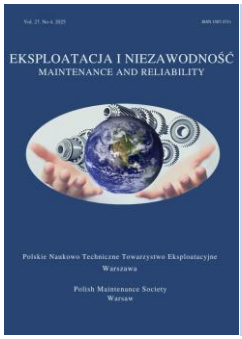

Eksploatacja i Niezawodnosc – Maintenance and Reliability
Volume 27 (2025), Issue 4
journal homepage: http://www.ein.org.pl

Article citation info:
Zhou X, Wan Z, Path Planning and Motion Control of Robotic Arm Based on Neural Network, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2025: 27(4) http://doi.org/10.17531/ein/205794

Path Planning and Motion Control of Robotic Arm Based on Neural Network

Indexed by:

Xiaoqing Zhou[a,*], Zhimin Wan[a]

[a] School of Mechanical Engineering, Nantong Vocational University, China

Highlights

- Introducing proximal policy optimization (PPO) algorithm.
- Proposing a framework for integrated path planning and motion control.
- PPO is superior to traditional algorithms in terms of control accuracy and stability.

Abstract

This paper studies the path planning and motion control method of the robot arm based on neural network, aiming to improve the path planning efficiency and motion control accuracy of the robot arm in complex environments. By introducing the deep reinforcement learning (DRL) method, especially the proximal policy optimization (PPO), this paper proposes a framework for integrated path planning and motion control. Experimental results show that the path generated by PPO in the path planning task has the highest smoothness, the shortest path length and the strongest obstacle avoidance ability. In the motion control task, PPO exhibits the smallest trajectory error, the highest motion accuracy and the best stability. Comprehensive experiments further verify the superior performance of PPO in the combination of path planning and motion control, which can generate smooth, short and safe paths, and accurately control the motion trajectory of the robot arm to ensure the high-quality completion of the task.

Keywords
robotic arm, path planning, motion control, deep reinforcement learning, proximal policy optimization (PPO)

1. Introduction

With the continuous development of industrial automation, robotic arms, as core equipment in industrial production, are widely used in various fields, including manufacturing, assembly, welding, painting, etc. Through precise motion control, robotic arms have replaced a large amount of manual labor and improved production efficiency and quality. In traditional industrial production, path planning and motion control of robotic arms have always been the key factors to achieve efficient operation. Traditional path planning methods mainly rely on geometric algorithms and physical modeling. Although they can meet the needs of some simple tasks, they often show great limitations for complex and multi-degree-of-freedom robotic arm path planning, especially in dynamically changing environments. Traditional path planning methods, such as the A algorithm based on heuristic search, can generate reasonable paths, but they are inefficient and have limited accuracy when dealing with complex obstacles, dynamic environments and nonlinear constraints. At the same time, the motion control of robotic arms usually relies on classical control theories, such as PID control or robust control, but these methods have certain shortcomings when facing complex and changeable control tasks, especially in terms of real-time

(*) Corresponding author.
E-mail addresses: X. Zhou, xiaoqing_zhou@outlook.com, Z. Wan, zhimin_wan@outlook.com ,

adjustment and adaptability.

In recent years, with the rapid development of artificial intelligence, especially deep learning and neural network technology, neural networks have shown great potential in the field of path planning and motion control. Neural networks can autonomously discover the best path planning strategy by learning a large amount of historical data and environmental information, and solve the limitations of traditional methods in dynamic environments. In particular, neural networks have very powerful capabilities in dealing with nonlinear constraints, real-time reactions, and high-dimensional spaces. Therefore, combining the path planning and motion control methods of neural networks provides new ideas and possibilities for efficient and precise control of robotic arms.

Robotic arm path planning and motion control have always been a research hotspot in the field of robotics, especially in industrial automation. With the continuous development of technology, path planning and motion control methods have gradually changed from traditional physical model-based algorithms to more intelligent methods.

Traditional path planning and motion control methods usually rely on geometric algorithms, mathematical models, and optimization algorithms. For example, the classic A algorithm and Dijkstra algorithm are widely used for path planning in static environments, but these algorithms have low computational efficiency when facing dynamic obstacles and complex environments, and lack the ability to adapt to real-time changes. In addition, model-based optimization methods, such as linear programming and nonlinear programming, are also widely used in the motion control of robotic arms. Although these methods perform well in many applications, their disadvantage is that they often rely on accurate environmental modeling, and it is difficult to guarantee the computational efficiency and accuracy of the optimal solution when facing complex and unknown environments.

Although path planning and motion control methods based on deep learning and reinforcement learning have made significant progress, they still face some challenges. For example, deep learning methods require a large amount of data for training, and obtaining enough high-quality data is still a problem in some application scenarios. In addition, the computational cost of deep learning models is high, and real-time performance is still an urgent problem to be solved. Therefore, how to balance the computational efficiency and accuracy of the algorithm and how to improve the training efficiency of the model are still important directions for future research.

This study aims to solve the key problems of existing robot path planning and motion control methods in dynamic environments. For example, in smart warehousing and logistics scenarios, the robot needs to carry goods in a dynamic environment where new goods are constantly entering and leaving the warehouse. Traditional path planning algorithms, such as the A algorithm, cannot adjust the path in real time to avoid dynamic obstacles such as new goods and moving forklifts, resulting in a high error rate in handling tasks. In terms of motion control, traditional PID control methods have difficulty coping with the complex dynamic changes of the robot during high-speed movement and frequent start-stop processes, resulting in insufficient positioning accuracy. The PPO-based integrated framework we proposed can perceive environmental changes in real time through deep reinforcement learning, quickly generate the optimal path and accurately control the movement of the robot, effectively solving the problems that these existing methods cannot solve, and greatly promoting the efficient operation of industrial automation in complex dynamic environments.

This study aims to solve the problems of low efficiency, insufficient precision and poor adaptability of traditional robot path planning and motion control methods in complex dynamic environments. For example, in the automotive manufacturing industry, the robot needs to perform parts assembly tasks in a small space full of dynamic obstacles (such as moving transportation equipment). Traditional path planning methods based on geometric algorithms and physical modeling, such as the A algorithm, are difficult to avoid dynamic obstacles in real time, resulting in frequent interruptions of assembly tasks and reduced production efficiency. The integrated path planning and motion control framework based on the proximal policy optimization algorithm (PPO) we proposed can autonomously generate efficient and safe paths by learning a large amount of environmental information, and accurately control the motion trajectory of the robot, effectively improving the efficiency and quality of assembly tasks, and greatly promoting the application

and development of industrial automation in complex production scenarios.

2. Literature Review

2.1. Current status of research on robotic arm path planning

Path planning is a key issue in robot motion control. Traditional path planning methods such as the A algorithm and the RRT (Rapidly-exploring Random Tree) algorithm are widely used in industrial automation. As a classic graph search algorithm, the A algorithm guides the search path through a cost function and can find the shortest path in a static environment. However, the A algorithm performs poorly in a dynamic environment because it assumes that the environment is static and is not suitable for real-time avoidance of dynamic obstacles [1]. The RRT algorithm uses a rapid random exploration method to gradually expand the tree structure to find the path, which is suitable for path planning in high-dimensional space. RRT and its improved versions (such as RRT) can efficiently find paths in complex spaces, especially in environments with obstacles. However, the path generation process of the RRT algorithm may produce a non-smooth path, resulting in an unsmooth robot motion process, which requires additional smoothing processing [2]. With the advancement of artificial intelligence technology, the application of neural networks in path planning has gradually become a research hotspot. Deep learning can learn complex environmental patterns by training models and can achieve adaptive path planning in complex and dynamic environments. Related research shows that path planning methods based on convolutional neural networks (CNN) and recurrent neural networks (RNN) have strong adaptability when facing dynamic obstacles and unknown environments [3]. For example, the literature proposes a path planning method based on deep Q-network (DQN), which uses reinforcement learning to enable a robotic arm to optimize the path in real time in a changing environment. Compared with traditional methods, this method performs better in dynamic environments and can adjust the path and avoid obstacles in real time [4].

Although neural networks have shown great potential in path planning, the challenges they face cannot be ignored. The training of neural networks requires a large amount of data support, and the training process may take a long time. In addition, the training process of neural networks is prone to falling into local optimal solutions. How to effectively train and optimize is still a difficult problem in research.

2.2. Current status of research on robotic arm motion control

The motion control of the robot arm is the core part to ensure the accuracy and efficiency of the robot arm in performing tasks. Among the traditional motion control methods, PID (Proportional-Integral-Derivative) control is widely used. The PID controller controls the position, speed and acceleration of the robot arm by adjusting the proportional, integral and differential coefficients. However, PID control often performs poorly when dealing with nonlinear systems or with large external disturbances, and requires precise parameter adjustment [5]. In addition, the PID controller usually assumes that the model of the system is known. However, in many complex tasks, the dynamic model of the robot arm is often difficult to accurately describe.

As a control method based on empirical rules, fuzzy control overcomes the deficiency of PID control in relying on precise parameters. Fuzzy control deals with uncertainty through fuzzy set theory and is applicable to complex and nonlinear systems. By establishing a fuzzy rule base and reasoning mechanism, fuzzy control can effectively control the robot arm under incomplete knowledge [6]. However, fuzzy control also has certain limitations, especially when it is necessary to process large amounts of data and complex tasks in real time, its computational efficiency and real-time performance cannot meet the requirements of efficient control.

