IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 January 2024, accepted 23 January 2024, date of publication 29 January 2024, date of current version 8 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3359268

RESEARCH ARTICLE

Enhancement of Control Performance for Degraded Robot Manipulators Using Digital Twin and Proximal Policy Optimization

SU-YOUNG PARK[1], CHEONGHWA LEE[2], HYUNGJUNG KIM[3], AND SUNG-HOON AHN[1,4]

[1]Department of Mechanical Engineering, Seoul National University, Seoul 08826, Republic of Korea
[2]Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Republic of Korea
[3]Department of Industrial Engineering, Konkuk University, Seoul 05029, Republic of Korea
[4]Institute of Advanced Machines and Design, Seoul National University, Seoul 08826, Republic of Korea

Corresponding authors: Sung-Hoon Ahn (ahnsh@snu.ac.kr) and Hyungjung Kim (hyungjungkim@konkuk.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grants funded by the Korean Government (MSIT) under Grant NRF-2021R1A4A2001824, Grant NRF-2021R1C1C2008026, and Grant NRF-2021R1A2B5B03087094.

ABSTRACT In this study, we propose a novel method for enhancing the control performance of a degraded robot manipulator by leveraging digital twins and proximal policy optimization, a specific deep reinforcement learning algorithm. Recently, various robotic technologies with high levels of controllability, safety, and reliability that incorporate the fourth industrial technology have been developed. Nevertheless, repairs or replacements owing to the performance degradation of sophisticated robot hardware or control systems are still time-, cost-, and manpower-consuming. To address these challenges, we propose a new strategy:1) approximate the unstable dynamic characteristics of six-degree-of-freedom low-performance robot manipulators to a digital twin with parameter tuning of a physics engine; 2) improve the accuracy and stability of reaching target points under diverse conditions through deep reinforcement learning using the domain randomization method; and 3) deploy a trained policy on an actual robot manipulator with degraded capabilities to validate the control performance improvement. Our method reduced the position error of a real robot manipulator by 63.0% and 39.0% compared to the built-in control method and the proportional-integral-derivative control method, respectively. Randomizing parameters in the physics engine of the digital twin during training allowed the method to simulate the imprecise motions of an actual degraded robot manipulator, facilitating the development of a more robust policy. Notably, our method has the potential to be applicable to all types of articulated robots, and presents a promising solution for maintaining performance while reducing long-term costs.

INDEX TERMS Degraded robot manipulator, control performance, digital twin, proximal policy optimization, reality-to-simulation reflection, reality gap.

I. INTRODUCTION

Robotics, traditionally limited to fields such as manufacturing, logistics, and services, has experienced an evolution driven by a rapid increase in technological innovation. Originally, these robotic entities were engineered with a primary focus on executing tasks characterized by their repetitive nature and required precision [1]. However, owing to recent robot technologies combined with Artificial Intelligence (AI), the Internet of Things, and Big Data technologies, the boundaries of robot use have disappeared [2], [3], [4], [5]. These next-generation entities offer unparalleled controllability, uncompromised safety standards, and reliable indicators. These characteristics enable flexible navigation and operation in complex, dynamic environments.

The associate editor coordinating the review of this manuscript and approving it for publication was Huiyu Zhou.

However, with improved robotics technologies, robots are increasingly at the forefront of increasingly demanding operational tasks, particularly those dominated by 3Ds, which are difficult, dangerous, and dirty [6]. Frequent repetition in these stressful environments can accelerate performance degradation, with many showing signs of mechanical fatigue before the expected life-cycle stage [7].

Because such a performance degradation of robot hardware has a negative impact on various productivity indicators, continuous condition monitoring and regular maintenance are essential. However, replacing or repairing the parts of deteriorated robots, such as motors, incurs enormous costs. The process downtime incurred while repairing robots further increases the losses. These maintenance processes rely on skilled experts, which leads to significant inefficiencies in robotic operation. Additionally, as the hardware structures and control systems of robots become more sophisticated, these tasks increasingly require more professionalism [8]. Several methods using simulation and optimization algorithms have been proposed to solve these problems in real robots [9], [10]. In particular, previous studies have emphasized bridging the simulation-to-reality (Sim2Real) gap, which is the error that occurs when simulation results are deployed as real robots. Many studies have been conducted to minimize the Sim2Real gap. In most cases, it was assumed that it is an ideal robot that accurately and stably follows the given commands. Simulation results that did not accurately reflect the dynamic characteristics of the actual degraded robot still had the Sim2Real gap problem. In other words, the reality-to-simulation (Real2Sim) gap has a significant impact on the Sim2Real gap [11].

To address these challenges, we introduce an efficient method to improve the control performance of degraded robot manipulators (DRMs) using a digital twin (DT) and deep reinforcement learning (DRL). This study makes several notable contributions, including the following.

1) Improvement of a DRM control performance based on a DT-DRL pipeline: The DRL-driven control policy trained in the DT pipeline improves the control performance of the DRM. The trained control policy ensured that the actual DRM, which had a large oscillation and static error owing to the performance degradation of the actuator, reached the target point stably and accurately. The DT-DRL pipeline also reduces the time required to train the DRL policy by using multiple agents. This can contribute to improving the productivity of industrial robots and popularizing service robots by simplifying the cost, time, and manpower-consuming replacement or repair processes of robot hardware.

2) Bridging Sim2Real gap by Real2Sim reflection process: This study introduced a novel approach to mimic the nonlinear dynamic characteristics of a six-degree-of-freedom (6)-DOF) robot manipulator in the DT. The complex dynamics of a DRM were effectively implemented by setting the physical parameter ranges within a physics engine. The control policy was trained using physical parameters randomly selected from a specific range that may contain the actual DRM dynamic characteristic values. This approach enhanced task performance during validation with the actual DRM. Our method streamlined the Real2Sim reflection process for articulated robots, particularly when dynamic modeling was not feasible. Through this process, we were able to bridge the Sim2Real gap.

3) Seamless deployment of the trained control policy in an actual DRM: We achieved direct, tuning-free deployment of our trained control policy into the actual DRM, leveraging position control through *damped least squares-inverse kinematics* (*DLS-IK*) for both the DT and real-world DRM. This seamless approach allows for the direct control of actuator joint values across different robotic platforms without the need for robot-specific task-space controls. Our method has been experimentally validated, demonstrating enhanced performance through stable and accurate operation of the actual DRM in real-world applications. This represents a universal solution with the potential to streamline the deployment of control policies in various types of robotic systems.

The remainder of this paper is organized as follows. Section II reviews previous studies on DRM and its applications in robot manipulators using DT and DRL. Section III introduces the proposed DT-DRL pipeline, which enhances robot control performance by bridging a Sim2Real gap using domain randomization. Section IV presents both qualitative and quantitative experimental evaluations of an actual DRM using a trained DRL control policy. Section V summarizes the conclusions of this study and discusses future work.

II. RELATED WORKS

A. DEGRADATION OF ROBOTS

The degradation of robot hardware, especially from wear and tear, affects the manipulability performance. Qiao et al. explored the effects of prolonged use of robotic joints and actuators, noting increased friction and decreased actuator efficiency during continuous operation [12]. Although preventive maintenance has addressed some issues, sudden mechanical failures still persist. Aliff et al. studied industrial robots under extreme conditions such as high temperatures and observed reduced operational efficiency and precision [13]. Despite the introduction of cooling systems, the feasibility of implementing additional hardware solutions in all the environments remains uncertain.

From a software perspective, Xiao et al. developed a framework to enhance the robustness of software for robotic control [14]. The seamless integration of hardware and software continues to be challenging. Aivaliotis et al. tackled the quality degradation from robot aging using machine learning to detect early signs of performance decline, thereby facilitating timely maintenance [15]. However, the effectiveness of the system depends on a massive amount of training data and only indicates when maintenance is required without performance enhancement.

In summary, algorithmic approaches to robot maintenance have not yet been extensively studied, and a more efficient and new approach is required.

B. ROBOT MANIPULATORS WITH DT AND DRL

Robot manipulators have been pivotal in numerous applications ranging from industry to healthcare. Recently, considerable research has been conducted on applying various new technologies of Industry 4.0, particularly DT and DRL, to robots.

DT, which provides virtual representations of physical objects, has proven to be significantly effective in real-time monitoring, system optimization, and predictive maintenance [16], [17], [18]. For instance, Kibira et al. used DT to predict wear and tear in robots [19]. Similarly, Aivaliotis et al. corrected errors in motion planning in real-time through synchronization between robots and DT [20].

DRL, where an agent trains through interaction with its environment, is particularly suited to robots with complex dynamics. Yang et al. successfully trained a 6-DOF robotic manipulator using heuristic DRL for tasks like object alignment [21], and Beltran-Hernandez et al. applied model-free DRL and transfer learning to high-precision tasks, demonstrating the transferability of simulated training to real-world operations [22].

Despite these advancements, the Sim2Real gap remains a significant challenge [9]. A policy that excels in simulation often underperforms in reality due to dynamic inconsistencies, external disturbances, or sensor inaccuracies. Tobin et al. argued that DRL through domain randomization in simulations has limitations in completely resolving the Sim2Real gap [23]. Additionally, the complexity of real-world environments, which often contain unpredictable variables and conditions do not present in simulations, further exacerbates the Sim2Real gap. Unlike controlled simulation settings, real-world scenarios may involve unmodelled interactions, non-linear dynamics, and varied environmental conditions, all of which can significantly impact the performance of DRL policies when applied outside of simulation environments.

In the field of robot manipulators, especially when integrating DT and DRL various methods have been explored, each with its own strengths and challenges. The Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy algorithm that combines ideas from Deterministic Policy Gradient (DPG) and Deep Q-Network (DQN). It excels in environments with high-dimensional action spaces, providing an efficient way to learn policies in continuous action domains. However, DDPG's performance is notably sensitive to its hyperparameter settings, such as the learning rate and the choice of the replay buffer size. This sensitivity can lead to challenges in achieving stable and consistent training results. Additionally, DDPG sometimes struggles with efficient exploration in the action space, as its deterministic policy can limit the diversity of actions explored during training.

