Eng Int Syst (2025) 5: 499–511

Engineering
Intelligent Systems

Collaborative Operation of Artificial Intelligence and Mechanical Arm Based on Transformer-XL

Wenzhi Zhang[1], Min Xia[2]* and Biao Yang[3]

[1] *Department of Mechanical and Electronic Engineering, Sichuan Vocational and Technical College of Communications, Chengdu 611130, Sichuan, China*
[2] *College of Engineering and Technology, Nanchang Vocational University, Nanchang 330000, Jiangxi, China*
[3] *School of Mechanical and Electrical Engineering, Gongqing Vocational College of Science and Technology, Jiujiang 332000, Jiangxi, China*

In response to the problems of insufficient flexibility, slow response speed, and poor long-term dependency processing ability in traditional mechanical arm collaborative operations, this article combined Transformer-XL and deep Q-learning (DQL) to improve the adaptability and decision-making efficiency of mechanical arms in complex dynamic environments, thereby achieving more efficient and accurate collaborative operations. Firstly, the collected sensor data can be cleaned and normalized to improve the effectiveness of model training. Secondly, building deep learning models based on the Transformer-XL architecture can improve the ability to capture long-term dependency relationships. Then, based on the DQL algorithm, the decision-making process of the mechanical arm in dynamic environments can be optimized. Finally, the output of Transformer-XL can be combined with DQL to form an efficient collaborative control strategy. The experimental results show that compared with traditional control methods, DQL (combined with Transformer-XL) exhibits significant advantages in various task environments. In more complex dynamic obstacle avoidance tasks, the decision time is only 220 milliseconds, while maintaining a success rate of 88%. It demonstrates faster response speed and higher success rate in complex dynamic environments. At the same time, it still maintains a success rate of 60% with a delay of 90 seconds, demonstrating stronger adaptability in dynamic environments compared to traditional control strategies. The outstanding performance in high dynamic environments also validates the reliability of the research content.

Keywords: Mechanical Arm; Transformer-XL Architecture; Deep Q-learning; Normalization Processing; Control Strategy

1. INTRODUCTION

In modern manufacturing and service industries, mechanical arms are playing an increasingly important role as a key automation tool [1–2]. With the rise of intelligent manufacturing and Industry 4.0, the application scope of mechanical arms continues to expand in areas such as assembly, welding, packaging, medicine and logistics [3–4]. However, traditional mechanical arms face problems such as insufficient flexibility, slow response speed, and poor ability to handle long-term dependencies in collaborative operations [5–6]. These limitations make the adaptability of mechanical arms particularly important in complex dynamic environments [7].

In highly complex work scenarios, mechanical arms need to quickly adapt to diverse tasks and environmental changes [8–9]. This requires mechanical arms to have the ability to instantly perceive the environment and make dynamic decisions. Currently, many existing systems are limited by fixed-control strategies and find it difficult to respond quickly

*Corresponding Author. Emails: 29445407@qq.com, 13171373325@163.com, 1439842660@qq.com

to rapid changes in the environment [10–11]. In multiple-station cooperative operation, the mechanical arm needs to integrate and analyze the information input [12–13] from multiple sensors, so as to develop the optimal operation scheme. The inability of traditional means to deal with this kind of multiple information fusion restricts the overall flexibility and operational efficiency of the system [14]. Long-term dependence is also a key problem in robotic arm collaboration [15]. When performing complex tasks, the action strategy of the robotic arm is not only affected by the current state, but is also closely related to its past action sequence. Traditional sequence modeling techniques cannot fully capture dependencies over the long term, leading to information omission or decision errors [16–17]. Due to the above problems, enhancing the mechanical arm's ability to understand the historical action sequence has become the key to improving its intelligence level.

To solve the above problems, this article introduces a cooperative operation scheme that integrates Transformer-XL model and DQL algorithm to improve the collaborative ability of the mechanical arm in a complex and variable dynamic environment. In the data preprocessing stage, it is necessary to implement a strict cleaning and normalization processing sequence for the collected sensor data to ensure the robustness and performance optimization of the subsequent model training. To build deep learning models, this study utilizes the excellent long-term dependence capture ability of Transformer-XL architecture to significantly improve the adaptability and response speed of the model to the dynamic changes of the environment. This study carefully designs the reinforcement learning strategy for the model's framework, adopts the DQL algorithm as the core, and makes detailed optimization and adjustment for the diversified operation tasks in order to strengthen the independent decision-making capability and adaptability of the mechanical arm. Finally, the feature information generated by the Transformer-XL model and the action strategy output by the DQL algorithm are effectively integrated to create an efficient cooperative control mechanism. This mechanism enables the robotic arm to respond quickly and precisely in a rapidly changing environment, greatly improving its operational flexibility and accuracy. In order to verify the effectiveness and superiority of the proposed method, this study established a series of evaluation indicators, including decision time, operation success rate and memory ability of the model and conducted a comprehensive experimental verification according to these indicators. The experimental results show that this study not only provides a new perspective and solution for the intelligent control field of the mechanical arm, but also provides a solid basis for improving the flexibility and efficiency of the mechanical arm in practical application scenarios.

2. RELATED WORK

In previous research on collaborative operation of mechanical arms, traditional studies have focused on issues of system safety and performance control. Yan et al. proposed a dynamic model and performance constraint control for a line driven soft mechanical arm, combining screw theory with Cosserat theory. The researchers established a dynamic model of a soft mechanical arm, which effectively solved the motion constraint problem of a line driven soft mechanical arm. However, this dynamic model still has limitations in terms of flexibility in complex and changing environments [18]. Nana et al. designed and implemented a safety detection system for robot control, which performs real-time safety detection of robot operations on industrial production lines to improve the safety and reliability of robot control systems. This has, to some extent, achieved network communication security for robot remote control systems [19]. Ning et al. proposed a unified dual IENT-ZNN scheme based on Integration-Enhanced Noise-Tolerant Zeroing Neural Network (IENT-ZNN) for the kinematic control problem of rigid and continuous mechanical arms, which improves the performance of the mechanical arm under noise interference but without solving the structural parameters of the mechanical arm [20]. Traditional research on mechanical arm control has focused mostly on the kinematic constraints of rigid structure mechanical arms or line driven soft mechanical arms. It has poor adaptability to dynamic changes in the environment and is difficult to cope with complex tasks. In order to improve the adaptability and collaborative efficiency of mechanical arms to dynamic changes in the environment, Wei et al. proposed an enhanced RRT (Rapidly-exploring Random Trees) algorithm, which corrects the node redundancy and path quality degradation problems unique to traditional RRT methods, optimizes the search space, and significantly improves efficiency. However, this algorithm is unable to capture long-term dependency information, which limits the performance of mechanical arms in handling complex collaborative tasks, especially when facing continuous decision-making problems. The limitations of traditional algorithms are particularly evident [21].

