This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

Reinforcement Learning-Based Optimization of Robotic Motion for High-Frequency Brazing Task of Copper Tube Joining*

Eugene Kim[1], Hyunrok Cha[1], Meyonghwan Hwang[1], and Younggon Kim[1,*]

***Abstract*— Achieving consistent performance, and high-quality brazed joints for copper tubes poses significant challenges due to the intricate interplay of thermal dynamics, material properties, and robot motion control. This paper presents a reinforcement learning (RL)–based framework for optimizing robotic trajectories and parameters in manufacturing brazing applications. The learned policy successfully guided the end-effector near the target region, as verified by the final pose.**

I. INTRODUCTION

Copper tube joint bonding technology is pivotal within the heat exchanger manufacturing industry, addressing the escalating demand for high-quality heat exchange applications. Specifically, high-frequency brazing for copper tube joints is considered as a core process in producing home appliances, Heating, Ventilation, and Air Conditioning (HVAC) systems, and precision-engineered products. Traditionally, brazing involves skilled operators manually positioning the heating coil, applying filler metal, and regulating contact angles and movements. However, manual joint brazing task faces inherent limitations in repeatability, and quality consistency. On the other hand, recent advances in industrial robotics and machine learning—particularly reinforcement learning (RL)—have significantly enhanced the feasibility of automating and optimizing tasks for industrial robots. By using RL, robotic systems can learn optimal motion trajectories to achieve precise heat transfer, high joining strength, and reduced processing time [1].

An automated copper tube brazing robot must perform two essential tasks: precise positioning and brazing. For positioning, a commonly employed strategy is used such as classical inverse dynamics to execute point-to-point (P2P) movements. However, even after an RGB camera detects the copper tube joint and the robot navigates to the optimal brazing location, it is challenging to continually refine the brazing posture based on real-time operational feedback. Moreover, in premium home appliances where a single heat exchanger may integrate multiple functionalities, the complexity of brazing points often renders purely rule-based approaches inadequate for handling edge cases. For upper reasons, the application of reinforcement learning not only can support autonomous positioning but also can be used as a real-time evaluation of brazing quality.

As illustrated in Fig. 1, the challenges of declining birth rates and an aging population are particularly significant in countries like South Korea, where the working-age population is projected to shrink by 30% by 2040. To address this issue, research on AI-driven autonomous manufacturing has been actively expanding, focusing on both task-specific AI models and more generalized models capable of handling diverse manufacturing processes. This study explores the development of AI models aimed at automating high-frequency brazing processes in industrial robotics. Specifically, we investigate reinforcement learning techniques for optimizing robot positioning using vision data, leveraging camera sensors to improve the efficiency and consistency of brazing operations.

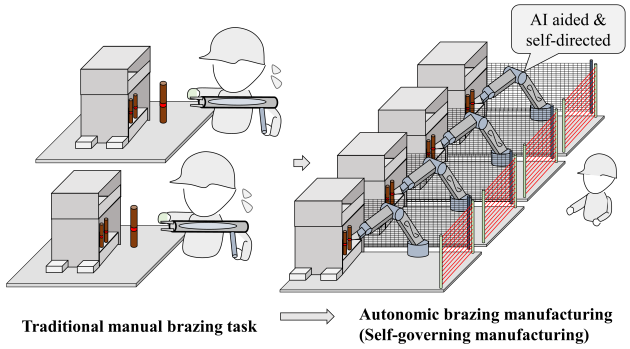

Fig. 1. This study discusses the autonomic high-frequency brazing welding using reinforcement learning to transfer heat exchanger welding tasks from humans to robots successfully

II. RELATED RESEARCH

A. Reinforcement Learning Algorithms

Reinforcement learning (RL) focuses on training neural network agents to optimize actions by interacting with a potentially complex environment. Early RL algorithms, such as Q-learning and SARSA, were tabular methods, effective only in relatively small or discrete state spaces. After emergence of deep neural networks (DNNs) led to Deep Q-Networks (DQN), which applied the Q-learning method with function approximation and demonstrated human-level control in Atari games. Moreover, a number of extensions improved stability and performance, including Double DQN (mitigating overestimation bias), Dueling DQN (separating state-value and advantage functions), and Prioritized Experience Replay (improving sample efficiency) [2].