To address some of these issues, the Twin Delayed Deep Deterministic policy gradient (TD3) was developed as an extension of DDPG. TD3 introduces key improvements like twin Q-networks to mitigate the overestimation of Q-values and delayed policy updates to reduce the variance of policy updates. While TD3 makes significant strides in improving stability over DDPG, it still inherits the fundamental challenge of hyperparameter sensitivity and can encounter difficulties in exploration, especially in complex environments with sparse rewards.

The Soft Actor-Critic (SAC) algorithm, another model-free, off-policy method, integrates an entropy-based approach for better exploration. The addition of entropy regularization to the reward encourages the policy to explore more diverse actions, making SAC particularly effective in environments where exploration is crucial for finding optimal policies. However, this added exploration comes at the cost of increased algorithmic complexity and computational overhead. The entropy term in SAC requires careful tuning, and the training process generally involves more sophisticated balance between exploration (i.e. entropy) and exploitation (i.e. reward maximization).

Despite the distinct advantages of these methods, they often encounter limitations in adaptation, particularly in environments with dynamic inconsistencies or when precise control is required. This is where Proximal Policy Optimization (PPO) distinguishes itself. PPO, with its policy update clipping mechanism, offers a stable and consistent training curve, which is beneficial in complex environments where maintaining a balance between exploration and exploitation is crucial. Additionally, the synchronous updates and objective functions in PPO contribute to its robustness and reliable convergence, making it a preferred choice in a broader range of applications.

The adaptability of PPO and its effectiveness in training under varying dynamic conditions or in high-precision scenarios have contributed to its growing use in the field. PPO demonstrates consistent policy improvement and the ability to handle a range of challenges, making it a strong choice for research and practical applications in advanced robotic systems.

III. METHODS

In this study, our goal was to train a control policy to move the tool center point (TCP) of the DRM to the target point. The primary challenges associated with DRMs include oscillation and static errors, which are often caused by aging and deterioration of the actuators. These issues can lead to significant errors in the TCP accuracy and result in unstable dynamic motion. In this context, we focused on motion generation as a target task to evaluate the control performance of the DRM.

To achieve this goal, we propose a DT-DRL pipeline (Figure 1). First, we assess the status of DRM and then replicate its unstable dynamics within the high-fidelity DT environment. A DRL-driven control policy is trained with domain randomization of various parameters to enhance the precision and stability of the DRM in reaching multiple target points. The results of our method are compared with those of a conventional proportional-integral-derivative (PID) control method by deploying a trained control policy on the DRM in the real-world.

Before adopting DRL, various algorithms were reviewed to improve the control performance of the DRM implemented

in our DT pipeline. Model predictive control (MPC) predicts the system behavior over a set range and makes control decisions based on optimization. By contrast, adaptive control adjusts the response to the observed behavior of the system by dynamically adjusting the parameters in real time. This is particularly useful in systems with uncertainties or changes [24]. MPC and adaptive control have their own advantages; however, for successful control, precise system modeling must be performed first. Therefore, in this study, we chose DRL, which trains the control system on its own by interacting with the environment, even though accurate modeling is not possible.

We selected PID control for performance comparison with our DRL control for two key reasons, aligning with the specific objectives and constraints of our study. Firstly, the widespread industry application of PID control provides a practical and established benchmark for our analysis [25]. This ensures our comparative study is grounded in a real-world context, crucial for evaluating the practical applicability of our DRL method in scenarios typically dominated by PID systems.

Secondly, considering the limitations in managing multiple control parameters and accurately modeling robot dynamics with DRMs, the straightforward yet robust characteristics of PID control become especially relevant. Unlike complex control algorithms, PID control requires minimal extensive parameter tuning or detailed system modeling, making it an ideal benchmark for assessing the performance of our DRL method. This simplicity enables focused evaluation of instability challenges in DRMs and the adaptive capabilities of DRL.

Our research aims to leverage DT physics engine parameters in DRL parameter randomization, aiding in the training and adaptation of the control system to the unstable characteristics of DRMs. By contrasting the outcomes of our DRL method with those obtained using PID control, we seek to showcase the advantages and enhancements our DRL approach brings to scenarios with complex and unstable systems. This comparison aims to underscore the enhanced capabilities of DRL and its potential for more refined control in comparison to conventional control methods (i.e. PID) in complex environments.

A. DT-DRL PIPELINE FOR ENHANCEMENT OF ROBOT CONTROL PERFORMANCE

In this study, we used NVIDIA® Omniverse Isaac Sim™ to implement the DT framework [26]. This comprehensive platform provides a high-fidelity physical simulation environment, making it suitable for simulating and assessing intricate robot dynamic systems. A notable advantage of this platform is its seamless integration with the NVIDIA® Isaac Gym, which significantly expedites the exploration and refinement of an optimal DRL policy [27]. Owing to their capability to engage in a parallelized simulation ecosystem, agents can rapidly train and explore optimal policies. Using Omniverse based on a Python backend, we can implement

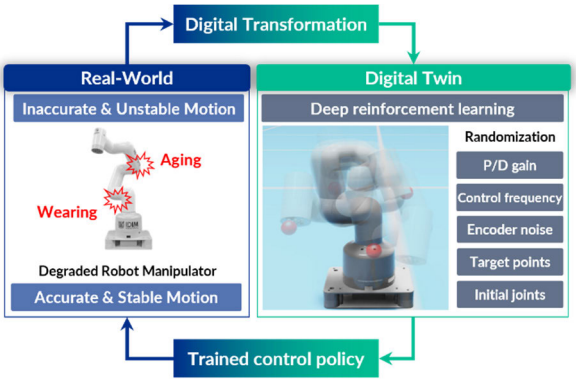

FIGURE 1. System overview of DT-DRL pipeline.

a DT-DRL pipeline by customizing various functions of the robots, control systems, and sensors.

The DT-DRL pipeline follows systematic steps (Figure 2): 1) conduct a comprehensive diagnosis of the proposed DRM hardware to discern its status and identify specific challenges that require resolution; 2) execute a digital transformation of the actual DRM within the DT platform and empirically set a range of the DRM control parameters in the DT based on the dynamic attributes of the actual DRM. 3) Randomization of both hardware and software parameters, along with the training conditions in the DT, to effectively address the dynamic properties of the DRM, which defies precise mathematical modeling; 4) training the DRL-driven control policy with multiple agents; 5) seamlessly deploying the trained control policy to a real robot manipulator, eliminating the need for any additional fine-tuning process; and 6) analyzing the result of the DRL policy to validate its control performance. The control performance of DRM can be easily and reliably improved using the DT-DRL pipeline.

B. HARDWARE DIAGNOSIS OF THE ROBOT MANIPUALTOR

In this study, we supposed Elephant Robotics· Mycobot 280-pi as the DRM because its inaccurate control performance and short lifespan. This robot manipulator consists of six servo motors, and the guaranteed payload of this equipment is only 250 g, which is very low compared with that of general industrial robot manipulators. Additionally, its lifespan is 500 h, which guarantees its low durability. Servo motors use a 12 binary digit encoder; thus, each joint angle resolution is 0.18°, and the theoretical control accuracy of the TCP is ±0.50 mm. Accurate angular velocity and control frequency data for elaborate control are not provided. Owing to unsophisticated geometrical modeling, steps between each link part, and insufficient torque of the motor, the control was slightly unstable (e.g., inaccurately reaching the input target points or causing oscillations during movement) (Figure 3). Considering these characteristics, we found that this robot manipulator was suitable as an extremely degraded robot manipulator to verify the improvement in the control performance.

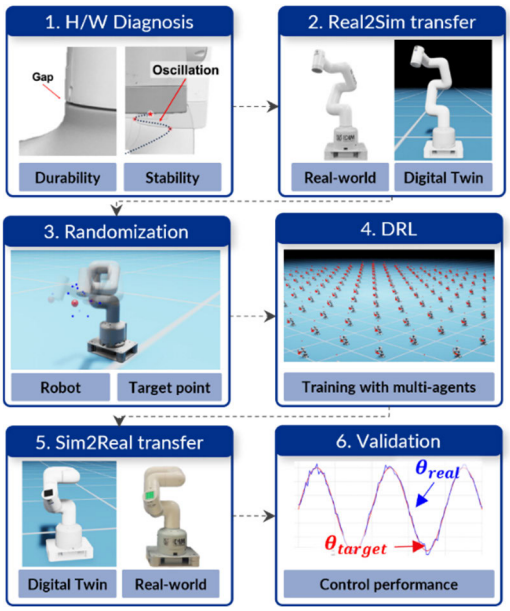

FIGURE 2. System workflow of the DT-DRL pipeline to enhance the control performance of the DRM.

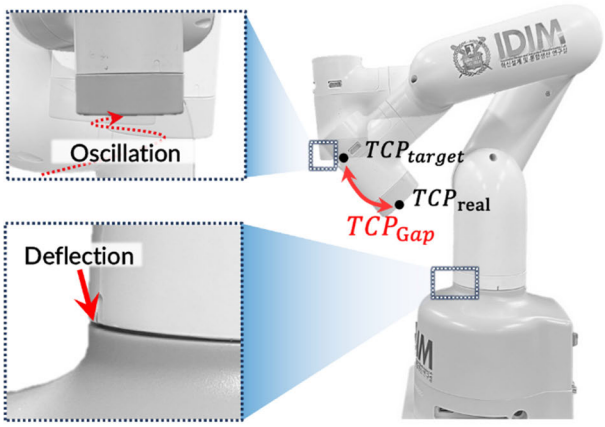

FIGURE 3. Critical problems of the DRM, deflection of links and oscillation around the target points.

