International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 2154-2170

https://doi.org/10.70517/ijhsa464184

A joint solution method for path optimization and kinematic constraints of robotic arm tasks under dynamic spatial constraints based on reinforcement learning

Yun Ge[1,*], Qiang Zhang[1], Shengqiang Fan[1], Xin Zhang[1] and Shushu Wang[1]
[1] Marketing Service Center, State Grid Shanxi Electric Power Company, Taiyuan, Shanxi, 030000, China
Corresponding authors: (e-mail: 18802989310@163.com).

Abstract Path planning has always been a key challenge in the field of robotic arm task execution and is a prerequisite for robotic arms to successfully complete specified tasks. This paper begins with the spatial pose and kinematic model of the robotic arm represented by DH, and solves the kinematic model of the robotic arm through forward and inverse kinematics. Starting from Cartesian space trajectory planning, the paper constructs a path optimization model for robotic arm task execution with the objective of minimizing the path execution time, while using kinematic metrics as constraints. Based on the DPPO algorithm in reinforcement learning, the paper introduces the CMA-ES mechanism to construct the DPPO-CMA algorithm and designs corresponding state-action and reward functions. Research shows that the average path length of the DPPO-CMA algorithm is 581.58 mm, which is 158.18 mm shorter than the average path length of the P-RRT* algorithm. The path search time decreases from the average of 163.25 seconds in the P-RRT* algorithm to 29.16 seconds. Additionally, the dynamic response results of the reward value are higher in this algorithm, and the task execution path planning results of the robotic arm exhibit higher stability and positioning accuracy. Reinforcement learning can better learn the task execution status of the robotic arm, thereby improving its efficiency during task execution and ensuring industrial production efficiency.

Index Terms spatial pose, kinematic model, Cartesian space, reinforcement learning, DPPO-CMA algorithm

I. Introduction

In today's high-end manufacturing industry, robots serve as a critical enabling technology, playing an indispensable role across various fields and in human life. Among these, robotic arms are an essential component of industrial robots, with their motion efficiency directly impacting the service quality and control precision of industrial robots [1], [2]. Robotic arms possess robust working capabilities and a wide operational range, enabling them to perform heavy, complex, and high-risk tasks while maintaining high precision [3], [4]. For human society, robots have become highly efficient tools for intelligent production and daily life. With the continuous advancement of scientific and technological innovation, research on robotic arm collaboration systems can be further explored [5]-[7]. Therefore, the application prospects of this system are extremely broad, and its research holds significant importance.

The control system of a robotic arm is a complex and advanced mechatronic system [8]. The target motion trajectory of a robotic arm can be obtained through manual calculation, and then the motion velocity curve of the robotic arm can be determined through trajectory planning [9], [10]. Currently, the phase plane method is generally used for robotic arm dynamics modeling, where, after setting the trajectory parameters, the dynamics model data is derived through formula derivation. However, this method has issues with low precision [11]-[13]. To address path planning issues during collaborative operations of robotic arms, it is necessary to find a path connecting the starting point and the target point within a given state space while ensuring that the path does not collide with any obstacles [14]-[16]. Therefore, exploring an efficient path planning algorithm to assist robots in performing obstacle avoidance tasks in environments with obstacles will further enhance efficiency and quality in practical applications [17], [18].

Currently, various path planning algorithms for robotic arm control systems have emerged in the academic community, each with its own advantages in different scenarios. Yu, X, et al. designed a robotic arm trajectory planning method aimed at time optimization, using a genetic algorithm to calculate the movement time of different points along the robotic arm's motion path, thereby obtaining the time-optimal robotic arm motion path [19]. Wang, L, et al. combined the TPBSO algorithm to explore the motion strategies of robotic arms under point-to-point path and fixed geometric path conditions. The robotic arm control model supported by this heuristic algorithm can achieve high computational speed and control performance without increasing computational complexity [20]. Cao, X et al.

established a motion trajectory planning model for robotic arms based on the multi-objective particle swarm optimization algorithm (GMOPSO), which ensures smooth operation of the robotic arm during task execution while maintaining high efficiency [21]. Kucuk, S. proposed an optimal trajectory generation algorithm (OTGA) for generating the shortest time smooth motion trajectories for serial and parallel robotic arms. This algorithm can optimize unreasonable motion path points, enabling smoother start and stop motions for robotic arm joints [22]. Zhang, Z et al. designed a novel three-criteria optimization-coordination-motion (TCOCM) strategy to address the redundancy issue in dual robotic arms. This strategy fully considers the physical constraints of robotic arm joints, effectively preventing excessive joint speeds and joint angle drift during robotic arm motion [23]. Chen, G et al. utilized the Probabilistic Landmark Method (PRM) to plan obstacle-avoidance paths for high-dimensional robotic arms, enabling the robotic arm to quickly generate a short and safe motion path in complex workspaces in response to existing obstacles. This method is applicable to robotic arms with any degree of freedom [24]. However, the aforementioned studies are all based on searches using predefined maps or environmental models. In dynamic environments, when the position or state of obstacles changes, the algorithms struggle to adjust in a timely manner. Additionally, they do not consider the various joint configurations and motion paths within the robot arm's reachable range, resulting in suboptimal path planning performance.

Path planning for robotic arm task execution is a critical step in completing operational tasks, and path optimization further enhances task completion quality while aligning with industrial development needs. This paper starts from the spatial pose of the robotic arm, models its kinematic behavior using the Denavit-Hartenberg model, and solves it using forward and inverse kinematics. Dynamic spatial constraints are imposed based on Cartesian space trajectory planning, and the shortest execution time for the robotic arm's task execution path is selected as the objective to construct a path optimization model for the robotic arm's task execution path. Starting from reinforcement learning, the DPPO algorithm is selected to solve the path optimization problem for the robotic arm's task execution path. To improve its solution efficiency, this paper introduces the CMA-ES mechanism to optimize the DPPO algorithm, thereby enhancing the quality of path planning for the robotic arm's task execution.

II. Mechanical arm task execution path optimization model

Robotic arms have been widely used in various fields of industrial production, especially in the automotive, aviation, and shipbuilding industries. Considering the complex nature of industrial environments, collisions may occur during the operation of robotic arms. To ensure the safety and efficiency of production, it is necessary to plan an obstacle-avoidance and optimized path in advance for task execution. However, current robotic arm path planning often ignores dynamics and other forms of differential constraints, focusing solely on search algorithms, and thus fails to effectively meet the requirements for task execution paths in industrial robotic arms.

II. A.Modeling of robotic arm kinematics

II. A. 1) Spatial Pose Description of Robotic Arms

The position and orientation description of a robotic arm primarily involves the specific position and state of the end-effector of the robotic arm in space. This is typically represented by two main parameters: position and orientation. Position description refers to the coordinates of the end-effector in space. This is typically represented by three coordinate values (x, y, z), which denote the position distribution of the end-effector along the x, y, and z axes in the Cartesian coordinate system, indicating the movement of the robotic arm in the lateral, longitudinal, and height directions. Attitude description refers to the orientation of the end-effector in space. This is typically represented by three rotational coordinate quantities R_x, R_y, R_z, which denote the rotational changes around the x th axis, y th axis, and z th axis. These three rotational angles collectively determine the orientation of the end-effector [25].

