Proceedings of the ASME 2023
International Mechanical Engineering Congress and Exposition
IMECE2023
October 29-November 2, 2023, New Orleans, Louisiana

IMECE2023-113131

AN ADAPTIVE PATH PLANNING APPROACH FOR DIGITAL TWIN-ENABLED ROBOT ARM BASED ON INVERSE KINEMATICS AND DEEP REINFORCEMENT LEARNING

Qi Zhou[1] **Sikai Li**[1] **Jingbo Qu**[1] **Jin Wu**[1]

Haomiao Xu[1] **Youyi Bi**[1*]

[1] University of Michigan – Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Efficient path planning methods for robot arms are crucial to ensure the quality and safety of their completing various tasks. Compared to traditional manual instruction, Reinforcement Learning (RL) based path planning methods show better adaptability for complex working scenarios. However, the training of RL is usually time-consuming with limited success rate. To tackle this problem, we propose an adaptive path planning approach for robot arm based on Inverse Kinematics (IK) and Deep Reinforcement Learning (DRL) in a pick-and-place context. A judgement mechanism is developed to adaptively select IK or RL based method according to the results of early-stage collision detection. We separate the pick and place task into three sequential curriculums (approaching, grabbing and placing) with modified reward functions to speed up the training process and achieve a higher success rate. The proposed approach is validated with a physical robot arm supported by a high-fidelity digital twin model. The experiment results show that our proposed approach outperforms traditional RL based method with improved training speed and guaranteed performance in collision avoidance and path accuracy. This work contributes to the practical deployment of RL based path planning method for digital twin-enabled robot arm in smart manufacturing.

Keywords: Robot arm, Path planning, Deep reinforcement learning, Inverse kinematics, Digital twin

1. INTRODUCTION

Robot arms are being used widely in smart manufacturing to replace human workers for those labor-intensive or dangerous work, especially in scenarios of assembly, palletizing and welding [1]. To ensure the quality and safety of completing these tasks, efficient path planning method for robot arms are needed. Traditional manual instruction for robots is easy to use, but laborious with poor flexibility. It cannot adapt to complex working conditions where robots have to process multi objects and avoid multi obstacles agilely [2]. Recent research explored to use Reinforcement Learning (RL) and Digital Twin (DT) based methods for robot arm path planning to meet the requirements of flexible manufacturing and adapt to fast switching of producing multi-category products [4-6]. RL is a machine learning approach that enables an agent to learn and make decisions by interacting with environment. DT is a cutting-edge technology that creates a virtual replica of a physical system, allowing real-time monitoring, analysis, and optimization for enhanced operational efficiency and decision-making. With the support of Digital Twin, RL based methods allow a robot arm to learn to find optimal movement trajectories by iteratively interacting with environment in a high-fidelity virtual space. These methods show better adaptability for complex working scenarios without the need of manual guidance [2, 3].

For example, researchers from Shanghai Jiao Tong University [4] introduced a human-robot cooperation framework based on the Proximal Policy Optimization (PPO) algorithm, a flexible RL method that could deal with high-dimensional state space problems. This framework is able to replan the collision part of robot path by combining the information of real working environment during the training process. A peg-in-hole assembly task was executed with a success rate of 90% achieved. Li et al. [5] developed a digital twin of robot to support the training of RL models and previewing the planned paths to avoid collision. This work contributes to a more feasible Sim-To-Real application framework of industrial robots. They [6] also realized a user-friendly control mode of robot arm by leveraging DT and Augmented Reality (AR) technologies to set up a multi-robot collaboration system where RL method was adopted for

[*] Corresponding author, Assistant Professor in Mechanical Engineering, Shanghai Jiao Tong University
Email: youyi.bi@sjtu.edu.cn

path planning. This work reduces redundant robot control learning costs and achieves an intuitive control mode of robot. Compared to manual instruction, these RL based methods for robot arm path planning show good adaptability to complex scenarios.

However, the training process for RL models is time-consuming which can take dozens of hours or days in laboratory, let alone in an actual factory [7]. In this regard, accelerating the training process without sacrificing model accuracy is crucial for RL based methods to be adopted widely in practice. Various researchers have made efforts in this area.

For instance, Zhou et al. [8] proposed an adaptive obstacle size adjustment method to simplify the modeling of obstacles in robot arm motion planning. Curriculum-based reinforcement learning is adopted to accelerate the training process. Wang et al. [9] optimized the PPO algorithm by introducing a value function update module. A Sim-to-Sim approach is proposed to shorten the training time, which materialized robot path planning in an unstructured environment. Matulis et al. [7] set ten curriculums for Pick-and-Place (P&P) task according to accuracy tolerance and achieved well performance in training speed.

To further improve the training speed in robot arm path planning, recent studies also explore to combine Inverse Kinematics (IK) with RL. In the field of robotics, IK can calculate the needed rotation angles of a robot arm's joints to ensure the robot reaches a given posture in a fast and accurate way. Luipers et al. [10] designed a human-like thinking-execution framework for the path planning problem of robot arms. In this framework, RL is used to plan waypoints first and then the IK solver calculates the trajectory between waypoints. This method greatly reduced the difficulty of task training, accelerated the convergence rate of RL models and achieved a 98% success rate. Zhang et al. [11] introduced a hierarchical reward function including motion accuracy based on robot inverse kinematics principle. This method speeds up the convergence of PPO algorithm and improves the motion accuracy of robot arm. In order to reduce the unnecessary exploration in state-action space, Zhong et al. [12] added an IK module into the RL algorithm and designed a gain module to avoid local optimality. The convergence speed and robustness of the RL models for robot motion planning are improved.

In the above studies, RL has been commonly employed to plan the entire motion path for collision avoidance. However, this approach may be inefficient for certain scenarios such as P&P task. In a P&P task, collisions mainly occur when the robot approaches to the target object, and fewer collisions exist in the initial starting of the robot arm and the final placement of the target object. For those motion segments with low risk of collision, inverse kinematics can provide a reasonable path for robot arm much more efficiently than reinforcement learning based methods. Thus, we propose an adaptive path planning approach for robot arm based on inverse kinematics and deep reinforcement learning in this paper. Our approach can reduce the training difficulty of the RL model, shorten its training time and facilitate the rapid deployment of RL-based method in robot arm path planning.

The core idea of our approach is a judgement mechanism that can adaptively select IK or RL based method according to the results of early-stage collision detection. We first leverage the Cyclic Coordinate Decent (CCD) algorithm from inverse kinematics to generate an initial robot arm path rapidly without considering obstacles. If collisions are detected in this path, the PPO algorithm from RL is applied to regenerate the collided part of the initial path. Finally, a collision-free path in an environment with multiple obstacles can be generated by fusing these two paths. In addition, we separate the P&P task into three sequential curriculums (approaching, grabbing and placing) with modified reward functions in the training of RL model to speed up the training process and achieve a higher success rate.