C. DIGITAL TRANSFORMATION OF A DRM FROM THE REAL-WORLD TO THE DT

This section explains the digital transformation process used to reflect the dynamic characteristics of a DRM in the DT. To approximate the robot dynamics in the DT, the forward kinematics (FK) and inverse kinematics (IK) of the DRM were first defined.

In robotics research, particularly in the context of manipulators with multiple degrees of freedom DOF, precise and reliable modeling of hardware data is essential for accurate digital transformation into DT. One of the most widely used techniques for achieving this level of understanding is kinematic modeling, which primarily employs Denavit-Hartenberg (DH) parameters coupled with transformation matrices [28]. This mathematical tool offers a robust framework for capturing the geometric and spatial attributes of robotic systems. These parameters encapsulate critical information regarding the joints and links of the robot manipulators, such as their relative orientations and distances.

The hardware configuration and DH parameters of the DRM used in this study were illustrated in Figure 4 and Table 1, respectively. Each row in the table provides the DH parameters θ_i, d_i, a_i, and α_i that govern the transformation from one joint to its adjacent joint. The corresponding DH parameter value was converted to FK using the transformation matrix, and each joint value of the DRM for the pose of the target point was inversely calculated using *DLS-IK*.

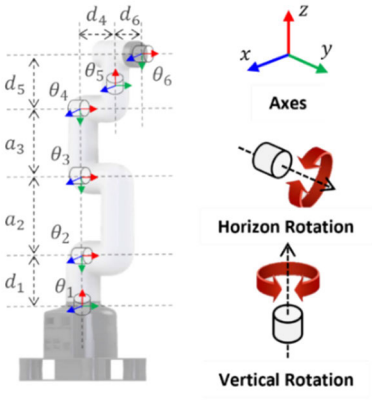

FIGURE 4. Hardware configuration of Mycobot 280-pi.

TABLE 1. DH-parameters of mycobot 280-pi.

Joint	Angle, θ_i (°)	Offset, d_i (mm)	Length, a_i (mm)	Twist, α_i (°)
1	q_1	131.2	0	-90.0
2	q_2	0	110.4	0
3	q_3	0	96.0	0
4	q_4	63.4	0	90.0
5	q_5	74.1	0	-90.0
6	q_6	45.6	0	0

The FK and IK of the robot manipulator are defined based on DH parameters [29], [30]. First, FK converts the input angle θ_{input} into the TCP pose in the Cartesian coordinate system based on a homogeneous transformation matrix T_i^{i+1} (1)–(5). $Rot_z(\theta_i)$ is the rotation matrix of θ_i around the z-axis, $Trans_z(d_i)$ is the translation matrix of d_i along the z-axis, $Trans_x(a_i)$ is the translation matrix of a_i along the x-axis, and $Rot_x(\alpha_i)$ is the rotation matrix of a_i around the x-axis. Each transformation matrix is in the form of 4×4, and the final homogeneous transformation matrix can be obtained by multiplying these matrices.

$$T_i^{i+1} = Rot_z(\theta_i) \times Trans_z(d_i) \times Trans_x(a_i) \times Rot_x(\alpha_i) \quad (1)$$

$$Rot_z(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$Trans_z\,(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$Trans_x\,(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$Rot_x\,(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_i) & -\sin(\theta_i) & 0 \\ 0 & \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The DRM, which consists of six joint drivers, can also be defined as FK (6)–(8).

$$T_{base}^{TCP} = T_{base}^{1} \times T_1^2 \times T_2^3 \times T_3^4 \times T_4^5 \times T_5^6 \times T_6^{TCP} \tag{6}$$

$$p_{TCP} = T_{base}^{TCP} * p_{base} \tag{7}$$

$$T_{base}^{TCP} = f\,(\theta) \tag{8}$$

where T_{base}^{TCP} is a homogeneous transformation matrix of the robot manipulator, $p = (x, y, z, \alpha, \beta, \gamma)$, and $\theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. The TCP pose in Cartesian coordinates TCP for the input joint value θ can be transformed by sequentially multiplying the transformation matrix of each joint from the base of the robot manipulator to the TCP pose in Cartesian coordinates p_{TCP} for the input joint value θ can be transformed. Thus, T_{base}^{TCP} can be expressed as a function of the joint value θ as $f\,(\theta)$.

In contrast to FK, IK can convert the TCP pose in the Cartesian coordinate p_{TCP} to each joint value θ using the Jacobian matrix $\mathbb{J}$ (9)–(10). Jacobian matrix $\mathbb{J}$ was calculated by taking the partial differentiation of each element of the TCP pose $x, y, z, \alpha, \beta, \gamma$. This provided a mathematical link between the joint and TCP velocities.

$$\Delta\theta = \mathbb{J}^{-1}\Delta p_{TCP} \tag{9}$$

$$\mathbb{J} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \cdots & \frac{\partial x}{\partial \theta_n} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \cdots & \frac{\partial y}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \gamma}{\partial \theta_1} & \frac{\partial \gamma}{\partial \theta_2} & \cdots & \frac{\partial \gamma}{\partial \theta_n} \end{bmatrix} \tag{10}$$

In this study, we used the *DLS-IK* method (11).

$$\Delta\theta_{DLS-IK} = \left(\mathbb{J}^T\mathbb{J} + \lambda^2\mathbb{I}\right)^{-1}\mathbb{J}^T\Delta p_{TCP} \tag{11}$$

where λ denotes the damping factor. The *DLS-IK* method introduces a damping factor λ to regularize the Jacobian matrix, particularly when it is near singular [31]. This ensures that the solution does not exhibit excessively high joint velocities. By adjusting the value of λ, this method can achieve a balance between the accuracy of the solution and its stability, thereby preventing erratic robot movements in challenging configurations. We set the value of λ to 0.05 for the *DLS-IK* of the proposed robot manipulator.

Finally, the parameters of the physics engine in the DT were set to mimic the unstable motion of an actual DRM. In this DT platform, we can control the joint angles of the robot manipulators with torque values τ based on the position control (12).

$$\tau = P_{DT} \cdot e + D_{DT} \cdot \dot{e} \tag{12}$$

where $P_{DT}, D_{DT}, e,$ and $\dot{e}$ are the stiffness gain, damping gain, joint angle error between θ_{target} and θ_{actual}, and angular velocity error between $\dot{\theta}_{target} - \dot{\theta}_{actual}$, respectively.

The stiffness P_{DT} and damping parameter D_{DT} of the angular drive were the most dominant parameters in the joint control system of *PhysX5*. This control method is similar to the conventional PID control method, but it should be noted that the torque control of a DT robot does not accurately reflect the torque control of an actual robot. This robot control method in a physics engine is merely a means of simulating the dynamic characteristics of an actual robot [32]. In the physics engine, the higher P_{DT}, the faster the joint values of the robot articulations reach the target values. If the P_{DT} value is extremely low, then the robot cannot withstand the weight of each link and rotates in the direction of gravity. The D_{DT} value determines the smoothness of the approach to a target point. If D_{DT} is significantly lower than P_{DT}, the TCP of the robot oscillates significantly near the target point. Conversely, if this value is higher than P_{DT}, the robot cannot move.

In the case of a general robot manipulator, when the target joint value is commanded, it reaches the target point smoothly and accurately, without oscillation. To implement these ideal dynamic characteristics of the robot manipulators in the DT, P_{DT} value was set to be very high (e.g., 10^{10}), and the D_{DT} value was set to this value divided by $10^1 \sim 10^3$. However, we need to set the P_{DT} and D_{DT} values outside the normal range to implement the phenomenon of not reaching the target point owing to the discrete motion and deflection caused by the oscillation of a DRM in the DT.

Nevertheless, it is difficult to accurately model the unpredictable dynamic characteristics of a DRM using only P_{DT} and D_{DT} values in the DT. Therefore, in this study, we selected an appropriate range of P_{DT} and D_{DT} values to behave similar to the unstable and inaccurate dynamic characteristics of an actual DRM. Through this process, we were able to implement the complex dynamic characteristics of the actual DRM in the DT pipeline, which bridges the Real2Sim gap by enabling more realistic simulation and DRL training.

D. PROXIMAL POLICY OPTIMIZATION IN DT-DRL PIPELINE

DRL blends deep learning and Markov decision processes (MDPs) to enable agents to optimize their decision-making capabilities by interacting with their environment [33]. MDPs define the states, actions, and rewards of the environment to minimize or maximize objective functions [34]. In DRL, agents aim to maximize cumulative rewards over time. Rewards, denoted by r, indicate the quality of action a, with positive values promoting good decisions and negative values deterring poor decisions. Using a policy π, which correlates states s to actions, and deep neural networks (DNNs) to capture intricate state-action relationships, the agent

continually refines its decisions by interacting with its environment, receiving feedback, and adjusting its DNN parameter θ and policy π.

In this study, we used PPO algorithm as the DRL to improve the control accuracy and stability of the DRM (Figure 5) [35]. Unlike DDPG and TD3, which are sensitive to hyperparameter settings, they occasionally struggle with exploration. PPO provides a more stable learning curve by clipping policy updates and ensuring consistent improvements. While A3C utilizes asynchronous updates that can introduce variance, synchronous updates, and objective functions of PPO lead to more robust and reliable convergence [36]. Considering the stability and simplicity of training, PPO was determined that using PPO was the most appropriate.