(1) Position description. In three-dimensional space, to accurately describe the position of a moving object, this paper establishes a fixed coordinate system at the base of the robotic arm as the base coordinate system and also establishes a coordinate system at the rigid body position. After establishing the base coordinate system, the position of any point in space can be precisely represented using a matrix. Assuming that a base coordinate system $\{A\}$ has been established in space and there is a point P, the position of point P in the base coordinate system $\{A\}$ can be represented by a matrix. That is:

$$P_A = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{1}$$

(2) Attitude description. When the attitude of a rigid body changes, the attitude of the coordinate system fixed to the rigid body also changes accordingly. Assume that a coordinate system $\{B\}$ is established on the rigid body,

with its origin at the center of the rigid body. The coordinate values of the rigid body's attitude along the three axes of coordinate system $\{B\}$, projected onto the three axes of the base coordinate system $\{A\}$, can be represented by matrix ${}^{B}_{A}R$ in terms of the cosine of their direction relative to the base coordinate system $\{A\}$. That is:

$$ {}^{B}_{A}R = \begin{bmatrix} {}^{B}_{A}X & {}^{B}_{A}Y & {}^{B}_{A}Z \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \tag{2} $$

(3) Pose description. The pose of a rigid body is determined by its position and orientation. Therefore, multiplying equations (1) and (2) yields the pose of the rigid body in the base coordinate system. That is:

$$ P_A = \begin{bmatrix} {}^{B}_{A}R & {}^{B}_{A}P \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

II. A. 2) Representation of the kinematic model of a robotic arm

In practical applications, the tasks of a robotic arm are executed by the end-effector in Cartesian space, while control is implemented by the joints in joint space. Typically, only the relative pose between the robotic arm's base coordinate system and the world coordinate system is known. However, to achieve robotic arm operation, it is also necessary to know the pose of the end-effector in the workspace. Therefore, it is necessary to establish the relative pose relationship between the end effector and the base coordinate system, as well as the mapping relationship between Cartesian space and joint space. This paper selects the six-degree-of-freedom rigid robotic arm RockyOne from a certain company as the research object. This robotic arm consists of six rotational joints and six rigid arms. The standard Denavit-Hartenberg model is adopted for its modeling. First, it is simplified to the structural diagram shown in Figure 1, where numbers 1 to 6 represent the six rotational joints of the robotic arm. Then, based on the principles of the coordinate systems of each joint of the robotic arm, the joint coordinate systems are constructed, where O_0 is the world coordinate system and $x_i y_i z_i$ is the coordinate system of each joint. Finally, the standard DH model parameters of the robotic arm are obtained.

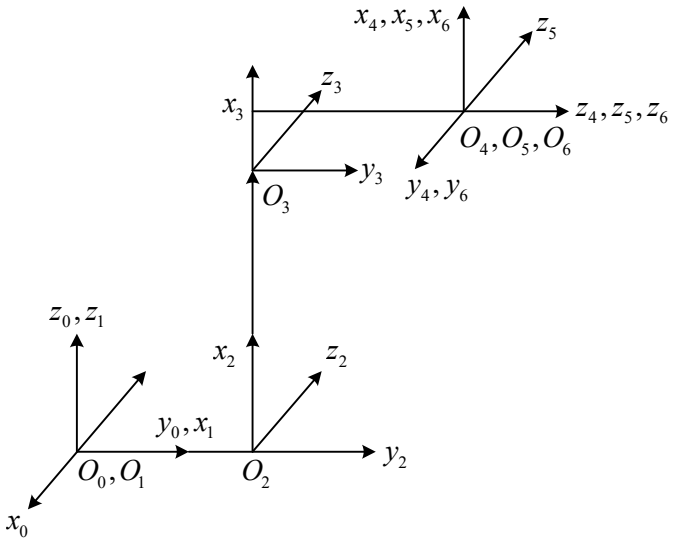


Figure 1: A simplified diagram of the coordinate structure of the robotic arm joint

In a rigid arm, the following relationship holds between two adjacent joint coordinate systems:

$$ {}^{i-1}_{i}T = \begin{Bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{4} $$

In the equation, $c\theta_i$ represents $\cos\theta_1$, $s\theta_i$ represents $\sin\theta_i, \theta_1, \alpha_{i-1}$, and d_i represents the standard DH model parameters. By successively multiplying the transfer matrix from the right, the transfer matrix from the end-effector coordinate system to the base coordinate system can be obtained as follows:

$$ {}^{0}_{n}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \cdots {}^{n-1}_{n}T \tag{5} $$

In the equation, n represents the number of joints in the robotic arm. This equation represents the forward kinematic model of the robotic arm. By substituting the standard DH model parameters, the pose of the robotic arm's end effector relative to the world coordinate system in the workspace can be obtained [26]. In robotics, the mapping from Cartesian space to joint space is typically achieved using the Jacobian matrix, which maps joint velocities to generalized Cartesian velocities at the end of the robotic arm, satisfying the following relationship:

$$ \dot{x} = J(q)\dot{q} \tag{6} $$

In the formula:

$$ \dot{x} = \begin{bmatrix} v_x & v_y & v_z & w_x & w_y & w_z \end{bmatrix}^T \tag{7} $$

where v is the Cartesian linear velocity at the end of the robotic arm, w is the Cartesian angular velocity at the end of the robotic arm, and $J(q) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix, then:

$$ \dot{q} = [\dot{\theta}_1 \quad \dot{\theta}_2 \quad \cdots \quad \dot{\theta}_n]^T \tag{8} $$

$\dot{\theta}$ is the speed of the robot arm joint.

II. A. 3) Solving forward and inverse kinematics of robotic arms

(1) Forward kinematics solution of the robotic arm

After completing the DH representation of the robotic arm, perform the forward kinematics solution of the robotic arm, i.e., obtain the end-effector pose and coordinate position through the known link joint variable values. The homogeneous transformation coordinate matrix between adjacent links of the robotic arm is:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos a_{i-1} & \cos\theta_i \cos a_{i-1} & -\sin a_{i-1} & -d_i \sin a_{i-1} \\ \sin\theta_i \sin a_{i-1} & \cos\theta_i \sin a_{i-1} & \cos a_{i-1} & d_i \cos a_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9} $$

Using the DH method, the parameters between the joints and linkages of the two robotic arms have been obtained. Next, we will solve the forward kinematics of the main robotic arm. Substituting the DH parameters of the main robotic arm into the above equation, we obtain:

$$ {}^{0}T_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10} $$

$$ {}^{1}T_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0.285 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11} $$

$$ {}^{2}T_{3} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 1.024 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12} $$

$$ {}^{3}T_{4} = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0.250 \\ 0 & 0 & 1 & 1.225 \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13} $$

$$ {}^{4}T_{5} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14} $$

$$ {}^{5}T_{6} = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0.225 \\ -\sin\theta_6 & -\cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15} $$

Multiplying the above matrices yields the pose transformation matrix of the end-effector coordinate system of the main robotic arm relative to the base coordinate system, i.e.:

$$ {}^{0}T_{6} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5}\,{}^{5}T_{6} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16} $$

In the equation, $\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$ represents the attitude vector, indicating the attitude of the coordinate system relative to the reference coordinate system; $[n_x \quad n_y \quad n_z]^T$ represents the X-axis direction of the coordinate system; $[o_x \quad o_y \quad o_z]^T$ represents the Y-axis direction of the coordinate system; $[a_x \quad a_y \quad a_z]^T$ represents the Z-axis direction of the coordinate system; and $[p_x \quad p_y \quad p_z]$ represents the current coordinate values of the end-effector.