While upper mentioned value-based algorithms improved, policy gradient methods also improved in parallel. Firstly, approaches such as REINFORCE, Advantage Actor-Critic

*Corresponding Author
[1]Authors are with the Purpose-Built Mobility group, Seonam Division, Korea Institute of Industrial Technology, Gwangju 61012, South Korea egkim@kitech.re.kr

This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

(A2C/A3C), and Trust Region Policy Optimization (TRPO) suggested a path to parameterize policies. However, computational complexity and hyperparameter sensitivity still remained issues. However, a Proximal Policy Optimization (PPO) emerged as a simpler yet effective variant of TRPO, constraining policy updates to avoid large performance degradation while using first-order optimization [3]. As a result, PPO quickly became a stable baseline in many RL tasks, especially in robotics and continuous control domains.

After PPO was reported, researchers explored further ideas and new directions. For instance, Soft Actor-Critic (SAC) introduced an entropy term into the objective function for maximum entropy RL, promoting exploration and improving sample efficiency in continuous action spaces [4]. Also, Twin Delayed Deep Deterministic Policy Gradient (TD3) introduced function approximation errors common in actor-critic methods by using two Q-networks to reduce overestimation biases [5]. More recent methods, such as Dreamer and DreamerV2, investigate model-based RL by learning a latent-space model of the environment, enabling long-horizon planning with reduced sample complexity [6]. Similarly, MuZero combined tree search with learned models of dynamics and rewards, achieving state-of-the-art performance in board games and classic Atari tasks. These developments highlight the ongoing trend of balancing exploration, sample efficiency, and robustness to expand RL's applicability in a variety of real-world scenarios.

B. AI-Based Brazing Research

Brazing is a process of joining different metal components using filler metal at high temperatures, requiring both high bond strength and efficient heat transfer. Traditionally, it has relied on experienced and skilled operators. However, it was difficult to ensure quality and high throughput. Therefore, recent research tried to use computer vision, sensor fusion, and robotics to improve and automate the brazing process. For example, infrared thermography can monitor the temperature distribution of the brazing joint in real time, or machine learning models can detect defects and automatically assess the quality of the joint. More recently, reinforcement learning was adopted to handle uncertainties such as minor deviations in component placement and changes in geometry, allowing robotic systems to adjust core parameters. By incorporating RL-based approaches, brazing systems are expected to achieve greater consistency, efficiency, and cost-effectiveness for fully automated and intelligent brazing operations.

III. Methodology

As mentioned earlier, Proximal Policy Optimization (PPO) has emerged as a robust baseline in the field of reinforcement learning (RL) owing to its balance between stable policy updates and practical implementation complexity. First, unlike vanilla policy gradient methods that are prone to large policy shifts, PPO constrains the update step, therefore decreasing the likelihood of catastrophic performance drops. Second, while more sophisticated algorithms such as Trust Region Policy Optimization (TRPO) provide theoretical guarantees on policy improvement, their computational overhead often limits real-time or large-scale applications. PPO captures many of TRPO's benefits—such as preventing excessively large policy updates—yet employs first-order optimization, making it more computationally efficient and straightforward to implement. These strengths position PPO as a compelling choice for robotic tasks, particularly where sample efficiency, training stability, and runtime performance are critical. Hence, we adopt PPO to enable stable policy optimization in our brazing robot control framework.