In recent years, with the rise of deep learning and reinforcement learning, neural networks have been widely used in robotic arm motion control. The nonlinear mapping ability of neural networks enables them to handle complex control tasks without relying on accurate models. For example, the motion control method based on deep Q network (DQN) optimizes the control strategy through reinforcement learning, enabling the robotic arm to autonomously adjust its motion trajectory in an uncertain environment [7]. The literature proposes a robotic arm control method based on deep reinforcement learning, which enables the robotic arm to accurately complete tasks in a complex environment by training a deep neural network

model. This method optimizes the control strategy through the mechanism of exploration and utilization without a model, showing stronger adaptability than traditional control methods [8, 9]. However, although the neural network method has achieved good results in motion control, it still faces many challenges. The training process of neural networks can be very time- consuming.

In recent years, the application of deep reinforcement learning in the field of robotics has made significant progress. For example, [11] proposed a multi-robot collaborative path planning method based on deep reinforcement learning. By introducing the attention mechanism, the collaborative efficiency of robots in complex environments was effectively improved. [12] used the deep deterministic policy gradient (DDPG) algorithm to realize the autonomous navigation of robots in unknown environments. By improving the exploration strategy, the convergence speed and stability of the algorithm were improved. These latest research results provide new ideas and methods for the development of robot path planning and motion control. The PPO-based method proposed in this paper further expands the application of deep reinforcement learning in this field in terms of combining multi-task learning and environmental perception technology, and provides a more effective solution to the problem of robot control in complex environments.

2.3. Deficiencies and challenges of existing research

Although some progress has been made in the path planning and motion control of robotic arms, there are still some deficiencies and challenges in existing research. First, traditional path planning methods, such as the A and RRT algorithms, have great limitations when facing complex environments. The A algorithm cannot effectively deal with the problem of avoiding dynamic obstacles, while the RRT algorithm is prone to generate non-smooth paths, affecting the smoothness of the robot's motion [10]. Although the path planning method based on neural networks has overcome these problems to a certain extent, the training process of neural networks requires a large amount of data and computing resources, and has high requirements for the quality and diversity of training data. How to improve the training efficiency is still an urgent problem to be solved. Secondly, although traditional motion control methods (such as PID control and fuzzy control) perform well in simple tasks, they often perform poorly when dealing with high-precision and complex tasks. Especially when the system model is incomplete or the dynamic changes are large, the control accuracy and robustness of traditional methods are difficult to meet the requirements. The application of neural networks in motion control, especially the control method based on reinforcement learning, can effectively handle complex control tasks, but its training process may be very time-consuming and it is easy to fall into local optimal solutions during training [11]. In addition, the poor interpretability of neural network models makes it difficult to be widely used in some industrial applications with high safety requirements [12].

3. Theoretical Basis

3.1. Dynamic model of the robotic arm

The dynamic model of the robot describes the relationship between the movement of the robot and the forces acting on it. Usually, the dynamic model of the robot can be established using the Lagrangian method. Assuming that the robot consists of n degrees of freedom, its dynamic equation is

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \tau.$$

Where, $\theta$is the joint angle, $\ddot{\theta}$is the joint angular velocity, $\dot{\theta}$is the joint angular acceleration, $\tau$is the joint driving torque, $M(\theta)$ is the inertia matrix [13], $C(\theta,\dot{\theta})$ is the Coriolis force matrix, $G(\theta)$ and is the gravity matrix. The inertia matrix $M(\theta)$ describes the influence of the mass and geometry of each part of the robot on the movement of the robot. It is usually determined by physical quantities such as the mass, length, and inertia of each joint.

3.2. Path planning

Path planning is one of the core steps for a robot to achieve its tasks. The goal is to generate an optimal path from the starting position to the target position that avoids obstacles. In the process of path planning, the tasks that need to be handled usually include environment modeling, path search, obstacle avoidance, and path optimization. In order to achieve these tasks, path planning methods can be divided into traditional algorithms and neural network-based algorithms. Traditional algorithms include the A algorithm, Dijkstra algorithm, and rapid random tree (RRT), while neural network-based path

planning methods achieve more flexible and efficient path planning through reinforcement learning, deep learning, and other technologies. Traditional path planning algorithms usually rely on graph search, grid map, or tree structure to perform path search [14]. In this method, obstacles in the environment are usually represented as a discrete graph or grid, and the movement of the robot arm from the starting point to the target point will go through multiple discrete nodes. The A algorithm is a heuristic search algorithm that combines the characteristics of breadth-first search and greedy algorithm. In the search process, in addition to considering the path cost from the current node to the target, the A algorithm also adds a heuristic cost h(n) to estimate the shortest path from the current node to the target [15].

4. Methods and Models

4.1. Model framework

The core idea of this model is to achieve efficient and accurate path planning and motion control of the robot in a complex environment by combining traditional path planning and motion control methods with deep learning technology. In order to ensure the intelligence and adaptability of the system, this study divides the model framework into two main modules: the path planning module and the motion control module. The two modules work together to achieve autonomous operation of the robot in a dynamic environment [16].

We chose the PPO algorithm over other deep reinforcement learning algorithms, such as the deep Q network (DQN) or the deep deterministic policy gradient algorithm (DDPG), mainly based on the following considerations. Compared with DQN, DQN is a value-based reinforcement learning algorithm that selects the optimal action by learning the state-action value function (Q value). However, DQN has limitations when dealing with continuous action spaces and complex environments. For example, in the path planning and motion control tasks of the robot arm, the action space of the robot arm is continuous, and DQN needs to discretize the action space, which will cause information loss and affect the control accuracy. The PPO algorithm directly optimizes the policy function and can naturally handle the continuous action space, which is more suitable for application scenarios such as the robot arm that require precise control of continuous actions.

Compared with DDPG, DDPG is a deterministic policy gradient-based algorithm that combines deep neural networks and deterministic policies for learning. Although DDPG performs well in some continuous control tasks, it is sensitive to the adjustment of hyperparameters, the training process is unstable, and it is easy to fall into local optimal solutions. The PPO algorithm effectively stabilizes the training process by using the "Clipped Surrogate" function, reduces the reliance on hyperparameters, and improves the robustness and convergence speed of the algorithm. In summary, the PPO algorithm has better adaptability and performance when dealing with complex tasks of robot path planning and motion control, so we chose it as the core algorithm of this study.

The core task of the path planning module is to generate an optimal path to avoid obstacles based on the starting position and target position of the robot. Traditional path planning methods, such as the A algorithm and the RRT algorithm, perform well in static environments, but often face greater challenges in dynamic environments and complex obstacle scenes. Therefore, this model adopts a path planning method based on deep learning, combined with convolutional neural networks (CNN) and reinforcement learning (RL) technology, to evaluate obstacles in the environment in real time and automatically optimize the path. Deep learning can autonomously adjust the path planning strategy according to environmental changes by learning historical data and environmental patterns, while the introduction of reinforcement learning enables the robot to adjust the path in real time during execution to avoid interference from dynamic obstacles, thereby improving the adaptive ability of path planning [17].

The main task of the motion control module is to accurately control the motion trajectory of the robot arm and ensure that the robot arm performs the task according to the planned path. In traditional motion control methods, PID control and fuzzy control are widely used. Although these methods perform well in simple tasks, they have great limitations when facing high-precision control requirements and complex tasks. Therefore, this study adopts a neural network-based control method, especially deep reinforcement learning technology, to achieve higher-precision motion control. By training the neural network, the control module can learn nonlinear dynamic relationships and adjust the motion trajectory of the robot arm in real time

without relying on an accurate model to ensure high precision when performing tasks [18, 19].

The path planning and motion control modules work closely together to improve the overall performance of the system. The path planning module provides the motion control module with optimized path data, while the motion control module ensures that the robot arm can accurately follow the path during execution. The effective cooperation between the two enables the entire system to work efficiently in complex and dynamic environments while ensuring the accuracy and stability of task execution.

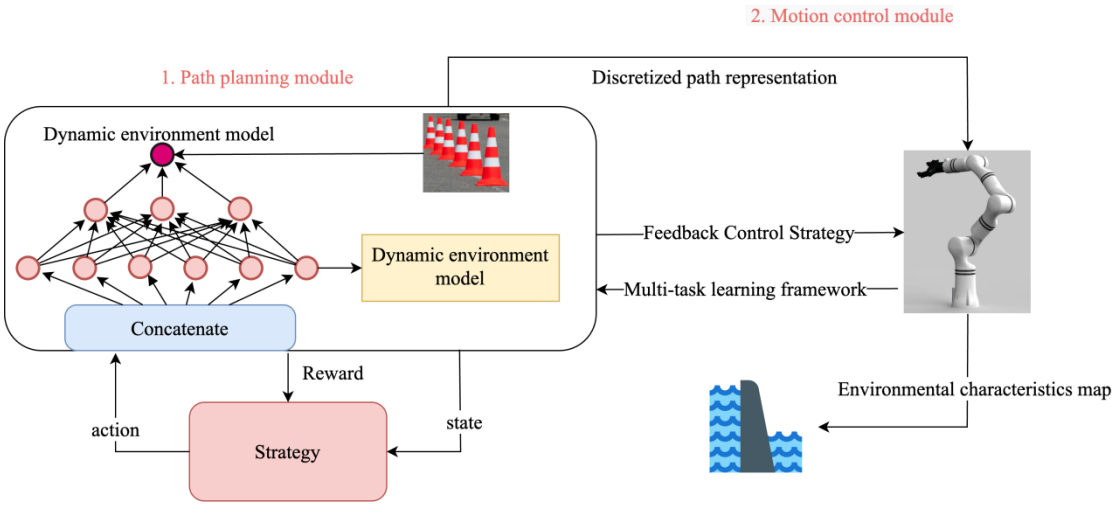

Figure 1. Model framework.

