processes

MDPI

Article

Deep Reinforcement Learning-Based Autonomous Docking with Multi-Sensor Perception in Sim-to-Real Transfer

Yanyan Dai and Kidong Lee *

Robotics Department, Yeungnam University, Gyeongsan 38541, Republic of Korea; yanyan_dai@ynu.ac.kr
* Correspondence: kdrhee@yu.ac.kr

Abstract

Autonomous docking is a critical capability for enabling fully automated operations in industrial and logistics environments using Autonomous Mobile Robots (AMRs). Traditional rule-based docking approaches often struggle with generalization and robustness in complex, dynamic scenarios. This paper presents a deep reinforcement learning-based autonomous docking framework that integrates Proximal Policy Optimization (PPO) with multi-sensor fusion. It includes YOLO-based vision detection, depth estimation, and LiDAR-based orientation correction. A concise 4D state vector, comprising relative position and angle indicators, is used to guide a continuous control policy. The outputs are linear and angular velocity commands for smooth and accurate docking. The training is conducted in a Gym-compatible Gazebo simulation, acting as a digital twin of the real-world system, and incorporates realistic variations in lighting, obstacle placement, and marker visibility. A designed reward function encourages alignment accuracy, progress, and safety. The final policy is deployed on a real robot via a sim-to-real transfer pipeline, supported by a ROS-based transfer node. Experimental results demonstrate that the proposed method achieves robust and precise docking behavior under diverse real-world conditions, validating the effectiveness of PPO-based learning and sensor fusion for practical autonomous docking applications.

Keywords: autonomous docking; proximal policy optimization (PPO); reinforcement learning; sim-to-real transfer; multi-sensor fusion

check for updates

Academic Editors: Iqbal M. Mujtaba, Hua Chen, Gang Wang and Yunfeng Ji

Received: 1 July 2025
Revised: 12 August 2025
Accepted: 1 September 2025
Published: 5 September 2025

Citation: Dai, Y.; Lee, K. Deep Reinforcement Learning-Based Autonomous Docking with Multi-Sensor Perception in Sim-to-Real Transfer. *Processes* **2025**, *13*, 2842. https://doi.org/10.3390/pr13092842

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In Automated Guided Vehicle (AGV) and Autonomous Mobile Robot (AMR) systems, autonomous docking technology plays a pivotal role in enabling efficient and fully unmanned operations, offering significant practical value. On one hand, it allows robots to autonomously return to charging stations [1,2] and perform precise docking for energy replenishment after completing tasks, greatly enhancing system autonomy and operational continuity. On the other hand, in smart factory [3,4] and logistics scenarios, docking capabilities enable robots to interface accurately with workstations, conveyor systems, or other equipment for precise material handover, serving as a foundation for flexible manufacturing and multi-robot collaboration [5,6]. Furthermore, by integrating vision, LiDAR, and other sensor data, robots can perform docking tasks reliably even in complex or dynamic environments, improving robustness and intelligence [2,7]. Autonomous docking reduces the need for human intervention, lowers maintenance costs, and provides critical support for building intelligent, efficient, and unattended industrial and logistics systems [8]. It is thus an essential capability in Industry 4.0 and smart logistics frameworks.