A. Proximal Policy Optimization Algorithm

We state our problem is within the standard Markov Decision Process (MDP) framework. Let

$$(\mathcal{S}, \mathcal{A}, P, r, \gamma), \tag{1}$$

denote the state space, action space, transition dynamics, reward function, and discount factor, respectively. Next, a policy $\pi_\theta(a_t \mid s_t)$ parameterized by θ specifies the probability of taking action a_t in state s_t. It should be noted that our goal is to find the policy π_θ that maximizes the expected sum of discounted rewards:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t\, r(s_t, a_t)\right], \tag{2}$$

where τ represents a trajectory $(s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1})$ generated by π_θ.

a) Clipped Objective: PPO limits how much a policy can change in one update by cutting off extreme changes in probability:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \tag{3}$$

where θ_{old} are the parameters of the policy on behalf of the current update. Then, the estimator of the advantage function at time step t can be written as $\hat{A}_t$. Therefore, the clip component of PPO can be written as:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\big(r_t(\theta)\hat{A}_t,\, \text{clip}\big(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\hat{A}_t\big)\right], \tag{4}$$

where ϵ (e.g., 0.1 or 0.2) controls the allowable deviation from the old policy. This design updates the penalty that move $r_t(\theta)$ outside of the interval $[\,1-\epsilon,\ 1+\epsilon\,]$, which reduces the risk of excessively large policy shifts and promotes more stable learning.

b) Critic and Overall Objective: It should be noted that the PPO typically employs an actor-critic architecture, where a value function $V_\phi(s)$ is used as a critic to estimate critical state values for advantage calculation. Next, the overall loss function combines the clip component of L^{CLIP} with a value loss term $\mathcal{H}$:

$$\begin{aligned} L(\theta, \phi) = \hat{\mathbb{E}}_t\Big[& L^{\text{CLIP}}(\theta) - c_1\big(V_\phi(s_t) - V_t^{\text{target}}\big)^2 \\ & + c_2\, \mathcal{H}\big(\pi_\theta(s_t)\big)\Big], \end{aligned} \tag{5}$$

where V_t^{target} is the target value, c_1 and c_2 are weighting coefficients, and $\mathcal{H}\big(\pi_\theta(s_t)\big)$ represents the entropy of the policy at state s_t.

This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

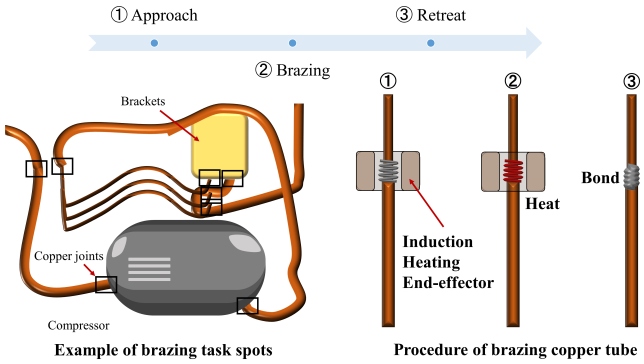

Fig. 2. Illustration of the copper brazing process for tube joints in a typical refrigeration or heat exchanger assembly. The red arrow highlights key components i.e., the compressor, a filter/drier (yellow), and the induction heating coil (brown blocks and coil) transitioning from an inactive (gray) to an active (red) state. The blue timeline at the top indicates the sequential process of the brazing task steps from left to right. By applying high-frequency current through the induction coil, precise heating of the joint is achieved, ensuring strong and consistent brazed connections

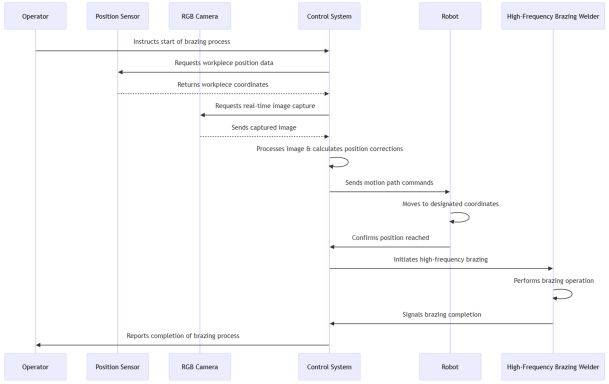

Fig. 3. Sequence diagram illustrating the autonomic brazing workflow. The control system coordinates data acquisition from the position sensor and RGB camera, then computes motion paths to guide the robot to the precise brazing position. Upon receiving a "position reached" confirmation, the control system sends signal to the high-frequency brazing controller to perform the operation. When the whole process is complete, a completion signal is returned, and the operator is notified

