




An Efficiently Convergent Deep Reinforcement Learning-Based Trajectory Planning Method for Manipulators in Dynamic Environments

Li Zheng¹ · YaHao Wang¹ · Run Yang¹ · Shaolei Wu² · Rui Guo³ · Erbao Dong¹ 

Received: 2 March 2022 / Accepted: 31 January 2023 / Published online: 27 March 2023
© The Author(s), under exclusive licence to Springer Nature B.V. 2023

Abstract

Recently, deep reinforcement learning (DRL)-based trajectory planning methods have been designed for manipulator trajectory planning, given their potential in solving the problem of multidimensional spatial trajectory planning. However, many DRL models that have been proposed for manipulators working in dynamic environments face difficulties in obtaining the optimal strategy, thereby preventing them from reaching convergence because of massive ineffective exploration and sparse rewards. In this paper, we solve the inefficient convergence problem at the two levels of the action selection strategy and reward functions. First, this paper designs a dynamic action selection strategy that has a high probability of providing positive samples in the pre-training period by using a variable guide item and effectively reduces invalid exploration. Second, this study proposes a combinatorial reward function that combines the artificial potential field method with a time-energy function, thereby greatly improving the efficiency and stability of DRL-based methods for manipulators trajectory planning in dynamic working environments. Extensive experiments are conducted using the *CoppeliaSim* simulation model with a freely moving obstacle and the 6-DOF manipulator. The results show that the proposed dynamic action selection strategy and combinatorial reward function can improve the convergence rate on the DDPG, TD3, and SAC DRL algorithms by up to 3-5 times. Furthermore, the mean value of the reward function increases by up to 1.47-2.70 times, and the standard deviation decreases by 27.56% to 56.60%.

Keywords Manipulator trajectory planning · Deep reinforcement learning · Autonomous navigation · Real-time obstacle avoidance · Dynamic action selection strategy · Combinatorial reward function

1 Introduction

In recent years, robot manipulators have become increasingly important in scientific research and engineering applications which have been used to measure the technological capacity of countries to some extent [1]. However, robot manipulators are confronted with increasingly complex environments and challenging tasks as their application becomes more widespread. For instance, robots are being used in highly dangerous tasks instead of humans and in some human-robot collaboration scenarios [2–5], which demand a greater accuracy in motion control in a multidimensional dynamic unstructured configuration space to

allow robots to work harmoniously with humans. Therefore, trajectory planning of robot manipulators is critical in completing a task.

A fundamental problem in robot manipulator motion control, trajectory planning aims to find a continuous trajectory without collision from the initial position to the target position in the configuration space. The result of trajectory planning can measure whether the robot manipulators complete the task well. Some researchers have performed trajectory planning of robot manipulators based on certain constraints, such as velocity and acceleration in the joint or Cartesian space, whereas others have performed trajectory planning based on sensor information [6–8]. Trajectory planning in the joint and Cartesian spaces generally computes trajectories via polynomial interpolation. While the time and complexity of calculating the trajectory in both these spaces increase dramatically, the accuracy of the calculated values decreases when trajectory planning is performed in a multidimensional configuration

✉ Erbao Dong
ebdong@ustc.edu.cn

Extended author information available on the last page of the article.

space for multi-joint robot manipulators [9–11]. Sensor-information-based trajectory planning usually employs an artificial potential field method and sampling-based algorithms, such as PRM and RRT, which tend to fall into the local minimal or cannot easily reach optimal convergence, thus preventing them from reaching optimal planning trajectories [12–15]. The above trajectory planning methods also have low intelligence, poor dynamic planning capabilities, and no self-learning capability, thereby restricting their application in unstructured environments.

With the development of artificial intelligence, deep reinforcement learning (DRL) has been applied in the trajectory planning of robot manipulators, thereby offering a new and viable solution in unstructured working environments [16–19]. The agent in DRL uses a “trial and error” mechanism to explore possible operations based on the current state of the robot manipulator, and then the manipulator completes the trajectory planning task in the environment that maximizes the accumulated reward. T. Zhao et al. [20] optimized redundancy analysis based on advanced neurodynamics in the manipulation space and proposed a dual-level planning approach based on low-level reinforcement learning in joint space for the grasping and placement tasks of manipulators. Fereidoun N. R. et al. [21] combined the artificial bee colony and reinforcement learning algorithms into a bee colony reinforcement algorithm for optimizing the trajectory of a three-DOF manipulator to reach a specific target. CZ Liu et al. [22] proposed a humanoid manipulator control scheme that combines deep reinforcement learning with the digital twin technology, which simulates the control and trajectory planning of a planar three-DOF manipulator and the 3D manipulator of a humanoid robot BHR via a twin synchronous control multitask training policy. YH Wu et al. [23] proposed a model-free reinforcement learning strategy for training a policy for online trajectory planning without establishing the dynamic and kinematic models of the space robot. S. Chen et al. [24] proposed a combined DPPO-DQN algorithm for the autonomous obstacle avoidance and navigation of manipulators for live working in a 2D flat space.

However, manipulator trajectory planning based on the DRL method still faces problems in reaching convergence and conducting efficient explorations, which are particularly pronounced when considering an unstructured work environment with dynamic and static obstacles. The core of this problem lies in dealing with numerous invalid explorations and sparse rewards, which always lead to failure ineffective convergence and in obtaining positive samples. At present, sparse reward functions that severely reduce algorithm efficiency are being used in most trajectory planning tasks.

To address these problems, the primary contributions of this paper consist of practical and theoretical aspects. From the theoretical aspect, an efficiently convergent DRL

algorithm is proposed with the following contributions to addressing the efficient convergence and rewarding sparsity issues:

- 1) This study proposes a dynamic action selection strategy, which selects exploration action consisting of a random action with interference noise and a numerical solution calculated by the inverse kinematic iterative method for the manipulator. The proportion of the two is adjusted by a variable scaling factor during the training process. This ensures effective explorations in a multidimensional space. The strategy also provides more informative positive samples in the early stages of training, avoids sparse rewards and can be applied to other robot models with better portability.
- 2) This study innovatively defines a combined reward function by combining the ideas of the artificial potential field method and the time-energy optimal function and demonstrates significant improvements in the convergence of DRL-based methods in dynamic working environments.

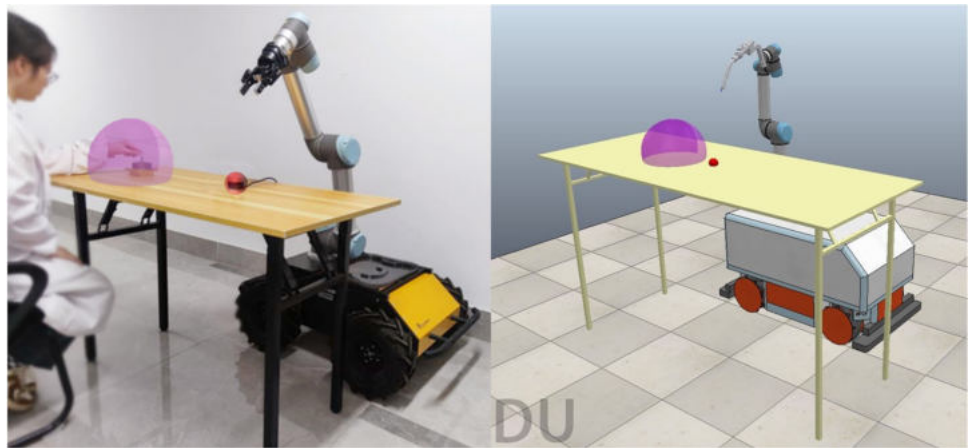
From the practical aspect, the proposed DRL algorithm is adopted to solve autonomous navigation and real-time obstacle avoidance of robot manipulator in dynamic obstacle environments. A simulation robot model close to real human-robot collaborative operation scenarios is set up on *CoppeliaSim*, and a co-simulation framework of *CoppeliaSim* and Python is designed. The efficient convergence of our proposed DRL algorithm in robot manipulator autonomous navigation and real-time obstacle avoidance tasks is validated and discussed in both extensive simulations and experiments.

This paper is organized as follows. Section 1 lists some conventional manipulator trajectory planning algorithms and DRL-based methods. Section 2 constructs a simulation robot model for a dynamic environment and establishes a co-simulation framework for *CoppeliaSim* and Python. Section 3 presents the theoretical foundations of reinforcement learning. Section 4 describes in detail the modified DRL-based trajectory planning method for robot manipulators. Section 5 presents the simulation experiments and results analysis. Section 6 concludes the paper and discusses some further works.

2 Robot Modeling in Dynamic Environment

2.1 *CoppeliaSim* Modeling

To simulate the real-time autonomous navigation and obstacle avoidance of robot manipulators in a dynamic environment with reference to a human-robot collaborative operation scenario, a simplified robot model was built in

Fig. 1 Robot modeling in dynamic environments

CoppeliaSim, as shown in Fig. 1. This model comprises two components, namely, a mobile robot with a manipulator and an operating platform. The manipulator is a UR5 manipulator that is fixed on the mobile robot with an end-effector at the end. The operation platform is fixed in front of the robot manipulator, where the target object (red ball) is stationary. The movable obstacle (purple ball) is made by enveloping the human arm and other parts of the body by simplifying the human-robot collaborative operation scenario. To improve the realism of the simulation environment and the robustness of the proposed algorithm, random Gaussian noise interference is added to the dynamic obstacle while moving. The relevant parameters of the model are shown in Table 1.