(2) Solving the inverse kinematics of the robotic arm

Inverse kinematics refers to the process of obtaining the rotation angles of each joint of the robotic arm by solving the inverse of the forward kinematics transformation matrix after the pose matrix of the end-effector coordinate system of the robotic arm is known. The process of solving the inverse kinematics solution using matrix inversion is as follows:

$$ \begin{cases} T = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5}\,{}^{5}T \\ T_1{}^{-1}T = {}^{1}T_{2}\,{}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5}\,{}^{5}T, \\ T_1{}^{-11}T_2{}^{-1}T = {}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5}\,{}^{5}T, \\ \vdots \\ T_1{}^{-11}T_2{}^{-12}T_3{}^{-13}T_4{}^{-1}T = T_5\,{}^{5}T_6 \end{cases} \tag{17} $$

Due to the complexity of the derivation process, the specific details will not be elaborated upon in this paper. According to Picper's principle, the solution to the inverse kinematics of a robotic arm is not unique. Therefore, in practical applications, the corresponding inverse solution for the robotic arm should be determined based on the specific task requirements and the motion trajectory of the path planning.

II. B. Task execution path optimization model

II. B. 1) Cartesian Space Trajectory Planning

To ensure that the joint positions, joint angular velocities, and joint angular accelerations of each rotational joint of the robotic arm are continuous and smooth, it is necessary to perform joint space trajectory planning for the robotic arm. Solving for the function describing how the pose of the robotic arm's end-effector changes over time in the workspace constitutes Cartesian space trajectory planning for the robotic arm. The basic approach involves first obtaining a sequence of paths in Cartesian space and using interpolation algorithms to implement them. Then, inverse kinematics calculations are performed on the spatial coordinates of each joint of the robotic arm to map the position and orientation of the robotic arm at each point in Cartesian space to the position and orientation of each joint in joint space [27].

Linear trajectory planning for a robotic arm in Cartesian space is achieved by knowing the base coordinate system B of the robotic arm, as well as the starting coordinates $P_s(x_s, y_s, z_s)$ and ending coordinates $P_g(x_g, y_g, z_g)$ of its end effector. The goal is to ensure that the end effector of the robotic arm moves along a straight-line trajectory between these two sets of coordinates. The following steps are used to plan the straight-line trajectory of the robotic arm in its workspace:

(1) When the end effector moves along a straight line, set its velocity v and the robot arm's control cycle T_s.

(2) Calculate the total movement time t and the number of interpolations N along the trajectory, i.e.:

$$t = \frac{|P_s P_g|}{v} \tag{18}$$

$$N = round(\frac{t}{T_s}) \tag{19}$$

Here, $|\cdot|$ represents the vector length, and $round(\cdot)$ represents the floor function. Therefore, the accuracy of the straight-line trajectory is determined by the number of interpolations. To make the end-effector's motion trajectory more closely approximate a straight line, more interpolations are required. However, the more interpolations there are, the higher the computational load required for the robot arm's trajectory generation.

(3) Calculate the coordinates $P_i(x_i, y_i, z_i)$ of the third interpolation point i, that is:

$$\begin{cases} x_i = x_s + i\dfrac{x_g - x_s}{N} \\ y_i = y_s + i\dfrac{y_g - y_s}{N} \\ x_i = x_s + i\dfrac{y_g - y_s}{N} \end{cases} \tag{20}$$

(4) Solve for P_s, P_g and the interpolation points using inverse kinematics, then convert the configuration of the interpolation points of the robotic arm's motion function in Cartesian space into the rotational joint configuration of the robotic arm in joint space. Then perform trajectory planning for the robotic arm in joint space to optimize the linear motion planning function of the robotic arm in Cartesian space.

II. B. 2) Task execution path optimization design

Assuming that the end-effector of the robotic arm needs to pass through n task points in three-dimensional space, and the coordinates of these task points in the workspace are given, the inverse kinematics of the robotic arm can be directly used to solve the problem, converting the spatial position coordinates into joint angle values for each joint of the robotic arm. Additionally, the path between any two adjacent points is a short segment. For n task points, there are $n-1$ path segments. Let the total time the robotic arm's end-effector takes to traverse each path segment be $T_i(i = 0,1,2,3,\cdots\cdots,n-1)$ and $T_i = t_{i+1} - t_i$, respectively. Here, t_i represents the time it takes for the robotic arm to move to the i th task point. Therefore, the total time required for the robotic arm's operation is:

$$T = T_1 + T_2 + \cdots + T_{i-1} = \sum_{i=1}^{n-1} T_i \tag{21}$$

Among them, the total time T is the objective function, and the time T_i used for each segment of the path is the variable. Based on the hardware parameters of the robotic arm, the maximum angular velocity, maximum angular acceleration, and maximum angular acceleration of each joint are used as constraint conditions. The objective function and constraint conditions for the six-degree-of-freedom robotic arm trajectory optimization problem used in this paper are as follows:

(1) The objective function can be expressed as:

$$\min T = \sum_{i=1}^{n-1} T_i \tag{22}$$

(2) The constraints are as follows:

The angular velocity constraint in kinematics is:

$$\begin{cases} \forall t \in [t_i, t_{i+1}] \\ |\dot{\theta}_i(t)| \le \dot{\theta}_{\max} \end{cases} \tag{23}$$

Among them, $\dot{\theta}_{\max}$ is the maximum angular velocity that each joint can reach when the robotic arm is working.

The angular acceleration constraint in kinematics is:

$$\begin{cases} \forall t \in [t_i, t_{i+1}] \\ |\ddot{\theta}_i(t)| \le \ddot{\theta}_{\max} \end{cases} \tag{24}$$

Among them, $\ddot{\theta}_{\max}$ is the maximum angular acceleration that each joint can reach when the robotic arm is working.

The angular acceleration constraint in kinematics is:

$$\begin{cases} \forall t \in [t_i, t_{i+1}] \\ |\dddot{\theta}_i(t)| \le \dddot{\theta}_{\max} \end{cases} \tag{25}$$

Among them, $\dddot{\theta}_{\max}$ is the maximum angular acceleration that each joint can achieve when the robotic arm is operating.

III. Optimization of task execution paths for robotic arms

Robotic arms are one of the earliest types of robots to be applied in actual production fields. Among them, serial robotic arms are the most widely used. They consist of a series of linked rods connected by joints, forming an open-chain link mechanism with multiple degrees of freedom. One end is fixed to a base, while the other end serves as the end-effector, primarily performing grasping operations. The intermediate section is composed of a series of driven rotating or moving joints connected in series. Robotic arm kinematics studies the mathematical relationships between the motion of each link joint and the position and orientation of various parts of the robotic arm (typically the end-effector). Understanding these relationships is essential for designing robotic arm motion controllers and is a key focus of research into optimizing task execution paths for robotic arms.

III. A. Reinforcement Learning and DPPO Algorithm

III. A. 1) Basic Principles of Reinforcement Learning

Reinforcement learning is a type of machine learning that primarily describes how an intelligent agent interacts with its environment through action selection to obtain rewards, thereby gradually learning the strategy that maximizes cumulative value. The structure of reinforcement learning consists of an intelligent agent, actions, states, rewards, value, and the environment.

(1) Agent. The agent can make different action selections based on its strategy, transitioning from the current state to the next state and thereby receiving a reward. Since the rewards obtained from transitioning to different states vary, the agent can gradually accumulate rewards from different selections through multiple training processes, ultimately learning the strategy that maximizes cumulative rewards. The agent is the basic unit of the reinforcement learning model.

(2) Action. The agent selects different actions to reach different states, with action selections based on the strategy.

(3) Rewards. Rewards are the feedback received by the agent after selecting an action and transitioning from the current state. Rewards reflect the impact of the selection on the overall task. The agent receives corresponding rewards for each state transition, and through the accumulation of multiple rewards, the agent gradually learns the

strategy that maximizes rewards.

(4) Value. Unlike immediate rewards, the value function reflects long-term cumulative benefits, including the agent's estimation of the total benefits from a series of subsequent actions. This prevents the agent from focusing solely on immediate gains, making its choices more globally oriented.

