environment is as follows:

- Operating system: Ubuntu MATE16.04
- CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
- GPU: Titan X
- Python version: 3.7.13
- Torch version: 1.10.1+cu111
- Torch vision version: 0.11.2++cu111
- Robot model: 7-axis collaborative robot of SIASUN SCR3.

The M2ACD adopts an ϵ -greedy strategy to achieve a better balance between exploration and exploitation. The exploration probability is defined by the formula $\epsilon + (1 - \epsilon)^{-Batch/EpsilonDecay}$, where $\epsilon = 0.01$. To optimize the network parameters, the Adam stochastic optimization method is utilized, with specific parameters detailed in Table 1.

The index of algorithm comparison includes: (1)The convergence speed directly reflects the efficiency of an algorithm in learning task objectives, typically measured by the time or number of training iterations required to reach a specified performance level. (2)Convergence stability refers to the degree of fluctuation in accuracy metrics during the later training or upon completion stages. Higher stability indicates greater robustness of the policy across different environments. (3) trajectory smoothness can effectively reduce robot vibrations and high-frequency variations during motion, minimizing the risk of collisions with obstacles and enhancing task safety. All the experimental data of trajectory planning are obtained directly from the position parameters of the driver. To ensure the accuracy of the experiment, this paper randomly generates the start and end points of trajectories, and some typical trajectories are selected for analysis.

Related experiment video: https://www.bilibili.com/video/BV1HkupeeETx/?share_source=copy_web

Trajectory planning and kinematics experiment

The correctness of kinematics algorithm is verified, as shown in Fig. 7. The M2ACD algorithm DRL trajectory planning under Cartesian coordinates will use Newton-MP iterative kinematics to convert the robot end trajectory into the joint coordinate system and send it to the joint drive unit through the bus. The absolute kinematic error shows the absolute error between the theoretical position value of the robot and the real

Method (M)	success rate
Newton-MP's inverse kinematics	100%
Energy optimal solution's inverse kinematics	87.62%
Deep learning's inverse kinematics	30.21%

Table 2. The success rate of the linear task of teaching reproduction (%).

Algorithms	Actors	Critics	Value Correction	Regularization	Cross Policy Update
DDPG	one	one	×	×	×
TD3	one	double	✓	×	×
DARC	double	double	✓	✓	×
M2ACD	double	double	✓	×	✓

Table 3. Structural comparison of related algorithms.

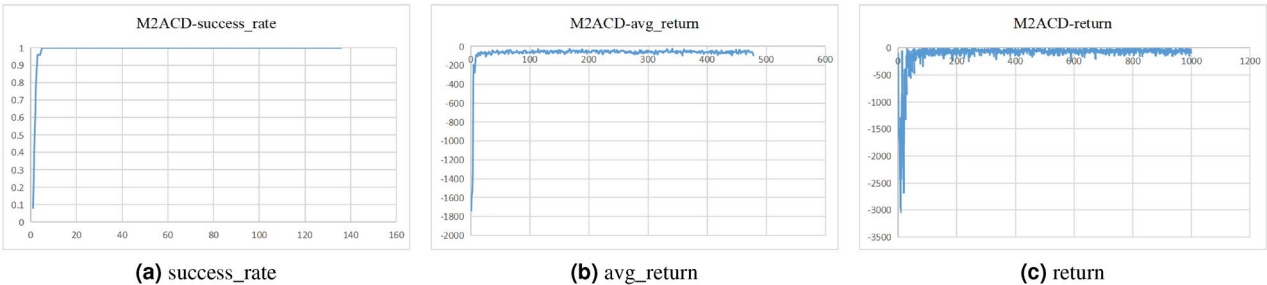


Figure 8. M2ACD algorithm training results.

kinematic position value. The horizontal axis is the absolute error unit: mm, and the vertical axis is the number of iterations unit: 16ms. The absolute error is about 0.00013mm, and the correctness of the kinematics is verified.

In industrial linear tasks with high precision and teaching reproduction, the joint values of the robot are required to be the same as the joint values calculated by the inverse solution. This paper compares Newton-MP's inverse kinematics, energy optimal solution's inverse kinematics, and deep learning's inverse kinematics concerning the linear task of teaching reproduction. According to Table 2, the Newton-MP algorithm proposed in this paper can effectively meet the above requirements and ensure the accuracy of the teaching reproduction task. In contrast, the inverse kinematics method of energy optimal solution and the inverse kinematics method based on deep learning rely on experience and energy model, which tend to cause the joint Angle of adjacent position points to change too much in the process of inverse solution, so it is challenging to meet the needs of practical application and accuracy.

Structural comparison of related trajectory planning algorithms, including the typical DDPG,DARC, M2ACD, and TD3 algorithms, is presented in Table 3 below.