2.2 CoppeliaSim Simulator Configuration

CoppeliaSim has four simulation physics engines, namely, Bullet, ODE, Vortex, and Newton. The Bullet physics engine is more suitable for the simulation of the manipulator compared with the other three simulation physics engines because of its special properties, such as friction, damping, and collision detection [25, 26]. Therefore, this physics engine was employed for this study. Considering the speed and accuracy of the simulation, the simulation time step dt of the simulator was set to 0.05 s, and the joint mode of the 6 joints in *CoppeliaSim* was set to torque or force mode.

Table 1 Simulation robot model parameters

Parameter	Value
Number of manipulator joints	6
Time steps dt	0.05 s
Joints action bound	$[-2\pi, 2\pi]$ rad
Size of target	$r = 4$ cm
Size of dynamic obstacle	$R = 25$ cm
Velocity of dynamic obstacle	$35 \text{ cm/s} \pm \mathcal{N}(0, \sigma^2)$

The joints can be manipulated by force or torque, velocity, and position information to move in the torque or force mode, hence allowing DRL models to control and train the simulation robot model.

2.3 Python and CoppeliaSim Co-simulation

CoppeliaSim has six simulation modes, namely, Child Script, Lua Script, Plugin, Remote API client, ROS node, and Customized client/server. Given that this experiment aims to combine different deep reinforcement learning algorithms to train the simulation robot model. The Remote API client mode was used for the simulation and interacted with the Python client by calling the Python API in the Remote API client mode. The co-simulation framework of CoppeliaSim and Python is illustrated in Fig. 2.

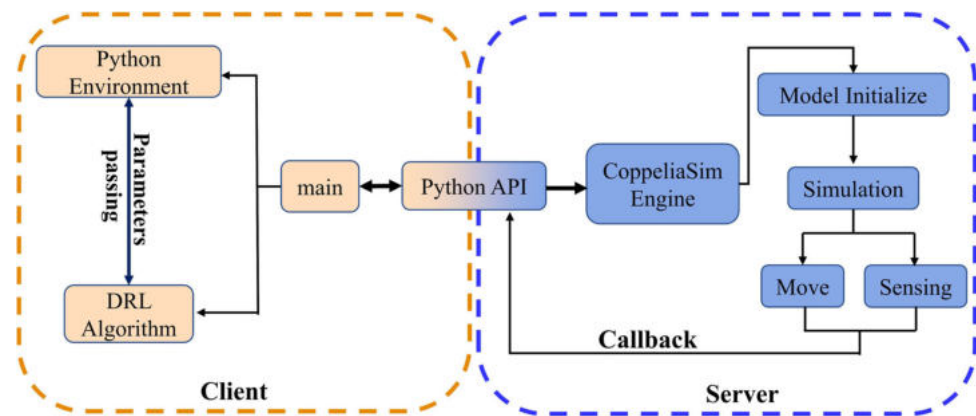
3 Basics on Deep Reinforcement Learning

3.1 Markov Decision Processes and Reinforcement Learning

The control problems addressed by RL generally can be formulated as Markov decision processes (MDPs) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma\}$ comprising a limited set of states, a limited set of actions, the probability of state transfer, reward functions, and a discount factor. At each moment, the agent will adopt the strategy $\pi(a_t | s_t)$ to select the corresponding action $a_t \in \mathcal{A}$ according to the current state $s_t \in \mathcal{S}$, and then interact with the environment to transit to the next state s_{t+1} with the probability $\mathcal{P}(s_{t+1} | s_t, a_t)$, and the agent obtains a reward r of this step. Given a strategy $\pi(a_t | s_t)$, the cumulative reward G_t can be calculated as $G_t = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k}$.

The objective of RL is to improve the strategy π by maximizing the sum G_t of the expected reward r_t for future steps. Nevertheless, the strategy π is stochastic, and the

Fig. 2 Co-simulation framework of *CoppeliaSim* and Python



cumulative reward G_t is a random variable, hence making it impossible to evaluate the value of state S_t . The Q-function or action-value function is defined as $Q^\pi(s_t, a_t) = \mathbb{E}[G_t | s_t, a_t]$. Let π^* denote an optimal policy i.e. any policy π^* s.t. $Q^{\pi^*}(s, a) > Q^\pi(s, a)$ for every $s \in \mathcal{S}$, $a \in \mathcal{A}$ and any policy π . All optimal policies have the same Q-function, which is called the optimal Q-function and denoted Q^* . It is easy to show that it satisfies the following Eq. 1 called the Bellman equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim p(\cdot | s, a)} [r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')] \quad (1)$$

3.2 Deep reinforcement learning

While the value function is subtle and well known, it still has many drawbacks when addressing complex problems. Specifically, the value function method evaluates certain states and actions, whereas the optimal value function cannot be solved effectively when a problem with a large or continuous action space is involved. To this end, Google DeepMind proposed the Actor-Critic algorithm framework that integrates value function and policy gradient methods into deep learning neural networks to output a specific action. The AC algorithm framework can be implemented for the task prediction of a continuous action, hence greatly improving its ability to solve multidimensional or highly complex problems.

The most representative and advanced AC algorithms include DDPG, TD3, and SAC. Lillicrap et al. [27] proposed the Deep Deterministic Policy Gradient (DDPG), which, unlike the AC algorithm, adds the *target C* and *target A* networks, and the experience replay technique from Deep Q Net [28, 29]. In this way, the stability and convergence of the AC algorithm can be improved by cutting off the correlation between the parameters when optimizing the Actor and the Critic networks by obtaining the parameters from a long

period. However, the Q value is potentially overestimated at each update in DDPG given that the policy network accomplishes parameter updates by maximizing the greedy target, and the overestimation bias will become increasingly obvious as the number of updates increases. Therefore, Twin Delayed Deep Deterministic Policy Gradient (TD3) [30] initially estimates the Q-value by using two networks and takes the relatively small Q-value as the target for updating. Meanwhile, the Gaussian noise $N(\mu, \sigma^2)$ is added to the prediction action of the *Target Actor* networks, and the *Target Actor* and *Target Critic* networks are updated by using the delayed soft update. And the Soft Actor-Critic (SAC) [31] takes a stochastic policy and obtains the optimal policy by optimizing the maximum entropy. Compared with the deterministic policy of DDPG, SAC has a stronger exploration ability generalizability, and robustness.

4 Manipulator trajectory planning algorithm based on deep reinforcement learning

This section initially determines the state and action spaces of the agent in the simulation environment. Afterward, a dynamic action selection strategy and a combinatorial reward function are established. The simulation robot model is eventually trained by adopting three deep reinforcement learning algorithms, namely, DDPG, TD3, and SAC. Figure 3 presents a scheme for training a manipulator with deep reinforcement learning.

4.1 State and action spaces

The state space S consists of the six joint angles $[q_1, q_2, q_3, q_4, q_5, q_6]$ of the UR5 manipulator, the spatial position $[x, y, z]$ of the end-effector, the Euclidean distance d_{target} from the end-effector to the target, and the minimum Euclidean distance $d_{\text{obstacle1}}$ and $d_{\text{obstacle2}}$ between each joint of the manipulator and the dynamic and static