c) Implementation Details: PPO algorithm collects sampling data through interaction with the environment (e.g., collecting trajectories) and conduct optimization through the clipped objective using mini-batch stochastic gradient descent for a fixed number of epochs. This learning process continues until it reaches stopping criteria—such as reaching a reward threshold or reaching a termination number of training timesteps—are satisfied. It is well knwon that the PPO has demonstrated strong performance in numerous continuous control tasks, including robotic manipulation and locomotion, making it a feasible candidate for real-world applications such as brazing robot control which is the target manufacturing process of this research.

IV. Experiment

A. Target task and manufacturing process

Figure 2 provides an overview of the induction brazing steps for copper tube assemblies in a typical refrigeration or heat exchanger system. In an autonomic manufacturing sites, precisely controlling coil placement, heat input, and process timing is critical for maintaining consistent product quality of the target manufacturing products.

Figure 3 illustrates interaction flow among the key components in target automnomic brazing system. First, the operator initiates the brazing task by issuing a start command to the main control system. Next, the control system requests the workpiece position data from the position sensor, which then returns the coordinates for precise alignment. Simultaneously, the control system requests a real-time image from the RGB camera module to validate and refine the positional data. After processing these inputs, the control system generates motion path commands and sends them to the robot. The robot moves to the received coordinates and confirms successful positioning. Subsequently, the control system sends signal the high-frequency brazing welder to begin the brazing operation, during which induction heating fuses the joint. Once brazing task is completed, the brazing controller sends a completion signal back to the control system, which, notifies the operator that the process has finished.

TABLE I
DH Parameter of Rainbow Robotics RB5-850e (unit: mm, rad)

[mm, rad]	α_i	a_i	θ_i	d_i
Joint 1	$\frac{\pi}{2}$	0	θ_1	169.2
Joint 2	0	425	$\theta_2 + \frac{\pi}{2}$	0
Joint 3	0	392	θ_3	0
Joint 4	$-\frac{\pi}{2}$	0	$\theta_4 - \frac{\pi}{2}$	110.7
Joint 5	$\frac{\pi}{2}$	0	θ_5	110.7
Joint 6	0	0	θ_6	94.7
End effector	0	0	0	0

B. General setup of digital-twin simulation

The simulationss were conducted using Python version 3.12.7, with TensorFlow 2.18.0 employed for deep learning applications. For physics-based simulations, we utilized PyBullet (version 3.2.6) alongside PyBullet-Industrial (version 1.0.3), ensuring a robust and integrated computational environment.

C. Robot description

Table I summarizes the Denavit–Hartenberg (DH) parameters for the RB5-850e manipulator. For each of the six joints, the link length (a_i), link offset (d_i), twist angle (α_i), and joint angle offset are provided. These parameters characterize the robot's kinematic configuration and serve as the foundation for both forward and inverse kinematics analyses.

D. Model description

Table II presents a detailed summary of the actor network architecture. The actor network processes an input RGB image of size $(240 \times 480 \times 3)$ through a series of convolutional layers followed by a flattening layer and a dense layer with