The main feature of the PPO algorithm is the simultaneous training of the policy and value networks; the agent's next action a_{t+1} is returned directly to the policy network. In the DT environment, the simulated results based on the physics engine are saved in Trajectory $=<s_t, a_t, r_{t+1}, s_{t+1}>$ and stored in the replay buffer. When N trajectories accumulate in the replay buffer, they are input into the policy and value networks in the form of a mini-batch. Each network is updated using mini-batch information and the policy network returns the next action a directly to the environment using the updated policy. In addition, each model was updated based on the return values of each DNN. First, the policy network is updated using the main objective functions $L^{clip}(s, a, \theta_{old}, \theta)$ in the PPO algorithm (13).

$$L^{clip}(s, a, \theta_{old}, \theta) = \hat{\mathbb{E}}\left[min\left(\begin{array}{l} \frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_{old}}(a\,|\,s)} A^{\pi_{\theta_{old}}}(s, a), \\ clip\left(\frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_{old}}(a\,|\,s)}, 1-\varepsilon, 1+\varepsilon\right) A^{\pi_{\theta_{old}}}(s, a) \end{array} \right) \right] \tag{13}$$

The term $\hat{\mathbb{E}}[\cdot]$ implies the calculation of the average based on certain probabilities. In this context, $\pi_t(a\,|\,s)$ represents the chance of selecting a specific action based on the current state and policy at time step t, and $\pi_{t+1}(a\,|\,s)$ indicates the next version of the policy. The parameters in θ facilitate deciding the action based on state s, whereas θ_{old} is a set of fixed parameters used to choose actions in the current policy step. These values remained the same when the policy was updated. Function $clip(\cdot)$ ensures that updates to the policy do not stray too far from the original using the probability ratio $R_t(\theta) = \frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_{old}}(a\,|\,s)}$ and the clipping ratio ε. It also uses advantage function $\hat{A}$ to adjust its policy updates (14). Advantage function $\hat{A}$ quantifies the relative efficacy of an agent's action by contrasting its expected outcome with the average outcome of that state.

$$\hat{A}^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \tag{14}$$

The generalized advantage estimator (GAE) λ balances the estimation accuracy, weighing immediate versus future rewards with discount factor γ. Higher λ values induce a focus on long-term returns, whereas lower λ values emphasize immediate returns. This balance is pivotal for the agent's exploration–exploitation strategy during policy optimization. Moreover, δ_{t+l}^V represents the difference between the projected value of the current state and the value of the succeeding state, with the exponent l indicating the foresight of the estimator. The updated policy π'_θ with gradient ∇L^{clip} is then returned to the policy network.

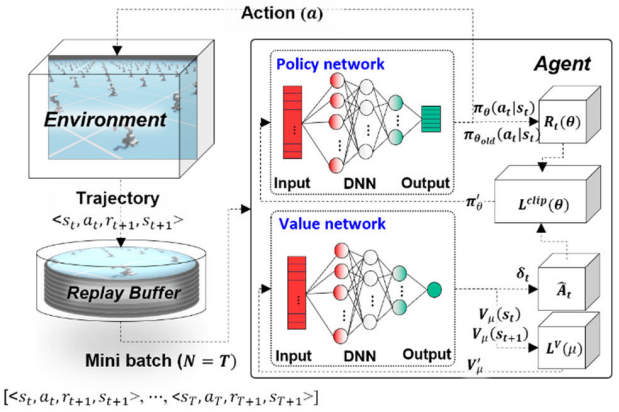

FIGURE 5. PPO architecture of DT-DRL.

The value network is updated using an objective function $L^V(\mu)$ (15).

$$L^V(\mu) = \hat{\mathbb{E}}\left[\,|\hat{V}_\mu^{target}(s_t) - V_\mu(s_t)\,|\,\right] \tag{15}$$

The objective function, $L^V(\mu)$ quantifies the discrepancy between the predicted and target values. The target value $\hat{V}_\mu^{target}(s_t)$ is computed using the immediate reward r_{t+1} and discounted estimated value of the subsequent state $V_\mu(s_{t+1})$. The difference between this target value and the current estimated value $V_\mu(s_t)$ is the temporal difference error, which aims to minimize the objective function. Subsequently, the updated value V'_μ with gradient $\nabla L^V(\mu)$ is returned to the value network. This process is repeated for a given time step and the agent is trained to determine the optimal policy.

E. DEPLOYMENT OF THE TRAINED DRL POLICY INTO A REAL ROBOT MANIPULATOR

We used the default communication method without constructing a separate pipeline to deploy the DRL policy trained in the DT-DRL to a real robot manipulator (Figure 6). The workstation where the control policy was trained with the DT-DRL was located, and the micro processing unit (MPU) of the robot (Raspberry pi-4) were connected via Wi-Fi and socket communication. When accessing a single serial port, we synchronized the threads of each control command and varied the access frequency between the commands to prevent conflicts as much as possible. Then, deployment proceeded in the following order:1) the target TCP points TCP_{target} received from the local server; 2) the input θ_{target} was converted into a control signal, and the servo motor of each joint was controlled through the encoder; 3) after each joint was controlled,

the current TCP pose TCP_t of the real robot was returned and transmitted to the DT through the MPU and server; and 4) the trained DRL control policy inserted TCP_t into the trained neural network, selected the next action, and delivered it back to the robot. This process uses a closed-loop control system with feedback control until the TCP position error between TCP_{target} and TCP_t converges below the threshold.

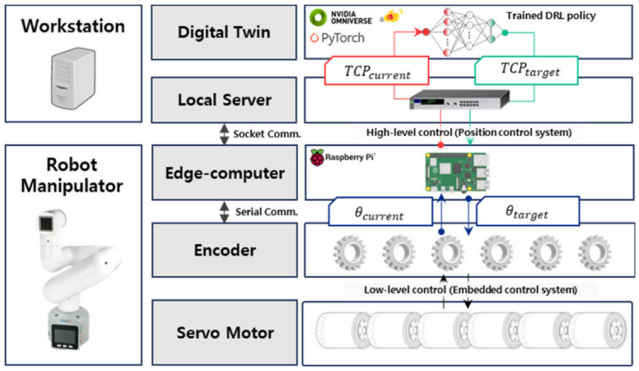

FIGURE 6. Control system architecture of the actual DRM operated by the pre-trained control policy.

IV. EXPERIMENTS AND RESULTS

In this study, we utilized high-performance computing hardware and software to accelerate high-fidelity physical simulations (Table 2). To use the Omniverse platform, a GeForce RTX 3090 graphics processing unit (GPU) was used, and a backend program was configured based on the compute unified device architecture (CUDA) library to accelerate rendering and DRL. In addition, we used the latest version of Isaac Sim, which provides the basic tools required for robot simulation.

A. TARGET TASK AND EXPERIMENTAL SET-UP

To validate the proposed DT-DRL system, we assigned a target task for the DRM in the DT and the real world. The most significant problems with DRMs are deflection and oscillation owing to aged and degraded motors, resulting in a large TCP error (i.e. accuracy issue) and unstable dynamic motion (i.e. stability issue). To solve these problems, an experimental environment was set (Figure 7). Based on the base of the DRM in the Cartesian coordinate system, target TCP points were randomly generated within the range of $130 \leq x \leq 230$ mm, $-100 \leq y \leq 100$ mm, $150 \leq z \leq 250$ mm (Figure 7(a)). In addition, considering the limitations of the robot hardware, a radius of 10 mm from the target TCP point was set as the threshold d_{thresh} (Figure 7 (b)). The TCP moved according to the same specific sequence to compare the control algorithms (Figure 7 (c)). A successful bonus was granted when $d_t < d_{thresh}$ was satisfied, and the target points were reset. When the target points were reset, the robot TCP moved from that location towards the target points, and when a collision occurred during this process, the positions of the targets and the pose of the DRM were initialized.

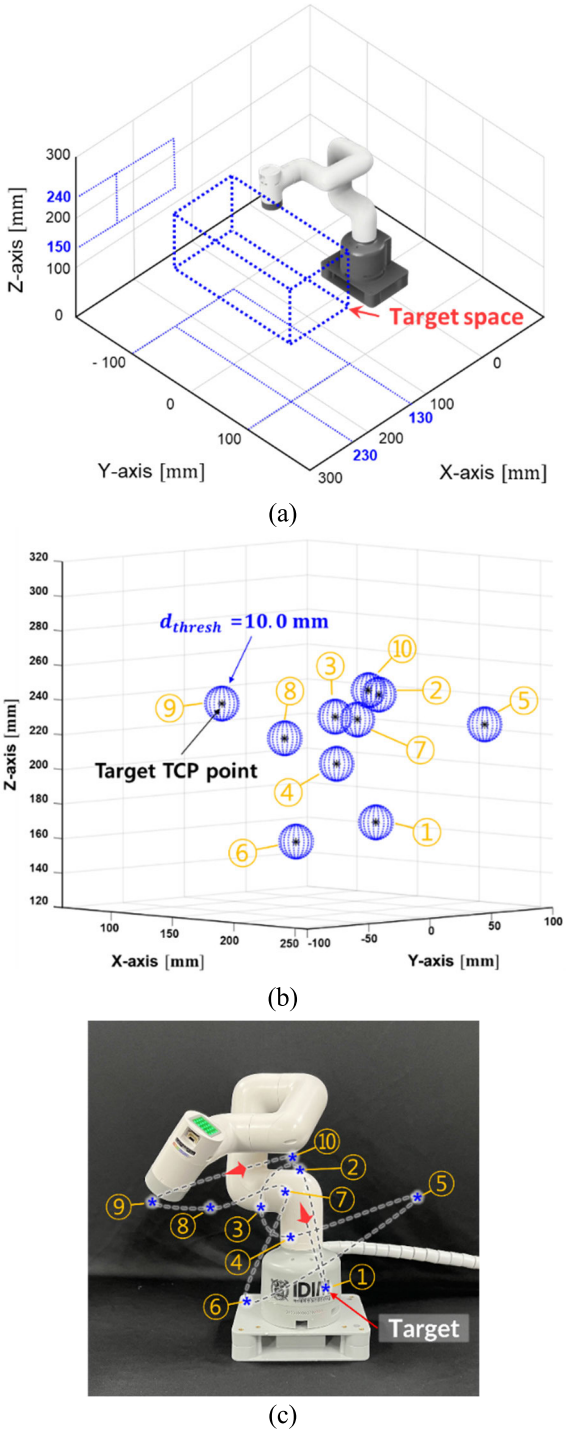

FIGURE 7. Experimental setup to validate the trained control policy in the real-world. (a) Target space where target points were randomly generated, (b) visualization of the targets (black star) and acceptable range (blue sphere) the in the Cartesian coordinate system, and (c) validation setup with random target points in the real-world.