(5) Policy. A policy is the mapping from the state space to the action space. The agent makes action selections based on the policy to change its current state. The ultimate goal of reinforcement learning is to obtain an optimal policy. During the model learning process, it is necessary to consider how to properly balance exploration and exploitation.

Reinforcement learning problems are typically described using Markov decision processes (MDPs). Most reinforcement learning algorithms use MDPs to build models, which are then solved to address reinforcement learning problems. A quintuple $M = (\mathrm{S}, \mathrm{A}, r, p, \gamma)$ can be used to describe an MDP, where S and A represent a set of states and actions, p represents the probability of state transitions, r represents the reward for state transitions due to environmental changes, and γ is a discount factor used to determine the priority of short-term rewards, typically a value between 0 and 1. The agent's policy is expressed as:

$$\mu = p(a \mid s) \tag{26}$$

Among these, p represents the probability of taking action a in state s. For each state, a deterministic policy only outputs one deterministic action, with a probability of 1, while the probabilities of all other possible actions are 0. A stochastic policy provides an action probability distribution, i.e., the probability of each possible action being selected. In an MDP, the policy is only dependent on the current state and does not consider historical states. Reinforcement learning agents learn how to maximize their long-term rewards through interaction with the environment. This long-term reward is typically the sum of discounted rewards, with rewards further in the future receiving larger discounts. Rewards can be represented as:

$$R_t = \sum_{i=t}^{r} \gamma^{i-t} r(s_t, a_t) \tag{27}$$

At each discrete time step t, given a state, the agent can select an action using policy μ to obtain a reward r, and then transition to the next state. In an MDP, the agent observes the environment state to select an action, which acts on the environment. The environment then provides a new state and reward, and the agent updates its policy based on this information. It then continues to observe the new state and select the next action. This process continues until a termination condition is met.

III. A. 2) Distributed proximal strategy optimization algorithm

The core of the Distributed Proximal Policy Optimization (DPPO) algorithm is the Proximal Policy Optimization (PPO) algorithm. The PPO algorithm is a deep reinforcement learning algorithm based on the Actor-Critic framework, designed to address the issues of slow network parameter updates and difficulty in determining the learning rate (learning step size) in the Actor's Policy Gradient (PG) algorithm. If the learning rate is set too high, the action strategy will experience significant fluctuations, making it difficult to converge. If the learning rate is too low, parameter updates will be slow, significantly increasing training time and failing to meet experimental requirements. The PPO algorithm limits the update magnitude of the new strategy based on the ratio of new to old strategies, enabling the PG algorithm to train and converge at a higher learning rate [28].

The objective function of the PG algorithm in the Actor is:

$$J(\theta) = E_t[A(s_t, a_t)\pi_\theta(a_t \mid s_t)] \tag{28}$$

In the equation, $\pi(\cdot)$ represents the policy function, θ represents the Actor network parameters, s_t, a_t represents the state and action at step t, $A(s_t, a_t)$ represents the advantage function, and E_t represents the empirical expectation of the time step length.

The advantage function compares the score obtained by selecting a_t under state s_t with the average score. If the score is higher, the advantage function is positive; otherwise, it is negative. The gradient ascent method is used to update θ, maximizing the objective function $J(\theta)$ to achieve the goal of optimizing the strategy score.

Since the PG algorithm uses online strategy updates, each parameter update requires resampling, making it difficult to determine the learning rate. The PPO algorithm converts online strategy updates to offline strategy updates, i.e., it uses a new and old Actor strategy, where the training data for the new Actor can be obtained from the old Actor. The action probability ratio $r_i(\theta)$ between the new and old strategies is used to represent the new strategy weight, which is expressed as:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)} \tag{29}$$

In the formula, θ' represents the old strategy network parameters.

The PPO algorithm objective function can be expressed as:

$$J^{CPI}(\theta') = E_t\left[A(s_t, a_t)\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)}\right] = E_t[A(s_t, a_t) r_t(\theta)] \tag{30}$$

In gradient ascent methods, to ensure that the expected return of the policy is monotonically increasing, the PPO algorithm restricts the difference between the old and new policy distributions to be small. That is, under the same state, the action probabilities obtained by the two networks cannot differ too greatly; otherwise, a large number of samples would be required to obtain an approximate result. To address the issue of significant changes in the policy, the PPO algorithm employs two solutions: one is to use a KL-penalty, and the other is to use a clip to limit the range of changes between the old and new policies within a specified interval. This paper adopts the clip method for training, with the objective function defined as:

$$J^{CLIP}(\theta') = E_t[\min(A(s_t, a_t) r_t(\theta), clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon) A(s_t, a_t))] \tag{31}$$

In the equation, ε is the hyperparameter, and $clip(\cdot)$ is the clipping function. This limits the value of $r_t(\theta)$ to within the range of $[-\varepsilon, 1+\varepsilon]$. The objective function ultimately selects the minimum value between the original value and the clipped value to prevent the parameters of the new policy network θ from updating too quickly.

The DPPO algorithm builds upon the PPO algorithm by introducing multi-threading. The multi-threading framework of the DPPO algorithm is shown in Figure 2. The main thread is responsible for updating the Actor and Critic parameters, while multiple auxiliary threads interact independently with the environment to collect data. By calculating gradients and aggregating them, they collectively update the network parameters. Specifically, multiple agents are added to the environment for parallel training. An agent can communicate with other agents and provide mutual feedback between strategies, significantly improving training efficiency and addressing issues such as slow convergence.

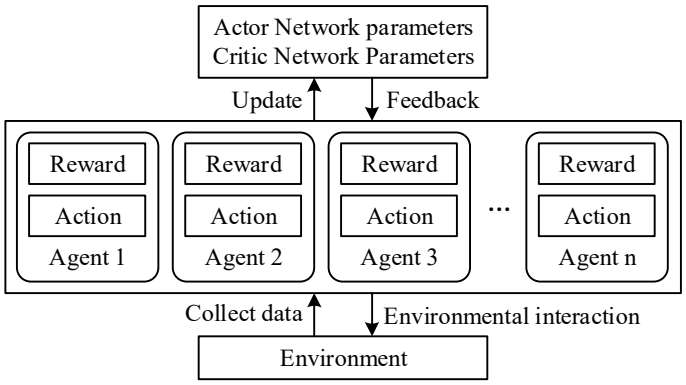


Figure 2: Multi-thread processing framework of DPPO

III. B. Optimization Design of Mechanical Arm Task Execution Path

III. B. 1) Task Execution Path Optimization Framework

Robot arm task execution path planning is a crucial component of robot control. Traditional robot arm trajectory planning methods are generally applicable to known structured environments and cannot address trajectory planning issues for robot arms in unknown working environments under dynamic spatial constraints. The emergence of deep reinforcement learning (DRL) has enabled robot arms to acquire autonomous learning capabilities, allowing them to independently complete trajectory planning in unknown environments. Based on this, this paper uses the DPPO algorithm as a foundation and introduces the CMA-ES mechanism to construct a robot arm task execution path optimization framework under Cartesian dynamic spatial constraints, as shown in Figure 3. The agent in the DPPO algorithm uses an “exploration-trial and error” mechanism. Based on the reward values provided by the reward function, it controls the robot arm to continuously explore the unknown working environment and ultimately plans a motion trajectory with the maximum cumulative reward through autonomous learning.