As shown in Fig. 8, the results of the experiments demonstrate that the M2ACD algorithm can rapidly achieve a high accuracy rate in the fourth epoch stage and stabilize the reward value within a narrow range close to zero. Following the attainment of 100% accuracy, there is no abrupt change in the accuracy rate during subsequent training. Moreover, the loss value change curves of M2ACD algorithms all demonstrate a decreasing trend, while the accuracy rates are all in a 100% trend. This indicates that the reward value obtained by the action process of collaborative robot trajectory planning has been enhanced through training.

The convergence stability of the algorithm is verified, as shown in Fig. 9. Experiments are conducted using the DDPG, TD3, DARC, and M2ACD algorithms proposed in this paper. The results of the experiments illustrates the performance comparison of the four methods. It can be observed that TD3, DDPG, DARC, and M2ACD exhibit comparable performance, except for a slight variation in the accuracy of the first two methods. The TD3 and DDPG algorithms exhibited a certain degree of instability in accuracy with increased training. In contrast, M2ACD and DARC, the proposed algorithm in this study, demonstrate consistent performance. But DARC showed inferior performance in *avg_return* and return.

The performance of DDPG, TD3, DARC, and M2ACD algorithms is compared and tested, as shown in Fig. 9. Although these algorithms are similar in overall performance, there are significant differences in accuracy and convergence speed. M2ACD algorithm can quickly make the accuracy reach 100% around the fourth epoch, while TD3 and DDPG algorithms have a long exploration stage and slow convergence speed in the early stage. The accuracy of TD3 and DDPG algorithms fluctuates significantly in the training process, and instability and sudden fluctuation occur frequently. In contrast, M2ACD and DARC algorithms show higher stability, with

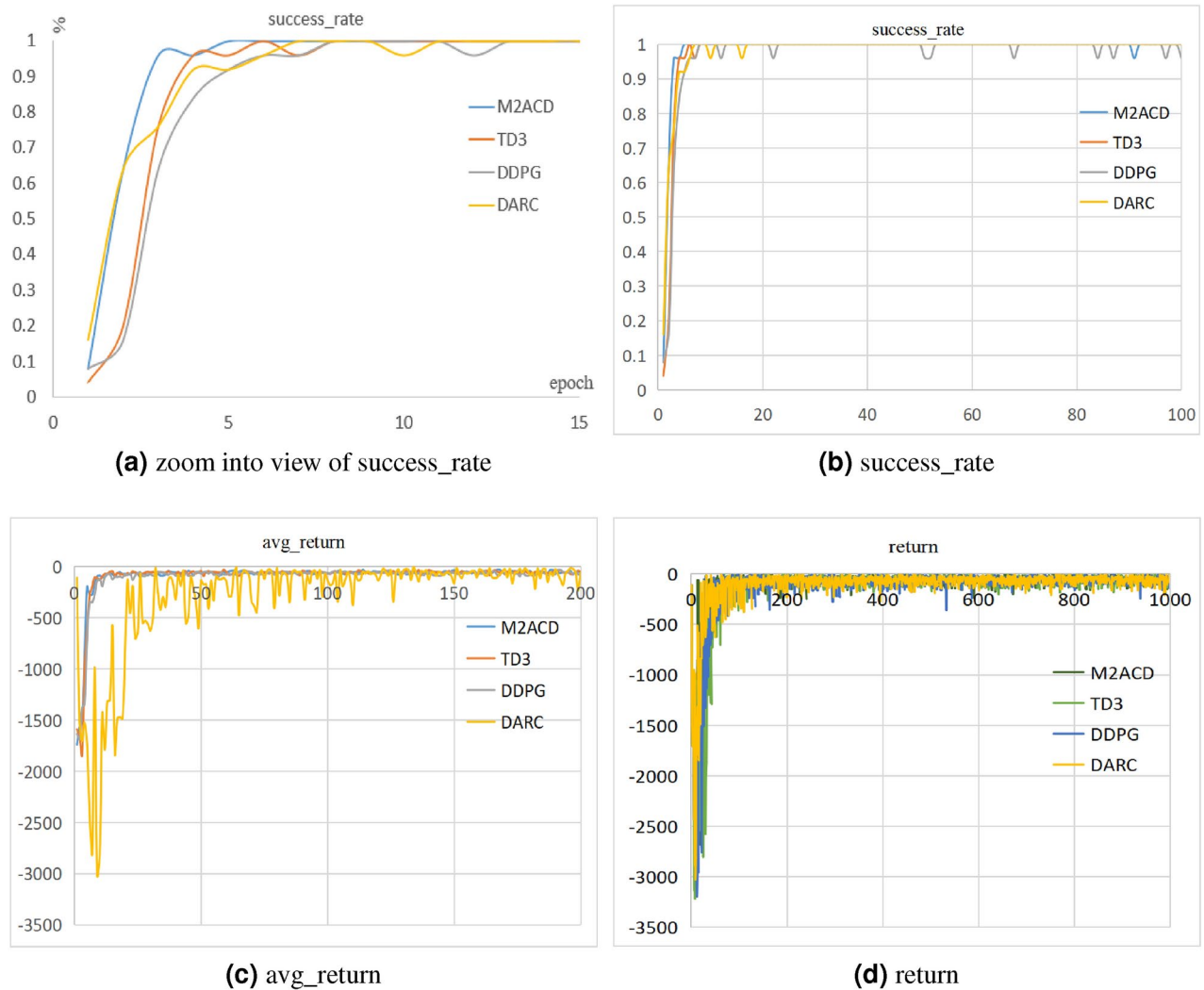


Figure 9. Comparison results of TD3, DDPG, DARC and M2ACD algorithms.