To validate the proposed approach, we built a high-fidelity digital twin model of robot arm to support the training of RL and conducted physical experiments with a real robot arm. The proposed approach was compared with traditional RL based method and IK based method. The experiment results show that our approach outperforms these methods with improved training speed and guaranteed performance in collision avoidance and path accuracy.

The rest of this paper is structured as follows. Section 2 introduces the overall structure of the proposed approach, and elaborates the key methods involved, including the adaptative path planning method, object detection method, inverse kinematics method and reinforcement learning method. Section 3 presents the simulation and physical experiments to examine the validity of the proposed approach and discusses the experiment results. Section 4 summarizes our work and provides potential future research directions.

2. METHODS

2.1 Overall Workflow of the Proposed Approach

Figure 1 presents the overall workflow of the proposed path planning approach for a digital twin-enabled robot arm. Here DT provides a high-fidelity environment for the visual representation of physical layer and the offline training of RL models. As shown in Fig. 1, the physical layer consists of hardware components such as robot arm, robot controller, electric gripper, RGB-D cameras and the workspace for the pick-and-place task. The virtual layer includes the high-fidelity digital twin model of the physical components built in Unity3D environment, a path planning module, virtual robot controller and a user interface. The physical and virtual layers are connected through socket communication based on TCP protocol, which enables the efficient modeling of real environment and transmission of control commands. To create a digital twin model with high-fidelity, dynamic properties and physical constraints of real robot arm such as joint weights, angular friction, angular damping, and joint angle limits are also incorporated in virtual space.

The overall workflow of our approach is as follows. First, the object detection module perceives the workspace information (i.e., the position and rotation of objects and obstacles) and transmits it to virtual layer. Then, the UI controller generates a virtual model of the workspace and receives the task commands

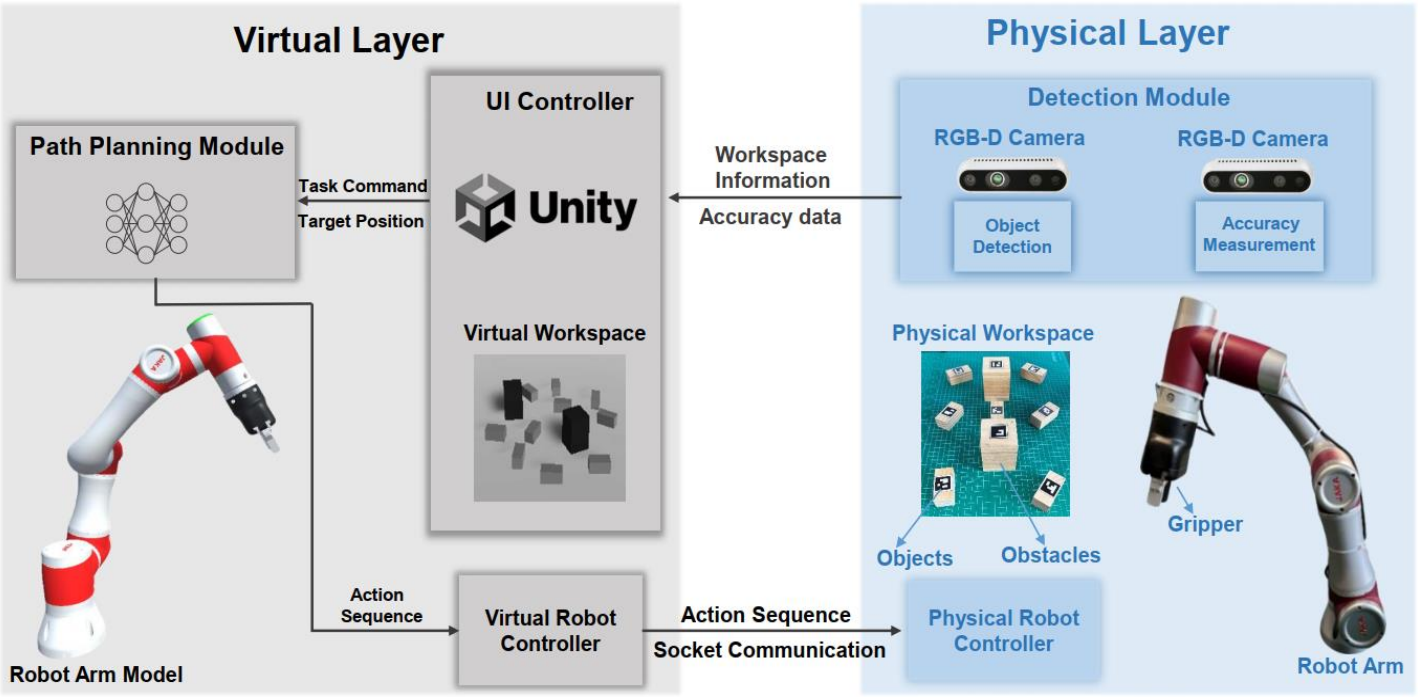

FIGURE 1: THE OVERALL WORKFLOW OF THE PROPOSED APPROACH

assigned manually (e.g., the ID of target object). Third, the path planning module utilizes the environment and object information to generate a proper path using the adaptive path planning based on inverse kinematics and deep reinforcement learning. The corresponding action sequence is then outputted to the virtual robot controller for replaying the robot's actions. Finally, the virtual robot controller passes the action sequence and joint angles to the physical robot controller for execution through socket communication. The details of the key techniques and methods involved in our proposed approach are provided in the following sections.

2.2 Object Detection and Location

Getting precise location information of the working environment and manipulating objects is the first step in the path planning for a robot arm. In this work, we set up a visual system consisting of two RGB-D cameras to detect and locate the working objects and obstacles due to the limited field of vision of a single camera. These two cameras are fixed at two sides of the working area utilizing an eye-to-hand approach. The front camera can provide the working environment information such as the position and rotation of all objects and obstacles. The rear camera is used to measure the position accuracy of the placed objects when a pick-and-place task completes. To overcome the occlusion problem in multi-object and multi-obstacle detection, ArUco markers are used as posture estimation feature. ArUco markers are fast to deploy with high accuracy and robustness of detection and location [13]. The output information from object detection and location is then transmitted to the virtual space for the use of visualization and path planning.