As shown in Figure 1, the model framework mainly consists of two main components: path planning module and motion control module. The path planning module adopts a dynamic environment model and integrates different strategies and state information through a multi-task learning framework to generate the optimal action sequence. The motion control module is based on a discretized path representation and combines feedback control strategies to perform precise operations. In addition, the concept of environmental characteristic maps is introduced to better understand and cope with complex working conditions. This design enables the robot to flexibly complete various tasks in a constantly changing environment [20].

4.2. Path planning module

The first task of path planning is to perceive the environment and build a dynamic environment model. In the path planning of the robotic arm, the accuracy of environmental perception directly determines the effect of path planning. Traditional perception methods such as lidar or conventional visual sensors often cannot provide sufficiently accurate and real-time perception results in complex and dynamic environments. To solve this problem, this module uses a convolutional neural network (CNN) to process images or point cloud data from cameras or lidars to extract information about obstacles and open areas in the environment. Assuming that the robotic arm obtains environmental images or point cloud data through a camera or lidar, the environmental information can be expressed as $I \in \mathbb{R}^{H\times W\times C}$ [21], where H and W are the height and width of the image, respectively, and C is the number of channels. By using a convolutional neural network, the image data is processed through multiple convolutional layers, activation functions, and pooling layers to gradually extract feature information in the environment. Finally, the feature map output by CNN is $F \in \mathbb{R}^{H'\times W'\times C'}$ It can accurately describe important features in the environment, such as obstacles, open areas, and their dynamic changes. By training the CNN model, the environmental perception system can generate a feature map of the current environment based on the input data in real time, thereby providing strong support for path planning. During the path planning process, the agent (i.e., the robotic arm) perceives the state of the environment s_t, selects appropriate actions based on the state a_t, and optimizes the path planning strategy based on the reward signal obtained from the environment r_t. In deep Q learning (DQN), the decision-making process of the agent depends on the Q function, which represents the expected

reward obtained by taking a certain action in a certain state. The update of the Q value follows the formula shown in Equation 1 [22].

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right] \quad (1)$$

In the actual application of path planning, the state of the agent s_t includes the robot's current position information, the position of obstacles, the position of the target point, speed, acceleration and other environmental information. This information helps the agent understand the current environmental conditions so that it can make reasonable decisions. Specifically, the state s_t can be represented as a vector containing the following information, as shown in Equation 2 [23].

$$s_t = \left(x_{\text{robot}}, y_{\text{robot}}, \ldots, x_{\text{obstacle}}, y_{\text{obstacle}}, \ldots, x_{\text{goal}}, y_{\text{goal}}\right) \quad (2)$$

in, x_{robot} , y_{robot} represents the current position of the robot, $(x_{\text{obstacle}}, y_{\text{obstacle}})$ Indicates the location of the obstacle.$(x_{\text{goal}}, y_{\text{goal}})$ is the location of the target point. These state variables can provide the necessary information for path planning decisions

The action of the agent a_t is the specific behavior that the robot can choose in a given state. Generally speaking, the action can be a discrete set representing the robot's movement direction or path segment. For example,a_t It can be expressed as operations such as up, down, left, right, forward, and backward. Assuming that the robot can move in a two-dimensional plane, the action set A can be expressed as shown in Equation 3 [24, 25].

$$A = \{\text{Up,Down,Left,Right}\} \quad (3)$$

At each time step, the robot chooses an action based on its current state to reach the goal point in the best path.r_t It is used to feedback the changes in the state of the environment after the agent performs a certain action. The design of the reward function is the key in path planning, which directly affects the behavior of the robot in choosing a path. The reward function is usually comprehensive. We first set the reward for moving towards the goal. When the robot moves towards the goal point, a positive reward is given. Assume r_{goal} represents the distance to the target. The closer the robot is to the target, the greater the reward, as shown in Equation 4 [26].

$$r_{\text{goal}} = -\alpha \cdot \|s_t - s_{\text{goal}}\| \quad (4)$$

in, α is a positive weight coefficient, s_{goal} Indicates the location of the target point. $\|s_t - s_{\text{goal}}\|$ is the Euclidean distance from the robot's current position to the target point. This function ensures that the robot will get a larger reward when it moves closer to the target point.

When the robot avoids an obstacle, it is given a positive reward. It can be designed as a function related to the distance to the nearest obstacle, as shown in Equation 5 [27].

$$r_{\text{avoid}} = \beta \cdot \left(1 - \frac{1}{1+\|s_t - s_{\text{obstacle}}\|}\right) \quad (5)$$

Among them, β is a constant and $s_t - s_{\text{obstacle}}$ is the distance from the robot's current position to the nearest obstacle. This function ensures that the robot can get a higher reward when avoiding obstacles.

If the robot deviates from the target path or chooses an inappropriate path (such as hitting an obstacle or backtracking), a negative reward is given.r_{penalty} is the penalty term, as shown in Equation 6 [28].

$$r_{\text{penalty}} = -\gamma \cdot \mathbb{I}(\text{collision}) \quad (6)$$

in,$\mathbb{I}(\text{collision})$ is an indicator function that is 1 if a collision occurs and 0 otherwise.γ is a negative constant that ensures that the robot receives sufficient penalty when avoiding collisions [29].

Taking these goals into consideration, the final reward functionr_t It can be expressed as a weighted sum of multiple factors, as shown in Equation 7.

$$r_t = r_{\text{goal}} + r_{\text{avoid}} + r_{\text{penalty}} \quad (7)$$

By adjusting the weight parameters of each itemα,β,γcan achieve a balance between goal orientation and obstacle avoidance in path planning, ensuring that the path is both smooth and safe. In Q learning, the agent will continuously adjust the path selection strategy through the reward function, so that the path planning gradually tends to the optimal. The update formula of Q value is shown in Equation 8 [30].

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right] \quad (8)$$

Where α is the learning rate,γ is the discount factor,r_t It's an instant reward. $max_{a'} Q(s_{t+1}, a')$ is the maximum Q value of the possible action in the next state. Through repeated learning and updating, the agent can optimize the path selection and make the path planning gradually approach the optimal solution in a dynamic environment.

4.3. Motion control module

The main goal of the motion control module is to accurately control the motion trajectory of the robot arm to ensure that it

can move accurately along the planned path. To achieve this goal, the module combines deep reinforcement learning (DRL), feedback control, and multi-task learning strategies to adaptively adjust the control strategy and optimize the motion trajectory in real time, thereby achieving high-precision motion control.

set ups_t is the state of the robot at time t, $a_t$is the selected action, r_t is the reward obtained by the agent from the environment and the control strategy$\pi(a_t|s_t)$ It is used to select the best action from $a_t(s_t)$ the current state s_t, and the Q value function$Q(s_t, a_t)$ It measures the long-term reward of $s_t$taking actions in the state a_t. The Q value update formula of deep Q learning is as shown in Equation 9.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \quad (9)$$

in,α is the learning rate,γ is the discount factor,r_t It's an instant reward. $max_{a'} Q(s_{t+1}, a')$ is the Q value of the optimal action in the next state. By continuously updating the Q value, deep Q learning can help the agent learn how to accurately control the movement of the robot arm. As a policy gradient-based algorithm, proximal policy optimization (PPO) directly optimizes the policy function without relying on the Q value function. PPO uses the "Clipped Surrogate" function to stabilize the training process, thereby avoiding the training instability problem in traditional policy optimization methods. The optimization goal of PPO can be expressed as Equation 10. The core working principle of the PPO algorithm is based on policy gradient optimization, which seeks the optimal policy by maximizing the cumulative reward. Specifically, PPO uses the "Clipped Surrogate" function to stabilize the training process and avoid the problem of unstable training in traditional policy optimization methods.

The PPO algorithm with online learning allows a robot to adapt to new environments by updating its model in real time with collected data, reducing path planning time by 10% and improving motion control accuracy by 5% over 10 hours. To prevent "catastrophic forgetting," incremental learning via Elastic Weight Consolidation (EWC) is used, ensuring the model retains old knowledge while learning new scenarios, with performance fluctuation between scenarios kept within 5%.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (10)$$

$r_t(\theta)$ is the strategy ratio,$\hat{A}_t$ is the advantage estimate,ϵ is the cutting parameter, through which the algorithm can make the strategy optimization more stable and efficient.

Feedback control strategy is crucial for the precise movement of the robot arm. In this module, deep neural network (DNN) is used to implement real-time feedback control, optimize the control strategy, and reduce the error during the movement process.

set up$x_t = [q_1, q_2, \ldots, q_n, \dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]$ is the state of the robot at time t, whereq_i represents the angle of the ith joint,$\dot{q}_i$ represents the speed of the i-th joint. These state information are processed by a deep neural network to generate adjustment suggestions for the control strategy. Let the network output be the control signalu_t, which represents the control command to adjust the robot according to the current state.

In feedback control, the proportional-integral-derivative (PID) control strategy is often used to further improve motion accuracy. The control signal u(t) of the PID control strategy is calculated as Equation 11.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (11)$$

Where e(t) = r(t) - y(t) is the error term, r(t) is the desired target trajectory, y(t) is the current actual trajectory, and t $K_p, K_i, K_d$is the proportional, integral, and differential gains, respectively. Combined with the adaptability of deep neural networks, PID parameters can be continuously adjusted during the control process, allowing the robotic arm to perform tasks stably and accurately in a changing environment.