no apparent fluctuations and mutations during training. The M2ACD algorithm proposed in this chapter has significant comprehensive advantages in convergence time, accuracy, stability, and resource utilization efficiency and shows high application potential.

This paper randomly generates the start and end points of trajectories, and some typical trajectories are selected for analysis. Fig. 10 shows the hopping trajectory contrast curves of three trajectory planning methods of DRL.

$p_x = 0.544768059$, $p_y = 0.022876087$, $p_z = 0.47057345$ are the starting position, $p_x = 0.646975827$, $p_y = 0.026603052$, $p_z = 0.04075532$ are the end position. The results show that during the trajectory planning of TD3, DARC, DDPG, and M2ACD without NURBS, there are significant position hopping jitter problem, which proves that it is difficult to cope with sudden obstacles and environmental changes effectively and lacks self-learning ability. Especially in a dynamic environment, the DRL algorithm is difficult to apply this algorithm to actual engineering projects.

Curve smoothing experiment

In order to solve the position hopping jitter problem, the trajectory planning curve is smoothed by NURBS spline. In Fig. 11a, the blue curve represents the expected trajectory of reinforcement learning, and the red dots represent the 11 critical points sampled. In Fig. 11b, the calculation process of control vertex d_i is calculated. The red vital points represent the calculated control vertex d_i , and the blue points represent the sampling position points of reinforcement learning planning. As shown in Fig. 11c, the red points are the critical points of the reinforcement learning trajectory, and the blue curve is the fitting result of the NURBS spline curve. In the straight line section, the fitted curve is consistent with the expected curve, while the error will occur in the region with significant curvature. Fine-tuning can be done by adjusting the weight factor. Although there are some errors, the fitting effect is significantly improved, and the error is small compared with the direct linear approximation method.