In the field of collaborative operation of mechanical arms, traditional methods focus mainly on improving system safety and performance control. With the development of automation technology, mechanical arms have been widely used in fields such as manufacturing, healthcare, and service robots. However, there are still significant bottlenecks in the adaptability and decision-making efficiency of traditional methods in complex dynamic environments. When dealing with complex or collaborative tasks, traditional control strategies often provide inadequate flexibility, slow response speed and poor environmental adaptability, making it difficult to meet the actual requirements. Most of the current studies focus on the kinematic limitations of the rigid and linear-driven soft robotic arms, without providing a comprehensive solution to the practical application challenges of the robotic arms in dynamic and complex environments. Although these studies have made progress in terms of safety and performance control in specific application scenarios, they still need to be strengthened to improve the collaboration efficiency of the mechanical arm and its adaptability to changes in the external environment. Also problematic is that traditional research approaches rely too heavily on the preset environmental parameters, making it difficult to adapt quickly to the changing environment in real-life scenarios. In particular, when the task requirements or environmental characteristics change significantly, the robotic arm often finds it difficult to respond quickly

and effectively. Existing robotic arm control methods also have limitations in responding to the dynamic requirements of complex tasks, especially in scenarios requiring multi-task collaborative operation. In order to significantly improve the decision efficiency of the robotic arm and its adaptability to complex dynamic environments, researchers need to develop control algorithms that can more effectively capture long-term dependent information and optimize real-time decisions in complex collaborative tasks. In view of this, this study is an attempt to overcome the limitations of traditional methods by combining Transformer-XL and DQL model, with the specific goal being to enhance the ability of robotic arm to capture long-term dependent information and optimize the collaborative control strategy.

3. METHOD

3.1 Data Preprocessing

3.1.1 Data Collection and Cleaning

Data is collected from various sensors, including position sensors, accelerometers, and gyroscopes. These sensors record data in real-time during the operation of the mechanical arm, ensuring accurate multidimensional information is obtained. During the data collection process, it is necessary to ensure that the sampling frequency of the sensor meets a certain standard in order to capture the dynamic changes of the mechanical arm under different operational tasks. All sensor data is stored in a unified database, forming a complete raw dataset.

In the data cleaning stage, a preliminary review of the raw data was conducted to identify missing values and outliers. The handling of missing values was done by applying the linear interpolation method. In the specific implementation process, for each missing value X_t, the nearest valid data points X_{t-1} and X_{t+1} were sought before and after it, and the missing value was calculated based on a linear relationship, as shown in formula 1:

$$X_t = \frac{X_{t-1} + X_{t+1}}{2} \tag{1}$$

This method ensures the smoothness and continuity of data, effectively preventing model performance degradation caused by missing values.

For features with a high proportion of missing values, a threshold is set. If the missing value ratio p_i of a certain feature F_i exceeds this threshold, that feature can be discarded from the dataset. In this study, the proportion of missing values was calculated using the following formula:

$$p_i = \frac{N_{missing}}{N_{total}} \tag{2}$$

N_{total} represents the total number of data points in the feature, and $N_{missing}$ represents the number of missing values in feature F_i. The decision to discard features is based on their potential negative impact on model training to avoid introducing too much noise. After the feature dropout operation is implemented, the structure of the dataset needs to be re-evaluated to ensure that it still maintains sufficient information in subsequent training.

Outlier detection can be performed on the collected data, using the interquartile range (IQR) method to identify outliers. Calculate the lower quartile Q_1 and upper quartile Q_3 of the dataset, and then calculate IQR based on the difference between Q_3 and Q_1. Determine the threshold range for outliers, as shown in formulas 3 and 4:

$$\text{Lower Bound} = Q_1 - 1.5 \times IQR \tag{3}$$

$$\text{Upper Bound} = Q_3 + 1.5 \times IQR \tag{4}$$

Data beyond this range can be marked as outliers and can be retained or deleted depending on the actual situation. This step improves the quality of the dataset, ensuring that subsequent model training is based on high-quality data.

After the data collection and cleaning stages, the obtained dataset has continuity and consistency, proving a solid basis for subsequent data preprocessing and model training.

3.1.2 Data Normalization Processing

For data normalization, the Min-Max normalization method was used to ensure that the sensor data were processed in the same dimension. The steps are as follows:

Statistical analysis can be performed on the collected sensor data to determine the minimum and maximum values of each feature. X_j is set as the original dataset for the jth feature, and minimum and maximum values, $X_{j,\min} = \min(X_j)$ and $X_{j,\max} = \max(X_j)$, are calculated.

Each data point X_{ij} (i represents sample index and j represents feature index) is normalized using these minimum and maximum values. Normalize as shown in formula 5:

$$X'_{ij} = \frac{X_{ij} - X_{j,min}}{X_{j,max} - X_{j,min}} \text{ for } j = 1, 2, \ldots, n \tag{5}$$

where X'_{ij} represents the normalized data points, and n represents the total number of features. This process scales all feature values to the interval of [0,1], eliminating the differences in magnitude between different feature classes.

Considering the possibility of outliers in the dataset, a modified normalization method is further introduced to enhance the robustness of data processing. Set the upper threshold $T_{j,up}$ and lower threshold $T_{j,low}$, as shown in formulas 6 and 7:

$$T_{j,up} = \mu_j + k \cdot \sigma_j \tag{6}$$

$$T_{j,low} = \mu_j - k \cdot \sigma_j \tag{7}$$

where μ_j and σ_j are the mean and standard deviation of feature X_j, respectively, and k is a constant (such as 1.5 or 2) used to control the flexibility of the threshold. Data beyond this range can be processed using the following formula:

$$X_{ij} = \begin{cases} T_{j,low} & \text{if } X_{ij} < T_{j,low} \\ T_{j,up} & \text{if } X_{ij} > T_{j,up} \\ X_{ij} & \text{otherwise} \end{cases} \tag{8}$$

After handling outliers, the minimum and maximum values can be recalculated and normalized again. After

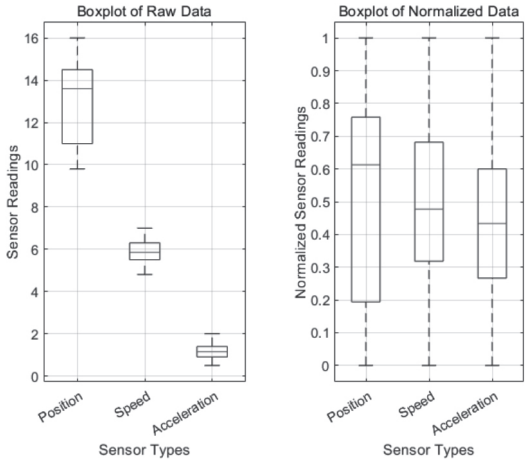

Figure 1 Comparison of normalized data.

normalization is completed, visual tools such as box plots can be used to compare and analyze the data before and after normalization, ensuring that the distribution range of the data has been successfully scaled to [0,1] and that the influence of each feature is balanced during the training process.