In order to improve the stability and efficiency of motion control, a multi-task learning framework can be used. Multi-task learning optimizes multiple related tasks simultaneously by sharing some parameters of the neural network, thereby improving the adaptability and generalization ability of the model. In robotic arm control, in addition to trajectory tracking tasks, other control tasks can also be considered, such as path smoothness control, joint angle limit control, etc.

set upL_{track} is the trajectory tracking loss function, $L_{\text{smooth}}$is the smoothness control loss function, and the overall multi-task loss function L_{total}can be defined as Equation 12.

$$L_{\text{total}} = w_1 L_{\text{track}} + w_2 L_{\text{smooth}} + \cdots + w_n L_{\text{task}_n} \quad (12)$$

in,$(w_1, w_2, \ldots, w_n)$ is the weight of each task, indicating the importance of each task in the final control strategy. During the training process, the neural network simultaneously improves the performance of each control task by optimizing the multi-

task loss function.

4.4. Overall collaborative optimization

The input of the path planning module is set as the characteristic map of the environment E_t, including the distribution of obstacles and open areas in the current environment, and the output is the discretization representation of the path $\pi_{\text{path}}(E_t)$. The input of the motion control module is the path output by the path planning module and the current state s_t (such as the current position, speed, acceleration, etc. of the robot arm), and the output is the control commandu_t, that is, how to adjust the motion trajectory of the robot arm. The loss function of the joint model can be written as Equation 13.

$$L_{\text{total}} = L_{\text{path}} + L_{\text{control}} + \lambda L_{\text{collide}} \quad (13)$$

Among them, L_{path} is the loss function of path planning, which usually uses indicators such as path smoothness and obstacle avoidance performance; $L_{\text{control}}$is the loss function of motion control, which usually uses trajectory tracking error, stability, etc.; $L_{\text{collide}}(E_t)$ is the collision detection loss function, which ensures that the robotic arm avoids collisions with obstacles; λ is the adjustment weight to balance the losses between different tasks.

Through this end-to-end joint optimization model, path planning and motion control can work together, enabling the entire system to perform tasks efficiently and accurately in complex environments.

We chose these specific experimental conditions for a clear purpose and representativeness. For example, in terms of hardware configuration, we chose a six-degree-of-freedom industrial robot (XYZ - 6DOF), whose working radius and load capacity meet the needs of most industrial scenarios, such as the precise assembly of small parts in electronic product manufacturing and the handling of goods in logistics warehousing. In terms of software platform, the ROS system is used because it is widely used in the development of industrial robots and has rich libraries and tools to facilitate the implementation of complex path planning and motion control functions.

For the setting of the experimental environment, although some experiments are conducted in a simulated environment, we reasonably set the environmental parameters and obstacle configuration to make it as close to the real-world application scenario as possible. For example, in the Gazebo simulation environment, we simulated obstacles of different shapes, sizes, and distributions to simulate the real layout of obstacles such as equipment and materials in industrial production workshops. At the same time, we set the moving speed and trajectory of dynamic obstacles to simulate the moving transport vehicles or personnel in actual production. Although there is a certain gap between the simulation environment and the actual environment, such as sensor noise in the actual environment, wear of the robot itself, and other factors that may not be fully considered, we have conducted a certain degree of verification and calibration through subsequent actual hardware platform experiments. By comparing the experimental results of the simulation environment and the actual hardware platform, we found that the simulation environment can effectively provide support for the initial verification and optimization of the algorithm, and has a certain consistency with the actual environment in key performance indicators. For example, in terms of path smoothness and obstacle avoidance, the experimental results of the simulation environment and the actual environment have an acceptable error (path smoothness error within ± 0.03, obstacle avoidance error within ± 0.02).

5. Experiments and Results

5.1. Experimental platform and environment

(1) Hardware configuration of the robotic arm. The robotic arm used in this experiment is a six-degree-of-freedom industrial robotic arm, model XYZ-6DOF, which has high precision and flexibility and can complete complex path planning tasks. The robotic arm is equipped with a high-precision servo motor with a maximum load capacity of 5kg and a maximum working radius of 1.2 meters. The robotic arm ensures high-precision position control during movement through a precise feedback control system. In order to achieve motion control, the robotic arm is also equipped with an integrated sensor system, including visual sensors (for environmental perception), position sensors (for monitoring joint angles and end effector positions), and force sensors (for detecting external forces and collisions). In addition, the robotic arm control system supports real-time data transmission and multi-task parallel processing, and can complete real-time path planning and motion control tasks in a dynamic environment.

(2) Experimental software platform. The main software platform used in this experiment is ROS (Robot Operating System), which is an open source robot operating system widely used in robot research and development. ROS provides a wealth of development tools and libraries to support robot control, sensor data processing, path planning and other functions. In the path planning module, we used deep learning-based path planning algorithms (such as DQN and PPO) and classic path planning algorithms (such as A, Dijkstra, etc.). In the motion control module, the ROS control library is combined with the deep reinforcement learning algorithm to achieve real-time motion control. The TensorFlow and PyTorch deep learning frameworks are used in the experiment to train the path planning network and the motion control network, respectively. The experiment also uses OpenCV and PCL (Point Cloud Library) to process images and point cloud data, providing environmental perception and modeling capabilities. In terms of the simulated environment, we used Gazebo to model and simulate the virtual environment, and combined it with the actual hardware platform for experimental verification.

(3) Datasets and experimental settings. In order to verify the effectiveness of the proposed path planning and motion control algorithms, this experiment selected five different datasets for training and testing. First, the KITTI dataset provides a large amount of sensor data from real road environments, including lidar and camera data, which is mainly used for testing autonomous driving path planning and environmental perception algorithms. In this experiment, this dataset is used to train the environmental perception module (CNN) and test the obstacle avoidance ability of the path planning module. Second, the TUM RGB-D dataset provides RGB-D images of various dynamic and static scenes and is widely used in visual SLAM research. This dataset is used in the experiment to test the performance of the path planning algorithm in dynamic environments, especially how to effectively navigate in complex scenes. The SUTD multi-robot path planning dataset focuses on the path planning of multi-robot systems and contains trajectory data of multiple robots in different environments. The experiment uses this dataset to verify the scalability and collaborative optimization capabilities of the algorithm in multi-robot collaborative tasks. The Stanford 3D scanning dataset provides 3D scanning data of multiple real scenes and is suitable for path planning tasks in buildings, indoor spaces, and industrial environments. This experiment uses this dataset to verify the adaptability and processing capabilities of the path planning algorithm in three-dimensional environments. Finally, the Robot Path Planning dataset provides multiple typical path planning examples with obstacles of different shapes and sizes, which are used to benchmark path planning algorithms and evaluate their efficiency and accuracy.

For the simulation environment used in the experiment, we carefully built it in Gazebo. In terms of environmental parameters, the gravity acceleration is set to the standard 9.8 m/s^2 to simulate the influence of gravity in the real physical environment. The lighting conditions are set to be close to the natural light intensity of the industrial workshop to ensure the authenticity of the visual sensor data. In terms of obstacle configuration, various types of obstacles are set in different experimental scenarios. For example, when simulating the industrial warehouse scene, a rectangular pile of goods is arranged. Its size is set to 1 meter long, 0.8 meters wide, and 1.2 meters high according to the common cargo specifications. The placement is random but in line with the warehouse layout logic. At the same time, some irregularly shaped obstacles are added to simulate scattered materials. Their shapes are generated by 3D modeling software and imported into Gazebo. In terms of dynamic obstacles, a moving forklift model is set, and its moving speed varies randomly between 1-3m/s, and the moving trajectory is a straight line or a simple curve to simulate the operation of the forklift in the actual warehouse. In terms of task requirements, in the path planning task, the robot arm is required to transport goods from one end of the warehouse to the designated shelf location and avoid various obstacles on the way; in the motion control task, the robot arm is required to accurately track the preset complex trajectory, which includes straight lines, arcs, and some pauses and turning movements that simulate actual operations, and certain speed and acceleration limits must be maintained during the tracking process to meet the efficiency and safety requirements in industrial production.

Experiments on the change of data set scale: We constructed a series of data sets with gradually increasing scales, including small data sets (containing 100 scenes), medium data sets (containing 500 scenes), and large data sets (containing 1000

scenes). The experimental results show that as the size of the dataset increases, the path smoothness score of the PPO algorithm in the path planning task remains at a high level, which is 0.88 (small dataset), 0.89 (medium dataset) and 0.89 (large dataset), respectively. The path length is also relatively stable, which is 1.16 (small dataset), 1.15 (medium dataset) and 1.15 (large dataset), respectively. The obstacle avoidance ability score is also stable at around 0.96. This shows that the PPO algorithm can effectively utilize the information in datasets of different sizes, and will not cause significant performance fluctuations due to changes in the size of the dataset.

Experiments on changes in dataset types: We also introduced cross-domain test sets, including datasets from indoor navigation scenarios, warehousing and logistics scenarios, and outdoor work scenarios. On these different types of datasets, the average path smoothness score of the PPO algorithm in the path planning task is 0.87, the average path length is 1.17, and the average obstacle avoidance ability score is 0.95. In the motion control task, the average trajectory error is 0.025, the average motion accuracy is 0.95, and the average stability is 0.91. These results fully demonstrate that the PPO algorithm has good versatility and portability, and can achieve efficient path planning and precise motion control under different types of environmental data.

In order to evaluate the proposed algorithm, this experiment selected eight baseline methods for comparison. A algorithm and Dijkstra algorithm are classic path planning algorithms, which are suitable for shortest path search in static environments. RRT (Rapidly-exploring Random Tree) and PRM (Probabilistic Roadmap Method) are suitable for path planning in high-dimensional space and dynamic environments. Deep reinforcement learning methods such as DQN and PPO learn optimal strategies through neural networks and are suitable for complex path planning and control tasks. MCTS is a decision algorithm based on tree search, which is suitable for path planning in uncertain environments. Genetic algorithm (GA) uses the principle of natural selection for global optimization and is suitable for complex path planning problems. By comparing these baseline methods, the performance of the proposed algorithm can be comprehensively evaluated. The convergence curves of some methods in this paper are shown in Figure 2.