Traditional rule-based docking strategies often suffer from poor adaptability in complex environments. These methods typically rely on hand-crafted thresholds or simple PID controllers [9,10], which are difficult to generalize across varying conditions. Vision-based localization methods, such as AprilTag or QR code detection, are prone to errors under lighting changes, partial occlusion, or reflective surfaces [11,12]. Reinforcement Learning enables agents to learn optimal behaviors through trial-and-error interactions with the environment, guided by a reward signal. For tasks such as docking, where the robot must align its pose with a docking target based on sensor input. RL offers a powerful framework for learning control policies that adapt to complex and dynamic conditions [13–15]. The paper [16] focuses on enhancing the autonomous capabilities of Autonomous Mobile Robots (AMRs) by integrating Deep Q-Network (DQN) reinforcement learning with AprilTag visual markers for docking and obstacle avoidance. However, DQN is effective for discrete action spaces but limited in handling the continuous control required for smooth docking. The paper [17] presents the application of reinforcement learning strategies of Deep Deterministic Policy Gradient (DDPG) for controlling the docking of an AUV onto a fixed platform in a simulation environment. DDPG is designed for continuous action spaces, it can learn efficient policies but may suffer from stability issues and sensitivity to hyperparameters. In contrast, Proximal Policy Optimization (PPO) is an on-policy algorithm that balances ease of implementation with robust performance, particularly in environments with high-dimensional observation spaces [18]. In addition, Ref. [19] evaluated PPO, TD3, and SAC for quadruped walking gait generation, finding that PPO achieved competitive performance with greater stability across various sensor configurations. To overcome these limitations, this paper proposes a deep reinforcement learning-based autonomous docking framework using Proximal Policy Optimization (PPO) combined with multi-sensor fusion. Our system leverages YOLO-based object detection for robust target identification, a depth camera for distance estimation, and LiDAR for angle correction. These sensor modalities are fused into a concise state representation that enables the robot to learn robust and generalizable docking behavior. Although the reward function in our PPO framework encodes docking-specific metrics such as distance, orientation, and alignment, the policy itself is not simply mimicking a hand-tuned controller. Unlike classical control schemes that rely on explicit modeling and fixed rules, our PPO-based policy learns a nonlinear strategy directly from high-dimensional multi-sensor input, including YOLO-based visual feedback, depth estimation, and LiDAR scans. This data-driven learning process allows the agent to adapt to sensor noise, occlusions, and environment variability without manual mode switching or controller redesign. Moreover, recent work [18,19] demonstrates that PPO performs more stably than traditional control pipelines in tasks requiring robust sensor fusion and generalization. Thus, our method offers a more flexible and scalable solution to real-world docking compared to rule-based controllers.

In the field of AMRs, digital twins are increasingly adopted to bridge the gap between simulation and real-world deployment [20–22]. A digital twin is a high-fidelity virtual replica of a physical system. It allows developers to simulate robot behavior, sensor feedback, and environmental interactions under controlled and repeatable conditions. This technology is particularly valuable in docking tasks, where precision, safety, and robustness are critical. By simulating the docking process in a digital twin environment, one can test various control policies, sensor fusion strategies, and environmental scenarios without risking hardware damage or incurring operational downtime.

In our work, we construct a Gym-compatible training environment in Gazebo, which acts as a digital twin of the real robot system. The digital twin enables us to train the PPO reinforcement learning policies in a safe, accelerated, and cost-effective manner. The trained model is then transferred to the real robot using a sim-to-real strategy, leveraging the

consistency between the simulated and physical domains. This approach not only reduces the need for extensive real-world trials but also improves generalization in complex docking scenarios. A carefully designed reward function guides the learning process, promoting smooth alignment, accurate approach, and avoidance of obstacles. Through continuous action control, the robot learns to fine-tune both linear and angular velocities for optimal docking performance.

The contribution of this paper is listed as: (1) A PPO-based reinforcement learning framework is proposed for autonomous docking, which learns smooth and precise docking behaviors through continuous control of linear and angular velocities. The reward function is carefully designed to penalize lateral offset, angular misalignment, and abrupt motions, enabling the policy to produce stable, safe, and goal-directed behaviors. (2) A multi-sensor fusion strategy is implemented, integrating YOLOv8-based AprilTag detection (robust even under partial occlusion), depth-based distance estimation, and LiDAR-based orientation correction. These sensor modalities are fused into a low-dimensional yet expressive state representation, enhancing perception robustness in challenging conditions such as variable lighting, tag occlusion, and asymmetric clutter. (3) A digital twin simulation environment in Gazebo is developed, compatible with OpenAI Gym interfaces. The training environment incorporates varying ambient lighting, diverse obstacle layouts, and randomized disturbances to improve policy generalization. This setup supports efficient training and robust sim-to-real transfer without the need for extensive real-world trials. (4) The trained PPO policy is deployed on a real AMR platform, where it achieves a 90% success rate over 50 physical docking trials. The robot is able to complete docking tasks even in the presence of partial tag occlusion and environmental obstacles. An emergency stop mechanism is introduced during deployment to enhance safety—automatically halting the robot if LiDAR detects obstacles within 0.1 m.