Figure 1 shows a comparison of box plots before and after normalization, displaying the raw data of three sensors: position sensor, speed sensor, and acceleration sensor. Before normalization, the horizontal axis represents the sensor type and the vertical axis represents the sensor reading. The range of position sensor data is from 9.8 to 16.0, showing high volatility, and the upper quartile is significantly higher than the median. Speed sensor: the data range is between 4.8 and 7.0, relatively concentrated, with small fluctuations, indicating that the reading of the speed sensor is relatively stable. Acceleration sensor: the data range is from 0.5 to 2.0, and there are significant differences, indicating that the acceleration changes greatly and is affected by environmental factors. In the normalized box plot, the same horizontal and vertical axes are set, but the data range of the vertical axis is reduced to between 0 and 1. After normalization, the data distribution of each sensor is uniform and concentrated, showing better comparability. After normalization, the influence of each feature during the training process is balanced, preventing the dominant influence of certain features on model training.

Finally, through the rigorous data normalization process mentioned above, a solid foundation was laid for the subsequent model construction, ensuring that the model can effectively capture potential patterns in the data during the training process, and improving the accuracy and stability of the model.

3.2 Model Construction

3.2.1 Transformer-XL Architecture Design

When building the basic architecture of Transformer-XL, it is necessary to define the dimensions and data format of the input sequence. The input data is organized into a time series containing data features from multiple sensors. The dimensions of each feature are preprocessed and normalized to ensure that the model responds consistently to different features. In this study, a self-attention mechanism module is designed. This module calculates the influence weight of each element in the input sequence on other elements, allowing the model to focus on relevant information. In the implementation process, three vectors are generated for the input sequence $X = [x_1, x_2, \ldots, x_n]$: Query, Key, and Value, represented as $Q = XW^Q, K = XW^K, V = XW^V$. Among them, W^Q, W^K, and W^V are learnable weight matrices. Then calculate the attention weight matrix A, as shown in formula 9:

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{9}$$

where d_k is the dimension of the key vector used for scaling to prevent gradient vanishing due to excessively large dot product values. Then, attention weights are applied to the value vector V, as shown in formula 10:

$$\text{Attention}(Q, K, V) = AV \tag{10}$$

Through this process, the model can dynamically adjust the level of attention to different parts of the input sequence, effectively capturing key information.

In the architecture, relative position encoding is introduced to overcome the limitations of traditional absolute position encoding. Relative position encoding is generated by considering the relative distance between elements. Assuming that the relative positions between elements in the input sequence are represented by PE_{ij}, the generation formula 11 for relative position encoding is as follows:

$$PE_{ij} = \begin{cases} \sin\left(\frac{j}{10000^{\frac{i}{d}}}\right) \\ \cos\left(\frac{j}{10000^{\frac{i-1}{d}}}\right) \end{cases} \tag{11}$$

where d represents the dimension of encoding. The relative position encoding can be added to the input embedding vector to ensure that the encoding of each position can adapt to the changing input length, as shown in formula 12:

$$X' = X + PE \tag{12}$$

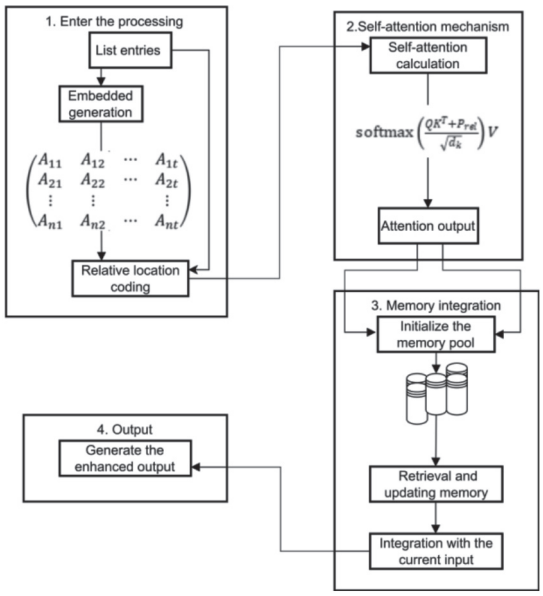

Figure 2 Position encoding and paragraph memory mechanism.

To increase the model's ability to capture long-term dependencies, a paragraph memory mechanism can be added. This mechanism includes a memory pool for storing state information from several past time steps. At each time step, the memory pool can be updated to retain useful historical information. In specific operations, an additional gating mechanism can be used to determine when to write new states into the memory pool and when to read information from it, defined as formula 13:

$$M_t = \alpha M_{t-1} + (\alpha)\text{NewState} \tag{13}$$

where M_t is the memory state of the current time step, and α is the hyperparameter that controls memory updates. When processing new inputs, the model dynamically retrieves relevant historical states by querying the memory pool M_t and updates them to formula 14:

$$\text{Read} = \text{Attention}(Q, K_M, V_M) \tag{14}$$

K_M and V_M represent keys and values from the memory pool, respectively.

In terms of setting the number of layers and hidden units, a structure of 6-layer encoder and 6-layer decoder can be used to improve the expression ability of the model. Each layer contains a multi-head self-attention mechanism and a feedforward neural network, optimizing the training process through residual connections and layer normalization. The output of the feedforward network is defined by formula 15:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{15}$$

W_1 and W_2 are learnable weight matrices, while b_1 and b_2 are bias terms. Through this layer configuration, the model can comprehensively explore complex patterns in the input sequence, ensuring decision-making efficiency in dynamic environments.

3.2.2 Position Encoding and Paragraph Memory

When implementing relative position encoding in the model, it is necessary to define the calculation method for relative position. For any two elements i and j in the input sequence, their relative distance $d = j - id$ can be calculated. Based on this relative distance, a learned embedding matrix can be used to map each relative distance to a vector representation for subsequent self-attention calculations. This embedding matrix is optimized through the backpropagation algorithm, enabling dynamic adjustment of relative position encoding to meet the requirements of different tasks.