This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

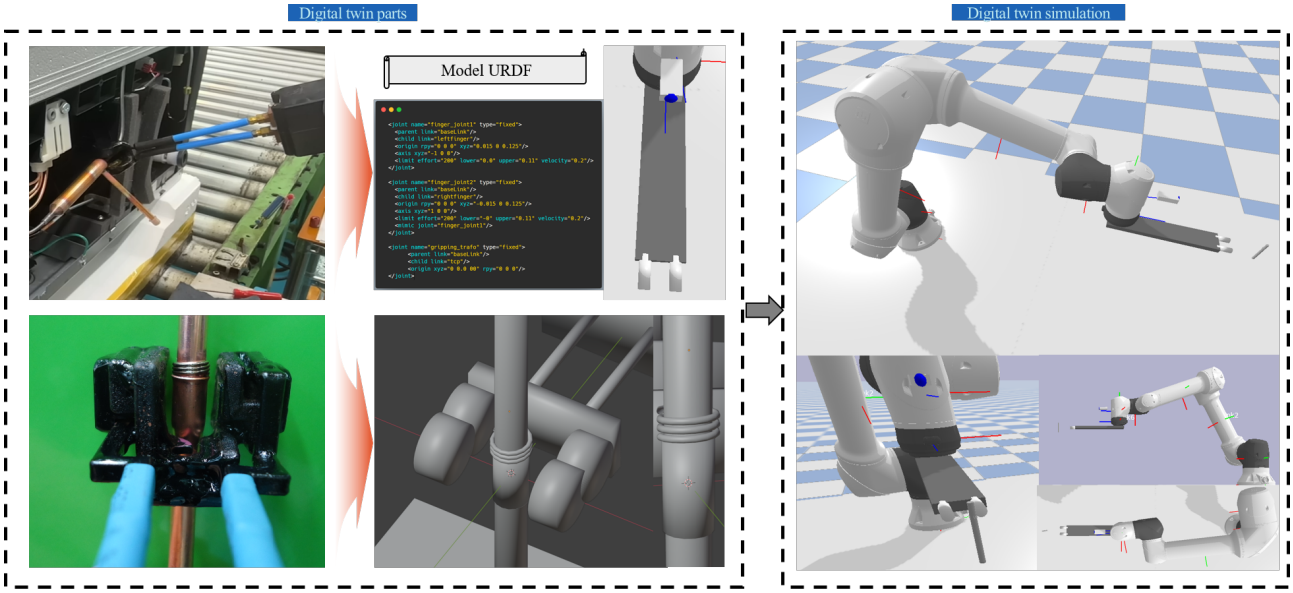

Fig. 4. Illustration of a digital twin workflow for a brazing application. The left column shows the physical setup, including the brazing end-effector approaching a copper tube, while the center highlights the URDF model representation of the end-effector and the robot configuration. On the right, the fully modeled system is simulated in a virtual environment (pybullet engine, and pybullet industrial add-on), enabling offline testing and validation of brazing operations before deployment

TABLE II

ARCHITECTURE SUMMARY OF THE ACTOR NETWORK

Layer	Output Shape	Parameter Count	Comments
`Input (input_states)`	$(240, 480, 3)$	0	RGB image input
`Input (input_action_matrixs)`	$(1, 6)$	0	Action mask input
`Input (input_advantages)`	$(1, 1)$	0	Advantage input
`Conv2D (32 filters, 8 × 8, stride 4)`	$(59, 119, 32)$	6,176	ReLU activation
`Conv2D (64 filters, 4 × 4, stride 2)`	$(28, 58, 64)$	32,832	ReLU activation
`Conv2D (64 filters, 3 × 3, stride 1)`	$(26, 56, 64)$	36,928	ReLU activation
`Flatten`	$(93, 184)$	0	—
`Dense (256 units)`	(256)	23,852,160	ReLU activation
`Dense (6 units)`	(6)	1,542	Softmax activation
Total	–	**23,929,638**	

TABLE III

ARCHITECTURE SUMMARY OF THE CRITIC NETWORK

Layer	Output Shape	Parameter Count	Comments
`Input (input_states)`	$(240, 480, 3)$	0	RGB image input
`Conv2D (32 filters, 8 × 8, stride 4)`	$(59, 119, 32)$	6,176	ReLU activation
`Conv2D (64 filters, 4 × 4, stride 2)`	$(28, 58, 64)$	32,832	ReLU activation
`Conv2D (64 filters, 3 × 3, stride 1)`	$(26, 56, 64)$	36,928	ReLU activation
`Flatten`	$(93, 184)$	0	—
`Dense (256 units)`	(256)	23,852,160	ReLU activation
`Dense (1 unit)`	(1)	257	Linear activation
Total	–	**23,928,353**	

This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

256 units. Finally, a fully connected layer with 6 units and a softmax activation outputs a probability distribution over the six available actions. The total number of parameters in the actor network was set as approximately 23.93 million.