B. PERFORMANCE COMPARISON IN NORMAL AND DEGRADADED STATUS

Before confirming the performance of control algorithms, the performance degradation of the target robot was quantitatively evaluated. The robots were used for less than 10 h

TABLE 2. Specifications of hardware and software for the DT-DRL pipeline.

Component		Specification
H/W	CPU	Intel Core i9-9900KF
	Memory	39.1 GB DDR4 RAM
	Graphics card	NVIDIA GeForce RTX 3090
	Storage	512 GB SSD
S/W	Operating system	Ubuntu 18.04 LTS
	Graphic driver	525.60.11
	CUDA	12.0
	Isaac Sim	2022.2.0

FIGURE 8. Movement results of normal and degraded robot manipulator for the same target point.

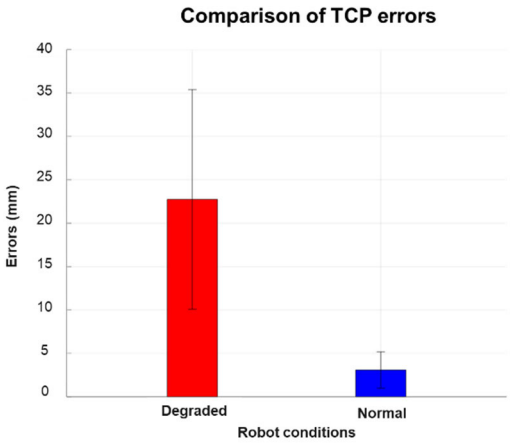

FIGURE 9. TCP error comparison between the degraded and normal robot manipulator.

(Normal) and more than 300 h (Degraded), respectively, and moved 10 positions in order. The TCP error at each target point was measured and these values were compared.

As a result of the experiment, even for the same target point, errors were so severe that they were visible. In the case of DRM, it was seen that the TCP error occurred significantly compared to the manipulator in normal status (Figure 8). Additionally, the TCP error was measured for all 10 target points and the average and standard deviation were compared. As a result, the average TCP error of the robot manipulator in normal status was 3.09 mm, while the average TCP error of DRM was 22.7 mm, a difference of about 7.34 times (Figure 9). Considering these results, we judged that the robot manipulator was clearly showing performance degradation, and that the algorithm performance comparison conducted later for the corresponding DRM was reasonable.

C. PERFORMANCE COMPARISON ACCORDING TO PD VALUES IN PhysX5

The unstable characteristics of DRM are illustrated by how the motion characteristics of an articulated robot manipulator change based on the PD value in *PhysX5*. To explore this, a suitable range of PD values was selected for domain randomization. This technique involves extracting random values within a specified range for use in DRL.

In *PhysX5*, the P value influences the stiffness of motion, while the D value affects its smoothness. For a standard robot, PD values are set to simulate accurate and smooth motion. However, in the case of DRM, there needs to be a TCP error, which is the discrepancy between the target point and the TCP, along with oscillation characteristics near the TCP. The TCP trajectory was measured under three conditions: 1) $P_{DT} = 10^4$ and $D_{DT} = 10^3$, 2) $P_{DT} = 10^3$ and $D_{DT} = 50.0$, and 3) $P_{DT} = 200$ and $D_{DT} = 10.0$ (Figure 10). In the first scenario, the motion closely resembled that of a normal robot with minimal TCP error and oscillation. The second scenario showed large oscillations due to a high P value relative to D, but the TCP error was low, enabling relatively accurate targeting. The third scenario, however, exhibited large TCP errors and oscillations, with the motion characteristics failing to reach the target point and moving to the next one instead.

To accurately simulate the dynamic characteristics of DRM, the PD values in *PhysX5* were chosen to be within $50.0 < P_{DT} < 200$, $(P_{DT}/50.0) < D_{DT} < (P_{DT}/10.0)$, a range that includes these specific values.

D. TRAINING OF CONTROL POLICY BY REFLECTING THE CHARACTERISTICS OF THE ACTUAL DRM

The outlined pseudocode describes the entire training process of the DRL control policy by reflecting the characteristics of an actual robot to stably and accurately move the TCP of the DRM to the target point (Algorithm 1). Domain randomization introduced variability in the simulation parameters, thereby enhancing the generalization ability of the model during the DRL training process. This technique allows the flexible and powerful adaptation of trained policies, especially when facing unpredictable real-world situations, contributing significantly to bridging the Sim2Real gap.

Initially, the algorithm randomizes the environmental parameters, such as control frequency, stiffness, and damping, and sets the stage for the manipulation task (Lines 1–3). The ranges of these parameters were selected by considering the dynamic characteristics of an actual robot. State observation s, action a and reward r are initialized. The values of P_{DT} and D_{DT} values were randomly assigned within a set range to imitate the oscillation and static error of the DRM. In addition, the control frequency $freq_{control}$ was randomized to ensure that the trained control policy could flexibly respond

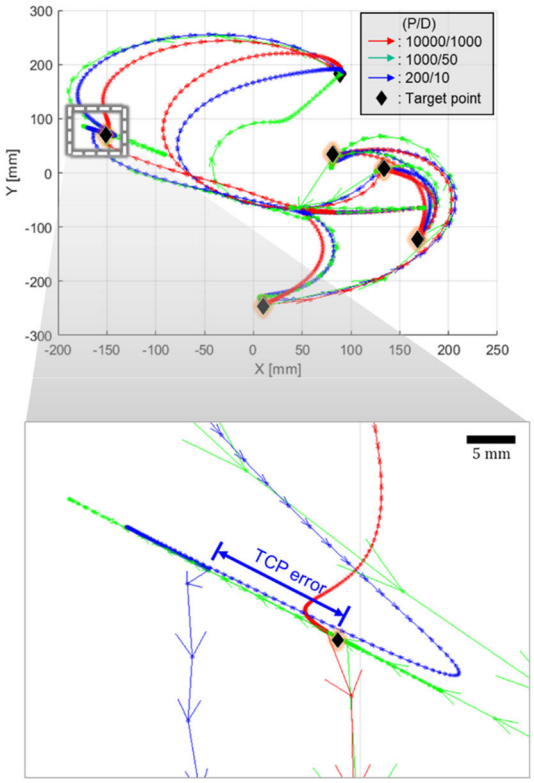

FIGURE 10. TCP trajectories of a robot manipulator according to PD values in PhysX5 (Top view).

to delays that may occur when transmitting commands to the actual DRM. For each episode, the algorithm followed these steps within the maximum timestep. At the beginning of each episode, the current joint angle θ_t is reset to θ_{init} (Lines 4-5). Additionally, episodic parameters, such as the target TCP points TCP_{target} and joint angle noises θ_{noise} are randomized, introducing variability and making the task more challenging, but reinforcing the DRL policy in unpredictable environments (Line 6). Throughout the episode, the algorithm used the *DLS-IK* method to compute the change in the joint angles $\Delta\theta_{DLS-IK}$ required to approach the target TCP. $\Delta\theta_{DLS-IK}$ is then combined with an action-based adjustment $\Delta\theta_t$ and joint noise θ_{noise} to update the joint angles of the DRM in the next step (Lines 7–10). If the DRM encounters a collision during its movement, a penalty is applied to the reward and the episode continues without updating the position. This prevents the robot from taking action, which leads to collisions (Lines 11–13). If the current TCP position error d_t is less than a predefined TCP position error threshold d_{thresh}, a success bonus $r_{success}$ is added to the reward, which promotes actions that reduce the error, and the episode is reset (Lines 14–16). If none of the above applies, penalties are applied based on the remaining error r_{error} between the current and target TCP position and the angular velocity r_{vel}. These penalties ensure that the robot moves without an excessive speed. After each action, the time counter increases (Lines 17–19). Finally, if the DRM does not reach its target TCP points within the maximum episode time T_{epi}, a reset penalty r_{epi} is applied, which encourages the robot to find the target faster in future attempts (Lines 20–21). Upon completing the maximum allowed time steps, the DRL algorithm would have trained an optimal policy $\pi_{opt}(s|a)$ that dictates the best actions in any given state to move the TCP towards its target TCP points with minimal penalties.

Algorithm 1 DRL-Driven training Process of the Robot Manipulator Joint Control Using

Input: Initial joint angles $\boldsymbol{\theta_{init}}$, Max total time step $\boldsymbol{T_{max}}$, Max episode time step $\boldsymbol{T_{epi}}$, TCP position error threshold $\boldsymbol{d_{thresh}}$, Collision penalty $\boldsymbol{r_{collision}}$, Success bonus $\boldsymbol{r_{success}}$, TCP position error penalty $\boldsymbol{r_{TCP}}$, Angular velocity error penalty $\boldsymbol{r_{vel}}$, Episode reset penalty $\boldsymbol{r_{epi}}$
Output: Optimal policy $\boldsymbol{\pi_{opt}(s|a)}$
Parameters: Control frequency $\boldsymbol{freq_{control}}$, Stiffness $\boldsymbol{P_{DT}}$ and damping $\boldsymbol{D_{DT}}$ in the physics engine, Target TCP point $\boldsymbol{TCP_{target}}$, Current time step $\boldsymbol{t}$, Current TCP position error $\boldsymbol{d_t}$,Current TCP pose $\boldsymbol{TCP_t}$, Joint angle noise $\boldsymbol{\theta_{noise}}$, action values of DRL $\boldsymbol{\Delta\theta_t}$, Change of joint angles by *DLS-IK* $\boldsymbol{\Delta\theta_{DLS-IK}}$
Procedure:
1: **DT-DRL** policy to reach the target points accurately and stably:
2: Randomize $\boldsymbol{freq_{control}}$, $\boldsymbol{P_{DT}}$, $\boldsymbol{D_{DT}}$
3: Initialize MDP parameters s, a, r
4: **while** $t < T_{max}$:
5: $\theta_t = \theta_{init}$
6: Randomize $\boldsymbol{TCP_{target}}$, $\boldsymbol{\theta_{noise}}$
7: **while** $t < T_{epi}$:
8: $\Delta\theta_{DLS-IK} = DLS - IK(TCP_{target}, TCP_t)$
9: $\Delta\theta_t = a_t$
10: $\theta_{t+1} = \theta_t + \Delta\theta_{DLS-IK} + \Delta\theta_t + \theta_{noise}$
11: **If** collision occurs:
12: $r- = r_{collision}$
13: **continue**
14: **If** $d_t < d_{thresh}$:
15: $r+ = r_{success}$
16: **continue**
17: $r- = r_{TCP}$
18: $r- = r_{vel}$
19: $t+ = 1$
20: $r- = r_{epi}$
21: **return** $\pi_{opt}(s|a)$