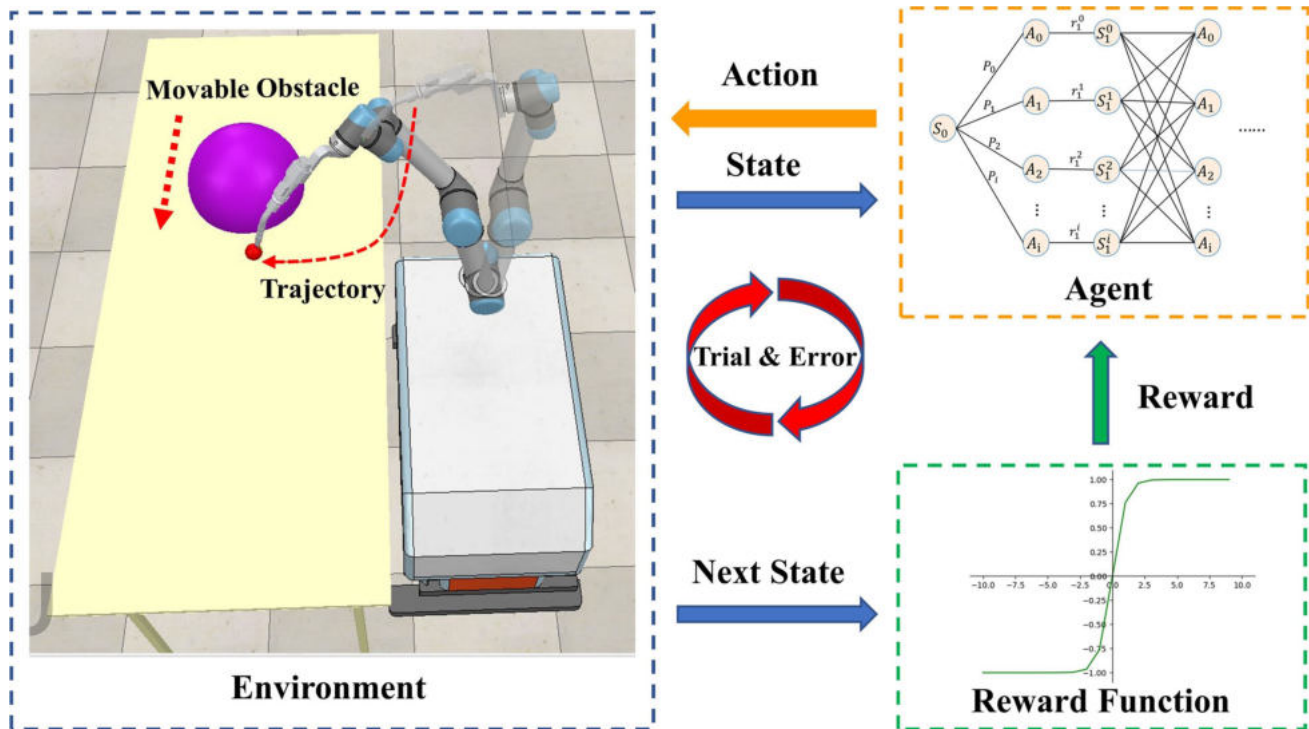


Fig. 3 DRL-based robot manipulator trajectory planning framework

obstacles, including the ground and the working platform, respectively.

$$S = [q_1, q_2, q_3, q_4, q_5, q_6, x, y, z, d_{\text{target}}, d_{\text{obstacle1}}, d_{\text{obstacle2}}] \quad (2)$$

Meanwhile, the action space A comprises the rotational velocity vector of each joint.

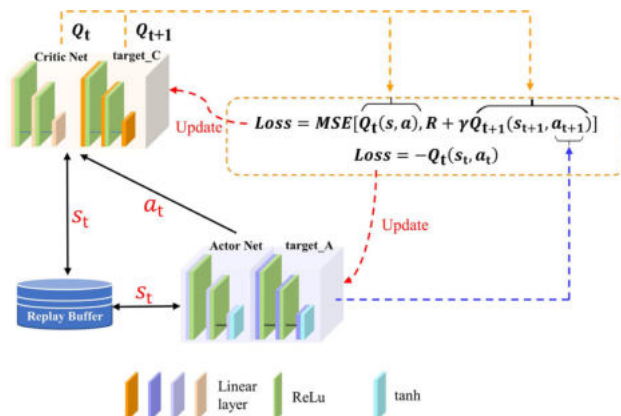
$$A = [\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6] \quad (3)$$

4.2 Deep Reinforcement Learning Algorithm Network

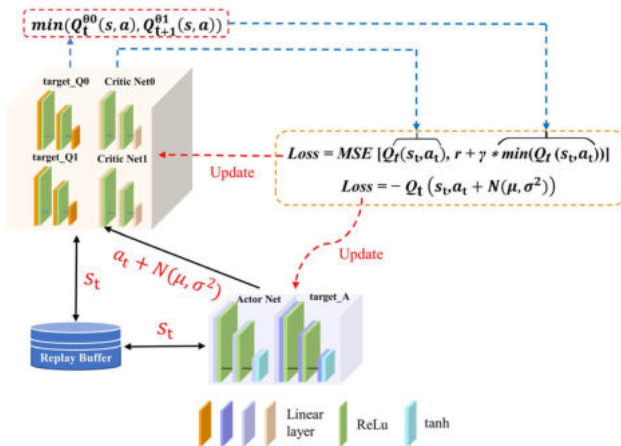
In this paper, the policy learning algorithms of the agent are DDPG, TD3, and SAC. The DDPG network includes the Critic and Actor networks, as well as *target C* and *target A* networks. These four networks have a similar structure, as shown in Fig. 4(a). Specifically, the Actor network and *target A* network comprise three linear layers, two ReLu activation layers, and one tanh activation layer, whereas the Critic and *target C* networks comprise three linear layers and two ReLu activation layers. All of these layers are connected in the form of a full connection layer. The inputs of the Actor network include the state space of the manipulator, which has 12 parameters, whereas those of the Critic network include the state and action spaces of the manipulator, which have 18 parameters. Meanwhile, the output of the proposed Action network is the rotational

velocity vector of each joint of the manipulator, which has 6 parameters, whereas that of the Critic networks is the state-action value function. In the hidden layers, the Action and Critic networks each use three layers. The activation functions of the hidden layers are the ReLu and *tanh* functions. When selecting the activation function of the output layer, if the ReLu function is used, then the output will either be a positive value or zero. Given that the control command of the manipulator requires a positive or negative value, the *tanh* function is selected as the activation function of the output layer of the Action network. This function has a range of $[-1, 1]$, which can prevent the output value of the Action network from being too large and the manipulator from moving too fast. In DDPG, the parameters of Actor networks are updated via minimizing the opposite of the state-action value, and the parameters of Critic networks are updated via minimizing the mean square error of the predicted state-action value and the actual state-action value.

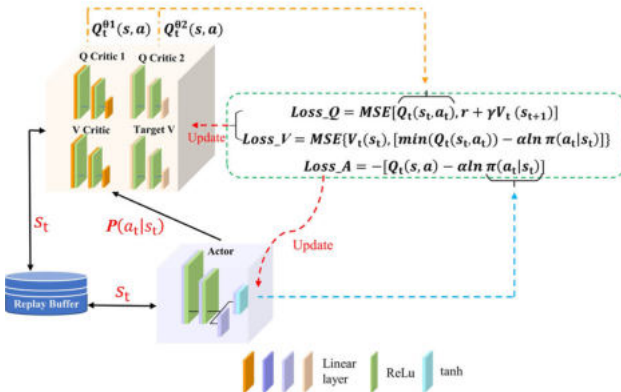
TD3 has more Critic and *target C* networks compared with DDPG as shown in Fig. 4(b). In TD3, the parameters of Actor networks are updated in the same way as DDPG, and the parameters of Critic networks are updated via minimizing the mean square error of the lower predicted state-action value and the actual state-action value. Meanwhile, the SAC algorithm network has only one Actor network, two *V Critic* networks, and two *Q Critic* networks. The outputs of the *V Critic* and *Q Critic*



(a) DDPG algorithm network framework



(b) TD3 algorithm network framework



(c) SAC algorithm network framework

Fig. 4 Diagram of the DRL algorithm network framework

networks are the state value function and state-action value function, respectively, as shown in Fig. 4(c). In SAC, the parameters of Actor networks are updated via minimizing the difference value of the state-action value and the entropy of action probabilities, the parameters of Q Critic networks are updated via minimizing the mean square error of the state-action value and the state value, and the V Critic

Table 2 Relevant parameters of the three deep reinforcement learning networks

Policy	Net	Num	Input	Output	Activation
DDPG	Action	2	State	Action	Relu,tanh
	Critic	2	State+Action	$Q(s, a)$	Relu
TD3	Action	2	State	Action	Relu,tanh
	Critic	4	State+Action	$Q(s, a)$	Relu
SAC	Action	1	State	Action	Relu
	V-Critic	2	State	$V(s, a)$	Relu
	Q-Critic	2	State+Action	$Q(s, a)$	Relu

networks are updated via minimizing the mean square error of the difference value of the state value and the entropy of action probabilities. The relevant parameters of the three deep reinforcement learning algorithm networks are shown in Table 2.

4.3 Dynamic Action Selection Strategy

The trajectory planning of a six-DOF manipulator is a multidimensional planning problem in itself, and this problem will become even more complicated when considering that the dynamic obstacle must be avoided in real-time. Therefore, if a random action selection strategy is used at the initial stage for training the agent, then this agent will easily perform massive invalid explorations and fall into the reward sparsity problem, which is difficult to converge. In this paper, a dynamic action selection strategy is proposed to solve the problem of invalid exploration. First, the current angle values of each joint of the manipulator and the spatial position of the target were obtained, and then the numerical solution $\vec{\Delta q}$ for the inverse kinematics of the manipulator was calculated by using the Jacobi matrix as a guide item to give the manipulator an appropriate action exploration direction. Afterward, the guide item was combined with the randomly sampled actions item with the exploration noise of the deep reinforcement learning algorithm to dynamically integrate the selected actions. The new action following this dynamic combination can be formulated as follows:

$$rate = \frac{Episode_now}{MAX_EPISODE} \quad (4)$$

$$\vec{Action_new} = rate \cdot \vec{action} + (1 - rate) \cdot \vec{\Delta q} \quad (5)$$

where $rate$ is the dynamic scale factor, $Episode_now$ is the current training episode, $MAX_EPISODE$ is the number of total training episodes, and \vec{action} is the random sampling action with exploration noise.

The manipulator can have a high probability of obtaining positive samples in the pre-training episodes by adjusting the dynamic scale factor, which can avoid invalid explorations and effectively solve the reward sparsity

problem in the multidimensional space. As the number of training episodes increases, the value of the scaling factor gradually grows, the proportion of the guide item $\vec{\Delta q}$ in the dynamic action progressively decreases, and the proportion of the randomly sampled action gradually increases, and the action selection of the manipulator is gradually randomized. With a sufficient number of positive samples at the early stage, the manipulator was trained to efficiently optimize convergence and learn how to effectively perform trajectory planning and real-time obstacle avoidance in a multidimensional space. Moreover, the dynamic action selection strategy has a strengthened migration capability and can be applied to different robot manipulator models because of its inverse solution using robot manipulator kinematics.