Table III summarizes the critic network architecture. Similar to the actor model, the critic model uses a series of convolutional layers to extract features from the same input image, followed by a flattening layer and a dense layer with 256 units. However, the final dense layer has only 1 unit with a linear activation function because the critic is designed to estimate a single scalar value representing the state value. This scalar output is essential for computing value losses and guiding the policy updates in the actor network. The critic network containd approximately 23.93 million trainable parameters.

E. Learning parameters

Table IV outlines the key model configurations and training hyperparameters used in the proposed reinforcement learning framework. This table details the dimensions of the input state (comprising position, velocity, angle, and angular velocity), as well as the image input dimensions (height, width, and channels). It also specifies the size of the action space and the scalar value output for the critic network. In addition, the table lists essential hyperparameters such as the learning rates for the actor and critic networks, the number of epochs per mini-batch update, the discount factor, the smoothing rate used in Generalized Advantage Estimation (GAE), the penalty applied under certain conditions (e.g., collisions), the total number of training episodes, the mini-batch update frequency, and the number of units in the dense layer following the convolutional layers. These parameters collectively define the operational settings and learning dynamics of the proposed model.

Table V provides an overview of the reward formulation and the termination criterion used during training. The positional reward is defined as the inverse of the distance between the end-effector and the target (with a small constant added for numerical stability), and it is clipped within the range of $[-10, 10]$. Yet, no reward is directly attributed to angular differences. A fixed penalty of -400 is applied when a collision is detected, and an episode terminates when the distance falls below 0.02, indicating that the target has been effectively reached.

V. Result and Discussion

A. Reward tracking data

The reward tracking results in Fig. 5 demonstrate that the PPO agent gradually improved its policy over the course of 500 training episodes. Early in training, the agent exhibited considerable variability in episodic rewards due to exploration and the stochastic nature of the environment. As learning progressed, however, the overall reward trend (as highlighted by the moving average) increased, indicating that the agent was successfully refining its actions to achieve higher returns.

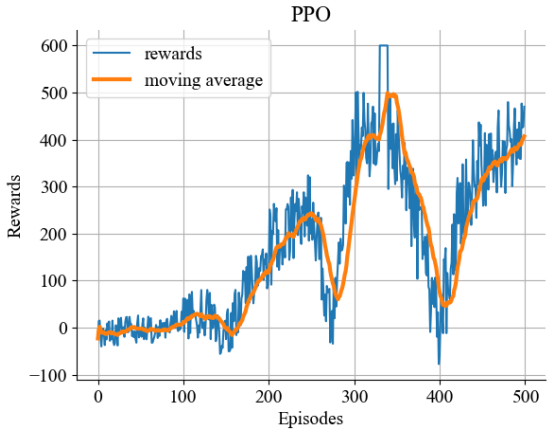

Fig. 5. Training performance of the PPO agent over multiple episodes. The blue curve represents the raw episodic reward, which exhibits fluctuations due to exploration and stochasticity in the environment. The orange curve is a moving average applied to the rewards, providing a smoother trend that highlights the overall improvement in the policy. As training progresses, the agent's actions become more optimal on average, reflected by the rising moving-average curve and generally increasing reward levels

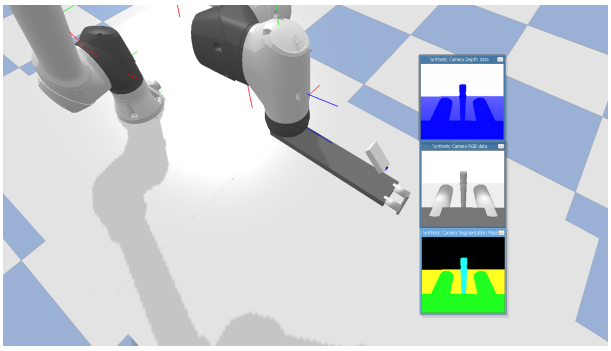

Fig. 6. Final position of end-effector after PPO agent training

Figure 6 shows the final end-effector position after the training process. By the end of learning, the agent could reliably guide the manipulator to a position near the target position without frequent collisions or misalignment.