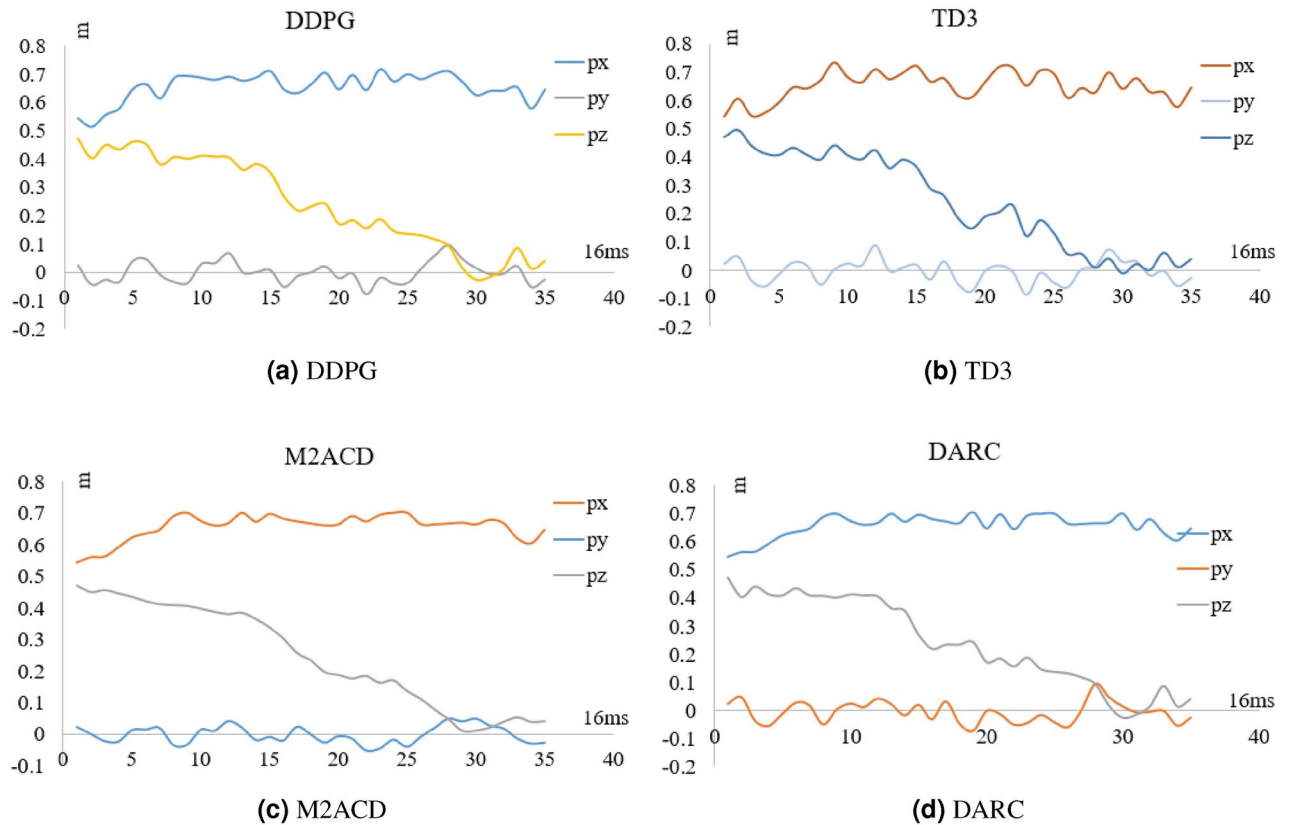


Figure 10. Trajectory planning comparison results of TD3, DDPG, DARC, and M2ACD algorithms.

The trajectory smoothness of the algorithm is verified. During the experiment, the target trajectory of the trajectory planning of the collaborative robot is randomly selected to perform the trajectory planning task. Figure 12 shows the results of four randomly selected trajectory planning of M2ACD without NURBS experiments. The trajectory planning is poor in smoothness, and the position hopping jitter problem occurs.

The NURBS is used to smooth the trajectory curve, and the results are shown in Fig. 13. Compare original curve without NURBS and the curve with NURBS, the M2ACD algorithm can effectively eliminate unnecessary fluctuations and sharp changes while preserving the overall shape of the curve. The experiment proves that the M2ACD algorithm successfully solves the trajectory hopping problem and can be effectively applied to the robot planning task in the actual scene.

The smoothness of NURBS curves can be evaluated through positional continuity G^0 and tangential continuity G^1 , with several quantitative metrics assessing curve smoothness from different perspectives:

1. Positional Continuity G^0 :

$$G^0 = C_i(u_{\text{start}}) = C_{i+1}(u_{\text{end}}) \quad (29)$$

Here, $C_i(u_{\text{end}})$ represents the tangent derivative at u_{end} . If this condition is satisfied, it indicates that the endpoints of adjacent curve segments are connected.

2. Tangential Continuity G^1 :

$$G^1 = (C_i(u_{\text{start}} + s) - C_i(u_{\text{start}})) / s \quad (30)$$

where s denotes the distance between two sequentially adjacent points, corresponding to the distance between key points in the NURBS curve. The variation in G^1 indicates tangential continuity, and the larger G^1 indicates less continuity.

The positional continuity metrics of the TD3, DARC, and DDPG algorithms fully meet the required standards. However, regarding tangential continuity, M2ACD with NURBS improves by 275% compared to the M2ACD without NURBS.

Real scene experiments

In the tasks of collaborative robots, medical thermal ablation requires high stability and accuracy of trajectories. The robot needs to move the ablation needle at the end precisely to the designated surgical site to ensure the accuracy of the treatment. The M2ACD algorithm parameters trained on the simulation platform are applied to the SCR3 collaborative robot to verify the effect of the planning algorithm in simulating the thermal