In the calculation process of self-attention mechanism, the relative position encoding is added to the input feature vector to ensure that the model can comprehensively consider the relative position relationship between elements in the input sequence. The calculation formula for attention weight in this process has been modified to formula 16:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + P_{rel}}{\sqrt{d_k}}\right)V \tag{16}$$

P_{rel} represents relative position encoding, while Q, K, and V represent query, key, and value, respectively. This modification enables the model to maintain positional consistency when processing inputs of different lengths, significantly improving the ability to capture long-range dependencies.

In order to enhance the long-term memory ability of the model, a paragraph memory mechanism is introduced. With this mechanism, a memory pool can be created to store the state information of the previous step. The operation involves extracting relevant information from the memory pool and performing calculations based on the current input each time a new sequence is inputted. When the memory pool is updated, a sliding window strategy is applied to ensure that old information is not completely discarded when processing new input, but is updated according to the preset window size.

Figure 2 starts with the input sequence data, which are preprocessed via a positional embedding layer. Based on the

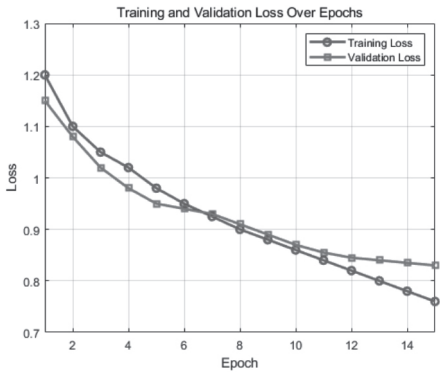

Figure 3 Changes in Training and Validation Losses.

relative position encoding, the relationship between adjacent elements is modeled precisely, thus ensuring that the model can capture information about the relative positions of any two elements in the sequence when dealing with sequences of different lengths.

The processed data can be input into the self-attention layer which is responsible for combining positional encoding information to improve the model's understanding of long-distance dependencies. The self-attention mechanism is not only dependent on the feature vector of the input, but also combines relative position coding, which enables the attention distribution to adapt with the input length.

The memory module is the implementation of a paragraph memory pool. When processing new sequences, the model can dynamically extract information from the memory pool to ensure that historical information is not lost during the decision-making process. The sliding window mechanism controls the content update in the memory pool, ensuring that only information related to the current task is retained.

After being processed by the self-attention mechanism and memory module, the sequence can generate an enhanced output sequence. These output sequences have richer contextual information and can effectively capture long-range dependencies in the sequence.

3.2.3 Model Training

During the model training phase, a sequence-to-sequence structure is adopted to adapt to the temporal characteristics of the mechanical arm control task. The input for training is a preprocessed sensor data sequence, including multidimensional features such as position, velocity, acceleration, etc. These features are normalized to ensure that the magnitudes of the inputs are consistent during the model training process, preventing certain features from having a significant impact on the loss function.

The output of training is the control instructions for the mechanical arm, which need to be represented in the form of multi-class classification, such as the motion instructions for each joint. To evaluate the performance of the model, the cross-entropy loss function is used as the optimization objective, as shown in formula 17:

$$L = -\sum_{i=1}^{N} y_i \log(p_i) \tag{17}$$

where y_i is the true label, and p_i is the probability of the model output. This loss function can effectively quantify the gap between the model output and the actual control instructions, providing clear guidance for optimizing model parameters.

To improve training efficiency, the Adam optimizer is selected for model parameter updates as it combines adaptive learning rate and momentum methods, which can effectively accelerate the convergence process. The initial learning rate is set to 0.001 and gradually reduced through a learning rate decay strategy. The specific method is to check the validation set loss value at the end of each epoch. If the validation set loss decreases within 5 consecutive epochs, the learning rate can be multiplied by 0.5 to make the model more stable in subsequent training.

The horizontal axis in Figure 3 represents training epochs, ranging from 1 to 15. The vertical axis represents the loss value, indicating the performance of the model on the training and validation sets. Generally, the lower the loss value, the better the performance of the model. As the number of training epochs increases, the training loss shows a gradually decreasing trend, from the initial 1.200 to 0.76, indicating that the model gradually adapts to the data and improves its learning ability on the training set. The validation loss also showed a downward trend, decreasing from 1.150 to 0.83, indicating that the model's generalization ability on the validation set is also improving.

The training loss and validation loss both significantly decrease with the increase of training epochs, indicating that the model is gradually learning and adapting to the input data. However, it is worth noting that the validation loss begins to stabilize after the 7th round, indicating that the model may face the risk of overfitting after this, although the training loss continues to decline. This requires attention to changes in validation loss in order to adjust training strategies in a timely manner, prevent overfitting of the model to the training data, and maintain its performance on unseen data.

Based on the changes in training and validation losses, the learning rate changes shown in Table 1 can be calculated. The learning rate values for each round gradually decrease from the initial 0.001 to 0.00025, enabling the model to converge more stably when approaching the optimal solution. By gradually decreasing the learning rate, the model can be optimized more effectively, especially when approaching convergence, which helps prevent instability and overfitting during the

Table 1 Learning rate attenuation.

Epoch	Learning Rate	Training Loss	Validation Loss
1	0.001	1.200	1.150
2	0.001	1.100	1.080
3	0.001	1.050	1.020
4	0.001	1.020	0.980
5	0.001	0.980	0.950
6	0.0005	0.950	0.940
7	0.0005	0.925	0.930
8	0.0005	0.900	0.910
9	0.0005	0.880	0.890
10	0.0005	0.860	0.870
11	0.00025	0.840	0.855
12	0.00025	0.820	0.845
13	0.00025	0.800	0.840
14	0.00025	0.780	0.835
15	0.00025	0.760	0.830

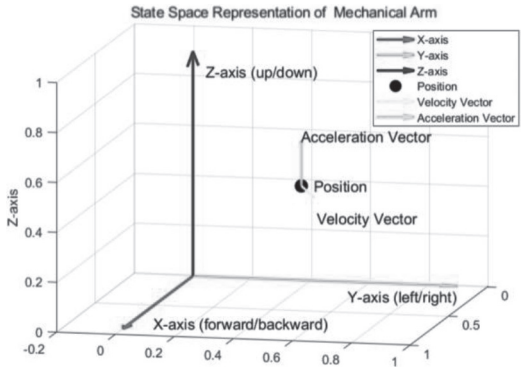

Figure 4 Three-dimensional state space.

training process. The data in Table 1 clearly demonstrates the relationship between learning rate and training and validation losses, providing a basis for understanding the learning strategies during the model training process.

During the training process, a batch training strategy is adopted to divide the training data into small batches in order to reduce memory consumption and accelerate training speed. After each batch undergoes forward and backward propagation, the model parameters are updated. To prevent overfitting, the dropout technique is used to randomly discard a certain number of neurons during training to improve the model's generalization ability.