4.4 Reward Function Design

To solve the sparse reward problem of DRL-based methods for manipulator trajectory planning in dynamic environments, a combinatorial reward function that incorporates the ideas of the artificial potential field method and the time-energy function was proposed. The reward function is a scalar function comprising a four-term sum and is defined as follows: the target attraction reward function R_{target} comprising the Euclidean distance between the end-effector and the target, the obstacle rejection reward function R_{obstacle} comprising the Euclidean distance between the manipulator and dynamic or static obstacles (e.g., the ground and operating platform), the energy reward function R_{energy} comprising the magnitude of actions, and the time reward function R_{time} comprising the movement time of the manipulator. This function can be formulated as follows:

$$R = R_{\text{target}} + R_{\text{obstacle}} + R_{\text{energy}} + R_{\text{time}} \quad (6)$$

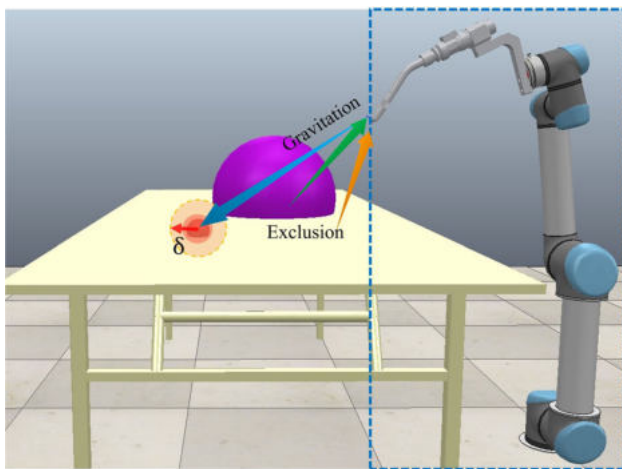


Fig. 5 Scheme of the target-obstacle reward function

To motivate the manipulator to approach the target as much as possible, the artificial potential field method was used to set the target as the gravitational potential, and the gravitational potential field reward function R_{target} was established along with the Euclidean distance between the end-effector and target, with the value of R_{target} being inversely proportional to that of d_{target} as shown in Fig. 5. The target attraction reward function was then constructed by means of the Huber-Loss function. The squared error was used when the d_{target} is less than δ , whereas the linear error was used when d_{target} is greater than δ . In this way, the robustness to some outlier action values can be improved, and the reward R_{target} was formulated as:

$$R_{\text{target}} = L_{\delta}(d_{\text{target}}) = \begin{cases} \frac{2}{d_{\text{target}}^2}, & \text{if } |d_{\text{target}}| < \delta \\ \frac{1}{\delta \cdot \left(|d_{\text{target}}| - \frac{1}{2}\delta\right)}, & \text{otherwise} \end{cases} \quad (7)$$

where d_{target} is the distance between the end-effector and target, and δ is the parameter of the Huber-Loss function that determines the smoothness.

The artificial potential field method was then adopted to help the robot manipulator achieve real-time obstacle avoidance by setting the obstacle as an exclusion potential. The excluded potential field reward function R_{obstacle} was then established based on the Euclidean distance between each joint of the manipulator and obstacles, and the value of R_{obstacle} was inversely proportional to the smaller value of the Euclidean distance $d_{\text{obstacle1}}$ from each joint of the manipulator to the dynamic obstacle and the Euclidean distance $d_{\text{obstacle2}}$ to static obstacles, as shown in Fig. 6. The obstacle rejection reward function R_{obstacle} can be described as:

$$d_{\text{obstacle}} = \min(d_{\text{obstacle1}}, d_{\text{obstacle2}}) \quad (8)$$

$$R_{\text{obstacle}} = - \left(\frac{d_{\text{ref1}}}{d_{\text{obstacle}} + d_{\text{ref1}}} \right)^p \quad (9)$$

where d_{ref1} is a constant parameter such that $-1 < R_{\text{obstacle}} < 0$, and p is the decay index of the reward.

The robot manipulator accomplishes the operation tasks while consuming the least amount of energy and in the shortest possible time, thereby making the optimized trajectory of the manipulator highly realistic. Therefore, the energy reward function R_{energy} and the time reward function R_{time} are designed based on the principle of the energy-time function. The joints of the manipulator are encouraged to move with a small amplitude not only to reduce the energy consumed by large movements but also

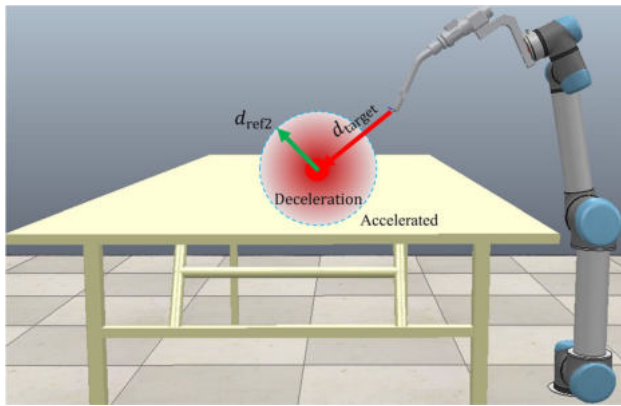


Fig. 6 Scheme of the time-energy reward function

to avoid instability and danger. Therefore, the value of reward R_{energy} is negatively correlated with the 2-norm of the amplitude of the action of each joint of the manipulator. Moreover, when the Euclidean distance between the end-effector and target is large, the magnitude of reward R_{time} is positively correlated with the amplitude of the motion angle of each joint. When the Euclidean distance between the end-effector and target is small, the magnitude of reward R_{time} is negatively correlated with the amplitude of the action of each joint, thereby improving the timeliness and stability of the robot manipulator trajectories as shown in Fig. 6. Rewards R_{energy} and R_{time} are formulated as:

$$R_{\text{energy}} = -\tanh(\|A\|_2) \quad (10)$$

$$R_{\text{time}} = \begin{cases} e^{-\|A\|_2} - 1, & d_{\text{target}} < d_{\text{ref2}} \\ \frac{1}{1 + e^{\|A\|_2}}, & d_{\text{target}} \geq d_{\text{ref2}} \end{cases} \quad (11)$$

where A is the rotational velocity vector of the six joints of the manipulator, and d_{ref2} is a constant parameter.

4.5 Hyperparameter

In machine learning, hyperparameters are parameters whose values are set optimally before the learning process, and choosing an optimal set of hyperparameters can improve the performance and effectiveness of learning. Therefore, the parameters associated with the agent and the exploration of algorithmic strategies during the training process are defined in Table 3.

5 Simulation and Experiment

5.1 Program Process and Simulation Testing

All simulation training processes for the entire model were run on a computer with an 8-core Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz, 16.0 GB RAM, and an NVIDIA GeForce RTX 3060 GPU. The program process of the manipulator for autonomous navigation target grasping and dynamic obstacle avoidance is shown in Fig. 7.

The simulation was run for 5000 episodes, with each episode having 300 steps and consuming approximately 18

Table 3 Hyperparameters of the DRL algorithm

Hyperparameter	Value	Description
MAX EPISODES	5000	Number of training episodes
MAX STEPS	300	Maximum number of steps per episode
Batch size	256	Number of samples in one training of the network
Action dim	6	Action space volume
State dim	12	State space volume
τ	0.005	Target network update rate
γ	0.99	Discount factor
η	0.0004	Learning rate
f	2	Frequency of policy delay updates
μ	1.0	Gaussian explore noise
κ	0.01	Noise decay factor
α	0.9	Reward coefficient of entropy
p	0.5	Reward function decay index
δ	8	Smoothness parameter of the Huber-Loss function
σ	2	Random Gaussian noise for dynamic obstacle
d_{ref1}	15	Constant parameter
d_{ref2}	5	

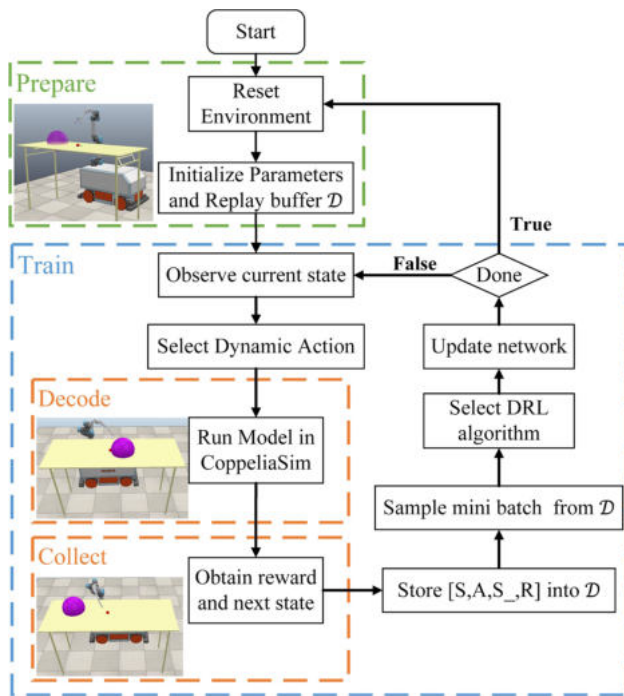


Fig. 7 The algorithm process diagram of manipulator DRL-based trajectory planning method for the manipulator

hours. First, each episode opens the simulation robot model in *CoppeliaSim* and runs the simulation while initializing the relevant hyperparameters and the replay buffer \mathcal{D} . Second, each episode enters the main function loop. The state space of a new episode is acquired, an action is then selected following the dynamic action selection strategy, the action state in the simulation environment is decoded afterward, and the current episode reward and the next

state are eventually acquired. The optimal action strategy is calculated via consecutive loop iteration. The manipulator is then controlled by the algorithm program to complete the target grasping and dynamic obstacle avoidance tasks. The overall process is summarized in Algorithm 1.