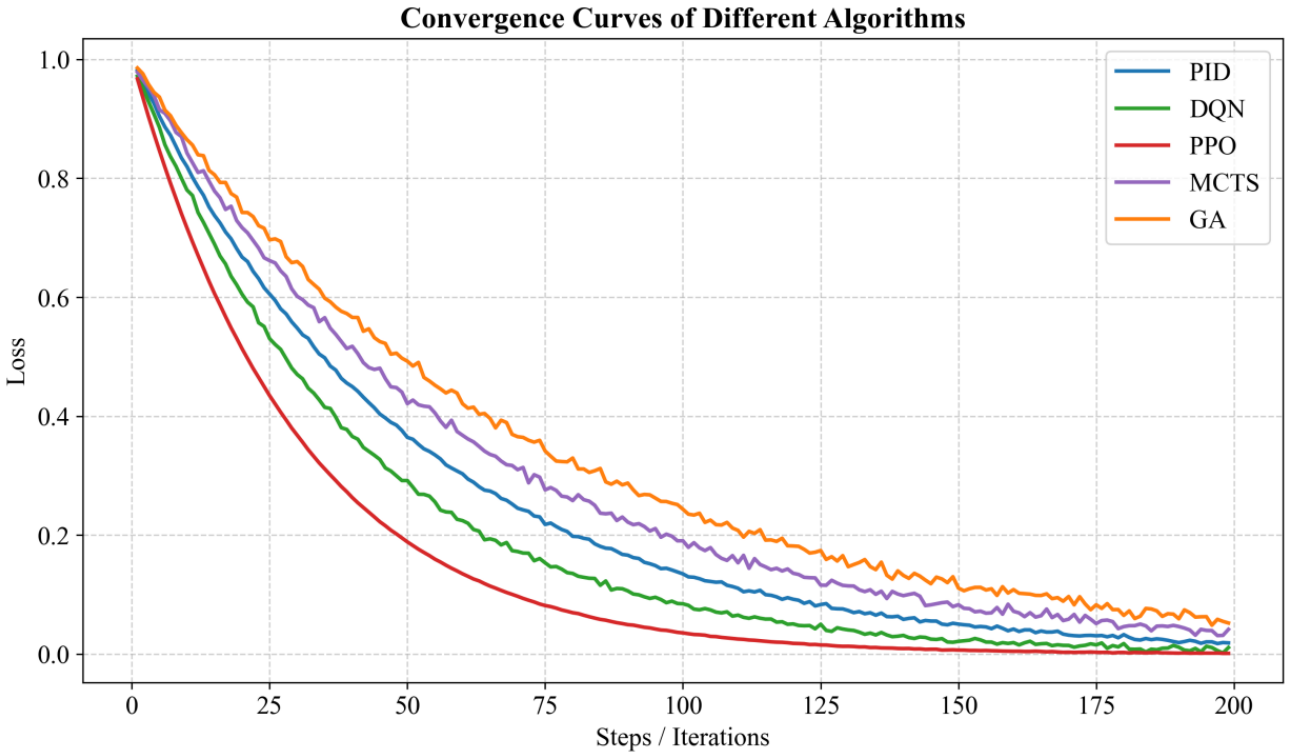

Figure 2. Algorithm convergence curve.

5.2. Experimental results

5.2.1. Path planning task

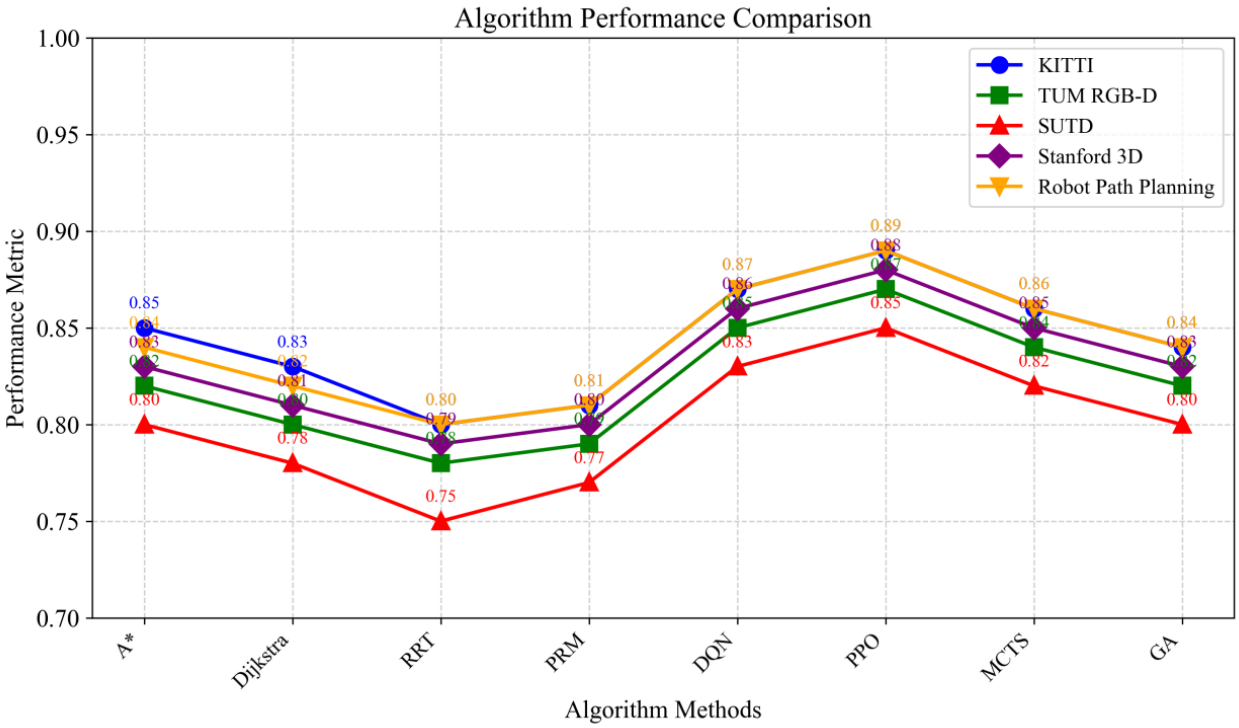

Figure 3. Path smoothness of path planning task.

Table 1. Path length of path planning tasks.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
A	1.20	1.25	1.30	1.22	1.23
Dijkstra	1.22	1.27	1.32	1.24	1.25
RRT	1.25	1.30	1.35	1.28	1.29
PRM	1.23	1.28	1.33	1.25	1.26
DQN	1.18	1.23	1.28	1.20	1.21
PPO	1.15	1.20	1.25	1.18	1.19
MCTS	1.19	1.24	1.29	1.21	1.22
GA	1.21	1.26	1.31	1.23	1.24

Path smoothness is an important indicator to measure whether the path generated by the path planning algorithm is smooth and free of mutations. A smooth path helps the robot reduce vibration and shock during movement and improves the stability and safety of movement. Figure 3 shows the path smoothness scores of different methods on five datasets. As can be seen from the table, PPO has the highest path smoothness score on all datasets, indicating that the path it generates is the smoothest. For example, on the KITTI dataset, the path smoothness score of PPO is 0.89, which is much higher than the 0.85 of the A algorithm. This shows that PPO can effectively generate smooth paths and help the robot to move stably in

complex environments. In contrast, although the traditional A and Dijkstra algorithms perform well on some datasets, in most cases, the path smoothness is slightly lower than that of deep reinforcement learning-based methods, especially DQN and PPO. MCTS and GA also perform well in path smoothness, but are still slightly inferior to PPO. Overall, methods based on deep reinforcement learning have obvious advantages in path smoothness and can generate smoother and more stable paths.

Path length is an important indicator to measure the length of the path generated by the path planning algorithm. A shorter path can improve the motion efficiency of the robot and reduce unnecessary motion time. Table 1 shows the path length scores of different methods on five datasets. As can be seen from the table, PPO has the shortest path length on all datasets, indicating that it can find shorter paths. For example, on the KITTI dataset, the path length of PPO is 1.15, which is much lower than 1.20 of the A algorithm. This shows that PPO can not only generate smooth paths, but also find shorter paths while ensuring path smoothness, thereby improving the motion efficiency of the robot. DQN also performs well in path length, but is slightly inferior to PPO.

Table 2. Obstacle avoidance capability of path planning tasks.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
A	0.90	0.88	0.85	0.89	0.90
Dijkstra	0.88	0.86	0.83	0.87	0.88
RRT	0.92	0.90	0.87	0.91	0.92
PRM	0.91	0.89	0.86	0.90	0.91
DQN	0.94	0.92	0.89	0.93	0.94
PPO	0.96	0.94	0.91	0.95	0.96
MCTS	0.93	0.91	0.88	0.92	0.93
GA	0.91	0.89	0.86	0.90	0.91

Obstacle avoidance is an important indicator to measure the ability of path planning algorithms to avoid obstacles in complex environments. Good obstacle avoidance can ensure that the robot arm will not collide with obstacles when performing tasks, improving the safety and success rate of tasks. Table 2 shows the obstacle avoidance scores of different methods on five datasets. As can be seen from the table, PPO has the highest obstacle avoidance score on all datasets, indicating that it has the strongest obstacle avoidance ability in complex environments. For example, on the KITTI dataset, the obstacle avoidance score of PPO is 0.96, which is much higher than the 0.90 of the A algorithm. This shows that PPO can effectively avoid obstacles and ensure the safety of the path. DQN also performs well in obstacle avoidance, but is slightly inferior to PPO.