Upon a more detailed examination of the previously mentioned process, we defined the components of the MDP and hyperparameters of the training process above. First, the P_{DT} and D_{DT} values were set 50.0 to 100 and $P_{DT}/10.0$ to 50.0 to mimic the motion similar to that of the actual DRM. Within these value ranges, the DRM in the DT oscillates near the target points and does not accurately reach the target joint values. The actual control frequency of the joint value through the serial port of DRM was set to 0.02s, and the actual control frequency was between 0.02 and 0.03s. Considering this, the control frequency $freq_{control}$ was randomly assigned between 30.0 and 60.0Hz.

The state observation value s_t consists of the current time step t, joint angle change $\Delta\theta$, current joint angle θ_t, joint angular velocity $\dot{\theta}_t$, current TCP pose TCP_t, TCP

pose at the target point TCP_{target}, collision buffer C, and success buffer S (16).

$$s_t = \left[t, \Delta\theta, \theta_t, \theta_{target}, \dot{\theta}_t, TCP_{target}, TCP_t, d_t, C, S\right] \quad (16)$$

where the size of the single state s_t was 28. Collision buffer C checks for DRM collisions, whereas success buffer S evaluates whether the TCP position error d_t between TCP_t and TCP_{target} falls below the TCP position error threshold d_{thresh}. Considering the performance decrease in the real-world, d_{thresh} was set to 5.00mm, which is difficult to reach for the DRM.

We also defined a control algorithm based on the robot joint control (17)–(19). We set a to be the change in the joint value $\Delta\theta$ per unit time dt. By adding $\Delta\theta$ to the current joint value of DRM θ_t, the PPO policy allows the direct control of the joints of the DRM to the next joint state θ_{t+1}. Thus, $\Delta\theta_{DLS-IK}$ was added, and the joint change was calculated using *DLS-IK* for more stable control. θ_{noise}, the joint angle noise, was also added to respond more flexibly to the unpredictable conditions.

$$\theta_{t+1} = \theta_t + \Delta\theta_{DLS-IK} + \Delta\theta_t + \theta_{noise} \quad (17)$$

$$a_t = \Delta\theta_t = [\Delta\theta_1, \Delta\theta_2, \Delta\theta_3, \Delta\theta_4, \Delta\theta_5, \Delta\theta_6] \quad (18)$$

$$\theta_{noise} \sim \mathcal{N}\left(0, \sigma^2\right) \in \mathbb{R}^6 \quad (19)$$

where $\sigma = 5.00$, indicating that each joint angle has a noise of ± 5.00 based on 0 and follows a Gaussian distribution.

Based on a 6-DOF robot manipulator with similar specifications, the maximum angular velocity was set to 30.0°150°/s; It was theoretically possible to rotate between 0.50°5.00°/frame in DT, where the control frequency was between 30.0 and 60.0Hz, and frames per second was 60. The reward values r_1 and r_2, represent the penalties for the Euclidean distance between the current TCP and target point at each time step and the angular displacement per unit time, respectively (20) and (21). Each of these penalties guides the TCP to move in the direction of the target point and prevents it from moving at an unnecessary or excessive velocity. r_3 is a penalty given when the time step exceeds the maximum episode time step and guides the DRM to reach as many target points as possible within the episode time step (22). r_4 and r_5 are the penalties and bonuses for collision and success, respectively, through which the DRM avoids collisions between its links or with the ground and allows accelerated reach to target points (23) and (24). To avoid excessive bias for a specific reward value, each r is multiplied by a scale factor w. The values of $w_1, w_2, w_3, w_4,$ *and* w_5 were set to 0.10, 0.10, −5.00, 10.0, and 10.0, respectively (25).

$$r_1 = -d_t \quad (20)$$

$$r_2 = -(\theta_t - \theta_{t-1})/dt \quad (21)$$

$$r_3 = \begin{cases} -1, & if\ t > T_{epi} \\ 0, & otherwise \end{cases} \quad (22)$$

$$r_4 = \begin{cases} -1, & if\ collision\ occurs \\ 0, & otherwise \end{cases} \quad (23)$$

$$r_5 = \begin{cases} +1, & if\ d_t < d_{thresh} \\ 0, & otherwise \end{cases} \quad (24)$$

$$r_t\left(s_t, a_t\right) = w_1 r_1 + w_2 r_2 + w_3 r_3 + w_4 r_4 + w_5 r_5 \quad (25)$$

Subsequently, hyperparameters were adjusted (Table 3). We set the learning epoch and mini-batch size to eight. The policy and value network of the PPO algorithm had three hidden layers, and the clipping rate ε was set to 0.10 to ensure stable training. We set the value of GAE λ to 0.95 To increase the exploration component of the PPO policy, we set the value of GAE λ to 0.95, enabling a wider range of actions to move flexibly under various conditions. A single episode consisting of 250 time steps out of 5000 time steps was executed during a single training session. DRL training was conducted simultaneously using 512 agents.

TABLE 3. Hyperparameters for PPO algorithm of the DT-DRL pipeline.

Hyperparameters	Values
Learning epochs	8
Mini-batch	8
Number of hidden layers of policy network	3
Number of hidden layers of value network	3
Clipping ratio (ε)	0.10
GAE (λ)	0.95
Discount factor (γ)	0.99
Max time step	250
Total time step	5000
Number of training agents	512

E. TRAINING FOR MOTION GENERATION OF A DRM USING DRL

To train the DRL policy, DRMs were virtualized in the DT pipeline and the target TCP points (blue points) were randomly generated within a specified range (Figure 11(a)). Subsequently, one shared policy and value network were simultaneously simulated and updated with 512 multi-agents for 5000 time steps (Figure 11(b)). The TCP trajectories of the DRM (black line) at the beginning of the training appeared to move in no specific direction (Figure 11(c)). When the control policy after 20 episodes was applied to the DRM after 20 episodes, the TCP reached the target TCP points accurately and stably, and no unnecessary or protruding behavior was detected (Figure 11(d)). The training results showed that the DRL policy was successfully trained to control DRM with accurate and stable motion.

To provide a more quantitative evaluation of DRL training results, we analyzed the accumulated rewards and time steps per episode (Figure 12). These metrics provide an intuitive understanding of how well a trained policy performs, as opposed to individual loss curves. Furthermore, DRL loss functions are often complex and can be challenging to interpret. Especially, PPO involves multiple components like policy loss, value loss, and entropy, making its interpretation less straightforward than in more conventional supervised learning scenarios. Therefore, to validate trained DRL policy,

it is generally more informative to look at two performance metrics that we selected rather than training dynamics.

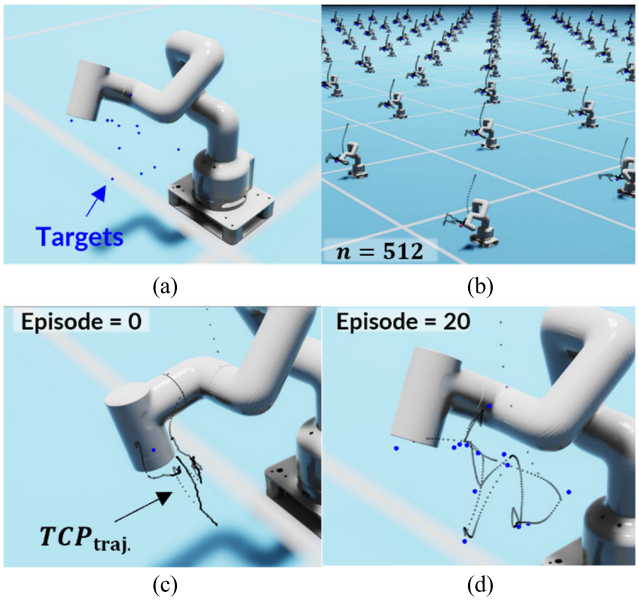

FIGURE 11. Training process of the robot manipulator using DT-DRL pipeline. (a) Random target points that the TCP should reach, (b) DRL-driven control policy training with multi-agents, (c) TCP trajectory at the initial training episode, and (d) TCP trajectory at the final training episode.

The training results were also compared based on the presence (Real2Sim) or absence (N/C Real2Sim) of the considered P_{DT} and D_{DT} values, which were set to generate motions similar to those of the actual DRM. First, the accumulated reward of the result trained with P_{DT} and D_{DT} values set to mimic the dynamic characteristics of the actual DRM (blue line) consistently increased as the time step progressed, converging to a value higher than -5.00 at approximately 2000 time steps. In contrast, when the P_{DT} and D_{DT} values were not considered (red line), the reward initially increased, but exhibited a continuous decline after approximately 1000 time steps (Figure 12(a)). Even in the results of the time step per episode, when trained with appropriate P_{DT} and D_{DT} values (red line), it can be observed that the episode time step continued to decrease as training progressed and converged within 400 time steps. This means that the target points were reached from the initial pose of the DRM at a faster rate than at the beginning of the training.