2.3 Adaptive Path Planning

Figure 2 depicts the workflow of the adaptive path planning method. After receiving the positions of all targets and obstacles in the working space, a virtual environment containing these objects (i.e., the digital twin model) is created in real-time. Following the pick-and-place command, the Inverse Kinematics (IK) Module generates the path sequence for the robot arm with corresponding joint angles. Then the virtual robot arm model moves according to the generated sequence of joint angles, and the collision detection module examines if any collision occurs. The collision detection module leverages the collision detection mechanism in Unity3D by adding colliders to the robot, gripper, manipulating objects, obstacles, and the working environment (e.g., table). The collision detection mechanism is based on the embedded physics engine in Unity3D, using colliders and rigid body components to simulate the shape and motion of objects and calculate the collision response. If no collision is detected in the generated path from the IK module, this path is considered to be safe and corresponding action sequence will be transmitted directly to the physical robot for task execution. Otherwise, the path interception mechanism intercepts the collided path sequence and the Reinforcement Learning (RL) module is invoked to re-plan the collided path. The RL path and truncated IK path are then fused to form the final feasible path and action sequence transmitted to the real robot controller.

Figure 3 shows an example of adaptive path planning. The grey block is the target object for pick-and-place, while the black blocks are obstacles. We superimpose the different postures of the robot arm in in this figure. P_0 and P_5 are the initial position and end position of the robot end effector, respectively. The

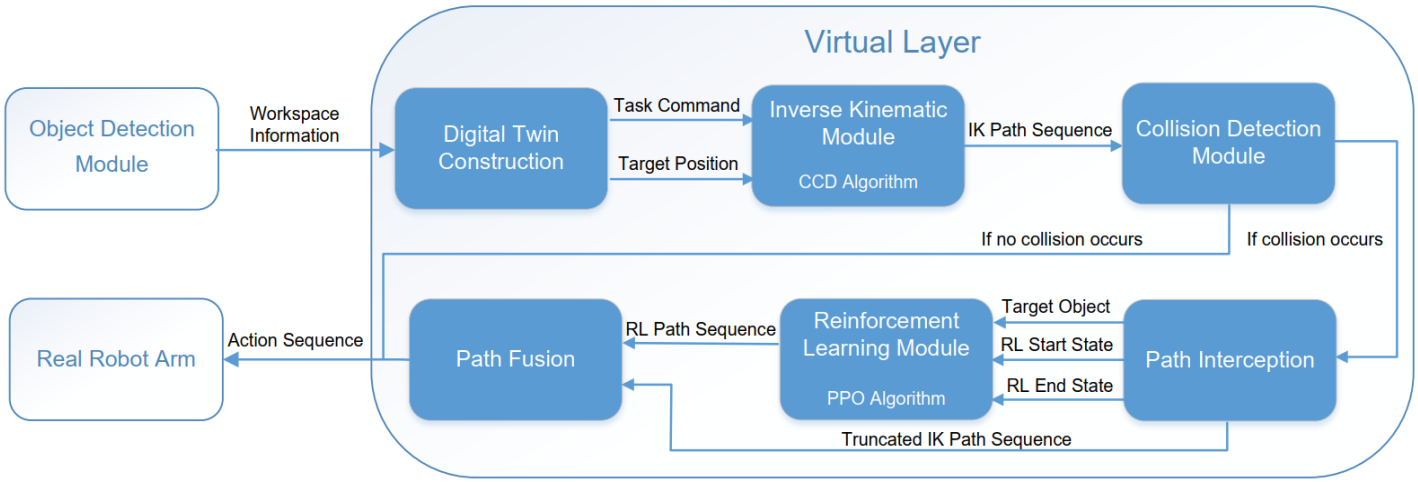

FIGURE 2: THE WORKFLOW OF THE ADAPTIVE PATH PLANNING METHOD

yellow line $P_0P_1P_2P_3P_4P_5$ is the path generated by IK module which may contains a collided part P_2P_3. To ensure a safe initial state for RL based path planning method and reduce the difficulty of RL training, a secure path is added into the initial and final segments of the intercepted collided path, represented by P_1P_2 and P_3P_4. The information of the first and last points (P_1 and P_4) of the truncated collided path, along with the target position, are inputted to the RL module. In RL module, we employ the PPO algorithm to re-plan the collided path and generate a safe path considering the obstacles, as illustrated by the red lines $P_1P_2P_6P_7P_4$. Finally, this re-planned path is fused with the truncated safe paths (P_0P_1, P_4P_5) from the IK module to obtain a complete path sequence $P_0P_1P_2P_6P_7P_4P_5$, which will be transmitted to the physical robot controller.

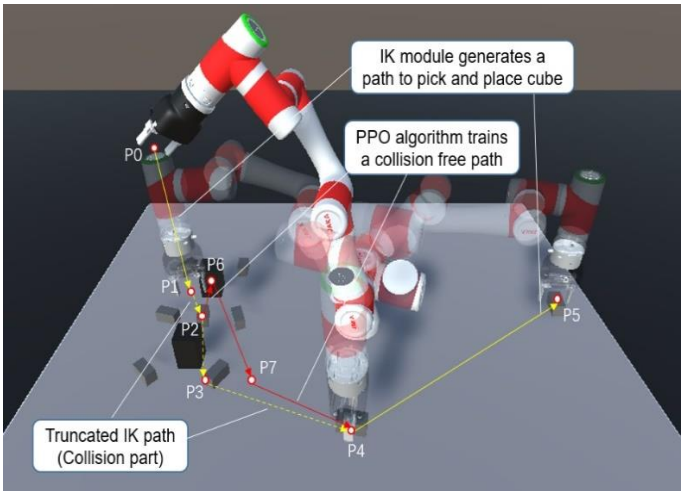

FIGURE 3: AN EXAMPLE OF ADAPTIVE PATH PLANNING

2.4 CCD-based Inverse Kinematics Algorithm for Path Planning

Inverse Kinematics is widely used in robotics to determine the needed rotation angles for each joint of a multi-joint robot arm given its initial and target postures. In our proposed approach, IK is used to quickly generate a viable path for the pick-and-place task in those environments with low risk of collisions, and simplify the training task of RL models. Here we employ the Cyclic Coordinate Decent (CCD) algorithm [14], which is a simple but effective IK method that converges quickly and is free from singularities. As shown in Algorithm 1, the CCD algorithm is a heuristic search technique that iteratively computes the angle between the vector from each joint to the end effector $\boldsymbol{v_{bt}}$ and the vector from each joint to the target point $\boldsymbol{v_{bp}}$. The algorithm calculates the rotation angle α_j starting from the end effector and rotates each joint sequentially until the end effector eventually reaches the target position after several iterations.