By observing and comparing the learning situation of the manipulator during the simulation training, Fig. 8 shows that in the first 50 episodes, the manipulator has just started learning and is not yet able to grasp the target and avoid dynamic and static obstacles. As the number of training episodes reaches approximately 5000 the manipulator learns how to grasp the target while avoiding dynamic and static obstacles in real time. When the manipulator approaches the target, the Euclidean distance between the end-effector and the target gradually decreases until reaching the target and maintains its position. When the obstacles are close to the manipulator, the manipulator adjusts its position relative to the target and moves farther away to ensure that the end-effector and the joints of the manipulator do not touch these obstacles. When the obstacles are located far away at a safe Euclidean distance, the manipulator then proceeds with the grasping task.

5.2 Simulation Results Analysis

Six groups of experiments were conducted to test the performance of the proposed dynamic action selection strategy and combinatorial reward function. The performance of these methods was evaluated using three indicators, namely, convergence rate, success rate, and mean reward value. The maximal reward in all experiments was set to 200. When the reward exceeds 0, the trajectory planning task is considered completed.

5.2.1 Dynamic Action Selection Strategy

To verify the effect of the action selection strategy on robot manipulator training, the random action selection strategy and the dynamic action selection strategy were set comparatively to train the robot manipulator, and three groups of experiments were set up to train by using the three deep reinforcement learning algorithms, namely, DDPG, TD3, and SAC. After 5000 episodes of training, the success rates of different action selection strategies and the average rewards of each episode were separately computed, as summarized in Table 4. The changing processes of the rewards in the training for each DRL algorithm are displayed in Fig. 9.

As shown in Table 4, the success rate and average reward of SAC were higher than those of DDPG and TD3. The success rate of the tasks for the three DRL algorithms trained using the random action selection strategy were only 1.6%, 4.04%, and 10.6% over 5000

Require: Environmental status S ;

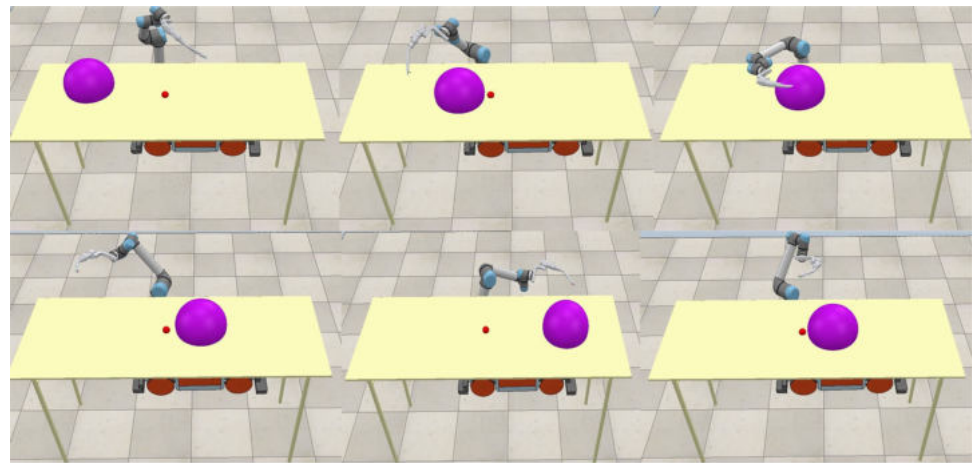
Ensure: Action

```

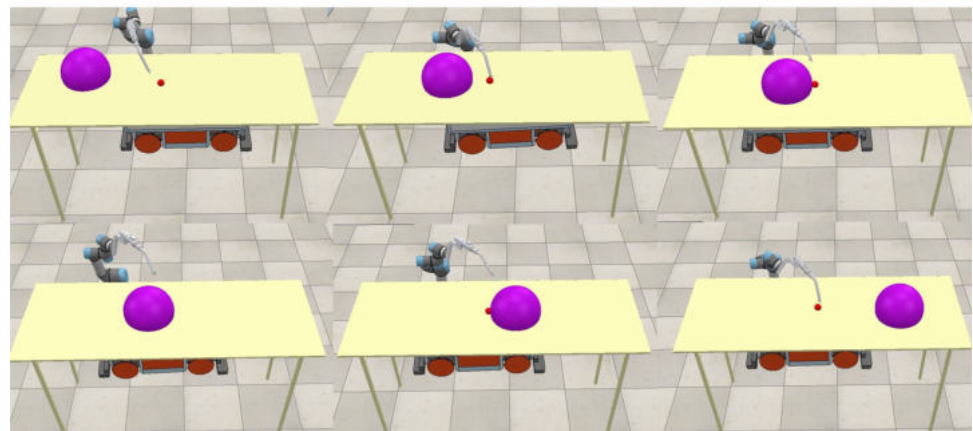
1: Initialize Networks parameters and Replay buffer  $\mathcal{D}$ ;
2: for current episode = 1 to MAX EPISODE do
3:   for current step = 1 to MAX STEP do
4:      $a_t \leftarrow a_t \cdot rate + \Delta q \cdot (1 - rate)$ 
5:      $s_{t+1} \leftarrow p(s_{t+1} | s_t, a_t)$ 
6:     Compute  $R_{target}$ ,  $R_{obstacle}$ ,  $R_{energy}$ ,  $R_{time}$ 
7:      $R = R_{target} + R_{obstacle} + R_{energy} + R_{time}$ 
8:      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R, s_{t+1})\}$ 
9:     if  $\mathcal{D}$  size > batch size then
10:      Select DRL Algorithm
11:      Update weigh of Networks
12:     end if
13:   end for
14: end for

```

Algorithm 1 Trajectory planning algorithm with dynamic action and reward function.

Fig. 8 Manipulator simulation training

(a) First 50 episodes



(b) Last 50 episodes

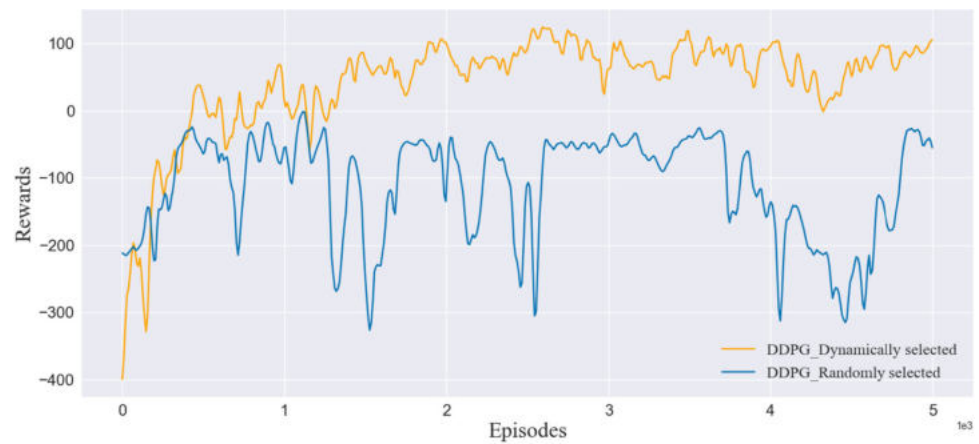
episodes, respectively. Meanwhile, the success rates of the tasks for the three DRL algorithms trained using the dynamic action selection strategy within 5000 episodes increased by 72.72%, 81.38%, and 82.62%, respectively. For the average reward, the three DRL algorithms trained using the dynamic action selection strategy improved by 1.47–2.70 times within 5000 episodes compared with those trained using the random action selection strategy.