At the end of each epoch, the training loss and validation loss are recorded to monitor the changes in model performance, in order to adjust the training strategy in a timely manner based on the learning progress of the model. By following the above steps, the effectiveness and efficiency of model training are ensured, establishing the foundation for subsequent control strategy optimization.

3.3 Strategy Design

3.3.1 Definition of State and Action Space

The definition of state and action space is the foundation for designing DQL algorithms. The state space of a mechanical arm includes multiple key parameters, which are monitored in real time by various sensors. The state space consists of the following dimensions: the three-dimensional position of the end effector of the mechanical arm (x, y, z coordinates), linear velocity (the velocity vector of the mechanical arm's movement), angular velocity (describing the rate of rotation of the mechanical arm), and acceleration (reflecting the rate of change in the mechanical arm's motion). In addition, it also includes the position of obstacles, the status of target objects, and other sensor feedback on the status information of the external environment. These pieces of information are uniformly encoded after data processing to create a state vector as input features for DQL algorithms.

Figure 4 shows the state and motion information of the mechanical arm in three-dimensional space, represented by the X, Y, and Z coordinate axes to indicate the position of the mechanical arm. The X-axis represents the forward and backward motion of the mechanical arm, the Y-axis represents the left and right motion, and the Z-axis represents the up and down motion. These coordinate axes constitute the state space of the mechanical arm in the workspace, covering all of its observable state variables.

In this coordinate system, a black mark point is given a specific meaning, which represents the current state of the robotic arm during the task, including the real-time observation value of its position, speed and acceleration.

This state point integrates the position information of the mechanical arm in three dimensions: X, Y and Z, and constitutes the core part of the input features of the Q learning algorithm. With this state point, the positional changes of the robotic arm in the time series are clearly presented. The yellow arrow extending outward from the state point is the movement speed vector of the arm. It not only indicates the travel direction of the arm, but also intuitively shows the current travel rate of the arm and the change of the direction through the length of the arrow, and it can intuitively grasp the movement trend of the arm in space. Another light blue arrow from the state point represents the acceleration vector. The arrow points to the direction of the arm acceleration, which reveals specific trends in the velocity. Acceleration information plays an important role in the decision-making optimization process of the Q learning algorithm because it provides an in-depth insight into the dynamic behavior of the mechanical arm, and helps the algorithm to more accurately predict and plan future action strategies. This design enables the algorithm to make full use of the motion state information of the robotic arm and optimize its decision-making process.

3.3.2 Construction of Q-value Function

In DQL, the construction of the Q-value function is crucial as it is responsible for determining the value of each state action pair. The Q-value function is defined as $Q(s, a)$, where s represents the current state and a represents the selected action. A deep neural network model can be constructed to approximate this function, with the input being the state feature $x \in \mathbb{R}^n$ and the output being the Q-value $Q(s, a; \theta)$ of all feasible actions $a \in A$, where θ is the network parameter. The network structure is generally designed as multiple fully-connected layers, each layer using a ReLU (Rectified Linear Unit) activation function to enhance the nonlinear expression ability of the model, as shown in formula 18:

$$h^l = \text{ReLU}(W^l h^{l-1} + b^l) \tag{18}$$

where h^l is the output of lth, W^l and b^l are the weights and biases, respectively.

During the training process, an experience replay mechanism is used to improve the learning efficiency of the Q-value function. The state transition records collected by the intelligent agent in the environment (including the current state, actions taken, rewards obtained, and next state) can be stored in the experience pool D. Each record format is (s_t, a_t, r_t, s_{t+1}). At each training stage, a small batch of samples $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^m$ is randomly selected from the experience pool for training to break the correlation between samples.

The update process of Q value is based on the Bellman equation, as shown in formula 19:

$$Q(s_t, a_t; \theta) \leftarrow Q(s_t, a_t; \theta) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)) \tag{19}$$

where a is the learning rate that controls the update step size, and γ is the discount factor used to adjust the impact of future rewards. The weights of the target network $Q(s_t, a_t; \theta)$ are updated every fixed number of steps r to maintain stability. This mechanism improves the convergence of the model by reducing the fluctuation of the target value.

To ensure the effectiveness of the Q-value function, periodic evaluations are conducted. The accuracy of the Q-value function can be verified by comparing the performance of the current strategy π and the target strategy π^* in the environment. Specifically, the estimation bias of Q value is calculated using mean square error, as shown in formulas 20 and 21:

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} (y_i - Q(s_i, a_i; \theta))^2 \tag{20}$$

$$y_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta^-) \tag{21}$$

where y_i is the target Q value. If the estimation deviation of Q value is found to be large, the learning rate α is adjusted or the reward mechanism is improved to optimize the model training effect.

This series of operations ensures that the Q-value function accurately reflects the potential value of the state-action pair, thus improving the decision-making ability of the robotic arm in a complex dynamic environment.

3.3.3 Reward Mechanism Design

In the strategy design of DQL, the construction of the reward mechanism can guide the learning process of the robotic arm. When defining a positive reward, the robotic arm was rewarded with $+10$ for successfully completing the task. During task execution, immediate rewards are set. When the robotic arm approaches the target object, small positive rewards can be given $(+1)$. For task failure cases, a negative reward was set to -5. Unexpected behaviors (collision or unexpected actions) during the task also give certain negative rewards (-2).

When designing the reward mechanism, a time penalty mechanism was also introduced. If the time spent by the mechanical arm during task execution exceeds the preset threshold, an additional negative reward (such as -1) can be given to encourage the agent to improve reaction speed and operational efficiency while ensuring task completion. This strategy effectively reduces task completion time and improves overall operational efficiency.

3.3.4 Implementation of Collaborative Control Algorithm

The implementation of collaborative control algorithms improves the operational efficiency of mechanical arms in complex dynamic environments by combining Transformer-XL output with DQL.

The Transformer-XL model can be trained to extract and capture long-term dependencies in input sensor data. When encoding historical state sequences, Transformer-XL can generate rich contextual information and provide environmental state representations for each time step. This representation can serve as an input feature for DQL algorithms, improving the model's understanding and responsiveness to

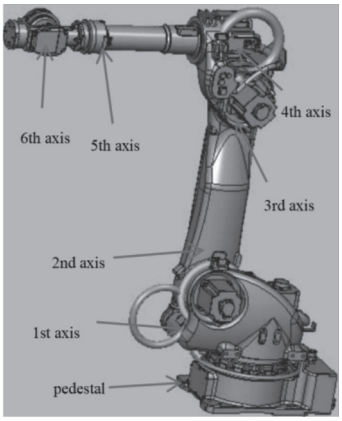

Figure 5 Schematic diagram of connecting rod.

Table 2 D–H parameters of mechanical arm.