To execute this iterative rotation, we create a robot bone $bone[j]$ that contains the initial position of the robot joints. Once the robot bone rotates to the target position after IK calculation, the robot joints will also rotate.

It is worth noting that the CCD algorithm cannot restrict the rotation angle of the end joint of the robot, which may result in misalignment between the electric gripper and the target object. Therefore, we introduce an end joint regulation mechanism to the CCD algorithm as shown in Line 12 of Algorithm 1. It can keep the gripper and target object in parallel, and enhance the success rate of grabbing.

Algorithm 1: CCD-based Inverse Kinematics Algorithm

Input: Target cube position $\boldsymbol{p}$, target cube rotation r
Output: Sequence of joint angles

1: Initialize robot bone position and robot joint position
2: **for** $i = 1$:MaxCcdDepth **do**
3: **for** $j = 1$:MaxBoneJoint -1 **do**
4: Transform boneTcp **t** and **p** to bone[j].localCoordinate

Calculate vector of bone $[j]$ to boneTcp: $\mathbf{v}_{\mathrm{bt}} \leftarrow (\mathbf{t}.x, 0, \mathbf{t}.z)$
Calculate vector of bone $[j]$ to $\mathbf{p}$: $\mathbf{v}_{\mathrm{bp}} \leftarrow (\mathbf{p}.x, 0, \mathbf{p}.z)$
Calculate rotation angle for bone$[j]$: α_j =Vector3.Angle($\mathbf{v}_{\mathrm{bp}}, \mathbf{v}_{\mathrm{bt}}$)
Calculate rotation axis for bone$[j]$: $\mathbf{l}$ =Vector3.Cross($\mathbf{v}_{\mathrm{bp}}, \mathbf{v}_{\mathrm{bt}}$). normalized
Rotate bone$[j]$ by α_j along $\mathbf{l}$
end for
end for
Calculate rotation angle for bone[6]: $\alpha_6 = r -$ bone[6].rotation
Rotate bone[6] by α_6 along axis $\mathbf{y}$
for $k = 1$:MaxJoint **do**
Rotate Joint$[k]$ to bone$[k]$.localRotation
end for

2.5 Reinforcement Learning Based Path Planning

As mentioned in Sec. 2.3, if collisions are detected in the generated path from the IK module, the reinforcement learning (RL) model is invoked to re-plan the collided path. The core framework of reinforcement learning is a sequential decision-making problem referred to as Markov Decision Process (MDP) [15]. In an MDP, the machine that learns and makes decisions is called agent, while the object that interacts with agent is the environment. At each time step t, the agent observes the environment state $S_t \in \mathcal{S}$ and chooses an action $A_t \in \mathcal{A}(s)$ based on the strategy $\pi(a|s)$. The agent then receives a reward $R_{t+1} \in \mathcal{R}$ and enters a new state S_{t+1}. This process forms a sequence $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$ between the MDP and the agent. In MDP, the occurring probability of reward R_t and environment state S_t depends only on the preceding state and action, as described in Equation (1):

$$p(s', r|s, a) = \Pr\{S_t = s', R_t = r|\, S_{t-1} = s, A_{t-1} = a\} \quad (1)$$

where $p: \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ is the probability of R_t and S_t occurring at time t. S_{t-1} and A_{t-1} are the state and action at time $t-1$. The goal of the agent is to maximize the expected return G_t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} \quad (2)$$

where $\gamma \in [0,1]$ is a discount factor to help maximize future expected return rather than the current reward. In the context of robot arm path planning, the agent of robot arm can learn to find optimal paths by iteratively interacting with environment to achieve maximum reward, i.e., arriving at the target position without collision. In the proposed approach, we employ Proximal Policy Optimization (PPO) algorithm as the RL method. Its basic idea, setting of environment states and actions, and the design of curriculum learning and reward functions are explained in the following subsections.

2.5.1 Proximal Policy Optimization

PPO is a deep reinforcement learning (DRL) method based on policy gradient [16]. Compared with other DRL method, PPO is more stable with high sampling efficiency for continuous motion space and high dimensional state space problems. It utilizes a new objective function known as the clipped surrogate objective, which maximizes the expected return of the new policy while keeping the difference between the old and new policies small. PPO algorithm avoids issues that may arise from overly large or small policy updates, thereby achieving more stable performance. The core of the PPO algorithm is the maximization of a constrained objective function defined in Equations (3) and (4):

$$L^{CLIP}(\theta) = \mathbb{E}[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)] \quad (3)$$

$$r_t(\theta) = \frac{\boldsymbol{\pi_\theta}(\mathrm{a_t}|\mathrm{s_t})}{\boldsymbol{\pi_\theta}_{\mathrm{old}}(\mathrm{a_t}|\mathrm{s_t})} \quad (4)$$

Where $\mathbb{E}$ is the expected value over all samples at time step t, $\boldsymbol{\theta}$ denotes the policy parameters, r_t is the ratio between the new and old policies. A_t represents the advantage function, which is the expected difference in returns when taking action $\boldsymbol{a_t}$ in state $\boldsymbol{s_t}$ compared to the old policy, and ε is a hyperparameter that controls the range of the ratio. In this study, we trained our robot agent using the PPO algorithm in ML-Agents, an open-source toolkit that supports the training of multiple deep reinforcement learning algorithms within the Unity3D environment [17].

2.5.2 Environment States and Actions

The state parameters $\boldsymbol{s_t}$ of the robot arm agent are shown as follows:

$$\boldsymbol{s_t} = \{J, P_g, P_t, P_a, O_g, O_t, X_{safety}\} \quad (5)$$

$J = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$	Joint angle of robot arm
$P_g = (P_g^x, P_g^y, P_g^z)$	Position of gripper catch center
$P_t = (P_t^x, P_t^y, P_t^z)$	Position of target object center
$P_p = (P_p^x, P_p^y, P_p^z)$	Position of placement area
$O_g = (O_g^x, O_g^y, O_g^z)$	Rotation of gripper
$O_t = (O_t^x, O_t^y, O_t^z)$	Rotation of target object
X_{safety}	Collision status, bool

X_{safety} indicates any possible collision during P&P task (e.g., collisions in Gripper-Table, Gripper-Obstacle, Gripper-Cube, Robot Arm-Table, Robot Arm-Obstacle, Cube-Obstacle, and Cube-Cube). The collision detection is realized using the collider component in Unity. The action $\boldsymbol{a}$ of robot arm agent is designed as:

$$\boldsymbol{a} = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \alpha) \quad (6)$$

The action $\boldsymbol{a}$ is a 7-dimensional vector that controls the rotation of six joints of the robot arm and the gripper. δ_i is controlled to rotate 1° in two different directions at each time step to reduce the control difficulty on the premise of ensuring accuracy. α is a binary-value variable controlling the opening and closing of the gripper.