Table 4 Results with different action selection strategies

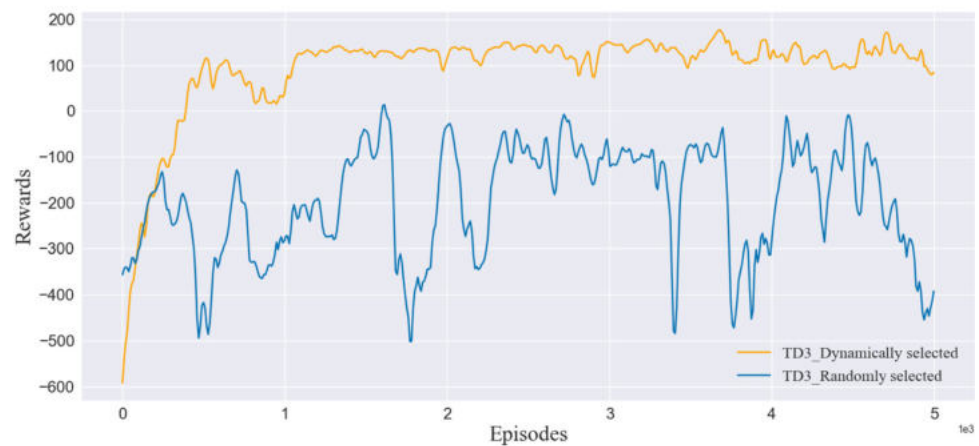
Policy	Action Select	Success Rate	Average Reward
DDPG	<i>Random</i>	1.60%	−111.25
	<i>Dynamic</i>	74.32%	80.70
TD3	<i>Random</i>	4.04%	−197.42
	<i>Dynamic</i>	85.42%	93.30
SAC	<i>Random</i>	10.60%	−78.26
	<i>Dynamic</i>	93.22%	133.16

As shown in Fig. 9, training with the random action selection strategy does not converge well on all three DRL algorithms, indicating that the manipulator cannot easily learn how to grasp and avoid dynamic obstacle in 5000 episodes. The dynamic action selection strategy selected for training started to converge approximately 3000, 1500, and 1000 episodes on the three DDPG, TD3, and SAC algorithms, respectively, and the maximum value of the accumulated rewards also stabilized. When faced with the trajectory planning problem of a six-DOF robot manipulator, the random action selection strategy causes massive invalid explorations at the early stage of the training, thereby resulting in sparse rewards and difficult convergence. As a consequence, the reward function curve greatly fluctuates and does not easily converge. In contrast, with the dynamic action selection strategy, the selected action in each step of the training episode is dynamically composed of a guide item and random action, which provides certain guidance to the agent at the early stage and reduces the invalid exploration problem. Meanwhile,

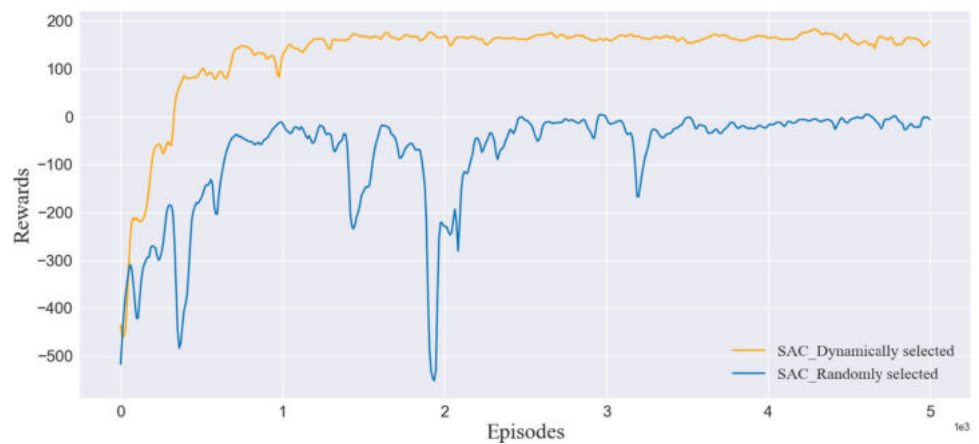
Fig. 9 Diagram of the reward convergence process with the different action selection strategies in the training



(a) DDPG



(b) TD3



(c) SAC

the action selection becomes randomized at the later stage, which can improve the robustness of the trajectory planning of the robot manipulator and contribute to autonomous navigation and real-time obstacle avoidance.

5.2.2 Combinatorial Reward Function

To verify the feasibility and generalizability of the previously proposed reward functions R_{target} and R_{obstacle}

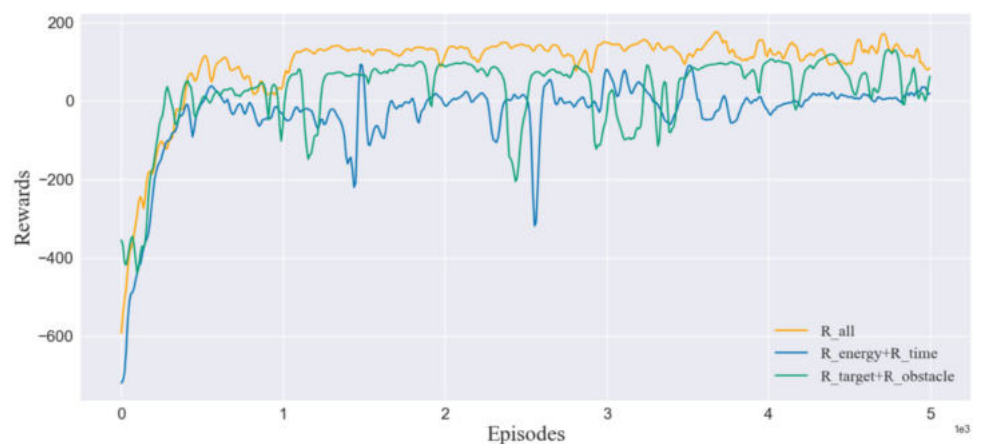
based on the artificial potential field method and the reward functions R_{energy} and R_{time} based on the idea of the time-energy function, three groups of comparison experiments were designed. The first group of experiments selected only the reward functions R_{target} and R_{obstacle} , the second group selected only the reward functions R_{energy} and R_{time} , and the

third group selected the sum of the four reward functions shown in Eq. 6. All three groups of experiments were trained by using the three deep reinforcement learning algorithms DDPG, TD3, and SAC, and the reward values of the training were obtained as shown in Fig. 10. After all groups of experiments converged, the success rate of different reward

Fig. 10 Diagram of the reward convergence process with the different reward functions for training in three DRL algorithms



(a) DDPG



(b) TD3



(c) SAC

Table 5 Results with the different reward functions

Policy	Reward Function	Success Rate	Average Reward	Standard Deviation
DDPG	$R_{\text{target}} + R_{\text{obstacle}}$	68.82%	53.92	66.23
	$R_{\text{energy}} + R_{\text{time}}$	17.58%	19.35	40.38
	R_{all}	74.32%	80.70	29.25
TD3	$R_{\text{target}} + R_{\text{obstacle}}$	76.86%	51.69	85.21
	$R_{\text{energy}} + R_{\text{time}}$	51.28%	33.02	65.00
	R_{all}	85.42%	93.30	36.98
SAC	$R_{\text{target}} + R_{\text{obstacle}}$	83.86%	87.14	41.45
	$R_{\text{energy}} + R_{\text{time}}$	73.76%	67.15	31.21
	R_{all}	93.22%	133.16	18.02

functions throughout the training process, the average reward of each episode, and the standard deviation of the reward were computed separately as summarized in Table 5. The changing process of the rewards in the training for each DRL algorithm is displayed in Fig. 10.

As shown in Table 5, the success rate and average reward of the tasks with the combinatorial reward function on the three DRL algorithms were higher than the target attraction and obstacle rejection reward functions and the time-energy reward function. Compared with adopting only the reward functions R_{target} and R_{obstacle} , the success percentages in 5000 episodes increased by 5.50%, 8.56%, and 9.36% for the three DDPG, TD3, and SAC algorithms, respectively, whereas the average rewards improved by 49.7%, 80.5%, and 52.8%, respectively. For the standard deviation, the combinatorial reward function group decreased by 27.5 to 56.60% compared with the previous two groups.

As shown in Fig. 10, during the training process, the group of target attraction and obstacle rejection reward functions rapidly converged on the three DRL algorithms yet greatly fluctuated after convergence. Meanwhile, the group of time-energy reward functions converged slowly and fluctuated greatly on the three DRL algorithms yet converged stably afterward. The combinatorial reward function group performed the best, thereby suggesting that the convergence speed and stability of the three DRL algorithms were better in this experiment than in the other two groups because the target attraction reward function R_{target} and obstacle rejection reward function R_{obstacle} guide the robot manipulator to navigate autonomously and avoid obstacles in real time under the strong constraint of target distance and obstacle distance, which are highly directional to complete the task. However, the stability was poor because the distance between the dynamic obstacle and robot manipulator changed constantly, thereby disturbing the robot manipulator trajectory planning. The time-energy reward functions R_{energy} and R_{time} were constrained with the movement amplitude of each joint, and given that

the action selection of the robot manipulator has some exploratory randomness, the manipulator greatly fluctuated at the early stage yet learned to be cautious and became stable after reaching convergence due to continuous training and learning. The combinatorial reward function, which combines the respective advantages of the target attraction and obstacle rejection reward functions and the time-energy reward functions, enables the robot manipulator to quickly and stably accomplish the tasks of autonomous navigation and real-time obstacle avoidance.

5.3 Experimental Evaluation and Validation

The simulation has demonstrated the convergence of the proposed algorithm and the robustness against different noise. Guided by the selection of hyper-parameters for the proposed algorithm and the networks designed in the simulation environment, the proposed dynamic action selection strategy and the combined reward function are evaluated and verified in a realistic dynamic obstacle environment.