However, when the policy was trained without considering P_{DT} and D_{DT} (red line), the time step per episode temporarily decreased at the beginning of the training and then increased rapidly after 1200 time steps, confirming that it was trained in the wrong direction (Figure 12(b)).

Based on these results, we verified the influence of similarly reflecting the dynamic characteristics of the actual DRM in the DT on training success. It was confirmed that the process of setting the range of parameters that determines the dynamic characteristics of a robot in a physics engine has a significant impact on the convergence of training.

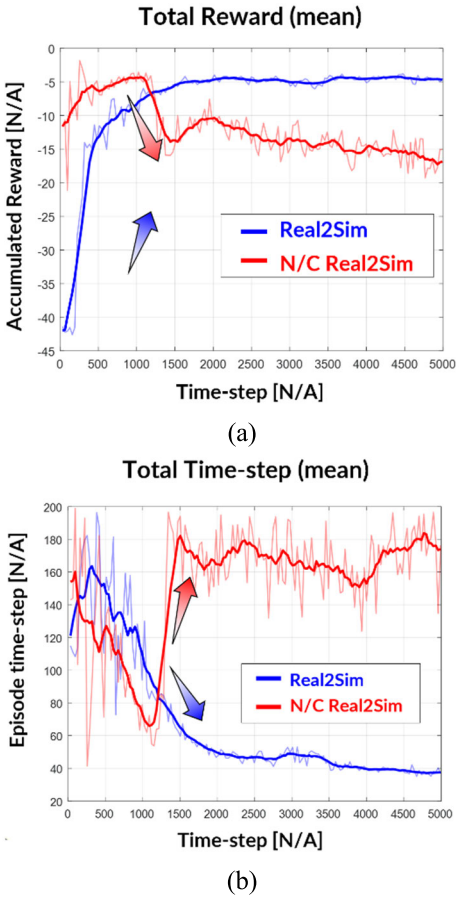

FIGURE 12. Training results of DRL with or without consideration of P_{DT} and D_{DT} values in the DT. (a) Accumulated reward and (b) total time step per episode.

F. REAL-WORLD VALIDATION ON THE DRM WITH THE TRAINED CONTROL POLICY

To validate the performance of the control policy trained with the DT-DRL pipeline, a comparative validation was conducted with other control methods: 1) the default control method of the DRM, 2) conventional PID control method, and 3) proposed DT-DRL control method.

The default control method is an open-loop control method in which only the joint value for the target TCP points is commanded once. The PID control is a closed-loop control that adds the control input θ_{input}, which is the sum of the error between the target and current joint angles e_t, accumulated error $\sum_{\tau=0}^{t} e_\tau \cdot dt$, and change in error $(e_t - e_{t-1})/dt$, to the current joint angle θ_t (26), (27). Each value is multiplied by the coefficients K_P, K_I, K_D to avoid bias towards one value.

$$\theta_{t+1} = \theta_t + \theta_{input} \tag{26}$$

$$\theta_{input} = K_P \cdot e_t + K_I \sum_{\tau=0}^{t} e_\tau \cdot dt + K_D \cdot \frac{e_t - e_{t-1}}{dt} \tag{27}$$

where $K_p = 0.40$, $K_I = 0.01$, $K_D = 5.00 \times 10^{-3}$, and dt was set to 0.02s. Each PID gain value was empirically tuned through iterative experiments. The gain values were fine-tuned by gradually adjusting them and observing the

response of the system until a proper balance between the stability and speed of the response was achieved.

The experimental data from the default control method showed that the trajectory of the TCP maintained stability. However, it was quantitatively observed that it failed to accurately reach almost all target points (Figure 13(a)). The PID control method exhibited a higher success rate in reaching the target than the default control method; however, many unnecessary movements occurred, such as passing the waypoints of P_1 and P_3 to converge below the threshold (Figure 13(b)). In contrast, the DT-DRL control method consistently reached all target points within the set success threshold. It was observed and quantified that the method minimized unnecessary movements, efficiently directing the TCP only to points closest to the threshold (Figure 13(c)).

The results were evaluated based on quantitative indicators (Table 4). The success rates of the control methods were quantified as 6.67% for the default method, 75.0% for the PID method, and 100% for the DT-DRL method. It was observed that the DT-DRL control method consistently achieved target point approach rates within the defined threshold for all instances. TCP position error for the default control method was measured at an average of 22.0 mm with a standard deviation of $\pm$14.6 mm. In contrast, the PID control method demonstrated an improved TCP position error, with a calculated average of 13.4 mm. In contrast, the PID control method demonstrated an improvement in TCP position error, recording a calculated average of 13.4 mm with a standard deviation of $\pm$12.2 mm. This higher standard deviation indicates a greater variability in performance, likely due to the complex and nonlinear dynamic characteristics of the DRM, resulting in less predictable control outcomes. The DT-DRL control method exhibited superior performance, achieving an average TCP position error of 8.21 mm, well below the defined threshold, and a notably low standard deviation of $\pm$1.11 mm, illustrating a more consistent and reliable performance across different trials compared to the other methods.

Finally, the cycle times required to reach the ten target points were compared. During the tests, a time interval of 20.0 s was allotted per target point for each of the default, PID, and DT-DRL control methods. If the TCP position error did not converge to below the threshold within this period, it proceeded to the next target point. The default, PID, and DT-DRL control methods required 197, 96.3, and 14.2s, respectively. This indicates that the DT-DRL method was approximately 13.9 times faster than the default control method and approximately 6.70 times faster than the PID control method.

There are several reasons for these differences in results. First, in this experiment, the target point value was converted into the target joint value using *DLS-IK*. The default control commanded the converted target joint value for the DRM controller.

Aged actuators in the DRM showed oscillations and static errors when the target joint value was changed. The lack of a supplementary control algorithm for error correction resulted in the system halting at the initial commanded joint value for 20s. Occasionally, success was achieved only when d_t was less than or equal to d_{thresh} during the oscillation. These results showed a correlation with the inherent characteristics of the DRM, notably reduced control reliability due to factors like actuator wear, aging, and backlash. In the case of PID control, the performance was clearly improved compared to the previous default control. Unlike the default control, the current joint value was monitored using an encoder built into the actuator, and this value was compared with the target joint value. Subsequently, the feedback control that calculates the next target joint value by substituting the difference into the PID control equation was repeated. Through this process, TCP was able to approach the target point. However, the observed instability and inaccuracy in results primarily stemmed from the inconsistent performance of the actuator, characterized by specific deviations in response times and precision, and the variable control frequency of the controller. First, the joint values were commanded to the controller, but the actuators did not respond within a certain range owing to the insufficient torque. The PID controller continued to accumulate error values owing to its integral component, which gradually increased the joint command value. When the command value finally exceeded the dead zone threshold, the actuators responded suddenly, demonstrating the dead zone phenomenon, in which the actuator only operates beyond a certain input level. This phenomenon occurred nonlinearly in the six actuators, causing the TCP of the DRM to become unstable and inaccurate. In addition, irregular control cycles contribute to this abnormal behavior. Although the command value transmission and encoder value reception frequencies were fixed through synchronization, these frequencies could not be accurately followed owing to the performance limitations of the controller. This irregular control frequency deteriorates the control performance of the PID control, which calculates the target joint value based on the current joint value and transmits it to the controller.

By contrast, the DRL-driven control policy responded robustly and flexibly to the problems mentioned above. In the training process, the noise was mixed with all joint values in each episode. Even in the presence of noise within the joint values, the DRL control policy was trained to return the action values that anticipated the subsequent state, thereby enabling the TCP to accurately reach the designated target point. In particular, because it was trained within the DT that replicated the dynamics of the actual DRM, the control policy was capable of generating action values that minimized potential oscillations and static errors. This solves the dead zone phenomenon that occurs in other control methods and enabled TCP to stably reach the target point. Furthermore, the control policy demonstrated flexibility in the face of control jitter within the actual system by adapting its action values to accommodate irregular control frequencies.

Based on these results, we confirmed that the proposed DT-DRL control method can improve the control accuracy and stability of the DRMs. In addition, the DT-DRL control policy was trained for less than 300s, confirming its advantages in terms of control system modeling.

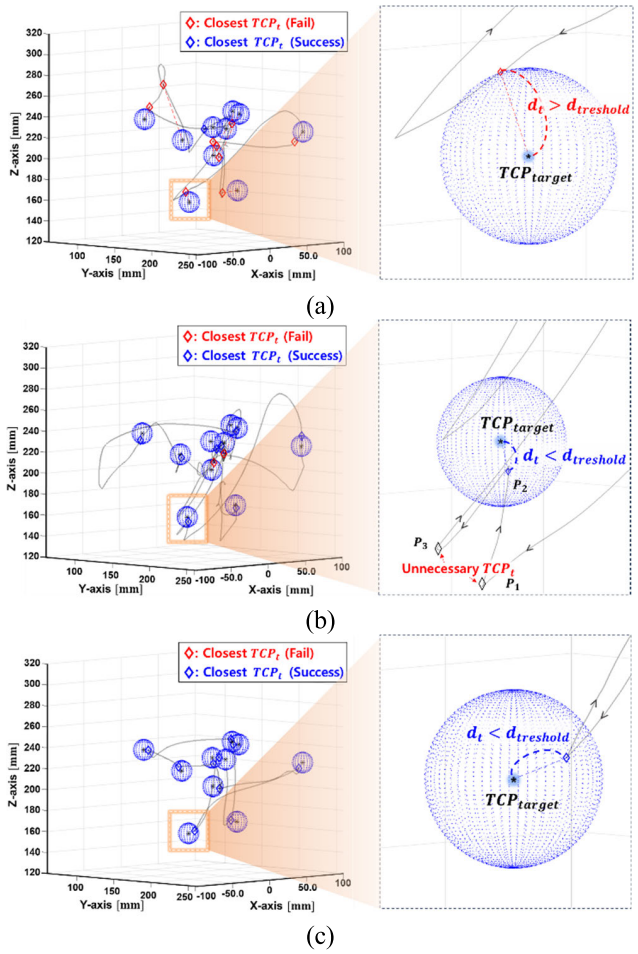

FIGURE 13. TCP trajectories in the Cartesian coordinates of the real robot manipulator based on control methods. (a) Default, (b) PID, and (c) DT-DRL (this study) control results.