2.5.3 Design of Learning Curriculums and Reward Functions

Curriculum learning is a technique to gradually introduce and increase the complexity of tasks which can effectively reduce the training difficulty in complex tasks and facilitate the model convergence of the RL methods. In this work, we divide the pick-and-place task into three curriculums for learning, including (1) Approaching (approach the target object), (2) Grabbing (grab the object), and (3) Placing (accurately place the object in the designated position). In each curriculum, corresponding reward functions are designed for better convergence of the RL model as shown in Table 1 and Equations (7) to (11).

Table 1. Three curriculums with associated reward functions for reinforcement learning-based path planning.

Curriculum	Reward Function
Approaching	$f_1 + f_2 + f_5$
Grabbing	$f_1 + f_2 + f_3 + f_5$
Placing	$f_3 + f_4 + f_5$

$$f_1 = -\lambda_1 D_{gt} \quad (7)$$

$$f_2 = -\lambda_2 O_{gt} \quad (8)$$

$$f_3 = -\lambda_3 \alpha,\ \alpha = \begin{cases} 0 & gripper\ is\ closing \\ 1 & gripper\ is\ opening \end{cases} \quad (9)$$

$$f_4 = \lambda_5(\lambda_4 - D_{tp}) \quad (10)$$

$$f_5 = -\lambda_6 X_{safety},\ X_{safety} = \begin{cases} 0 & collision\ free \\ 1 & collision\ occurs \end{cases} \quad (11)$$

We introduce five reward functions here from f_1 to f_5, and λ_1 to λ_6 are the scale parameters for each reward function. In each curriculum, different combinations of reward functions are used to improve the training of RL model.

In the first curriculum Approaching, the main goal is to approach the target object and adjust the posture of the gripper to keep it parallel to the target object. Two criteria are designed, the relative distance D_{gt} and the relative rotation O_{gt}. The D_{gt} is the distance between the gripper catch center and the target object center. The O_{gt} is the rotation difference between the gripper and the target object. The distance reward f_1 and the rotation reward f_2 will increase when the gripper approaches the target and rotates to the posture parallel with the target object, respectively. Besides, the collision punishment f_5 is introduced in all three curriculums to avoid any possible collision.

In the second curriculum Grabbing, the robot needs to grab the target object without collision. The relative distance and rotation criteria (f_1 and f_2) are used to fine-tuning the posture of gripper. A grabbing action reward function f_3 is designed to encourage the agent to close the gripper and grab the target object when the gripper is in a proper posture (i.e., $D_{gt} < 10mm$ and $O_{gt} < 5^{\circ}$).

In the third curriculum Placing, the agent is supposed to place the target object in the designated position. Different from other reward functions, the distance reward f_4 is set to positive by introducing a distance threshold value λ_4 and D_{tp} indicating the distance between the target cube center and the placing position. This is because if the target position is far away, the robot arm needs to take a large number of steps to reach it. The negative reward will cause the reward to shrink continuously. When the cumulative reward is smaller than the penalty of collision, the agent will collide to obtain a relatively large reward. The grabbing action reward function f_3 is also used to hold the object during the placing process.

3. EXPERIMENT AND RESULTS

To validate the proposed approach, we built a high-fidelity digital twin model of a real robot arm and conducted physical experiments in the context of pick-and-place task. In this task, the robot arm needs to grab cubes initially located in random positions and then move and place them in target positions. We expect to examine and compare the performance of the proposed approach with other approaches in this task. The experiment settings and results are presented in the following subsections.

3.1 Experiment Settings

Figure 4 shows the setting of the physical equipment and working space. A robot arm (JAKA Zu3) with an electric gripper (CTEK CTP2F50) is fixed to a $1.2\text{m} \times 1.2\text{m}$ table. The plane of the table was set as the plane with $z = 0$ in world coordinate system. The center of the robot arm base was set as the origin point (0,0). Two RGB-D cameras (RealSense D435i) are deployed at (-0.45, 0.65) and (-0.45, -0.65) to detect the cubes and obstacles in the picking area and measure the accuracy of cube placements in the placing area, respectively. The target cubes and obstacle cubes are wooden blocks with sizes of $3\text{cm} \times 3\text{cm} \times 6\text{cm}$ and $6\text{cm} \times 6\text{cm} \times 9\text{cm}$. Seven target cubes were placed randomly in the rectangular area defined by the four vertices (-0.25, 0.35), (0.25, 0.35), (-0.25, 0.55) and (0.25, 0.55). Two obstacle cubes were placed on (-0.10, 0.45) and (0.10, 0.45). To ensure the safety during tasks, adjacent cubes must keep a minimum spacing of 5 cm in the initial setting. This distance also considers the width of the gripper (9 cm).

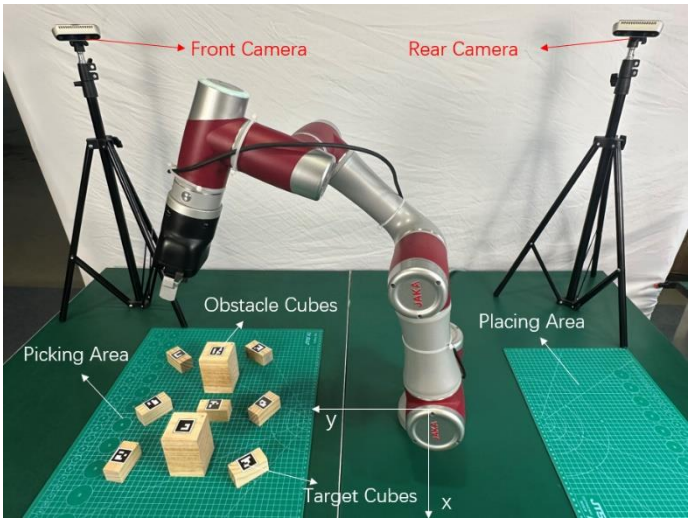

FIGURE 4: OVERALL EXPERIMENT SETTING

A high-fidelity digital twin model of the robot and the working environment is established in Unity3D as shown in Fig. 5. Besides displaying the visual representation of these physical entities, DT also provides the status data of environment and robot (e.g., the position of objects and obstacles, the joint angle and end effector position of robot) for offline RL training. In fact, the training of RL model needs the real-time status of the agent (i.e., robot arm) and environment. DT can provide such information in a virtual space without acquiring data from physical space which is obviously expensive and time-consuming. Besides, high-fidelity DT model is the significant foundation for collision detection which enables the adaptive switch of two path planning methods (i.e., IK and RL). Meanwhile, DT bridges the virtual environment and real entity, and the action sequence generated in virtual space could be transmitted to the real robot controller for execution in physical space. A user interface is established for a more convenient way to control. On the interface, three path planning methods (Adaptive method, IK method, RL method) are available for users to choose and all the generated paths are recorded. Here the IK method refers to the CCD algorithm-based IK method, and the RL method is the PPO algorithm-based method. Besides, the commands with operating physical robot arm and cameras are also integrated in the user interface for Sim-To-Real control. The positions of the virtual cubes and obstacles are updated in real time according to their actual position information in physical world captured by the front RGB-D camera. The training process of the proposed adaptive path planning method and RL method runs on a workstation with Intel Core i7-12700F CPU, 16GB RAM, and RTX3080 GPU.