5.3.1 Experiment Setup

In our experiments, the autonomous navigation and dynamic obstacle avoidance tasks are executed by a 6-DOF UR5 robot manipulator, and the distances between the robot and the target, dynamic obstacle and static obstacles during obstacle avoidance are detected by a 3D Motion Capture System *Qualisys*. The architecture of the dynamic obstacle environment platform is shown in Fig. 11. The target object is fixed on the experimental table in front of the robot manipulator, and the human arm and body constitute the dynamic obstacle with a simplified envelope as a sphere, which can move randomly and freely. The objective of the robot manipulator task is to approach the target autonomously and avoid static and dynamic obstacles in real-time.

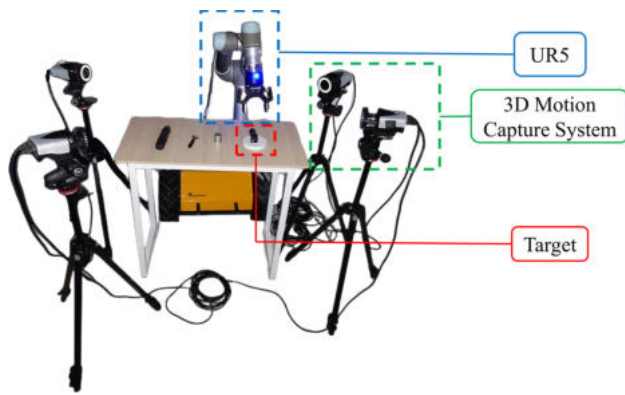


Fig. 11 Experimental environment platform

5.3.2 Experiment Analysis

To evaluate and validate the effectiveness and stability of the proposed dynamic action selection strategy and the combined reward function, we executed the target navigation and obstacle avoidance tasks 50 times using the three deep reinforcement learning algorithms, DDPG, TD3 and SAC, to test their success rates. In each experiment, the robot manipulator performs for 250 episodes, and the networks are trained for 200 steps in each episode. To improve the safety and robustness of the algorithm,

exploration action noise is added in the first 50 episodes and then the parameter soft update noise is injected into the parameters of the actor network in the remaining episodes. One of the episodes of interaction between the robot manipulator and the dynamic obstacle is shown in Fig. 12.

The reward values obtained from the three sets of experiments are shown in Fig. 13. The success rate, the average reward for each set, and the reward and its standard deviation were calculated separately for each set of experiments, as shown in Table 6. With the same initial position and training process, the proposed dynamic action selection strategy and the combined reward function achieve success rates of 72.85%, 83.28%, and 91.60%, respectively.

The stability and robustness of the proposed algorithm are verified through the above experiments. Compared to traditional trajectory planning algorithms such as RRT and PRM, the proposed learning-based algorithm does not require a complex dynamics model, which not only improves the success rate of dynamic obstacle avoidance, but also provides greater intelligence and transferability. In addition, the proposed dynamic action selection strategy and the combined reward function can be adapted with various DRL algorithms, which can also be applied to many practical applications beyond robot target navigation and autonomous obstacle avoidance tasks.

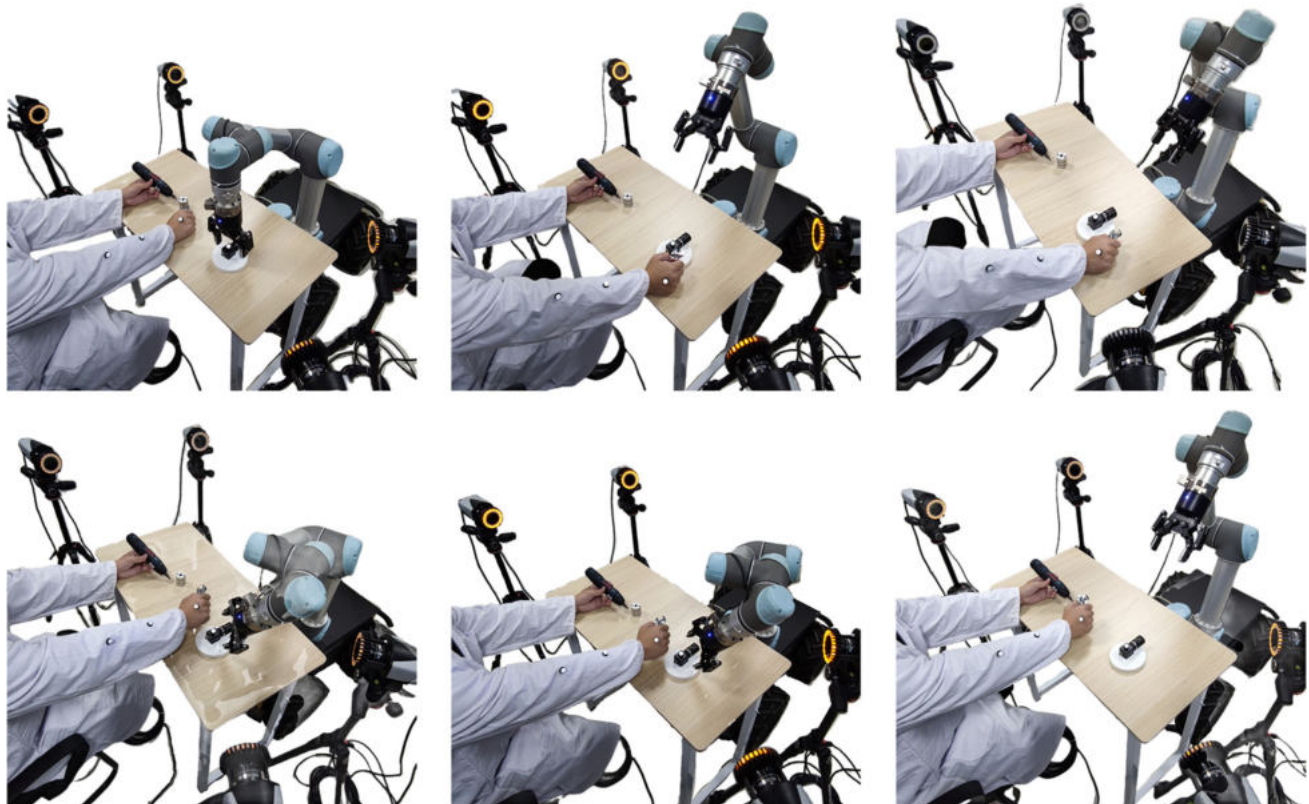
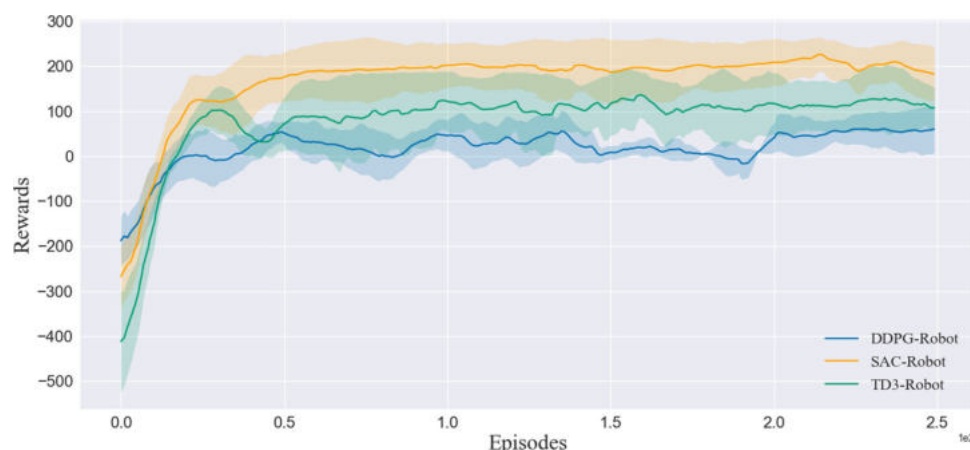


Fig. 12 Experimental results of robot manipulator for autonomous navigation and dynamic obstacle avoidance task

Fig. 13 Diagram of the reward convergence process with the dynamic action selection strategy and the combinatorial reward function for trains in three DRL policies



6 Conclusion

In this paper, an efficiently convergent deep reinforcement learning-based trajectory planning method was proposed for six-DOF manipulators in dynamic environments. This method effectively provides a fast convergence effect for DRL-based and was applied in autonomous navigation and real-time obstacle avoidance tasks. In addition, this method can avoid complex dynamics modeling and time-consuming parameter optimization processes by trial and error. The main algorithmic contributions of this paper are twofold. First, a dynamic action selection strategy with variable factors was proposed, which enables the agent to obtain more positive samples to avoid invalid explorations in the pre-training period. Second, a combinatorial reward function was proposed by combining the ideas of the artificial potential field method and the time-energy function, which can significantly accelerate convergence and improve stability after convergence. For the purpose of experimental verification, a simulation robot model simplified by the human-robot collaborative operation scenario was constructed using *CoppeliaSim*, and a co-simulation framework was built for simulation training. Additionally, we performed experimental verification on the real robot manipulator. Multiple experiments on three DRL algorithms were performed for comparison, and the results show that the proposed dynamic action selection strategy and combinatorial reward function can greatly improve the

convergence rate by 3-5 times and increase the success rate by 72.72%-82.62%.