Joint i	θ_i	$d_i(m)$	$a_i(m)$	α_i(degrees)
1	θ_1	0.1	0.25	90
2	θ_2	0	0.35	0
3	θ_3	0	0.3	90
4	θ_4	0.2	0	0
5	θ_5	0	0	-90
6	θ_6	0.15	0	0

environmental changes. When introducing a multi-head self-attention mechanism, Transformer-XL can effectively capture the correlation between different states, ensuring that the decision-making process fully utilizes historical data.

Then a fusion layer is constructed to combine the state representation output by Transformer-XL with the Q-value function of DQL. In this fusion layer, a fully connected neural network is used to perform linear transformation on the output of Transformer-XL, mapping the generated high-dimensional state features to the Q-value space. This approach combines the long-term dependency capture capability of Transformer-XL with the decision optimization mechanism of DQL to improve the adaptability of intelligent agents to dynamic environmental changes.

For specific control tasks, the fused features are input into a DQL model for decision-making. The intelligent agent executes a collaborative control strategy by selecting the action corresponding to the maximum Q value. During the execution process, the Q-value is updated by combining real-time feedback data to continuously optimize the decision-making strategy of the mechanical arm. To ensure accuracy and flexibility in execution, the output of the predictive model is used as a reference for action selection, dynamically adjusting the priority of actions to ensure stability in complex scenarios.

4. METHODS EVALUATION

4.1 Experimental Simulation Platform

In this study, the experimental simulation platform was constructed using ROS (Robot Operating System) and the Gazebo simulation environment to support the simulation and performance evaluation of the mechanical arm. Based on the ROS architecture, the motion model and sensor model of the mechanical arm, including LiDAR and camera, were configured to obtain real-time information about the surrounding environment. The schematic diagram of the linkage of the 6-degree-of-freedom mechanical arm in the simulation experiment is shown in Figure 5.

Using Gazebo, multiple test scenarios were constructed, corresponding to static, low dynamic, and high dynamic environments. Different obstacles and target objects were set in each scene to simulate various complex situations that may be encountered in the real world. By adjusting environmental parameters, the speed, shape, and number of obstacles can be flexibly changed to comprehensively evaluate the adaptability of the mechanical arm. During the simulation process, the DQL algorithm (combined with Transformer-XL) was integrated into ROS to process sensor data in real-time and generate control instructions. The algorithm adjusts the action strategy of the mechanical arm under different environmental conditions through continuous iterative learning, optimizing the efficiency and accuracy of task execution.

The D–H (Denavit–Hartenberg) parameter of a mechanical arm was used to describe the geometric relationships of its joints and to compute the forward and inverse kinematics of the mechanical arm. Table 2 shows the D-H parameters of a 6-degree-of-freedom mechanical arm.

The D–H parameters in Table 2 provide a detailed description of the geometric relationship of each joint. Initial joint 1 describes the rotation of the mechanical arm base, with a linkage offset of $d_1 = 0.1$ meters, a linkage length of $a_1 = 0.25$ meters, and a torsion angle of 90 degrees, indicating that its rotation axis is perpendicular to the linkage. Similarly, there is no twisting at joints 2, 4, and 6, while joints

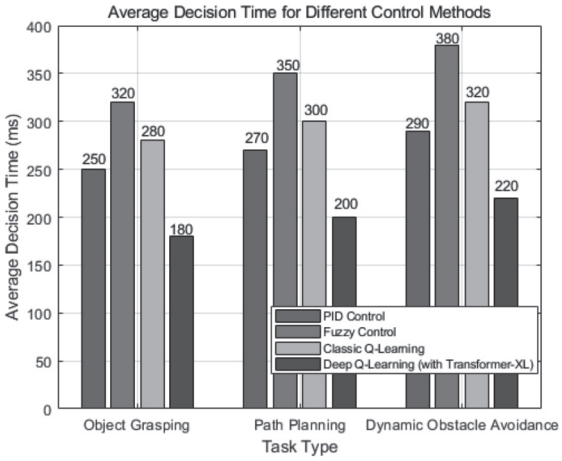

Figure 6 Comparison of average decision time for control methods.

3 and 5 can twist 90 degrees and −90 degrees respectively. These parameters can be used for kinematic analysis of the mechanical arm. Through these parameters, the pose of each joint of the mechanical arm relative to the previous joint can be calculated for later evaluation in the experiment.

4.2 Decision Time

To evaluate the decision-making time of the mechanical arm during task execution, including the time delay from receiving instructions to actual operation execution, this study adopts the following evaluation methods and indicators.

The measurement of decision time is achieved by recording the response process of the mechanical arm in different task scenarios. After receiving the operation command, the mechanical arm can immediately start a timer to record the starting time. The timing stops when the mechanical arm completes the command and starts executing the task, and the end time can be recorded.

In order to comprehensively evaluate the decision-making efficiency of the mechanical arm, traditional control methods were selected for comparison, including PID control, fuzzy control, and classical reinforcement learning algorithms. These methods represent different decision-making strategies.

PID control (proportional-integral-derivative control) regulates the motion of a mechanical arm through proportional (P), integral (I), and derivative (D) control. The decision time of the mechanical arm when executing instructions can be measured to determine its efficiency in rapid response tasks. Fuzzy control utilizes fuzzy logic for decision-making, with strong adaptability but relatively slow response speed. It can record the decision time of fuzzy control in the same task scenario and evaluate its delay compared to intelligent algorithms. The classic reinforcement learning algorithm records its decision time in the decision process based on state action value updates, in order to compare it with the method where DQL is combined with Transformer-XL.

When conducting evaluations, each control method must be applied under the same operating conditions to avoid external factors interfering with the results. This study set up multiple experimental scenarios, covering different task complexities and environmental dynamics, to ensure the comprehensiveness of the evaluation.

Statistical analysis was conducted on the decision time of different methods, and the advantages of collaborative control strategies in response speed was evaluated by comparing the average decision time with the standard deviation. Through this series of evaluations, the researchers ensured that the decision-making time of the mechanical arm in dynamic environments was scientifically and accurately measured, providing a data foundation for subsequent performance optimization.