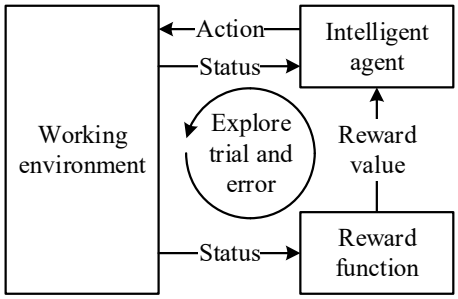


Figure 3: Optimal design of the task execution path for the robotic arm

The DPPO algorithm is used for trajectory planning tasks of robotic arms in unknown environments. Its penalty term mechanism is utilized to keep the update amplitude of the planning strategy within a reasonable range. In addition, this paper comprehensively considers factors such as the relative direction and relative position of the robotic arm in relation to the task objective during the learning and planning process. A new reward function design method is proposed to accurately evaluate the quality of the planned trajectory and reduce ineffective exploration by the robotic arm.

III. B. 2) Action Selection and Reward Function Design

For a discrete point P_d on the task trajectory, once the six joint angle offsets of the robotic arm gripping the heterogeneous component are determined, the pose of point P_d in the world coordinate system can be derived using the DH parameters. Then, based on the relationship between the two robotic arm base coordinate systems T_{B2}^{B1}, the processing pose of point P_d is transformed into the robotic arm base coordinate system. By solving the inverse problem, the six joint angle offsets of the robotic arm are determined, thereby obtaining the information of the twelve joint angles of the dual robotic arm (the inverse solution may be unsolvable, i.e., the joint angle offsets are “null”).

The six joint angle offsets of the robotic arm are used as the state, i.e., $s=(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$. The action is the change in the pose of the end of the sixth axis of the robotic arm, i.e., $e=(x,y,z,\theta_x,\theta_y,\theta_z)$, and x,y,z is the change in the coordinates of the end of the sixth axis in the base coordinate system, with a range of [−30mm, 30mm]. θz, θy, and θx are the three Euler angle coordinate axis angle changes of the end of the sixth axis rotating in the ZYX order, ranging from [−Π/5, Π/5].

For a task execution trajectory line, given the initial state, i.e., the joint angle offset of the robotic arm at the starting point, control the robotic arm to perform a certain action to enter the next discrete point. The joint angle offset of the robotic arm corresponding to the solution is obtained. Based on the information of the twelve joint angles, analyze the robotic arm's reachability, collision, and other metrics. When a collision occurs or all discrete points on the trajectory are completed, the round ends; otherwise, continue to control the robotic arm to perform a certain action and move to the next discrete point.

The reward function guides the learning process to make the results tend toward the optimization goal. The goal of reinforcement learning is to obtain the maximum cumulative reward value, find the optimal robotic arm pose combination corresponding to the processing trajectory on the heterogeneous part, and achieve path optimization. For the reward value r obtained from moving from the previous discrete point to the current discrete point, it must first be ensured that the robotic arm's position is reachable, that the robotic arm has an inverse solution, and that there is no collision between robotic arms. Otherwise, a low reward value is directly assigned, and the round is terminated, prompting the robotic arm to gradually avoid such positions in subsequent rounds. If the robotic arm position is reachable and there are no collisions, the reward function for the robotic arm moving from the current discrete point to the next discrete point is defined as:

$$r=\frac{1}{f} \tag{32}$$

Among them, is the stability optimization function, i.e., $f=\sum_{i=1}^{12}K_i(\theta_i-\theta_i')$. K_i is the weighted coefficient of the i-th joint of the dual robotic arm, and its value is selected based on the impact of joint displacement changes on stability. θ_i,θ' represents the angular displacement of the i-th joint of the robotic arm at the current discrete point and the previous discrete point, respectively.

III. B. 3) Adaptive DPPO-CMA Algorithm

The Distributed Proximal Policy Optimization (DPPO) algorithm is integrated into industrial robot systems, employing parallel training to train robotic arms in the same environment. Drawing on existing relevant research, the core mechanism of CMA-ES is introduced into the DPPO framework to improve the process of strategy parameter optimization. By expanding the exploration range of strategy variance, the update mechanism for strategy parameters is refined, thereby enhancing the ability to search for local optima. In CMA-ES, the update of strategy parameters is based on multiple candidate models generated by perturbing internal parameters. The mean and standard deviation of the strategy for the robotic arm system are generated by independent networks, and their parameters are updated independently. The strategy is expressed as:

$$\sum_k \left[\frac{(a_i^t - \mu_\theta(o_i^t))^2}{\sigma_\theta(o_i^t)} + 0.5 \log \sigma_\theta(o_i^t) \right] \sim \pi_\theta(a_i^t \mid o_i^t) \mid \tag{33}$$

where $\pi_\theta(a_i^t \mid o_i^t)$ represents the current policy. "~" indicates that the distributions on both sides are identical. Policy parameter θ has two forms of representation: θ^μ and θ^σ, representing the parameters of the mean network and variance network, respectively. The update process for θ involves independent updates of θ^μ and θ^σ. Specifically, during the update of θ^σ, gradient optimization of the policy mean is paused, and similarly, during the update of θ^μ, gradient optimization of the policy variance is paused. This sequence helps to expand the exploration distribution in the optimal search direction and avoid premature variance contraction.

The hyperparameter ε (typically $\varepsilon = 0.2$) specified in the original DPPO algorithm has been adjusted and changed to a dynamic parameter η. The new objective function is defined as:

$$L^P(\theta) = E_t[\min(r^t(\theta)A_i^t, clip(r^t(\theta), 1-\eta, 1+\eta))A_i^t] \tag{34}$$

Among them, $\tilde{r}(\theta) = \frac{\pi_\theta(\alpha_i^{'} \mid o_i^{'})}{\pi_\theta(\alpha_i^{'} \mid o_i^{'})}$ is defined as the strategy ratio, which quantifies the change in strategy before and after parameter update. η is a dynamic parameter adjusted according to the difference in KL divergence between the old and new strategies. After the strategy update, the KL divergence between the old and new strategies is calculated as follows:

$$D_{KL}(\pi_{\theta_{add}} \| \pi_\theta) = E_{\tau \sim \pi_{add}} \left[\log \frac{\pi_{\theta_{add}}(a \mid o)}{\pi_\theta(a \mid o)} \right] \tag{35}$$

At this point, the cut range of the strategy can be expressed as:

$$\eta = \begin{cases} \varepsilon \times \max(1 - \alpha(\frac{D_{KL}}{D_{target}} - 1), 0.5) & if D_{KL} > D_{target} \\ \varepsilon \times \max(1 + \beta(1 - \frac{D_{KL}}{D_{target}}), 2.0) & if D_{KL} < D_{target} \end{cases} \tag{36}$$

Among them, ε is the initial shear range, and α and β are sensitivity parameters for adjusting the shear range. The max function is used to limit the variation amplitude of η to ensure training stability. D_{target} is the target value of KL divergence.

IV. Simulation results of robotic arm task execution experiments

Since its inception, the robotic arm has been widely applied in various fields such as industrial production and military applications. In the field of robotic arm research, path planning is one of the key areas of study, and the level of path planning is also an important indicator of the robotic arm's intelligence. This paper conducts path optimization for robotic arm task execution under the constraints of Cartesian dynamic space and kinematic constraints, with the objective of minimizing the time required for the robotic arm to execute its task. The aim is to further enhance the application effectiveness of the robotic arm.

IV. A. Adaptability of task execution path optimization algorithms

IV. A. 1) Adaptive verification results of the algorithm

To validate the advantages and effectiveness of the DPPO-CMA algorithm proposed in this paper, the algorithm was verified through MATLAB simulation. The computer processor used was an Intel(R) Core(TM) i3-10100 CPU

@ 3.60GHz 3.60 GHz, with 8GB of RAM, and the simulation experiment platform was MATLAB R2023.