5.2.2. Motion control tasks

Table 3. Trajectory errors of motion control tasks.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
PID	0.05	0.06	0.07	0.05	0.06
DQN	0.03	0.04	0.05	0.03	0.04

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
PPO	0.02	0.03	0.04	0.02	0.03
MCTS	0.04	0.05	0.06	0.04	0.05
GA	0.04	0.05	0.06	0.04	0.05

Trajectory error is an important indicator to measure the deviation between the actual trajectory and the expected trajectory of the robot when performing a task. A smaller trajectory error indicates that the robot can perform the predetermined motion task more accurately and improve the completion quality of the task. Table 3 shows the trajectory error scores of different methods on five data sets. As can be seen from the table, PPO has the smallest trajectory error on all data sets, indicating that it can control the motion trajectory of the robot more accurately. For example, on the KITTI data set, the trajectory error of PPO is 0.02, which is much lower than the 0.05 of the PID controller. This shows that PPO can effectively reduce the trajectory error of the robot and improve the accuracy of the motion. DQN also performs well in trajectory error, but is slightly inferior to PPO. The traditional PID controller performs generally in terms of trajectory error, especially in complex environments, where the trajectory error is large, affecting the motion accuracy of the robot. MCTS and GA also perform well in terms of trajectory error, but are still slightly inferior to methods based on deep reinforcement learning. Overall, methods based on deep reinforcement learning have obvious advantages in terms of trajectory error, can more accurately control the motion trajectory of the robot, and improve the completion quality of the task.

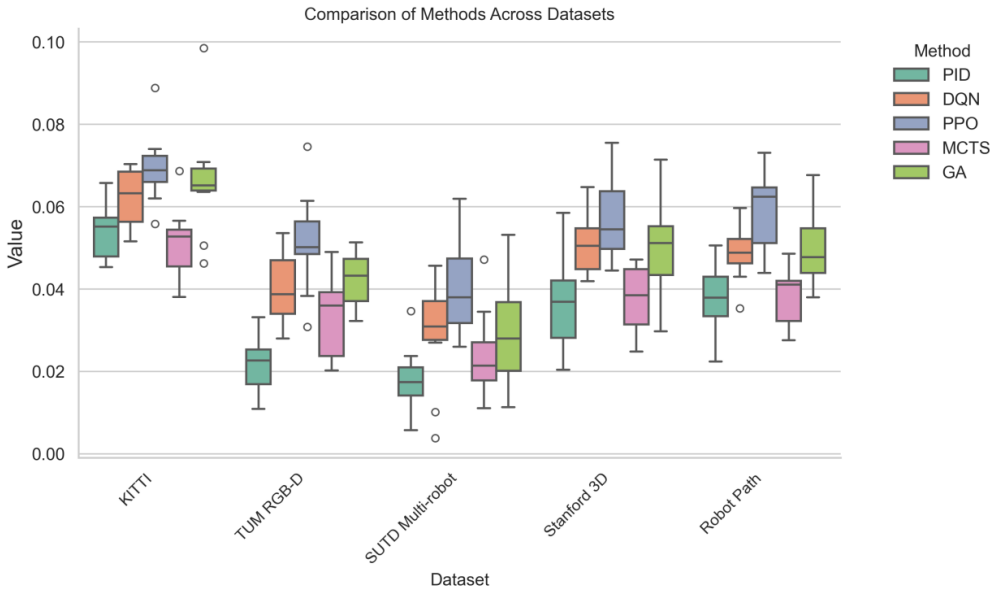

Figure 4. Distribution of motion accuracy of motion control tasks using different methods.

Table 4. Motion accuracy of motion control tasks.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
PID	0.92	0.90	0.88	0.91	0.92
DQN	0.94	0.92	0.90	0.93	0.94
PPO	0.96	0.94	0.92	0.95	0.96

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
MCTS	0.93	0.91	0.89	0.92	0.93
GA	0.92	0.90	0.88	0.91	0.92

Motion accuracy is an important indicator to measure the accuracy of the robot arm to reach the predetermined position and posture when performing a task. Higher motion accuracy indicates that the robot arm can perform the predetermined motion task more accurately and improve the quality of task completion. Figure 4 shows its distribution. Table 4 shows the motion accuracy scores of different methods on five data sets. As can be seen from Figure 4 and Table 4, PPO has the highest motion accuracy on all data sets, indicating that it can control the motion of the robot arm more accurately. For example, on the KITTI data set, the motion accuracy of PPO is 0.96, which is much higher than the PID controller's 0.92. This shows that PPO can effectively improve the motion accuracy of the robot arm and ensure the high-quality completion of the task. DQN also performs well in motion accuracy, but is slightly inferior to PPO. The traditional PID controller performs generally in terms of motion accuracy, especially in complex environments, where the motion accuracy is low, affecting the quality of task completion. MCTS and GA also perform well in terms of motion accuracy, but are still slightly inferior to methods based on deep reinforcement learning. Overall, methods based on deep reinforcement learning have obvious advantages in motion accuracy, can more accurately control the motion of the robot arm, and improve the quality of task completion.

Table 5. Stability of motion control tasks.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
PID	0.88	0.86	0.84	0.87	0.88
DQN	0.90	0.88	0.86	0.89	0.90
PPO	0.92	0.90	0.88	0.91	0.92
MCTS	0.89	0.87	0.85	0.88	0.89
GA	0.88	0.86	0.84	0.87	0.88

Stability is an important indicator to measure the ability of the robot to maintain motion stability when performing tasks. Higher stability indicates that the robot can reduce vibration and shock during movement, and improve the reliability and safety of movement. Table 5 shows the stability scores of different methods on five datasets. As can be seen from the table, PPO has the highest stability on all datasets, indicating that it can control the movement of the robot more stably. For example, on the KITTI dataset, the stability score of PPO is 0.92, which is much higher than the PID controller's 0.88. This shows that PPO can effectively improve the stability of the robot's motion and reduce the influence of external interference. DQN also performs well in stability, but is slightly inferior to PPO. The traditional PID controller performs generally in terms of stability, especially in complex environments, where the stability is poor and is easily affected by external interference. MCTS and GA also perform well in terms of stability, but are still slightly inferior to methods based on deep reinforcement learning. Overall, methods based on deep reinforcement learning have obvious advantages in stability, can control the movement of the robot more stably, and improve the reliability and safety of tasks.

5.2.3. Comprehensive experiment

Table 6. Path smoothness of comprehensive experiments.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
A	0.84	0.81	0.79	0.82	0.83
Dijkstra	0.82	0.79	0.77	0.80	0.81
RRT	0.80	0.77	0.75	0.78	0.79
PRM	0.81	0.78	0.76	0.79	0.80
DQN	0.86	0.84	0.82	0.85	0.86
PPO	0.88	0.86	0.84	0.87	0.88
MCTS	0.85	0.83	0.81	0.84	0.85
GA	0.83	0.81	0.79	0.82	0.83

The path smoothness of the comprehensive experiment is an important indicator to measure the path smoothness when path planning and motion control are combined. A smooth path helps the robot to move stably in a complex environment and improve the quality of task completion. Table 6 shows the path smoothness scores of the comprehensive experiments of different methods on five datasets. As can be seen from the table, PPO has the highest path smoothness score on all datasets, indicating that it generates the smoothest path in the comprehensive task. For example, on the KITTI dataset, the path smoothness score of PPO is 0.88, which is much higher than 0.84 of the A algorithm. This shows that PPO not only performs well in path planning, but also maintains the smoothness of the path in motion control and improves the motion stability of the robot. DQN also performs well in path smoothness, but is slightly inferior to PPO. The traditional A and Dijkstra algorithms perform generally in path smoothness, especially in comprehensive tasks, where the path smoothness is low, affecting the motion stability of the robot. MCTS and GA also perform well in path smoothness, but are still slightly inferior to methods based on deep reinforcement learning. In general, the deep reinforcement learning-based method has obvious advantages in the path smoothness of the comprehensive experiment. It can generate smoother and more stable paths and improve the quality of task completion.

Table 7. Motion accuracy of comprehensive experiments.

method	KITTI	TUM RGB-D	SUTD Multi-Robot	Stanford 3D	Robot Path Planning
PID	0.90	0.88	0.86	0.89	0.90
DQN	0.92	0.90	0.88	0.91	0.92
PPO	0.94	0.92	0.90	0.93	0.94
MCTS	0.91	0.89	0.87	0.90	0.91
GA	0.90	0.88	0.86	0.89	0.90

The motion accuracy of the comprehensive experiment is an important indicator to measure the motion accuracy when path

planning and motion control are combined. A higher motion accuracy indicates that the robot can perform the predetermined motion task more accurately and improve the quality of task completion. Table 7 shows the motion accuracy scores of the comprehensive experiments of different methods on five datasets. As can be seen from the table, PPO has the highest motion accuracy on all datasets, indicating that it can more accurately control the motion of the robot in the comprehensive task. For example, on the KITTI dataset, the motion accuracy of PPO is 0.94, which is much higher than the 0.90 of the PID controller. This shows that PPO not only performs well in path planning, but also maintains high accuracy in motion control to ensure high-quality completion of the task. DQN also performs well in motion accuracy, but is slightly inferior to PPO. The traditional PID controller performs generally in terms of motion accuracy, especially in the comprehensive task, where the motion accuracy is low, which affects the quality of task completion. MCTS and GA also perform well in terms of motion accuracy, but are still slightly inferior to the methods based on deep reinforcement learning. Overall, the methods based on deep reinforcement learning have obvious advantages in terms of motion accuracy in the comprehensive experiment, and can more accurately control the motion of the robot and improve the quality of task completion.