Figure 6 shows the average decision time of different control methods in three specific tasks (object grasping, path planning, dynamic obstacle avoidance). The horizontal axis represents task type, and the vertical axis represents average decision time (unit: milliseconds). Each group of bars represents a control method, including PID control, fuzzy control, classical Q-learning, and DQL (combined with Transformer-XL). In object grasping tasks, the average decision time of PID control is 250 milliseconds,significantly faster than fuzzy control (320 milliseconds) and classical Q-learning (280 milliseconds). DQL (combined with Transformer-XL) is even more superior, requiring only 180 milliseconds, demonstrating its efficiency in simple tasks. In path planning tasks, the decision time of fuzzy control increases to 350 milliseconds, while the average decision time of DQL is 200 milliseconds. This change indicates that in situations requiring more complex decisions, the response time of fuzzy control is longer, while DQL still demonstrates higher efficiency. The decision time for dynamic obstacle avoidance tasks is more obvious, with PID control reaching 290 milliseconds and fuzzy control reaching up to 380 milliseconds. In contrast, the decision time of DQL is 220 milliseconds, demonstrating its excellent adaptability and decision-making ability in complex and dynamic environments. In summary, Figure 6 compares the performance differences of different control methods in various specific tasks, highlighting the advantages of DQL (combined with Transformer-XL) in various scenarios, indicating that this method can provide faster response speed and better decision-making ability in practical applications, and is suitable for use in complex dynamic environments.

Table 3 Comparison of operation success rates.

Control Method	Object Grasping Success Rate (%)	Path Planning Success Rate (%)	Dynamic Obstacle Avoidance Success Rate (%)
PID Control	85	80	75
Fuzzy Control	78	75	70
Classic Q-Learning	82	79	76
DQL (with Transformer-XL)	95	90	88

Table 4 Comparison of model memory abilities.

Model	30s Delay Success Rate (%)	60s Delay Success Rate (%)	90s Delay Success Rate (%)
PID Control	70	55	40
Fuzzy Control	65	50	35
Classic Q-Learning	75	60	45
DQL (with Transformer-XL)	90	75	60

4.3 Operation Success Rate

The evaluation of operational success rate is achieved through quantitative analysis of the performance of different control methods applied to specific tasks. The evaluation process involved PID control, fuzzy control, classical Q-learning, and DQL (combined with Transformer-XL), and experiments were conducted on the same three tasks (object grasping, path planning, and dynamic obstacle avoidance) for each method. Each task comprised 100 test cases to ensure the representativeness and reliability of the data.

The successfully completed task is one where the mechanical arm correctly executes the predetermined target. For this purpose, the number of successful cases for each control method is recorded for each experiment, and then the success rate of the operation is calculated.

Table 3 lists the success rates of four control methods applied to three different tasks. In object grasping tasks, the success rate of DQL (combined with Transformer-XL) is 95%, demonstrating high accuracy. In contrast, the success rate of fuzzy control is 78%, indicating that its performance in simple tasks is slightly inadequate. The success rate of classical Q-learning is 82%, which is similar to PID control but still slightly inferior. In the path planning task, DQL (combined with Transformer-XL) once again demonstrated a success rate of 90%, while fuzzy control decreased to 75%. The success rate of classical Q-learning has decreased to 79%, which is not much different from PID control. DQL (combined with Transformer-XL) has demonstrated its advantages in complex path computation in this task.

In dynamic obstacle avoidance tasks, the success rate of PID control drops to 75%, while fuzzy control is even lower at only 70%. The success rate of classical Q-learning is 76%, while DQL (combined with Transformer-XL) still performs at 88%. DQL achieved an 88% success rate in this complex task, demonstrating its excellent adaptability and decision-making efficiency in dynamic environments. Moreover, it has enormous potential in the collaborative operation of mechanical arms.

4.4 Model Memory Ability

The model memory ability can quantify the performance of DQL models in handling long-term dependencies. The model memory ability is evaluated by measuring the ability of a mechanical arm to accurately perform previously learned tasks even after a certain time delay.

In the experiment, firstly, the mechanical arm was trained for a specific task; this was followed by the introduction of a time delay (30 seconds, 60 seconds, and 90 seconds), and then the arm was required to perform the same task. This process simulates the time intervals caused by environmental changes in real scenes. It is possible to record the number of times that the mechanical arm successfully completes tasks during each delay period, and calculate the success rate. This indicator shows that the mechanical arm can still retain the previously learned task information after delay. By comparing different delay times, it is possible to determine how long the mechanical arm can maintain memory of the task, and analyze the advantages of different methods in terms of memory ability. The expected results show that DQL can maintain a high task success rate over longer delay times (such as a success rate of 60% after a 90 second delay), while traditional methods (such as PID control and fuzzy control) typically experience significant declines.

Table 4 lists the task success rates of four control models under different delay times. For the PID control model, the success rate is 70% with a 30-second delay, dropping to 55% with a 60-second delay, and further decreasing to 40% with a 90-second delay. This indicates that the model has relatively weak memory ability when dealing with long-term dependency tasks. Even the performance of fuzzy control is inferior, with a success rate of 65% at a delay of 30 seconds, dropping to 50% at 60 seconds, and 35% at 90 seconds. This indicates the inadequacy of fuzzy control in terms of memory retention ability, especially in the case of longer delays, where performance decreases significantly. The success rate of classical Q-learning is 75% at a 30 second delay, 60% at a 60 second delay, and 45% at a 90 second delay. Compared

to PID control and fuzzy control, classical Q-learning has certain advantages in terms of memory retention ability, but still has time-extension limitations.

DQL (combined with Transformer-XL) performs well under all latency times. Its success rate is as high as 90% under a 30-second delay, 75% under a 60-second delay, and still remains at 60% under a 90-second delay. This indicates that DQL has significantly better memory ability in long-term dependency tasks than other models, and can effectively retain previously learned information.

The data comparison showed that DQL (combined with Transformer-XL) has significant advantages in terms of memory ability and its adaptability in dynamic environments. Furthermore, it provides a solid foundation for the practical application of mechanical arm collaborative operation.

5. CONCLUSIONS

This article proposes a solution that combines Transformer-XL and DQL to address the issues of insufficient flexibility, slow response speed, and poor ability to handle long-term dependencies in traditional mechanical arm collaborative operations. In this study, the sensor data was normalized to improve the training effectiveness of the model. A deep learning model based on Transformer-XL was constructed to improve the capture of long-term dependencies, and combined with deep Q-learning algorithms to optimize the decision-making efficiency and adaptability of the robotic arm in complex dynamic environments. The experimental results show that, compared with traditional control methods, the control strategy proposed in this article exhibits significant advantages in various task environments. In object grasping, path planning, and dynamic obstacle avoidance tasks, the combination of DQL and Transformer-XL not only significantly improves the success rate of operations, but also effectively shortens decision time, especially in high dynamic environments. In addition, the adaptability assessment further confirms the reliability and flexibility of this method in complex environments. In summary, the research findings of this study provide an efficient intelligent control strategy for collaborative operation of mechanical arms, with broad application prospects and important references for the further development of intelligent manufacturing and automation systems.

AVAILABILITY OF DATA AND MATERIALS

No data were used to support this study.