TABLE 4. Comparison of task performance of the three control methods.

Metrics ($n = 6$)	Control methods		
	Default	PID	**DT-DRL (Ours)**
Success rate (%)	6.67	75.0	**100**
TCP position error (mm, mean ± std)	22.0 (±14.6)	13.4 (±12.2)	**8.21 (±1.11)**
Cycle time (s)	197	96.3	**14.2**
Training time (s)	N/A	N/A	**< 300**

V. CONCLUSION AND FUTURE WORKS

In this study, we developed a DT-DRL pipeline to enhance the accuracy and stability of the DRM control policy. The parameters of the physics engine in the DT were tuned to synchronize the dynamic characteristics of the DRM in both virtual and real settings. Our method, incorporated into the Real2Sim reflection process, yielded enhanced control accuracy and stability in the training outcomes, surpassing the performance typically achieved through general DRL training processes. The Sim2Real gap was bridged by randomizing and training hardware and control parameters in the DT environment, eliminating the need for intricate dynamic modeling of the DRM. By deploying the trained control policy in an actual DRM, we proved the effectiveness of the DT-DRL pipeline.

This study highlighted that the algorithmic approach could improve the performance of aging robots without hardware replacement, thereby providing long-term sustainability and reliable operation. Our method is also versatile for use on all articulated robots, promising reduced maintenance costs and sustained performance.

Future studies should broaden the applicability of the DT-DRL pipeline to a wider range of robotic types and tasks. Additionally, we aim to employ engineering strategies such as optimization theory to further bridge the Sim2Real gap. We also plan performance comparisons with optimization and other DRL algorithms for more difficult tasks, including object handling tasks.

ACKNOWLEDGMENT

(Su-Young Park and Cheonghwa Lee contributed equally to this work.)

REFERENCES

[1] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, pp. 841–844, Jun. 2019.
[2] C.-C. Wong, M.-Y. Chien, R.-J. Chen, H. Aoyama, and K.-Y. Wong, "Moving object prediction and grasping system of robot manipulator," *IEEE Access*, vol. 10, pp. 20159–20172, 2022.
[3] Z. Liu, K. Wang, D. Liu, Q. Wang, and J. Tan, "A motion planning method for visual servoing using deep reinforcement learning in autonomous robotic assembly," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 6, pp. 3513–3524, Dec. 2023.
[4] H. Su, C. Yang, H. Mdeihly, A. Rizzo, G. Ferrigno, and E. D. Momi, "Neural network enhanced robot tool identification and calibration for bilateral teleoperation," *IEEE Access*, vol. 7, pp. 122041–122051, 2019.
[5] A. Rassõlkin, V. Rjabtšikov, V. Kuts, T. Vaimann, A. Kallaste, B. Asad, and A. Partyshev, "Interface development for digital twin of an electric motor based on empirical performance model," *IEEE Access*, vol. 10, pp. 15635–15643, 2022.
[6] F. Sherwani, M. M. Asad, and B. S. K. K. Ibrahim, "Collaborative robots and industrial revolution 4.0 (IR 4.0)," in *Proc. Int. Conf. Emerg. Trends Smart Technol. (ICETST)*, Mar. 2020, pp. 1–5.
[7] Y. Song, M. Liu, B. Lian, Y. Qi, Y. Wang, J. Wu, and Q. Li, "Industrial serial robot calibration considering geometric and deformation errors," *Robot. Comput.-Integr. Manuf.*, vol. 76, Aug. 2022, Art. no. 102328.
[8] J. G. Enríquez, A. Jiménez-Ramírez, F. J. Domínguez-Mayo, and J. A. García-García, "Robotic process automation: A scientific and industrial systematic mapping study," *IEEE Access*, vol. 8, pp. 39113–39129, 2020.
[9] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153171–153187, 2021.
[10] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51416–51431, 2021.
[11] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. K. Liu, J. Peters, S. Song, P. Welinder, and M. White, "Sim2Real in robotics and automation: Applications and challenges," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 398–400, Apr. 2021.
[12] G. Qiao and B. A. Weiss, "Accuracy degradation analysis for industrial robot systems," in *Proc. Manuf. Equip. Syst.*, Jun. 2017.
[13] M. Aliff, N. Samsiah, M. Yusof, and A. Zainal, "Development of fire fighting robot (QRob)," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, 2019.

[14] B. Xiao, L. Cao, S. Xu, and L. Liu, "Robust tracking control of robot manipulators with actuator faults and joint velocity measurement uncertainty," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 3, pp. 1354–1365, Jun. 2020.
[15] P. Aivaliotis, Z. Arkouli, K. Georgoulias, and S. Makris, "Degradation curves integration in physics-based models: Towards the predictive maintenance of industrial robots," *Robot. Comput.-Integr. Manuf.*, vol. 71, Oct. 2021, Art. no. 102177.
[16] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.
[17] Y. H. Son, G.-Y. Kim, H. C. Kim, C. Jun, and S. D. Noh, "Past, present, and future research of digital twin for smart manufacturing," *J. Comput. Design Eng.*, vol. 9, no. 1, pp. 1–23, Dec. 2021.
[18] I. Raouf, H. Lee, Y. R. Noh, B. D. Youn, and H. S. Kim, "Prognostic health management of the robotic strain wave gear reducer based on variable speed of operation: A data-driven via deep learning approach," *J. Comput. Design Eng.*, vol. 9, no. 5, pp. 1775–1788, Oct. 2022.
[19] D. Kibira, G. Shao, and B. A. Weiss, "Buiding a digital twin for robot workcell prognostics and health management," in *Proc. Winter Simul. Conf. (WSC)*, Dec. 2021, pp. 1–12.
[20] P. Aivaliotis, K. Georgoulias, and G. Chryssolouris, "The use of digital twin for predictive maintenance in manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 32, no. 11, pp. 1067–1080, Nov. 2019.
[21] Z. Yang, S. Yang, S. Song, W. Zhang, R. Song, J. Cheng, and Y. Li, "PackerBot: Variable-sized product packing with heuristic deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 5002–5008.
[22] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Appl. Sci.*, vol. 10, no. 19, p. 6923, Oct. 2020.
[23] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, May 2017, pp. 23–30.
[24] S. Cheng, L. Li, H.-Q. Guo, Z.-G. Chen, and P. Song, "Longitudinal collision avoidance and lateral stability adaptive control system based on MPC of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2376–2385, Jun. 2020.
[25] M. A. Johnson and M. H. Moradi, *PID Control*. London, U.K.: Springer-Verlag, 2005, pp. 47–107.
[26] S. ISAAC. *Extensions API*. Nov. 2, 2023. [Online]. Available: https://docs.omniverse.nvidia.com/py/isaacsim/index.html
[27] A. Serrano-Muñoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, "Skrl: Modular and flexible library for reinforcement learning," 2022, *arXiv:2202.03825*.
[28] P. I. Corke, "A simple and systematic approach to assigning Denavit–Hartenberg parameters," *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 590–594, Jun. 2007.
[29] D. Chwa and H. Kwon, "Nonlinear robust control of unknown robot manipulator systems with actuators and disturbances using system identification and integral sliding mode disturbance observer," *IEEE Access*, vol. 10, pp. 35410–35421, 2022.
[30] S. Kucuk and Z. Bingul, *Robot Kinematics: Forward and Inverse Kinematics*. London, U.K.: InTech, 2006.
[31] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *J. Intell. Robotic Syst.*, vol. 14, no. 1, pp. 43–68, Sep. 1995.
[32] J. Tan, K. Liu, and G. Turk, "Stable proportional-derivative controllers," *IEEE Comput. Graph. Appl.*, vol. 31, no. 4, pp. 34–44, Jul. 2011.
[33] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends, Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.
[34] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[36] J. W. Mock and S. S. Muknahallipatna, "A comparison of PPO, TD3 and SAC reinforcement algorithms for quadruped walking gait generation," *J. Intell. Learn. Syst. Appl.*, vol. 15, no. 1, pp. 36–56, 2023.

SU-YOUNG PARK received the M.S. degree in mechanical engineering from the Pohang University of Science and Technology, in 2019. He is currently pursuing the Ph.D. degree in mechanical engineering with Seoul National University. His current research interests include robotic automation, smart manufacturing systems, and digital twin and optimization.

CHEONGHWA LEE received the M.S. degree in mechanical engineering from the Kumoh National Institute of Technology, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University. His current research interest includes AI-based robotic automation control and applications.

HYUNGJUNG KIM received the Ph.D. degree in mechanical engineering from Seoul National University, in 2011. He is currently an Assistant Professor with the Department of Industrial Engineering, Konkuk University. Before joining Konkuk University, he was with Doosan Infracore, Doosan Robotics, and Samsung Electronics. His research interests include digital twin/smart factory, industrial AI, smart sensor/vision, collaborative robots, and appropriate manufacturing technologies for small- and medium-sized enterprises (SMEs).

SUNG-HOON AHN received the Ph.D. degree from Stanford University, in 1997. He was a Postdoctoral Researcher and a Lecturer with the University of California in Berkeley and an Assistant Professor with Gyeongsang National University. He is currently a full-time Professor with the Department of Mechanical Engineering, Seoul National University. His research interests include smart factories, sensors/actuators, appropriate technology, robotics, smart grids, 3-D/4-D printing, composite materials, and micro/nanofabrication. He is a fellow of CIRP (the International Academy for Production Engineers) and the Korean Academy for Science and Technology and an Associate Member of the National Academy of Engineers of Korea.