The algorithm's adaptability metrics in complex state spaces are measured by path search time, path length, and iteration stability. In this paper, four different types of environments were set up in the dynamic space of a robotic arm: sphere (E1), rectangular prism (E2), cylinder (E3), and rectangular prism, cylinder, and sphere (E4). The main comparison algorithms selected were RRT*, Bi-RRT*, and P-RRT*. Each comparison experiment involved searching for paths within different obstacle state spaces, with a search step size of 4 mm, a threshold of 15 mm, a radius of 20 mm, and a maximum iteration count of 3,000. Each experiment was conducted 50 times, and the average was taken. Table 1 presents the results for path length, search time, and iteration count across different algorithms.

Analysis of the data in the table shows that in complex obstacle sampling spaces, the DPPO-CMA algorithm converges significantly faster in terms of path planning time and iteration count. In four environments, the average iteration counts for P-RRT* and DPPO-CMA were 964.05 and 534.55, respectively. The average number of iterations for the DPPO-CMA algorithm was reduced by 44.55%. Additionally, the average path lengths for the P-RRT* and DPPO-CMA algorithms were 739.76 mm and 581.58 mm, respectively. The DPPO-CMA algorithm reduced the average path length by 158.18 mm, improving path planning length quality by 20.47%. Furthermore, the path search time decreased from an average of 163.25 seconds for the P-RRT* algorithm to 29.16 seconds, improving the path search time by 82.14%. In summary, compared to the P-RRT* algorithm, the DPPO-CMA algorithm demonstrates stronger environmental adaptability and can plan progressively optimal task execution paths for the robotic arm in a shorter time.

Table 1: Comparison of performance indicators of algorithms in four environments

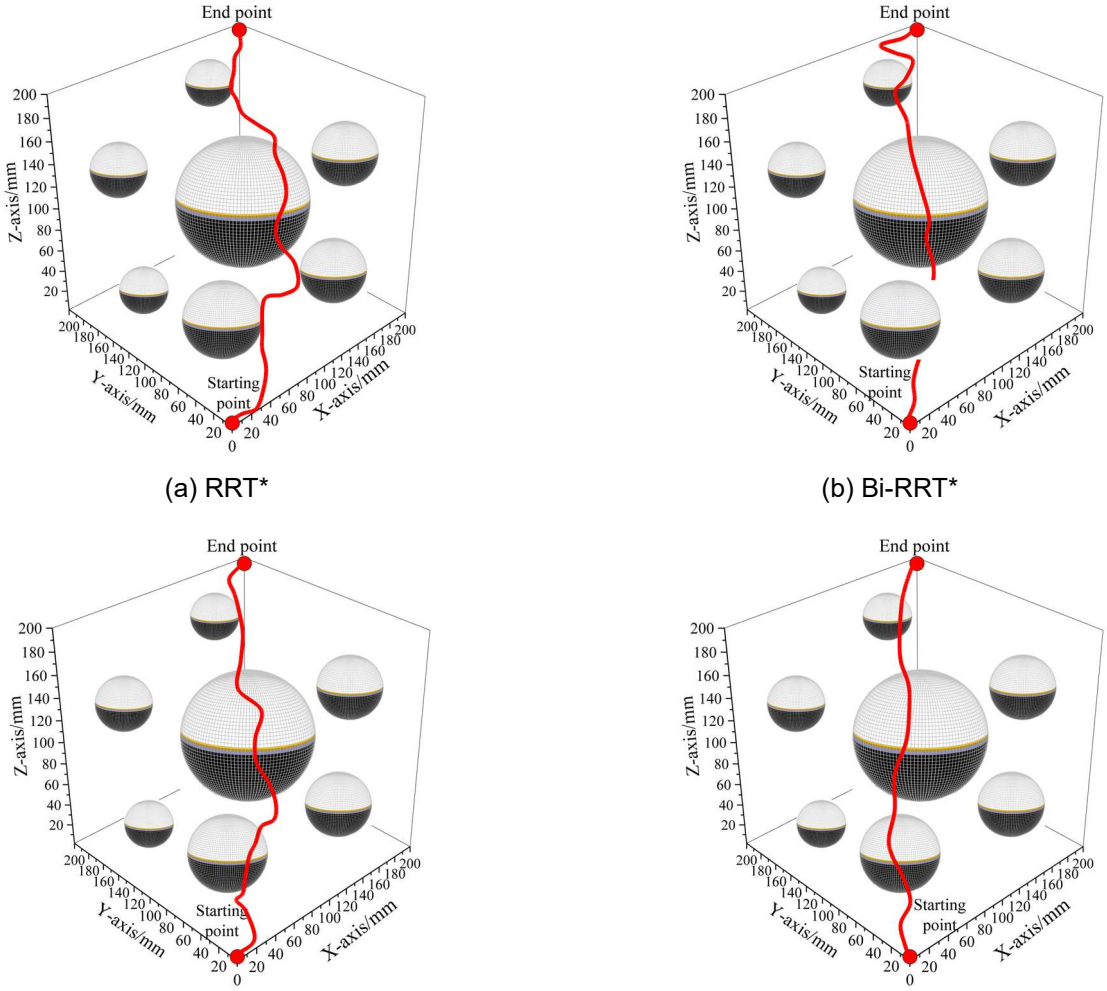
Algorithms	-	Iteration number	Path length/mm	Search time/s
RRT*	E1	94.1	336.24	6.51
	E2	132.9	385.37	8.19
	E3	149.8	412.68	9.03
	E4	4135.7	1984.36	753.19
Bi-RRT*	E1	82.6	376.52	4.03
	E2	90.1	412.85	5.15
	E3	98.4	454.27	5.63
	E4	2374.5	2155.43	552.13
P-RRT*	E1	79.3	316.39	4.74
	E2	86.7	367.92	6.23
	E3	142.9	431.73	8.16
	E4	3547.3	1842.98	633.87
DPPO-CMA	E1	42.1	263.41	0.92
	E2	43.9	273.58	1.53
	E3	62.7	365.37	2.06
	E4	1989.5	1423.95	112.13

IV. A. 2) Comparison of task execution path planning

To visually demonstrate the advantages of the DPPO-CMA algorithm in path planning for robotic arm tasks, a data comparison was conducted between the DPPO-CMA algorithm and the RRT*, Bi-RRT*, and P-RRT* algorithms in a predefined sampling space with multiple spherical obstacles. The maximum number of experimental iterations was set to 2000, and the target bias threshold was set to 0.5. The map range is set to [200,200], the starting point coordinates are [10,10,0], the target point is [180,180,180], the step size and threshold are 6 and 12, respectively, and each experiment is executed 20 times. The search path results are shown in Figure 4, where Figures 4(a) to (d) represent the path planning results of each algorithm. The distances between the nodes obtained during the iteration process and the target point for the four algorithms are shown in Figure 5.

In the path planning for the robotic arm task, it can be observed that the RRT* algorithm exhibits randomness in its path planning sampling, with the search path dominated by predefined sampling spaces, resulting in low search efficiency. The Bi-RRT* algorithm generates random trees at the start and end points for bidirectional search. Although the search path exhibits divergence, the final path shows high path cost and high path curvature. The P-RRT* algorithm partially addresses the sampling blindness of the RRT* algorithm, causing the search path to be biased toward the target point. However, the final path of this algorithm also exhibits path non-smoothness. The DPPO-CMA algorithm in this paper builds upon the DPPO algorithm by introducing the CMA-ES mechanism to

adaptively adjust the step size coefficient and accelerate the search. This reduces the number of path iterations required to converge to a local minimum from 172 in the P-RRT* algorithm to approximately 86, resulting in faster path generation.