REFERENCES

1. Ng N, Gaston P, Simpson P M, et al. Robotic arm-assisted versus manual total hip arthroplasty: a systematic review and meta-analysis. The Bone & Joint Journal, 2021, 103(6): 1009–1020. DOI: https://doi.org/10.1302/0301-620X.103B6.BJJ-2020-1856.R1
2. Zhang J, Ndou W S, Ng N, et al. Robotic-arm assisted total knee arthroplasty is associated with improved accuracy and patient reported outcomes: a systematic review and meta-analysis. Knee Surgery, Sports Traumatology, Arthroscopy, 2022, 30(8): 2677–2695. DOI: https://doi.org/10.1007/s00167-021-06464-4
3. Carron A, Arcari E, Wermelinger M, et al. Data-driven model predictive control for trajectory tracking with a robotic arm. IEEE Robotics and Automation Letters, 2019, 4(4): 3758–3765. DOI: 10.1109/LRA.2019.2929987
4. Batailler C, Fernandez A, Swan J, et al. MAKO CT-based robotic arm-assisted system is a reliable procedure for total knee arthroplasty: a systematic review. Knee Surgery, Sports Traumatology, Arthroscopy, 2021, 29(11): 3585–3598. DOI: https://doi.org/10.1007/s00167-020-06283-z
5. Hampp E L, Chughtai M, Scholl L Y, et al. Robotic-arm assisted total knee arthroplasty demonstrated greater accuracy and precision to plan compared with manual techniques. The journal of knee surgery, 2019, 32(03): 239–250. DOI: 10.1055/s-0038-1641729
6. Flesher S N, Downey J E, Weiss J M, et al. A brain-computer interface that evokes tactile sensations improves robotic arm control. Science, 2021, 372(6544): 831–836. DOI: 10.1126/science.abd0380
7. Jeong J H, Shim K H, Kim D J, et al. Brain-controlled robotic arm system based on multi-directional CNN-BiLSTM network using EEG signals. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2020, 28(5): 1226–1238. DOI: 10.1109/TNSRE.2020.2981659
8. Mahmoodabadi, M. J. (2020). Moving least squares approximation-based online control optimised by the team game algorithm for Duffing-Holmes chaotic problems. Cyber-Physical Systems, 7(2), 93–113.
9. Chen X, Huang X, Wang Y, et al. Combination of augmented reality-based brain-computer interface and computer vision for high-level control of a robotic arm. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2020, 28(12): 3140–3147. DOI: DOI: 10.1109/TNSRE.2020.3038209
10. Marchand R C, Sodhi N, Anis H K, et al. One-year patient outcomes for robotic-arm-assisted versus manual total knee arthroplasty. The journal of knee surgery, 2019, 32(11): 1063–1068. DOI: 10.1055/s-0039-1683977. DOI: 10.1055/s-0039-1683977
11. Mahoney O, Kinsey T, Sodhi N, et al. Improved component placement accuracy with robotic-arm assisted total knee arthroplasty. The journal of knee surgery, 2022, 35(03): 337–344. DOI: 10.1055/s-0040-1715571
12. Cio Y S L K, Raison M, Menard C L, et al. Proof of concept of an assistive robotic arm control using artificial stereovision and eye-tracking. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2019, 27(12): 2344–2352. DOI: 10.1109/TNSRE.2019.2950619
13. Clement N D, Gaston P, Bell A, et al. Robotic arm-assisted versus manual total hip arthroplasty: a propensity score matched cohort study. Bone & Joint Research, 2021, 10(1): 22–30. DOI:https://doi.org/10.1302/2046-3758.101.BJR-2020-0161.R1
14. Khlopas A, Sodhi N, Hozack W J, et al. Patient-reported functional and satisfaction outcomes after robotic-arm-assisted total knee arthroplasty: early results of a prospective multicenter investigation. The journal of knee surgery, 2020, 33(07): 685–690. DOI: 10.1055/s-0039-1684014
15. Kayani B, Konan S, Tahmassebi J, et al. An assessment of early functional rehabilitation and hospital discharge in conventional versus robotic-arm assisted unicompartmental knee arthroplasty: a prospective cohort study. The Bone & Joint Journal,

2019, 101(1): 24–33. DOI:https://doi.org/10.1302/0301-620X.101B1.BJJ-2018-0564.R2
16. Z. Ma, "Visual Servo Control of Robot Arm Based on Image Features", Engineering Intelligent Systems, vol. 31 no. 6, pp. 501–511, 2023.
17. Cool C L, Jacofsky D J, Seeger K A, et al. A 90-day episode-of-care cost analysis of robotic-arm assisted total knee arthroplasty. Journal of comparative effectiveness research, 2019, 8(5): 327–336. DOI: https://doi.org/10.2217/cer-2018-0136
18. Yan Yang, Jiangtao Han, Zhijie Liu, Zhijia Zhao, Keum-Shik Hong. Modeling and Adaptive Neural Network Control for a Soft Robotic Arm with Prescribed Motion Constraints. IEEE/CAA Journal of Automatica Sinica, 2023,10(2):501–511. DOI:10.1109/JAS.2023.123213
19. Nana Li, Yong Wang, Pengfei Shen, Shuangfei Li, Lin Zhou. A DoS Attacks Detection Algorithm Based on Snort-BASE for Robotic Arm Control Systems.Journal of Computer and Communications,2022,10(4):1–13. DOI:10.4236/jcc.2022.104001
20. Ning Tan, Peng Yu, Zhiyan Zhong, Fenglei Ni. A New Noise-Tolerant Dual-Neural-Network Scheme for Robust Kinematic Control of Robotic Arms With Unknown Models. IEEE/CAA Journal of Automatica Sinica,2022,9(10): 1778–1791. DOI: 10.1109/JAS.2022.105869
21. Zhu, Qixiang. Design of a PLC DC Motor Speed Regulation System Based on Fuzzy Control Algorithm. Engineering Intelligent Systems, v 32, n 2, p 127–137, March 2024

Wenzhi Zhang, Department of Mechanical and Electronic Engineering, Sichuan Vocational and Technical College of Communications, Chengdu 611130, Sichuan, China Email: 29445407@qq.com

Min Xia, College of Engineering and Technology, Nanchang Vocational University, Nanchang 33000, Jiangxi, Email: 13171373325@163.com

Biao Yang, male, Han nationality, born in August 1995, Bazhong, Sichuan, CPC member, postgraduate degree, master's degree in engineering, mold design and manufacturing (intermediate), enterprise human resource manager (senior), published three papers, presided over and participated in four projects, presided over and participated in invention patents, utility patents, 3, and guided the students to participate in vocational competitions, career planning contests, social practice, etc., and won nine awards.