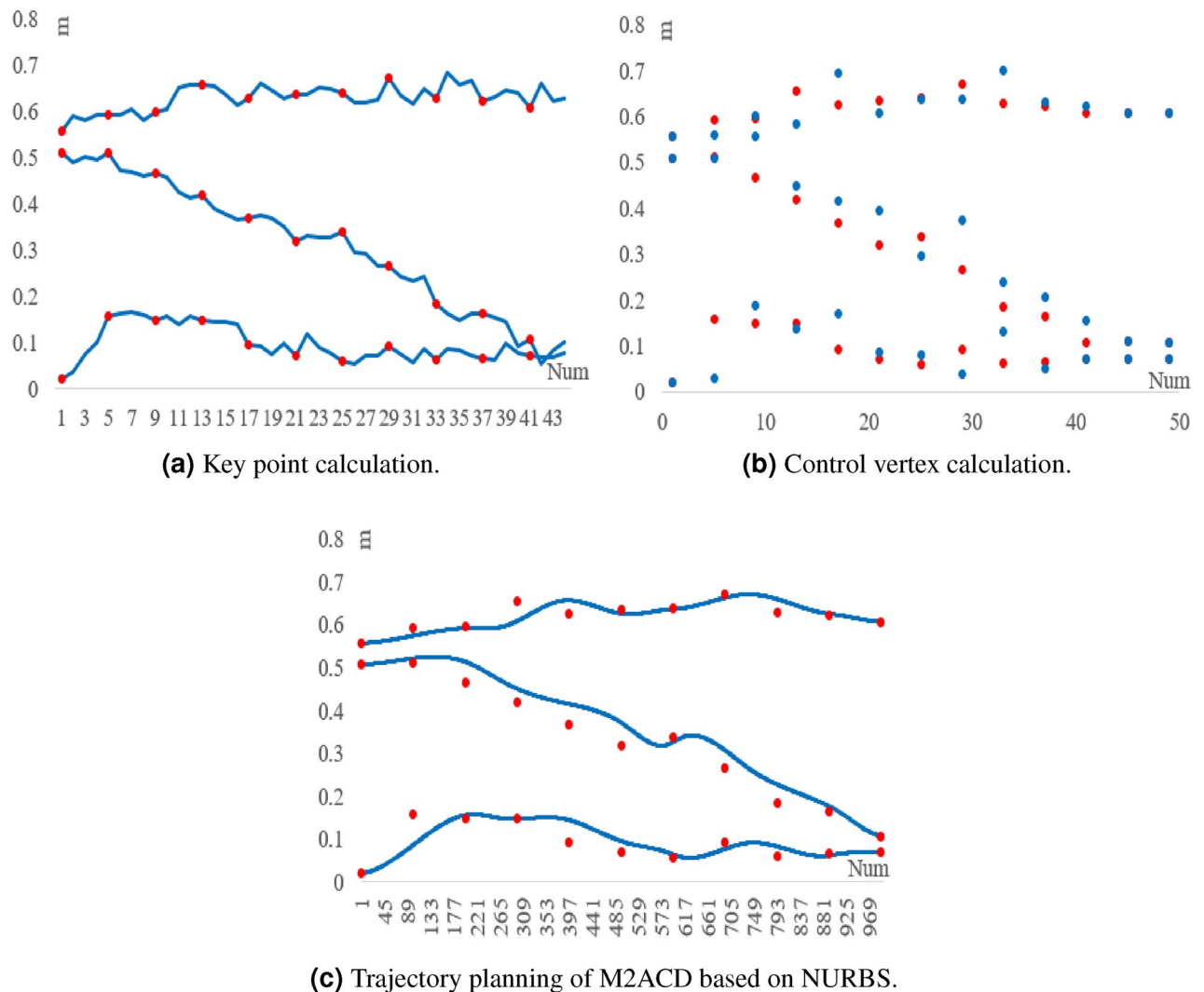


Figure 11. NURBS spline curve sampling fits reinforcement learning trajectory planning.

ablation task of the narrow position of the human skeleton. When the trajectory planning task is executed, the tasks and objectives of the trajectory planning of the cooperative robot are randomly published so that the end of the cooperative robot moves from a position to a specified target position. As illustrated in Fig. 14, collaborative robots can smoothly and accurately reach the marked ablation position on the gray ball. The experiment verifies the accuracy and smoothness of the inverse kinematics and trajectory planning algorithms. The experiment confirmed that the M2ACD algorithm does not need to rely on a real training and validation platform, significantly reducing training costs and potential physical damage and security risks. In the face of a new environment or unseen scenes, the algorithm shows strong generalization ability and performs excellently in practical application.

The validity of TSR strategy is verified, as shown in Table 4, the efficacy of distinct algorithms is evaluated in the context of trajectory planning tasks. The M2ACD algorithm modified with a reward strategy can further improve the training time of the M2ACD algorithm and satisfy real-time requirements with minimal planning time. The experiment confirmed that the M2ACD algorithm introduces the TSR (two-stage reward) strategy, which speeds up the training speed and overcomes the learning complexity caused by sparse or delayed reward signals.

In summary, Several discrepancies and challenges typically arise when transferring DRL trajectory planning algorithms from a simulated environment to real-world conditions. First, the simulated environment's control parameters and sensor data are idealized. In contrast, in the real world, variability in the environment may lead to instability or errors in trajectory planning. The robot may encounter unforeseen obstacles or disturbances in the real environment, requiring real-time adaptation and strategy adjustment. Second, sensor data in simulated environments is usually accurate, whereas the robot in real-world operations may face unpredictable external interference or object changes. During the simulation training phase, noise can be introduced in the simulated environment to improve stability. Finally, differences in the inference speed and computational resource requirements between the simulated training environment and the real world may affect the algorithm's real-