Table 2 shows the setting of hyperparameters for the RL training process. The batch size is the experience samples used for each training batch. The buffer size refers to the capacity of replay buffer used to store the training experience. The learning rate controls the training speed, which is set to 0.0003 to avoid reward divergence. Epsilon is set to 0.2, which is a key parameter influencing the range of policy updating. A small value of

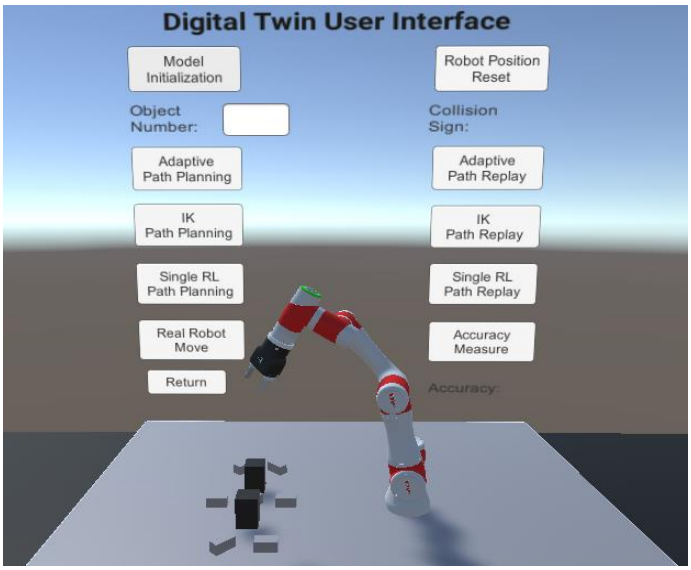

FIGURE 5: THE DIGITAL TWIN MODEL OF THE ROBOT AND WORKING ENVIRONMENT WITH A USER INTERFACE.

Epsilon can slow down the rate of convergence while big values may lead to divergence. Lambda is a discount factor calculating cumulative return. We set lambda to 0.95 to prioritize current rewards. Beta controls the advantage function in PPO algorithm and the epoch number indicates rounds of the training model in each training batch. Linear learning rate schedule makes the learning rate to decrease linearly before max steps. These parameters are chosen based on previous work and preliminary experiments.

Table 2. Hyperparameters for the PPO training process.

Hyperparameter	Value
Batch size	512
Buffer size	10240
Learning rate	0.0003
Epsilon	0.2
Lambda	0.95
Beta	0.005
Epoch number	3
Learning rate schedule	Linear

3.2 Experiment and Results

3.2.1 Convergence Speed Analysis in the Simulated Environment

We first compare the convergence speed of our proposed adaptive path planning approach and RL method (original PPO algorithm) for the pick-and-place tasks in the simulated environment. In each task, 7 target cubes and 2 obstacle cubes are placed on the picking area. A target cube is randomly assigned to the robot arm and then transferred to the target position by the robot arm. The positions of target cubes are randomly shuffled after completing each task. In order to improve the success rate, curriculum learning is applied in both

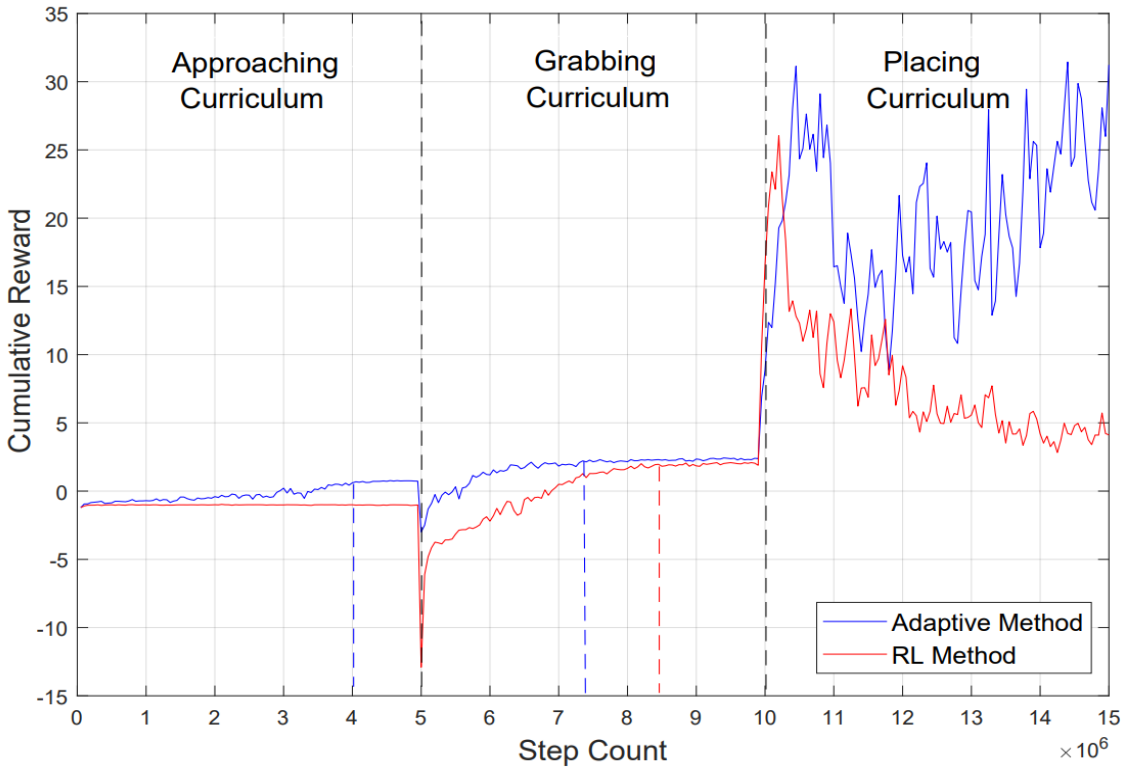

FIGURE 6: THE OBTAINED AVERAGE CUMULATIVE REWARD OF PROPOSED METHOD AND RL METHOD

methods. The training of our proposed approach and the RL method takes 15 million steps and each curriculum takes 5 million steps respectively. The obtained average cumulative reward of the two methods in the training process for approaching, grabbing and placing curriculums is shown in Fig. 6.