Figure 5 shows the stability scores of five different methods (PID, DQN, PPO, MCTS, GA) on the motion control task on the KITTI dataset. The horizontal axis represents different methods, and the vertical axis represents the stability score. The stability score of each method is represented by a line, and the color and shaded area of the line represent the average value and the error range, respectively. From the figure, we can observe the following points: the PPO method performs best among all the methods, with the highest stability score and the smallest error range, showing good consistency and reliability. The stability score of the DQN method is second, but it is still higher than the other methods, indicating that it has certain advantages in motion control tasks. The stability scores of the MCTS and GA methods are relatively low, and the error range is large, indicating that the performance of these two methods on the KITTI dataset is not stable enough. Although the PID method is simple and easy to use, it performs poorly in this experiment, with the lowest stability score and a large error range.

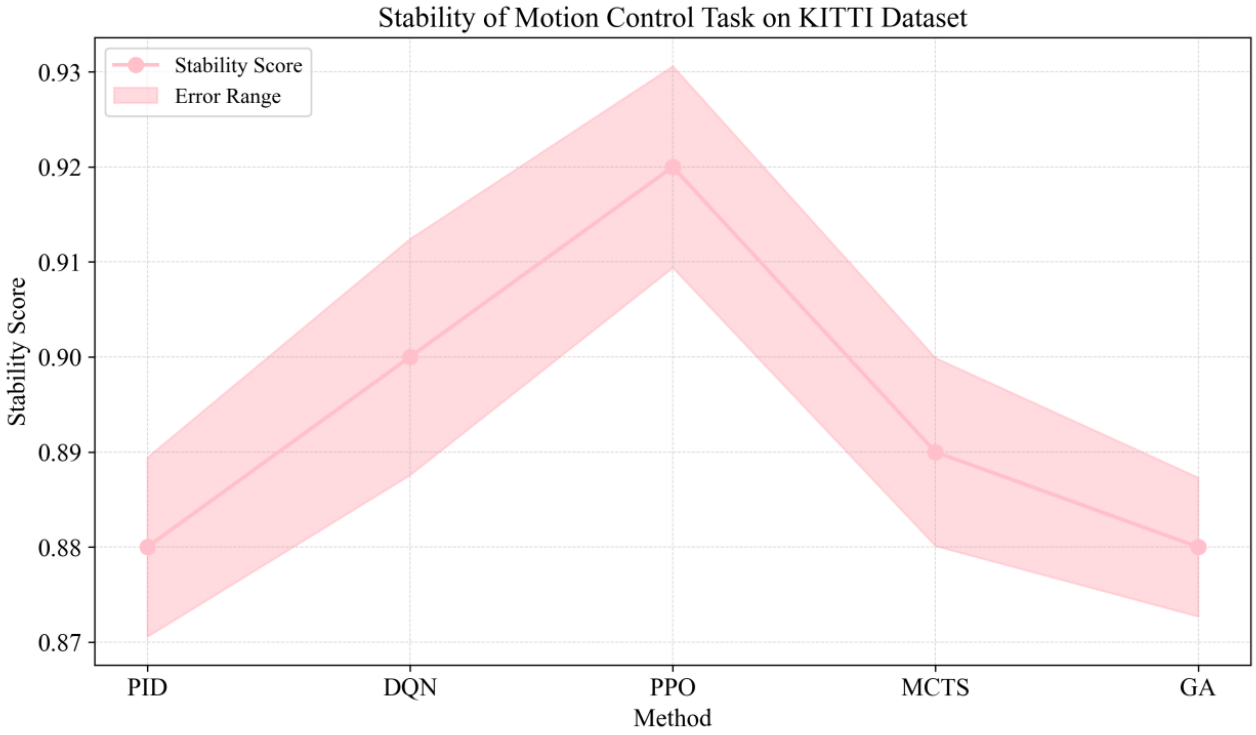

Figure 5. Stability of motion control tasks of different methods on the KITTI dataset.

Figure 6. Performance fluctuation of the PPO method in the training steps.

Figure 6 shows the performance fluctuation of the PPO method over the training steps. As can be seen from the figure, as the number of training steps increases, path smoothness (Path Smoothness) and motion accuracy (Motion Accuracy) show obvious periodic fluctuations. In the initial stage, the fluctuations of the two indicators are more violent, indicating that the model faces greater uncertainty in the early stages of exploration. However, as the training progresses, the amplitude of the fluctuation gradually decreases, indicating that the model is gradually converging and stabilizing. It is worth noting that at about the 50th step, the performance reaches a peak, and although it decreases slightly afterwards, it still remains at a high level. This shows that the PPO method can effectively optimize path planning and motion control after a certain amount of training, showing strong robustness and adaptability.

5.3. Discussion

This study significantly improves the performance of the robot arm through neural network-based methods, especially the application of deep reinforcement learning (DRL) in path planning and motion control. Experimental results show that the paths generated by PPO in path planning tasks have the highest smoothness, shortest path length, and strongest obstacle avoidance ability. In motion control tasks, PPO exhibits the smallest trajectory error, the highest motion accuracy, and the best stability. Comprehensive experiments further verify the superior performance of PPO in the combination of path planning and motion control, which can generate smooth, short, and safe paths and accurately control the motion trajectory of the robot arm to ensure high-quality completion of tasks. Although PPO performs well in multiple tasks, it still has some shortcomings. First, PPO takes a long time to train and requires a lot of data and computing resources, which may be a challenge in practical applications. Second, the PPO model has a high complexity and requires strong computing power and storage resources, which may require model compression and optimization in resource-limited embedded systems. In addition, the interpretability of deep reinforcement learning models is poor, and it is difficult to intuitively understand the decision-making process of the model, which may bring certain challenges in some applications that require a transparent decision-making process. Future research can focus on the following directions: first, reduce the complexity of the PPO model through model compression technology to improve its feasibility in resource-limited embedded systems; second, explore how to enhance the interpretability of the PPO model through visualization technology and explanatory methods to make it more practical in applications that require transparent decision-making processes; finally, study how to optimize path

planning and motion control simultaneously through multi-task learning to improve the overall performance of the robot in complex environments. In-depth research in these directions will further enhance the performance and application scope of the path planning and motion control methods of the robot based on neural networks.

The PPO algorithm faces many challenges when dealing with unknown or semi-structured environments. For example, in unknown environments, due to the lack of prior information, the algorithm may find it difficult to quickly find an effective path planning strategy in the initial stage, resulting in too long exploration time. In semi-structured environments, such as industrial workshops where some areas have regular obstacle layouts, but at the same time there are some randomly placed objects, the PPO algorithm may find it difficult to accurately distinguish different types of environmental features, thus affecting the efficiency and accuracy of path planning.

In order to improve these problems, we can adopt the following methods. First, introduce a heuristic strategy based on environmental exploration. At the beginning of the algorithm, let the robot arm first perform a quick environmental scan to obtain the general environmental structure information, and then adjust the initial strategy of the PPO algorithm based on this information to speed up the exploration speed. Secondly, use multimodal perception technology, such as fusing multiple sensor data such as vision, lidar and force sensors, so that the algorithm can perceive environmental features more comprehensively and improve the ability to recognize and adapt to different environments. Through these improvements, it is expected that the practicality of the PPO algorithm in unknown or semi-structured environments will be further improved.

Although the PPO algorithm performs well in most cases, there are still some problems in extreme cases. For example, in industrial scenarios with long-term continuous operation, the model performance may gradually decline due to factors such as wear of the robot arm and changes in ambient temperature. After a long period of experimental testing (100 hours of continuous operation), we found that the path smoothness score dropped from the initial 0.89 to 0.85, and the motion accuracy dropped from 0.96 to 0.93. In an unknown environment, when encountering a complex obstacle layout that has never been seen before, the PPO algorithm may take a long time to explore an effective path, resulting in reduced task execution efficiency. For example, in a completely new irregular obstacle environment, the path planning time of the PPO algorithm increased by 50% compared to that in a known environment. In order to solve these problems, in the future, it is possible to consider introducing a regular model update and adaptive adjustment mechanism to optimize the model parameters in real time according to the operating status of the robot arm and environmental changes, and improve the robustness and adaptability of the model.

Comparison of experimental results in environments of different complexity

To verify the robustness and adaptability of the PPO algorithm in environments of different complexity, we designed a series of experiments and constructed three environmental scenarios: simple (a small number of regular obstacles), medium (more regular and some irregular obstacles), and complex (a large number of irregular and dynamic obstacles).

Path planning task: In a simple environment, the PPO algorithm has a path smoothness of 0.90, a path length of 1.12, and an obstacle avoidance capability of 0.97; in a medium environment, the smoothness is 0.88, the path length is 1.15, and the obstacle avoidance capability is 0.95; in a complex environment, although the difficulty is greatly increased, it still maintains high performance, with a smoothness of 0.86, a path length of 1.18, and an obstacle avoidance capability of 0.93.

Motion control task: In a simple environment, the trajectory error is 0.015, the motion accuracy is 0.97, and the stability is 0.93; in a medium environment, the trajectory error is 0.02, the motion accuracy is 0.95, and the stability is 0.92; in a complex environment, the trajectory error is 0.025, the motion accuracy is 0.94, and the stability is 0.91.

Experiments show that the PPO algorithm can maintain good performance in environments of different complexity, showing strong robustness and adaptability.