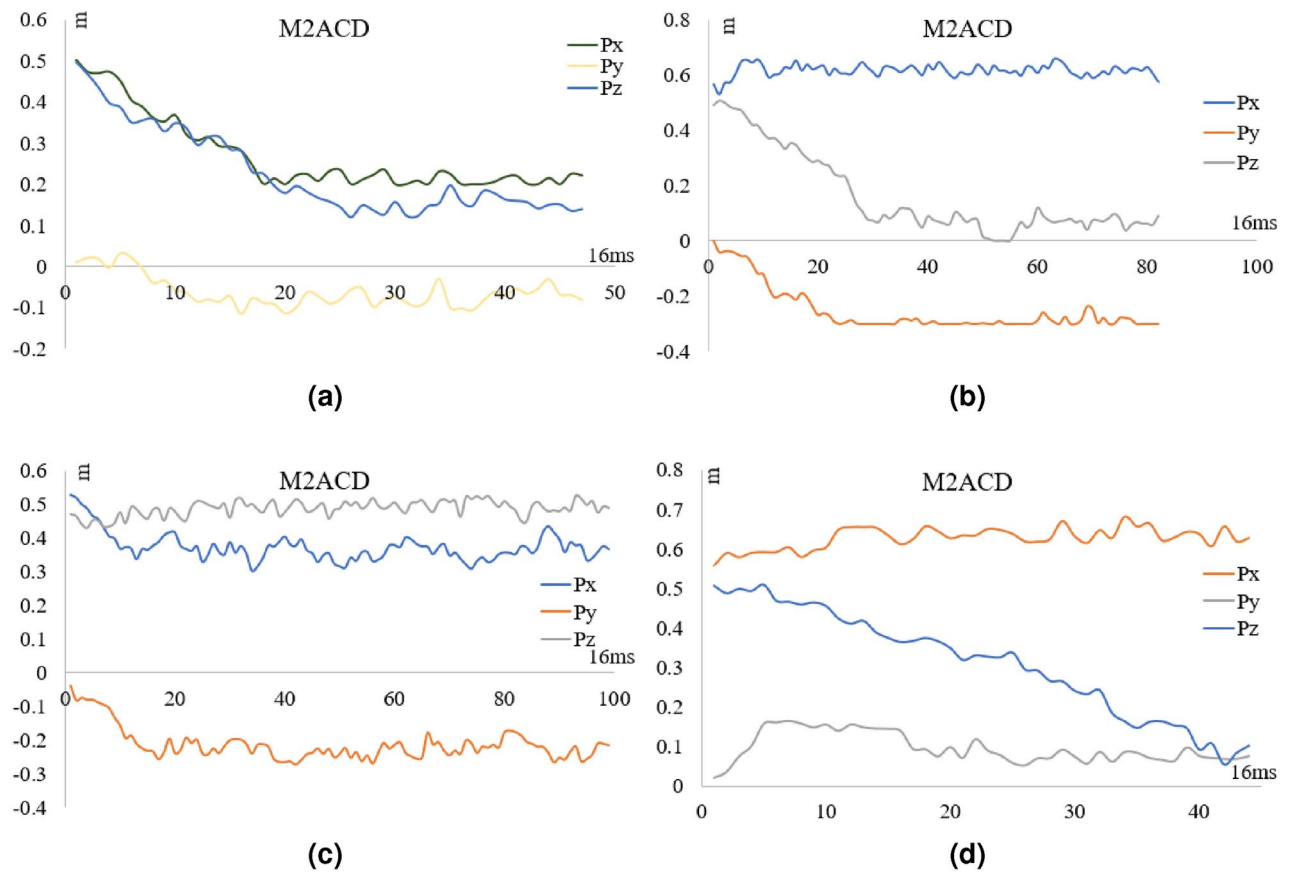


Figure 12. Four times reinforcement learning trajectory planning for randomly selected collaborative robots.

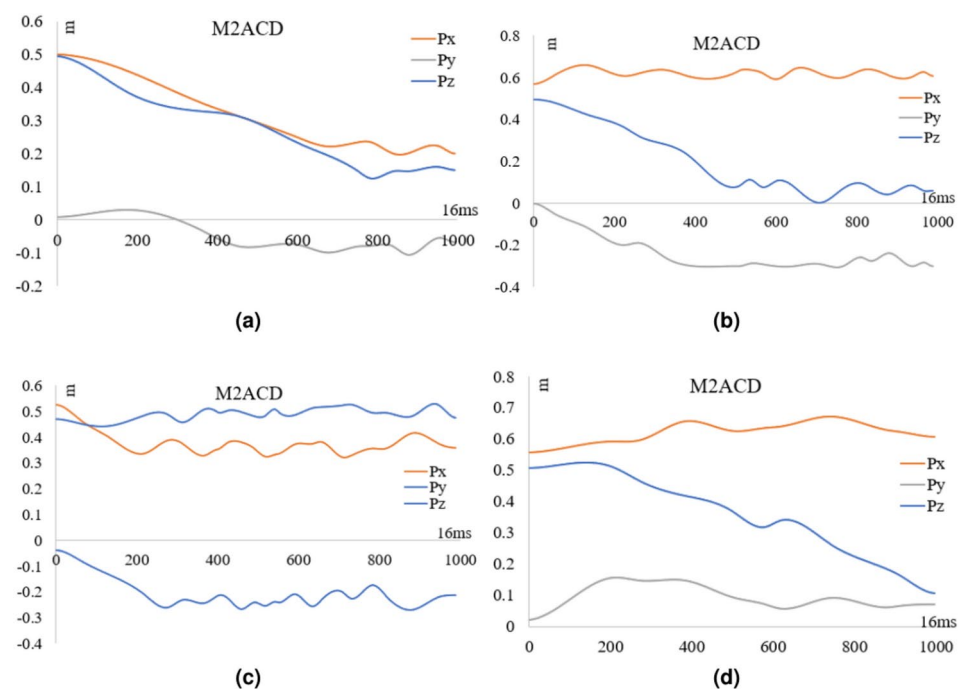


Figure 13. Experimental results of trajectory planning of the cooperative manipulator.