As we can see in Fig. 6, the rewards of both methods converge at the end, indicating that the agents in both methods complete the approaching, grabbing and placing tasks. In the first 5 million steps, the RL method's reward function remains unchanged, indicating that the agent has not learned how to approach the target object. This is mainly because the initial position of the gripper in the RL method is far away from the target object, making the task more difficult and leading to collisions in the training process all the time. In contrast, the adaptive method's reward keeps increasing and converges at 4 million steps, as the starting position of the gripper assigned by IK module is quite close to the target. Which makes the task easier to complete and the agent learns the curriculum quickly.

In the process from 5 million to 10 million steps, the rewards of the two methods keep increasing. The adaptive method converges at 7.3 million steps, while the RL method converges at 8.5 million steps. This is probably because the approaching experience learned from the adaptive method in the first curriculum can be directly applied to the second curriculum, speeding up the training speed. However, the experience of RL method learned in the first curriculum has poor effect, so it is necessary to re-train this action in the second curriculum, which increases the difficulty of the task.

In the last 5 million steps, the cumulative reward of RL method gradually shrinks and converges to 5. In contrast, the cumulative reward of the adaptive method begins to rise after a period of oscillation and converges to 25 in the end. One possible reason is that the placing task of adaptive method is easier to accomplish since the placing position is assigned by IK module which is closer than that of RL method. Thus, the agent of adaptive method could learn this curriculum better.

In summary, the proposed adaptive path planning method simplifies the P&P task by introducing the IK module, speeds up the training process and achieves a higher effectiveness.

3.2.2 Generalization Performance and Robustness Analysis in Physical Environment

In order to test the generalization performance and robustness of the proposed approach, we designed a pick-and-

FIGURE 7: PICK AND PLACE EXPERIMENT (10 GROUPS IN TOTAL, 7 TARGET CUBES AND 2 OBSTACLE CUBES ARE PLACED RANDOMLY IN EACH GROUP.)

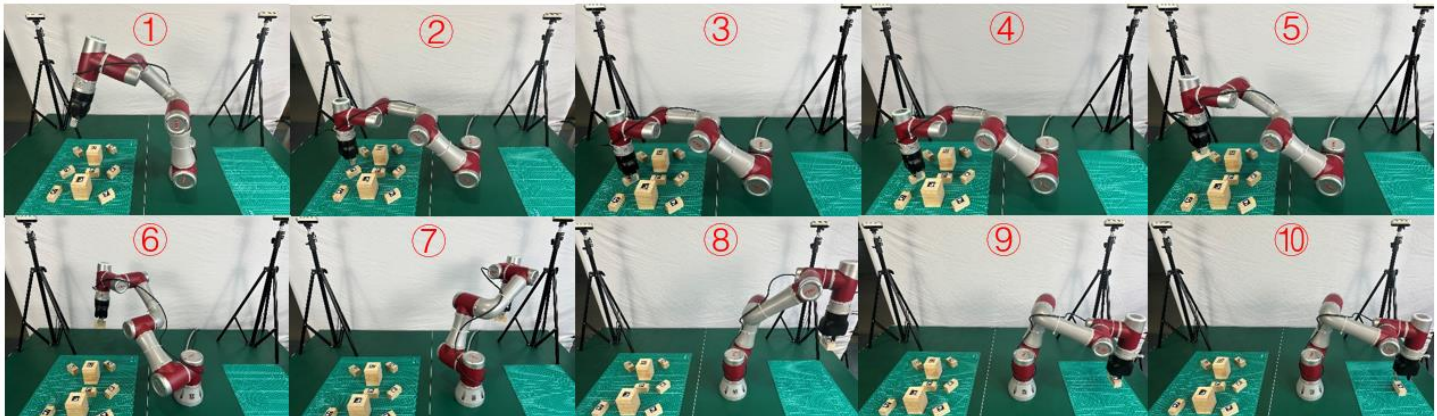

FIGURE 8: AN EXAMPLE OF PICK AND PLACE TASK IN PHYSICAL ENVIRONMENT. ① AND ②: APPROACHING STAGE; ③ AND ④: GRABBING STAGE; ⑤ TO ⑩: PLACING STAGE.

place scene as shown in Fig. 4. The proposed adaptive path planning approach, RL method and IK method were employed respectively to pick up 7 target cubes and place them at the target position (0, -0.50), which was 50 cm directly behind the robot. Ten groups of pick-and-place tasks were conducted for each method. In each group of tasks, the target cubes were placed randomly to examine the generalization performance of three methods, as shown in Fig. 7. Figure 8 shows a complete cycle of the pick-and-place task in physical environment.

In the physical experiments, we use the collision avoidance rate, first attempt success rate and placing accuracy as the three metrics to evaluate the performance of different methods. Collision avoidance rate is defined as the ratio of the cases with no collision occurring during P&P task to the total cases. Note that any improper contact between robot and other objects (e.g., cube, obstacle, table), gripper and obstacle, gripper and other cube, gripper and table, target cube and obstacle or other cube (during moving), will be considered as collision. First attempt success rate means the rate of successfully placing the target in the target area without any collision for the first try. Here the target area is defined as a circular shape within 5 cm around the target position (0, -0.50). Placing accuracy is calculated as the difference between target position and the actual center position of placed cube obtained from the rear RGB-D camera.

Figure 9 shows the comparison results of three methods. The collision avoidance rate of the IK method is 43%, while that of the RL method and adaptive method are 100%, since IK method is unable to avoid collisions automatically. Both RL method and the adaptive method can generate a collision-free path after several attempts. The first attempt success rate of the IK method is equal to its collision avoidance rate. This is mainly because of the target cube in the same position, the paths planned by the IK method are the same each time. The first attempt success rate depends on the effect of approaching, grabbing, and placing actions. The first attempt success rate of the RL method and the adaptive method is 41% and 53%, respectively. This result indicates that the proposed approach has a higher success rate for the three curriculums in the P&P task. It is worth noting that the first attempt success rate is measured in the simulation environment. When collision occurs in the simulation environment, both the RL method and the adaptive method will re-plan the path until a safe path is generated. Therefore, both methods can generate a safe path after several attempts, and the actual task success rates of both methods are 100% although their first attempt success rates are not high.