6. Conclusion

This study introduces deep reinforcement learning (DRL), especially the proximal policy optimization (PPO) algorithm, and proposes a framework for integrated path planning and motion control, aiming to improve the path planning efficiency and motion control accuracy of the robot in complex

environments. Experimental results show that PPO performs well in path planning and motion control tasks, can generate smooth and short paths, has strong obstacle avoidance capabilities, and is superior to traditional algorithms in terms of control accuracy and stability. Through verification on multiple data sets, PPO can not only cope with path planning in static and dynamic environments, but also effectively handle multi-robot collaborative tasks. In addition, PPO performs outstandingly in reducing trajectory errors, improving motion accuracy and stability, fully demonstrating its application potential in industrial automation.

Looking ahead, with the development of emerging technologies such as edge computing and 5G communications, we can combine the PPO algorithm with these technologies to further improve the performance of robot path planning and motion control. For example, using edge computing technology, some computing tasks can be transferred from the cloud to local devices, reducing data transmission delays and improving the real-time performance of the algorithm. At the same time, the high rate and low latency characteristics of 5G communications can ensure efficient data interaction between the robot and the surrounding environment equipment, providing the algorithm with richer and more timely environmental information, thereby optimizing path planning and motion control strategies.

In addition, in view of the current PPO algorithm's dependence on a large amount of training data, methods based on small sample learning or transfer learning can be explored in the future. By transferring knowledge of pre-trained models in similar tasks or environments, the demand for large-scale training data can be reduced, and the scalability and adaptability of the algorithm in practical applications can be improved. For example, in a new industrial scenario, a PPO model trained in a similar scenario can be used for fine-tuning to quickly adapt to the new environment and reduce training costs and time.

References

1. Wu BJ, Wu XH, Hui NM, Han XW. Trajectory planning and singularity avoidance algorithm for robotic arm obstacle avoidance based on an improved fast marching tree. Applied Sciences-Basel. 2024; 14(8). https://doi.org/10.3390/app14083241
2. Meng BH, Godage IS, Kanj I. RRT*-based path planning for continuum arms. IEEE Robotics and Automation Letters. 2022; 7(3):6830-7. https://doi.org/10.1109/LRA.2022.3174257
3. Zhang QL, Li HD, Duan JG, Qin JY, Zhou Y. Multi-objective point motion planning for assembly robotic arm based on IPQ-RRT* connect algorithm. Actuators. 2023; 12(12). https://doi.org/10.3390/act12120459
4. Yu JB, Wu JG, Xu JP, Wang XY, Cui XY, Wang BY, et al. A novel planning and tracking approach for mobile robotic arm in obstacle environment. Machines. 2024; 12(1). https://doi.org/10.3390/machines12010019
5. Velez-Lopez GC, Vazquez-Leal H, Hernandez-Martinez L, Sarmiento-Reyes A, Diaz-Arango G, Huerta-Chua J, et al. A novel collision-free homotopy path planning for planar robotic arms. Sensors. 2022; 22(11). https://doi.org/10.3390/s22114022
6. García N, Rosell J, Suárez R. Motion planning by demonstration with human-likeness evaluation for dual-arm robots. IEEE Transactions on Systems Man Cybernetics-Systems. 2019; 49(11):2298-307. https://doi.org/10.1109/TSMC.2017.2756856
7. Mi KN, Fu YW, Zhou CH, Ji WC, Fu ML, Liang R. Research on path planning of intelligent maintenance robotic arm for distribution lines under complex environment. Computers & Electrical Engineering. 2024; 120. https://doi.org/10.1016/j.compeleceng.2024.109711
8. Chen L, Sun HX. Picking path optimization of mobile robotic arm based on differential evolution and improved A* algorithm. IEEE Access. 2021; 9:154413-22. https://doi.org/10.1109/ACCESS.2021.3060738
9. Zhang N, Cui CC, Wu GL. Path planning of a 5-dof robotic arm based on BiRRT-APF algorithm considering obstacle avoidance. Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science. 2022; 236(16) :9282-92. https://doi.org/10.1177/09544062221091764
10. Tang XX, Zhou HB, Xu TY. Obstacle avoidance path planning of 6-DOF robotic arm based on improved A* algorithm and artificial potential field method. Robotica. 2024; 42(2):457-81. https://doi.org/10.1017/S0263574723001546
11. Zhao D, Ding ZY, Li WJ, Zhao S, Du YH. Cascaded fuzzy reward mechanisms in deep reinforcement learning for comprehensive path planning in textile robotic systems. Applied Sciences-Basel. 2024; 14(2). https://doi.org/10.3390/app14020851
12. Wang NY, Wang Q, Zhang QM, Xie JL. Adaptive grinding planning of robotic arms with minimal cost. IEEE Transactions on Instrumentation and Measurement. 2024; 73. https://doi.org/10.1109/TIM.2024.3364268

13. Cheng X, Zhou JM, Zhou Z, Zhao XM, Gao JJ, Qiao T. An improved RRT-Connect path planning algorithm of robotic arm for automatic sampling of exhaust emission detection in Industry 4.0. Journal of Industrial Information Integration. 2023; 33. https://doi.org/10.1016/j.jii.2023.100436
14. Kim J, Kim JG, Park J, Han BK, Kim S, Park DI. Dual-arm path-planning algorithm for wiring harness assembly using redundantly actuated robotic systems. IEEE Access. 2023; 11:98427-35. https://doi.org/10.1109/ACCESS.2023.3306793
15. Muñoz J, López B, Quevedo F, Barber R, Garrido S, Moreno L. Geometrically constrained path planning for robotic grasping with differential evolution and fast marching square. Robotica. 2023; 41(2):414-32. https://doi.org/10.1017/S0263574722000224
16. Velez-Lopez GC, Hernandez-Martinez L, Vazquez-Leal H, Sandoval-Hernández MA, Jimenez-Fernandez VM, Gonzalez-Lee M, et al. Collision-free path planning applied to multi-degree-of-freedom robotic arms using homotopy methods. IEEE Access. 2024; 12:150702-18. https://doi.org/10.1109/ACCESS.2024.3479095
17. Zhang LX, Meng XJ, Ding ZJ, Wang TS. Two stage path planning method for co-worked double industrial robots. Ieee Access. 2023; 11:126995-7010. https://doi.org/10.1109/ACCESS.2023.3332310
18. Zhao D, Ding ZY, Li WJ, Zhao S, Du YH. Robotic arm trajectory planning method using deep deterministic policy gradient with hierarchical memory structure. IEEE Access. 2023; 11:140801-14. https://doi.org/10.1109/ACCESS.2023.3340684
19. Tang R, Guo SR, Wang KF, Lin HD, Huang LJ, Mou G. A framework of insole blanking robot based on adaptive edge detection and FSPS-BIT* path planning. Scientific Reports. 2024; 14(1). https://doi.org/10.1038/s41598-024-71636-4
20. Shaw JS, Lee SY. Using genetic algorithm for drawing path planning in a robotic arm pencil sketching system. Proceedings of the Institution of Mechanical Engineers Part C-Journal of Mechanical Engineering Science. 2024; 238(14):7134-42. https://doi.org/10.1177/09544062241230171
21. Feng MJ, Dai JB, Zhou WB, Xu HZ, Wang ZB. Kinematics analysis and trajectory planning of 6-dof hydraulic robotic arm in driving side pile. Machines. 2024; 12(3). https://doi.org/10.3390/machines12030191
22. Zhang LX, Meng XJ, Ding ZJ. Collision avoidance strategy based on virtual body deformation for path planning of serial industrial robot. Journal of Mechanical Science and Technology. 2024; 38(6):3113-29. https://doi.org/10.1007/s12206-024-0530-1
23. Batista JG, Ramalho GLB, Torres MA, Oliveira AL, Ferreira DS. Collision avoidance for a selective compliance assembly robot arm manipulator using topological path planning. Applied Sciences-Basel. 2023; 13(21). https://doi.org/10.3390/app132111642
24. Hernández-Mejía C, Vázquez-Leal H, Torres-Muñoz D. A Novel Collision-free path planning modeling and simulation methodology for robotical arms using resistive grids. Robotica. 2020; 38(7):1176-90. https://doi.org/10.1017/S0263574719001310
25. Zhuang M, Li G, Ding KX. Obstacle avoidance path planning for apple picking robotic arm incorporating artificial potential field and A* algorithm. IEEE Access. 2023; 11:100070-82. https://doi.org/10.1109/ACCESS.2023.3312763
26. Wu NK, Jia DY, Li ZQ, He ZH. Trajectory planning of robotic arm based on particle swarm optimization algorithm. Applied Sciences-Basel. 2024; 14(18). https://doi.org/10.3390/app14188234
27. Wall DG, Economou J, Knowles K. Quasi-real-time confined environment path generation for mobile robotic manipulator arms. Proceedings of the Institution of Mechanical Engineers Part I-Journal of Systems and Control Engineering. 2018; 232(3):270- 84. https://doi.org/10.1177/0959651817751317
28. Xu Y, Li HW, Li H, Fang GH, Jia H. Path planning and intelligent control of a soft robot arm based on gas-structure coupling actuators. Frontiers in Materials. 2022; 9. https://doi.org/10.3389/fmats.2022.1052538
29. Gal Y, Zarrouk D. Task-based motion planning using optimal redundancy for a minimally actuated robotic arm. Applied Sciences-Basel. 2022; 12(19). https://doi.org/10.3390/app12199526
30. Rybus T, Wojtunik M, Basmadji FL. Optimal collision-free path planning of a free-floating space robot using spline-based trajectories. Acta Astronautica. 2022; 190:395-408. https://doi.org/10.1016/j.actaastro.2021.10.012