(a) RRT* (b) Bi-RRT*

(c) P-RRT* (d) DPPO-CMA

Figure 4: The path planning results of various algorithms

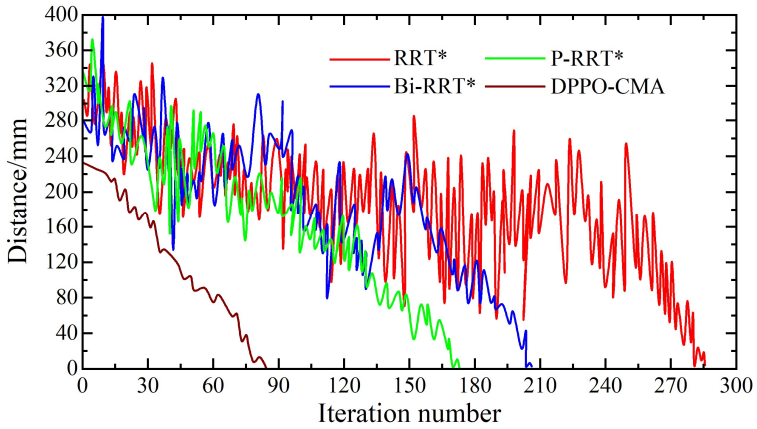


Figure 5: The relationship between the number of iterations and distance

IV. B. Effectiveness of the task execution path optimization algorithm

IV. B. 1) Dynamic Response and Planning Stability

To further validate the effectiveness of the DPPO-CMA algorithm designed in this paper, we analyze the reward values obtained during the training cycles. During the algorithm iteration process, we statistically analyze the reward values obtained in each cycle, with the specific results shown in Figure 6. The horizontal axis and vertical axis represent the number of cycles in a single training session and the reward values obtained in each cycle, respectively. A small reward value indicates that the algorithm made an incorrect path planning decision in that cycle, while a large reward value indicates that the algorithm made a correct path planning decision in that cycle. Since the DPPO-CMA algorithm uses offline training, it can effectively shorten the data accumulation phase and quickly enter the learning phase after training begins. As shown in the figure, as training progresses, the reward value obtained in each cycle gradually increases, indicating that the neural network correctly modifies its parameters through interaction with the environment, gradually making correct control decisions, and ultimately stabilizing the reward value. This indicates that the neural network's parameters have converged, achieving stable control performance. During the training process, reward values may fluctuate. This is primarily due to the DPPO algorithm generating outputs based on the probabilities of different actions after receiving state variables, resulting in a small probability of outputting incorrect actions, thereby causing fluctuations in reward values. Since the DPPO algorithm uses a neural network to generate probabilities for different control decisions, it may not always select the optimal decision within a single cycle. Therefore, the curve is not smooth and may exhibit step-like changes in reward values.

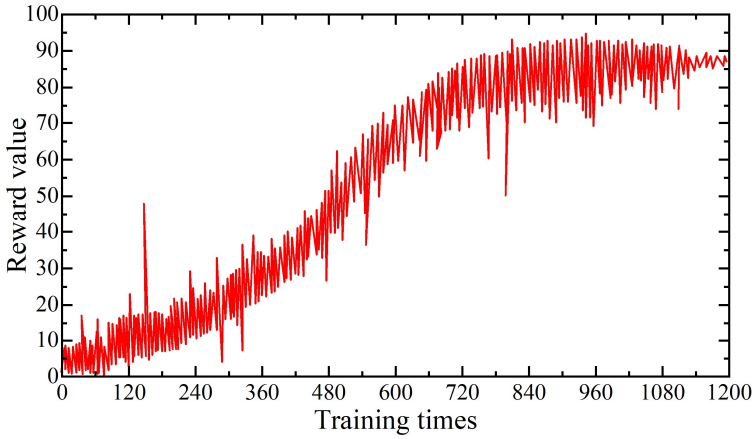


Figure 6: Dynamic response of sudden plus load

Additionally, after training was completed, the path planning stability of the DDPG algorithm and the DPPO-CMA algorithm proposed in this paper was compared when the neural network parameters were stable. The relative distance between the robotic arm gripper and a specific position below the object was measured for 50 path optimization commands, and the comparison results are shown in Figure 7. As shown in the figure, the relative distance between the path of the robotic arm task execution planning based on the DPPO-CMA algorithm and a specific position below the object has a smaller fluctuation range, indicating that after training, the algorithm in this paper can achieve more stable control performance. Due to the refinement of the reward function, the algorithm effectively suppresses the jitter phenomenon in control, achieving precise alignment between the robotic arm and the object.

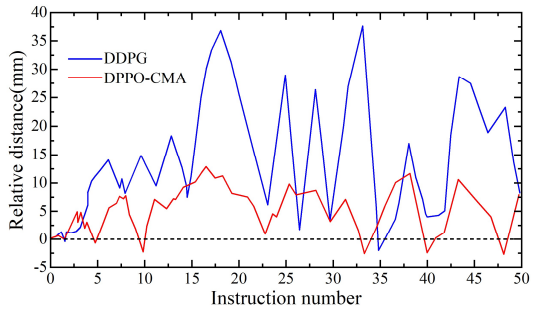


Figure 7: Task execution path planning stability

IV. B. 2) Positioning Accuracy and Loss Function Value

(1) Positioning accuracy of the robotic arm's task execution path

Based on the analysis of simulation results, it can be seen that the motion path of the cutting claw at the end of the robotic arm can be iteratively planned as the number of reinforcement training sessions increases. To verify the reliability of the motion path, the effectiveness of the DPPO-CMA algorithm proposed in this paper can be judged by analyzing the relationship between the distance changes between the cutting point at the end of the robotic arm and the object below. Figure 8 shows the curve relationship of positional error changes.

This paper primarily compares and analyzes the changes in position error during training sessions ranging from 100 to 500 times. As the number of training sessions increases, the rate of change in position error accelerates. At 100 training sessions, the change in prediction time is relatively slow between 1 and 4 seconds, but accelerates between 4 and 10 seconds. However, the stable prediction time at 100 training iterations is 10.2 seconds, while the time for the trend to stabilize at 500 training iterations is approximately 9.8 seconds. The training results show that the positional accuracy error of the cutting point stabilizes within the range of 4×10^{-3} cm at 500 training iterations.

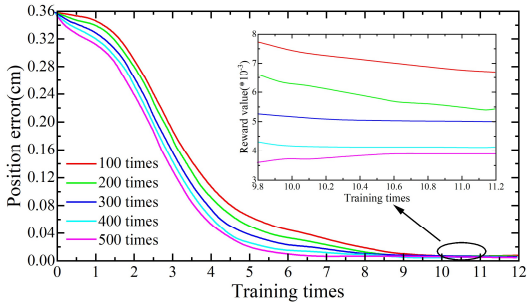


Figure 8: The relationship of the curve of position error variation

In addition, during the dynamic change process, this paper analyzes the errors in relative position, relative attitude, relative linear velocity, and relative angular velocity. The study found that the relative velocity error exhibits three stages as the velocity gain matrix is dynamically adjusted: smooth startup, rapid approach, and near-capture stages. During the smooth startup stage, the error increases smoothly from the initial value. In the rapid approach stage, it maintains a large value, enabling the robotic arm to approach the target at a faster speed. In the near-capture stage, it rapidly converges to zero, avoiding excessive residual velocity. The relative pose error is continuously smooth and converges synchronously with the relative velocity error in the near-capture stage. At the end of the capture, all relative errors converge to the specified precision, meeting the capture task requirements.