The proposed method has a significant effect in improving the learning rate of the manipulator. However, its generalization ability is reduced due to the specificity of the model scenario. In the future, we will further optimize the action selection strategy and explore the mechanisms of reward construction to improve the generalizability of the DRL-based trajectory planning method. We have performed experimental validation on the real robot manipulator and plan to apply the DRL method in a human-robot collaborative operation scenario with multiple target objects.

Acknowledgements This work was supported by the National Key R&D Program of China (Grant number:2018YFB1307400).

Author Contributions All authors contributed to the study conception and design. Conceptualization, methodology, software, data curation, writing-original draft preparation, investigation were performed by Li Zheng. Data curation was performed by YaHao Wang, Run Yang, Shaolei Wu, Rui Guo and Erbao Dong. Supervision, writing, reviewing and editing were performed by Erbao Dong. All authors read and approved the final manuscript.

Funding This work was supported by the National Key R&D Program of China (Grant numbers[2018YFB1307400]).

Availability of Code and Data The codes and datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for Publication The authors affirm that human research participants provided informed consent for publication of the images in Fig. 1.

Competing interests All authors disclosed no relevant financial or nonfinancial interests to disclose.

Table 6 Results with different DRL policies

Policy	Action Select	Reward Function	Success Rate	Average Reward
DDPG	<i>Dynamic</i>	R_{all}	72.85%	52.34 ± 5.56
TD3	<i>Dynamic</i>	R_{all}	83.28%	91.72 ± 8.98
SAC	<i>Dynamic</i>	R_{all}	91.60%	142.17 ± 7.82

References

1. Brogårdh, T.: Present and future robot control development—an industrial perspective. *Annu. Rev. Control.* **31**(1), 69–79 (2007)
2. Wonsick, M., Long, P., Önl, A.Ö., Wang, M., Padir, T.: A holistic approach to human-supervised humanoid robot operations in extreme environments. *Front. Robot. and AI* **8**, 148 (2021)
3. Gonçalves R.S., Carvalho, J.C.M.: Review and latest trends in mobile robots used on power transmission lines. *Int. J. Adv. Robot. Syst.* **10**(12), 408 (2013)
4. Mgbemena, E.: Man-machine systems : a review of current trends and applications. *FUPRE J. Sci Ind. Res. (FJSIR)* **4**(2), 91–117 (2020)
5. Robla-Gómez, S., Becerra, V.M., Llata, J.R., Gonzalez-Sarabia, E., Torre-Ferrero, C., Perez-Oria, J.: Working together : a review on safe human-robot collaboration in industrial environments. *IEEE Access* **5**, 26754–26773 (2017)
6. Ata, A.A.: Optimal trajectory planning of manipulators : a review. *J. Eng. Sci. Technol.* **2**(1), 32–54 (2007)
7. Wang, T., Wang, W., Wei, F.: An overview of control strategy and trajectory planning of visual servoing. In: *Recent Featured Applications of Artificial Intelligence Methods. LSMS 2020 and ICSEE 2020 Workshops*, pp. 358–370. Springer (2020)
8. Gasparetto, A., Boscaroli, P., Lanzutti, A., Vidoni, R.: Path planning and trajectory planning algorithms: a general overview. *Motion Oper. Plan. Robot. Syst.* 3–27 (2015)
9. Guan, Y., Yokoi, K., Stasse, O., Kheddar, A.: On robotic trajectory planning using polynomial interpolations. In: *2005 IEEE International Conference on Robotics and Biomimetics-ROBIO*, pp. 111–116. IEEE (2005)
10. Fang, S., Ma, X., Zhao, Y., Zhang, Q., Li, Y.: Trajectory planning for seven-dof robotic arm based on quintic polynomial. In: *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2, pp. 198–201. IEEE (2019)
11. Wang, H., Wang, H., Huang, J., Zhao, B., Quan, L.: Smooth point-to-point trajectory planning for industrial robots with kinematical constraints based on high-order polynomial curve. *Mech. Mach. Theory* **139**, 284–293 (2019)
12. Guldner, J.R., Utkin, V.I., Hashimoto H.: Robot obstacle avoidance in n-dimensional space using planar harmonic artificial potential fields (1997)
13. Guernane, R., Belhocine, M.: A smoothing strategy for prm paths application to six-axes motoman sv3x manipulator. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4155–4160. IEEE (2005)
14. Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.P.: Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.* **17**(5), 1105–1118 (2009)
15. Sepehri, A., Moghaddam, A.M.: A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields. *IEEE Access* **9**, 26059–26070 (2021)
16. Qureshi, A.H., Nakamura, Y., Yoshikawa, Y., Ishiguro, H.: Robot gains social intelligence through multimodal deep reinforcement learning. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 745–751. IEEE (2016)
17. Kahn, G., Villafior, A., Ding, B., Abbeel, P., Levine, S.: Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5129–5136. IEEE (2018)
18. Tai, L., Paolo, G., Liu, M.: Virtual-to-real deep reinforcement learning : Continuous control of mobile robots for mapless navigation. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36. IEEE (2017)
19. Chen, X., Ghadirzadeh, A., Folkesson, J., Björkman, M., Jensfelt, P.: Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3110–3116. IEEE (2018)
20. Zhao, T., Deng, M., Li, Z., Hu, Y.: Cooperative manipulation for a mobile dual-arm robot using sequences of dynamic movement primitives. *IEEE Trans. Cogn. Dev. Syst.* **12**(1), 18–29 (2018)
21. Rahatabad, F.N., Rangraz, P.: Combination of reinforcement learning and bee algorithm for controlling two-link arm with six muscle: simplified human arm model in the horizontal plane. *Phys. Eng. Sci. Med.* **43**(1), 135–142 (2020)
22. Liu, C., Gao, J., Bi, Y., Shi, X., Tian, D.: A multitasking-oriented robot arm motion planning scheme based on deep reinforcement learning and twin synchro-control. *Sensors* **20**(12), 3515 (2020)
23. Wu, Y.-H., Yu, Z.-C., Li, C.-Y., He, M.-J., Hua, B., Chen, Z.-M.: Reinforcement learning in dual-arm trajectory planning for a free-floating space robot. *Aerosp. Sci. Technol.* **98**, 105657 (2020)
24. Chen, S., Yan, D., Zhang, Y., Tan, Y., Wang, W.: Live working manipulator control model based on dppo-dqn combined algorithm. In: *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, pp. 2620–2624. IEEE (2019)
25. Rohmer, E., Singh, S.P., Freese, M.: V-rep : a versatile and scalable robot simulation framework. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326. IEEE (2013)
26. Freese, M., Singh, S., Ozaki, F., Matsuhira, N.: Virtual robot experimentation platform v-rep : a versatile 3d robot simulator. In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pp. 51–62. Springer (2010)
27. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. *arXiv:1509.02971* (2015)
28. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
29. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: combining improvements in deep reinforcement learning. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
30. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: *International Conference on Machine Learning*, pp. 1587–1596. PMLR (2018)
31. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International Conference on Machine Learning*, pp. 1861–1870. PMLR (2018)

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Li Zheng received the B.Eng degree from Dalian University of Technology, in 2020. He is currently a Ph.D. student at the University of Science and Technology of China (USTC). His research interest covers reinforcement learning, manipulator motion planning, and intelligent mobile robot.

Yahao Wang received the B.Eng degree from Anhui Engineering University, in 2019. He is currently a Ph.D. student at the University of Science and Technology of China (USTC). His research interests include the motion planning of the manipulator.


Run Yang he is currently a Ph.D. student at the University of Science and Technology of China (USTC). His research interest covers intelligent mobile robot.

Shaolei Wu is a professor-level senior engineer, engaged in power distribution technology management and power robotics research.

Rui Guo is the chief scientist of State Grid Intelligence Technology Co., Ltd. His research interests focus on robotics for the electric power industry, live-line working, non-destructive testing.

Erbao Dong received the Ph.D. degree from the Department of Precision Mechanics and Precision Instruments, University of Science and Technology of China (USTC), in 2010. From 2010 to 2012, he was a postdoctoral fellow at the Mechanics Postdoctoral Station of USTC. He was appointed as the Distinguished Associate Professor of the Department of Precision Machinery and Precision Instruments, USTC, in 2012 and was transferred to associate professor in 2018. His research interests include Artificial muscles and bionic robots, intelligent mobile robots, dual-arm collaborative robots.

Affiliations

Li Zheng¹ · YaHao Wang¹ · Run Yang¹ · Shaolei Wu² · Rui Guo³ · Erbao Dong¹ 

Li Zheng
zlsy@mail.ustc.edu.cn

YaHao Wang
wyh218@mail.ustc.edu.cn

Run Yang
892613895@qq.com

Shaolei Wu
wushaolei@sina.com

Rui Guo
guoruihit@qq.com

- ¹ CAS Key Laboratory of Mechanical Behavior and Design of Materials, Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China, 96 Jinzhai Road, Hefei, 230026, Anhui Province, China
- ² State Grid Anhui Electric Power Company Electric Power Research Institute Hefei, 230601, Anhui Province, China
- ³ State Grid Intelligent Technology Co, Ltd, Jinan, 250001, Shandong Province, China