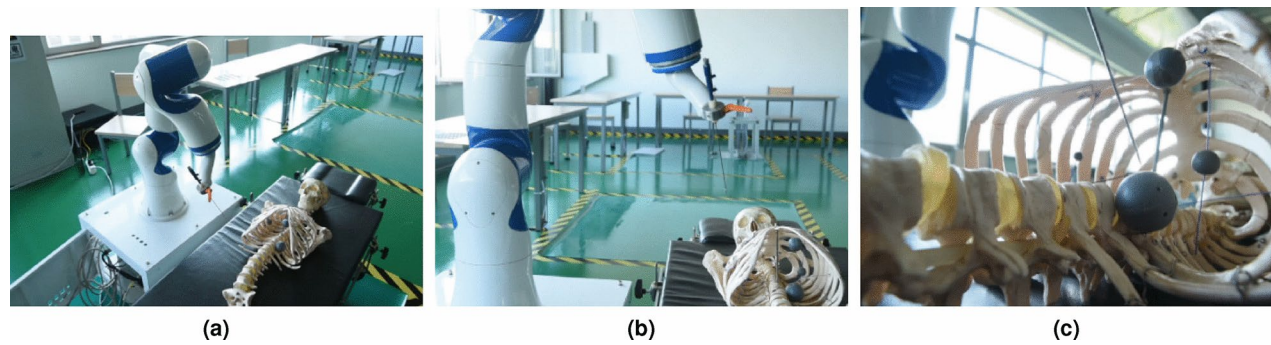


Figure 14. Experimental results of trajectory planning of the cooperative manipulator.

Algorithms	Time (s/episode)	Mutation of success rate
DDPG	1.26	✓
TD3	1.24	✓
M2ACD without TSR	1.19	×
M2ACD	0.98	×

Table 4. Correlation algorithm comparison.

time performance and effectiveness. Therefore, transitioning from simulation to the real world often requires domain adaptation techniques, additional field data, and hardware fine-tuning to overcome these discrepancies.

Conclusion

This paper presents a novel M2ACD algorithm for robotic arm trajectory planning in robots based on the structure of a multi-strategy network and a multi-value network. In addition, the paper proposes a reward strategy based on reward value prioritization and divides the robotic arm grasping task into the approach and close based on the TSR strategy. In order to improve the smoothness of reinforcement learning trajectory planning, NURBS spline is used to smooth the trajectory of M2ACD reinforcement learning output. The M2ACD algorithm is employed in a simulation environment to train the robot’s trajectory planning strategy, which is then transferred to a real experimental platform for grasping experiments. Simulation and real robotic arm experiments show that the proposed M2ACD algorithm is significantly more effective than TD3 and DDPG algorithms, and the M2ACD algorithm takes less time and converges faster than TD3 and DDPG algorithms in the early exploration stage. The planned curve has a good smooth line, which can effectively eliminate unnecessary fluctuations while ensuring accuracy.

Future research will focus on combining vision sensor technology to explore real-time obstacle avoidance planning methods of the M2ACD algorithm in complex obstacle environments. Specifically, the obstacle image information captured by the vision sensor is replaced with the environmental feature information suitable for the state space of the M2ACD algorithm. Regarding reward mechanism design, a positive reward is provided for each successful collision avoidance behavior, and enormous punishment is imposed on the collision behavior to restrain the potential risk. Use real-time feedback from the environment for dynamic adjustment to improve obstacle avoidance in complex and dynamically changing environments. To further enhance its adaptability and robustness in practical application scenarios.

Data availability

The datasets used and analysed during the current study are available from the corresponding author on reasonable request. Corresponding. Bin Zhao email: tech_zhaobin@126.com

Received: 3 December 2024; Accepted: 5 March 2025

Published online: 10 March 2025

References

1. Wang, L. et al. Supervised meta-reinforcement learning with trajectory optimization for manipulation tasks. *IEEE Trans. Cogn. Dev. Syst.* **16**, 1–1 (2024).
2. Luo, R., Tian, H., Ni, W. & Cheng, J. E. A. Deep reinforcement learning enables joint trajectory and communication in internet of robotic things. *IEEE Trans. Wireless Commun.* 1–1 (2024).
3. Han, J., Chai, J. & Hayashibe, M. Synergy emergence in deep reinforcement learning for full-dimensional arm manipulation. *IEEE Trans. Med. Robotics Bionics* **3**, 498–509 (2021).
4. Li, X., Liu, H. & Dong, M. A general framework of motion planning for redundant robot manipulator based on deep reinforcement learning. *IEEE Trans. Industr. Inf.* **18**, 5253–5263 (2021).