(2) Changes in loss function values

By analyzing the convergence and volatility of the loss function values, the effectiveness of the algorithm proposed in this paper can be assessed. Figure 9 shows the comparison results of the loss functions. As shown in the figure, the loss function of the DPPO-CMA algorithm designed in this paper exhibits significant volatility before 150 training iterations, but the volatility gradually decreases between 150 and 200 iterations and approaches stability after 250 iterations. The volatility of the traditional DDPG algorithm does not stabilize before 200 iterations, but it gradually converges and stabilizes after 200 iterations. Through comparison, it can also be seen that the DPPO-CMA algorithm designed in this paper converges faster, making it more feasible for application in path planning for robotic arm tasks.

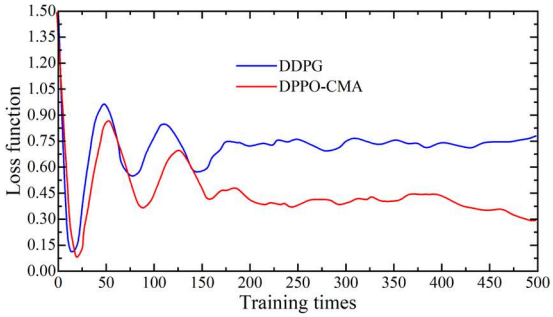


Figure 9: The comparison result of the loss function

V. Conclusion

The paper constructs a path planning model for robotic arm task execution under dynamic spatial constraints based on the kinematic model of the robotic arm and Cartesian space trajectories, and designs a path optimization algorithm for robotic arm task execution based on the adaptive DPPO-CMA algorithm. From the simulation results, the DPPO-CMA algorithm designed in this paper demonstrates strong adaptability, with its task execution path planning results aligning with the kinematic requirements of the robotic arm. Additionally, the task execution planning results exhibit high stability and significant reward values, resulting in lower error margins in robotic arm motion and object grasping outcomes. This better meets the precision requirements for task execution path planning under dynamic Cartesian space constraints.

Funding

This work was supported by State Grid Shanxi Electric Power Company Marketing Service Center (No: 52051L250008).

References

[1] Lee, H. W. (2020). The study of mechanical arm and intelligent robot. IEEE Access, 8, 119624-119634.
[2] Chen, J. I. Z., & Chang, J. T. (2020). Applying a 6-axis mechanical arm combine with computer vision to the research of object recognition in plane inspection. Journal of Artificial Intelligence, 2(02), 77-99.
[3] Leal-Naranjo, J. A., Miguel, C. R. T. S., Ceccarelli, M., & Rostro-Gonzalez, H. (2018). Mechanical Design and Assessment of a Low-Cost 7-DOF Prosthetic Arm for Shoulder Disarticulation. Applied Bionics and Biomechanics, 2018(1), 4357602.
[4] Shi, T. W., Chen, K. J., Ren, L., & Cui, W. H. (2023). Brain computer interface based on motor imagery for mechanical arm grasp control. Information Technology and Control, 52(2), 358-366.
[5] Sundaralingam, B., & Hermans, T. (2017). Relaxed-rigidity constraints: In-grasp manipulation using purely kinematic trajectory optimization. planning, 6(7), 10-15607.
[6] Lei, X., & Wu, Y. (2022). Vibration and trajectory tracking control of engineering mechanical arm based on neural network. Computational Intelligence and Neuroscience, 2022(1), 4461546.
[7] Ma, H. (2021). Application of an improved ant colony algorithm in robot path planning and mechanical arm management. International Journal of Mechatronics and Applied Mechanics, (10), 196-203.
[8] Rawat, D., Gupta, M. K., & Sharma, A. (2023). Intelligent control of robotic manipulators: a comprehensive review. Spatial Information Research, 31(3), 345-357.
[9] Baressi Šegota, S., Anđelić, N., Lorencin, I., Saga, M., & Car, Z. (2020). Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. International journal of advanced robotic systems, 17(2), 1729881420908076.
[10] Sandakalum, T., & Ang Jr, M. H. (2022). Motion planning for mobile manipulators—a systematic review. Machines, 10(2), 97.
[11] Elsisi, M., Zaini, H. G., Mahmoud, K., Bergies, S., & Ghoneim, S. S. (2021). Improvement of trajectory tracking by robot manipulator based on a new co-operative optimization algorithm. Mathematics, 9(24), 3231.
[12] Shen, H., Xie, W. F., Tang, J., & Zhou, T. (2023). Adaptive manipulability-based path planning strategy for industrial robot manipulators. IEEE/ASME Transactions on Mechatronics, 28(3), 1742-1753.
[13] Wei, K., & Ren, B. (2018). A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. Sensors, 18(2), 571.
[14] Jin, L., Li, S., La, H. M., & Luo, X. (2017). Manipulability optimization of redundant manipulators using dynamic neural networks. IEEE Transactions on Industrial Electronics, 64(6), 4710-4720.
[15] Ram, R. V., Pathak, P. M., & Junco, S. J. (2019). Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling. Mechanism and Machine Theory, 131, 385-405.
[16] Reiter, A., Müller, A., & Gattringer, H. (2018). On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. IEEE Transactions on Industrial Informatics, 14(4), 1681-1690.
[17] Muhammed, M. L., Humaidi, A. J., & Flaieh, E. H. (2023). A comparison study and real-time implementation of path planning of two arm planar manipulator based on graph search algorithms in obstacle environment. ICIC Express Lett, 17(1), 61-72.
[18] Chen, D., Li, S., Wu, Q., & Luo, X. (2019). New disturbance rejection constraint for redundant robot manipulators: An optimization perspective. IEEE Transactions on Industrial Informatics, 16(4), 2221-2232.
[19] Yu, X., Dong, M., & Yin, W. (2022). Time-optimal trajectory planning of manipulator with simultaneously searching the optimal path. Computer Communications, 181, 446-453.
[20] Wang, L., Wu, Q., Lin, F., Li, S., & Chen, D. (2019). A new trajectory-planning beetle swarm optimization algorithm for trajectory planning of robot manipulators. IEEE access, 7, 154331-154345.
[21] Cao, X., Yan, H., Huang, Z., Ai, S., Xu, Y., Fu, R., & Zou, X. (2021). A multi-objective particle swarm optimization for trajectory planning of fruit picking manipulator. Agronomy, 11(11), 2286.
[22] Kucuk, S. (2017). Optimal trajectory generation algorithm for serial and parallel manipulators. Robotics and Computer-Integrated Manufacturing, 48, 219-232.
[23] Zhang, Z., Lin, Y., Li, S., Li, Y., Yu, Z., & Luo, Y. (2017). Tricriteria optimization-coordination motion of dual-redundant-robot manipulators for complex path planning. IEEE Transactions on Control Systems Technology, 26(4), 1345-1357.
[24] Chen, G., Luo, N., Liu, D., Zhao, Z., & Liang, C. (2021). Path planning for manipulators based on an improved probabilistic roadmap method. Robotics and Computer-Integrated Manufacturing, 72, 102196.
[25] M. Bailova,M. Beres,P. Beremlijski,J. Koziorek,M. Prauzek & J. Konecny. (2025). Method for estimating the pose of a robotic arm using a camera and calibration pattern. Ain Shams Engineering Journal,16(9),103525-103525.

[26] Yongqi Shuai. (2025). Optimizing forward kinematics of a 6-DOF robotic arm for precision and efficiency. Journal of Physics: Conference Series,3019(1),012020-012020.
[27] Serhat Obuz,Enver Tatlicioglu & Erkan Zergeroglu. (2024). Adaptive Cartesian space control of robotic manipulators: A concurrent learning based approach. Journal of the Franklin Institute,361(6),106701-.
[28] Yu Cao,Guoxing Wen,Baoshuo Feng & Bin Li. (2025). Optimized consensus control of multi-manipulator system having actuator fault using reinforcement learning approximation strategy. Information Sciences,712,122141-122141.