As shown in Fig. 9, the placing accuracy of the IK method, adaptive method, and RL method are 3mm, 19mm, and 50mm, respectively. The accuracy of the proposed method is lower than the IK method since the grabbing process mainly relies on the RL module, which has certain positioning errors. The accuracy of the proposed method is higher than the RL method since the path in the last part of the placing action is generated by the IK module, which can guarantee the placing accuracy to a certain extent. In summary, the proposed adaptive path planning approach has high collision avoidance rate, first attempt success rate and relatively good placing accuracy compared to the IK method and RL method. The experiment results validate the effectiveness of our proposed approach in realistic application scenarios.

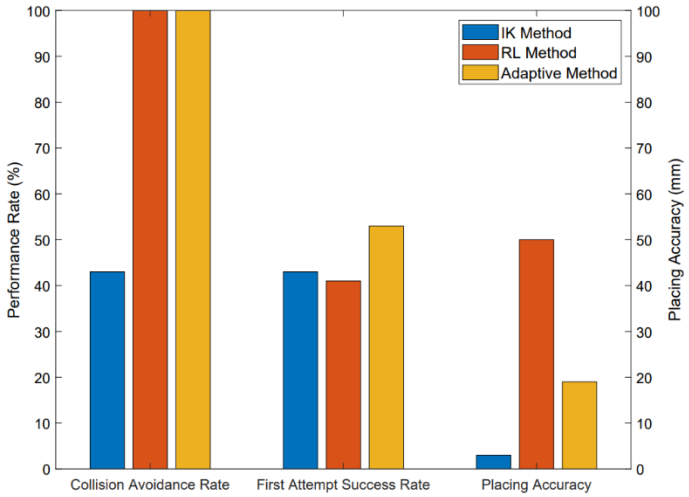

FIGURE 9: PERFORMANCE COMPARISON OF IK (CCD), RL (PPO) AND ADAPTIVE METHOD (CCD+PPO) IN P&P TASKS.

4. CONCLUSION

In this paper, we propose an adaptive path planning approach for robot arm based on inverse kinematics and deep reinforcement learning. We divide the movement path of a robot arm into several segments, and adaptively employ IK-based or RL-based method for segments with different risk levels of potential collisions. Our approach integrates the fast calculation speed and high accuracy of IK-based method and the flexible collision avoidance ability of RL-based method. A high-fidelity digital twin model of the robot arm is built to support the training of our approach in the context of pick-and-place task, a commonly seen scenario in automatic assembly line. Compared to traditional reinforcement learning based method, our approach can reduce the difficulty of training RL models, improve the training speed with guaranteed performance in collision avoidance and path accuracy. Our work contributes to the practical deployment of reinforcement learning based path planning methods for digital twin-enabled robot arm in smart manufacturing.

One limitation of this work is that the geometric model of each object must be built in advance in our approach, thus the current object detection and location method may not be able to handle objects with complex shapes efficiently. Future work will focus on developing advanced object detection methods to extend the application of our approach in more complex environments.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support from National Key R&D Program of China (2022YFB4702400).

REFERENCES

[1] Bragança, S., Costa, E., Castellucci, I., & Arezes, P. M. (2019). A brief overview of the use of collaborative robots in industry 4.0: human role and safety. Occupational and environmental safety and health, 641-650.

[2] Bae, H., Kim, G., Kim, J., Qian, D., & Lee, S. (2019). Multi robot path planning method using reinforcement learning. Applied Sciences (Switzerland), 9(15). https://doi.org/10.3390/app9153057

[3] Bhuiyan, T., Kästner, L., Hu, Y., Kutschank, B., & Lambrecht, J. (2023). Deep-Reinforcement-Learning-based Path Planning for Industrial Robots using Distance Sensors as Observation. http://arxiv.org/abs/2301.05980

[4] Li, J., Pang, D., Zheng, Y., Guan, X., & Le, X. (2022). A flexible manufacturing assembly system with deep reinforcement learning. Control Engineering Practice, 118. https://doi.org/10.1016/j.conengprac.2021.104957

[5] Li, C., Zheng, P., Yin, Y., Pang, Y. M., & Huo, S. (2023). An AR-assisted Deep Reinforcement Learning-based approach towards mutual-cognitive safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, *80*. https://doi.org/10.1016/j.rcim.2022.102471

[6] Li, C., Zheng, P., Li, S., Pang, Y., & Lee, C. K. M. (2022). AR-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. Robotics and Computer-Integrated Manufacturing, 76. https://doi.org/10.1016/j.rcim.2022.102321

[7] Matulis, M., & Harvey, C. (2021). A robot arm digital twin utilising reinforcement learning. Computers and Graphics (Pergamon), 95, 106–114. https://doi.org/10.1016/j.cag.2021.01.011

[8] Zhou, D., Jia, R., & Yao, H. (2021). Robotic Arm Motion Planning Based on Curriculum Reinforcement Learning. 2021 6th International Conference on Control and Robotics Engineering, ICCRE 2021, 44–49. https://doi.org/10.1109/ICCRE51898.2021.9435700

[9] Wang, Y., & Kasaei, H. (2022). IPPO: Obstacle Avoidance for Robotic Manipulators in Joint Space via Improved Proximal Policy Optimization. http://arxiv.org/abs/2210.00803

[10] Luipers, D., Kaulen, N., Chojnowski, O., Schneider, S., Richert, A., & Jeschke, S. (2022). Robot Control Using Model-Based Reinforcement Learning With Inverse Kinematics. 2022 IEEE International Conference on Development and Learning, ICDL 2022, 244–249. https://doi.org/10.1109/ICDL53763.2022.9962215

[11] Zhang, Z., & Zheng, C. (2022). Simulation of Robotic Arm Grasping Control Based on Proximal Policy Optimization Algorithm. *Journal of Physics: Conference Series*, *2203*(1). https://doi.org/10.1088/1742-6596/2203/1/012065

[12] Zhong, J., Wang, T., & Cheng, L. (2022). Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics. *Complex and Intelligent Systems*, *8*(3), 1899–1912. https://doi.org/10.1007/s40747-021-00366-1

[13] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005

[14] Besl, P. J., & McKay, N. D. (1992, April). Method for registration of 3-D shapes. In Sensor fusion IV: control paradigms and data structures (Vol. 1611, pp. 586-606). Spie.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018, pp. 45-56.

[16] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. http://arxiv.org/abs/1707.06347

[17] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., &

Lange, D. (2018). *Unity: A General Platform for Intelligent Agents*. http://arxiv.org/abs/1809.02627