5. Su, H. et al. An incremental learning framework for human-like redundancy optimization of anthropomorphic manipulators. *IEEE Trans. Industr. Inf.* **18**, 1864–1872 (2020).
6. Guo, Q., Ye, Z. & Wang, L. e. a. Imitation learning and model integrated excavator trajectory planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5737–5743 (2022).
7. Leng, S. et al. Flexible online planning based residual space object de-spinning for dual-arm space-borne maintenance. *Aerospace Sci. Technol.* **130**, 107907 (2022).
8. Fang, Z. & Liang, X. Intelligent obstacle avoidance path planning method for picking manipulator combined with artificial potential field method. *Ind. Robot.* **49**, 835–850 (2022).
9. Ewerton, M., Maeda, G. & Koert, D. e. a. Reinforcement learning of trajectory distributions: applications in assisted teleoperation and motion planning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4294–4300 (2019).
10. Xue, W. et al. Inverse reinforcement learning for trajectory imitation using static output feedback control. *IEEE Trans. Cybern.* **54**, 1695–1707 (2024).
11. Kar, R. et al. Eeg-induced autonomous game-teaching to a robot arm by human trainers using reinforcement learning. *IEEE Trans. Games* **14**, 610–622 (2021).
12. Leottau, L., Ruiz-del, D., Solar, J. & Babuška, R. Decentralized reinforcement learning of robot behaviors. *Artif. Intell.* **256**, 130–159 (2018).
13. He, H. et al. Deep reinforcement learning-based distributed 3d uav trajectory design. *IEEE Trans. Commun.* **72**, 3736–3751 (2024).
14. Wu, Y., H., Y., Z., C. & Li, Y. E. A., C. Reinforcement learning in dual-arm trajectory planning for a free-floating space robot. *Aerospace Sci. Technol.* **98**, 105657 (2020).
15. Liu, Y. et al. A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping. *Robotics Comput. Integr. Manuf.* **78**, 102365 (2022).
16. Man, H., Ge, N. & Xu, L. Intelligent motion control method based on directional drive for 3-dof robotic arm. In *2021 5th International Conference on Robotics and Automation Sciences (ICRAS)*, 144–149 (IEEE, 2021).
17. Pore, A., Corsi, D. & Marchesini, E. e. a. Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4025–4031 (IEEE, 2021).
18. Cui, Y., Xu, Z. & Zhong, L. e. a. A task-adaptive deep reinforcement learning framework for dual-arm robot manipulation. *IEEE Trans. Autom. Sci. Eng.* (2024).
19. Lindner, T. & Milecki, A. Reinforcement learning-based algorithm to avoid obstacles by the anthropomorphic robotic arm. *Appl. Sci.* **12**, 6629 (2022).
20. Liu, H., Ying, R., F. and Jiang & Shan, Y. e. a. Obstacle-avoidable robotic motion planning framework based on deep reinforcement learning. *IEEE/ASME Trans. Mechatronics* 1–12 (2024).
21. Peng, G., Yang, J. & Li, X. Deep reinforcement learning with a stage incentive mechanism of dense reward for robotic trajectory planning. *IEEE Trans. Syst. Man Cybern. Syst.* **53**, 3566–3573 (2023).
22. Al-Gabalawy, M. A hybrid mpc for constrained deep reinforcement learning applied for planar robotic arm. *ISA Trans.* (2021).
23. Sekkat, H. et al. Vision-based robotic arm control algorithm using deep reinforcement learning for autonomous objects grasping. *Appl. Sci.* **11**, 7917 (2021).

Acknowledgements

This research was funded by the National Natural Science Foundation of China under Grants (U20A20197), the Provincial Key Research and Development for Liaoning under Grant (2020JH2/10100040). Development and application of autonomous working robots in large scenes (02210073421003). Research and development of automatic inspection flight control technology for satellite denial environment UAV (02210073424000)

Author contributions

B.Z. and Y.W. conceived the experiment(s), Y.W. and R.S. conducted the experiment(s), B.Z. and C.W. analysed the results. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Permission

Bin Zhao, Yao Wu, Chengdong Wu, hereby grant Scientific Reports permission, from the copyright holder of the SIASUN logos to publish them under a CC BY open access license to use the logos displayed in Figs. 2 and 5. This work © 2 by B is licensed under CC BY-NC-SA 4.0. Should you require any further information, please do not hesitate to contact me at Email: tech_zhaobin@126.com

Additional information

Correspondence and requests for materials should be addressed to B.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025