Furthermore, during training, the agent had to overcome the penalty from collisions with specimen. The preliminary determined penalty value helped elaborating the policy away from undesirable actions, encouraging more precise positioning of the end-effector. However, the policy also introduced large reduction in rewards when collisions occurred, resulting in sharp downward spikes in the raw reward curve. As the learning proceeds, the change of reduction in rewards decreased, illustrating that the agent increasingly avoided hazardous actions. However, in practice, although it was not applied in the simulation, factors such as camera noise, partial observability, and the complexity of the robotic environment can influence final performance.

VI. Conclusion

In this proceeding study, we presented a reinforcement learning approach based on Proximal Policy Optimization

This is a preprint version.The final authenticated publication is available at https://doi.org/10.1109/ARSO64737.2025.11124970

TABLE IV
MODEL CONFIGURATIONS AND TRAINING HYPERPARAMETERS

Parameter	Value	Description
State Dimension	4	[position, velocity, angle, angular velocity]
Image Height	240	Height of input RGB image (pixels)
Image Width	480	Width of input RGB image (pixels)
Image Channels	3	Number of channels (RGB)
Action Size	6	Number of discrete actions
Value Size	1	Scalar value output (critic)
Learning Rate (Actor)	1×10^{-4}	Actor network learning rate
Learning Rate (Critic)	1×10^{-4}	Critic network learning rate
Epoch Count	10	Epochs per mini-batch training
Discount Rate	0.90	Future reward discount factor
Smooth Rate	0.95	GAE smoothing factor
Penalty	-40	Penalty applied for collisions and termination
Episode Count	500	Total number of training episodes
Mini-batch Step Size	10	Steps between mini-batch updates
Dense Layer Units	256	Units in the dense layer following the CNN
Moving Average Size	20	Window size for performance averaging

TABLE V
REWARD CONDITIONS AND TERMINATION CRITERIA

Component	Formula/Threshold	Description
Positional Reward	$r_{\text{pos}} = \frac{1}{d+\epsilon}$	d: distance between end-effector and target; $\epsilon = 1 \times 10^{-6}$. Clipped to [-10,10].
Angular Reward	0	No reward contribution from angular difference
Collision Penalty	-40	Applied when a collision is detected
Termination Criterion	$d < 0.02$	Episode terminates when the distance is below 0.02

(PPO) for controlling a robotic manipulator in a vision-based environment. The agent learned to navigate the end-effector toward a designated target position while minimizing collisions and misalignment. Empirical results showed a progressive increase in the average reward and a stable convergence trend over multiple training episodes.

Specifically, our experimental setup demonstrated that even with partial observability and noise in camera input, the PPO agent could discover effective action policies through iterative exploration and exploitation. The inclusion of a high collision penalty steered the agent away from risky maneuvers, though occasional reward drops were observed when collisions occurred. Over time, these drops became less frequent, illustrating the agent's ability to internalize collision avoidance.

These findings highlight the potential of PPO to handle high-dimensional image inputs and complex robotic tasks under uncertain conditions. Future extensions could focus on refining hyperparameters for more sample-efficient training, integrating additional sensor modalities for improved perception, or adopting curriculum learning strategies to further enhance the robustness and adaptability of the policy.

ACKNOWLEDGMENT

This work was supported by "Development of AI-based Equipment Control and Autonomous Manufacturing Operation Technology for High-quality Management of Non-standard Production Products in Home Appliance Factories" (KM-240409), and "Development of Core Technologies for a Working Partner Robot in the Manufacturing Field" (KITECH EO-250005). The authors express their gratitude to project leader Bonggu Kim, Changhoon Ryu of DH Global for his valuable support and contributions.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, Pmlr, 2018.

[5] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[6] M. Okada and T. Taniguchi, "Dreamingv2: Reinforcement learning with discrete world models without reconstruction," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 985–991, IEEE, 2022.