2. Methodology

This section presents the proposed framework for autonomous docking using deep reinforcement learning with Proximal Policy Optimization (PPO) and multi-sensor fusion. The system is trained in simulation using a digital twin built in Gazebo and deployed to a real mobile robot through sim-to-real transfer.

2.1. PPO-Based Reinforcement Learning

Figure 1 shows the PPO-based Reinforcement Learning (RL) framework. In the framework, the Proximal Policy Optimization (PPO) algorithm serves as the learning core of the intelligent agent, forming a closed-loop RL structure that enables the robot to learn optimal docking behavior through interaction with its environment.

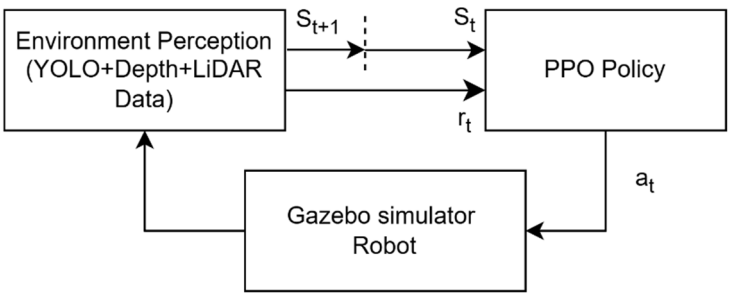


Figure 1. PPO-Based Reinforcement Learning Framework.

In the environment perception block of the framework, the state vector $s_t = [x_t, z_t, d_{t1}, d_{t2}]$ is generated from a fused multi-sensor pipeline to represent the spatial relation between the robot and the AprilTag docking target. Here, x_t denotes the horizontal

offset of the tag in the camera frame, and z_t represents the front distance from the camera to the tag, both derived from YOLO-based detection and depth image projection. The detection module identifies the center pixel coordinates (u_t, v_t) of the AprilTag using a YOLO object detector. Combining the corresponding depth value z_t from the depth camera and intrinsic camera parameters, the system computes the 3D position (x_t, y_t, z_t), where x_t and z_t are used in the final state vector. In parallel, the system processes LiDAR scan data by selecting symmetrical angular regions to the left and right of the robot's front. The average distances in these regions are recorded as d_{t1} and d_{t2}, which reflect orientation deviation and are used to assist angular correction. This 4D state vector is then fed into the PPO policy network, enabling the agent to learn continuous control actions for precise and robust docking.

In this framework, the PPO policy network takes the current state vector $s_t = [x_t, z_t, d_{t1}, d_{t2}]$ as input and outputs a continuous action vector $a_t = [v, \omega]$ where v represents the robot's linear velocity and ω represents its angular velocity. This action vector serves as the velocity command that directly controls the robot's movement in either the Gazebo simulation or the real-world environment. By continuously adjusting these control outputs, the robot learns to perform smooth and accurate docking maneuvers.

The training environment is built using the Gym interface combined with the Gazebo simulator, forming a high-fidelity digital twin system that closely mirrors the real robot setup. After each action a_t is executed, the environment returns the next observation s_{t+1} along with a reward r_t. The reward r_t is calculated as (1):

$$r_t = r_t^{base} + r_t^{bonus} + r_t^{success} + r_t^{penalty} \tag{1}$$

The reward r_t includes four terms. The base reward term r_t^{base} is designed to continuously guide the robot toward the ideal docking position at every step. It comprises three key components: the forward distance error $\mid z_t - z_{define} \mid$, which measures how close the robot is to the target along the longitudinal axis. The lateral offset is $\mid x_t \mid$, indicating how centered the robot is. To suppress lateral oscillations during docking, we replace the absolute value of the lateral offset $|x_t|$ with a quadratic term x_t^2. This modification penalizes larger deviations more strongly and encourages smoother convergence toward the docking centerline. The LiDAR-based angular deviation is $\mid d_{t1} - d_{t2} \mid$, reflecting the misalignment in orientation. The term r_t^{base} is described as:

$$r_t^{base} = - \mid z_t - z_{define} \mid -\lambda_1 x_t^2 - \lambda_2 \mid d_{t1} - d_{t2} \mid \tag{2}$$

The base reward function r_t^{base} uses weighting coefficients $\lambda_1 = 2.0$ and $\lambda_2 = 0.3$, which balance the influence of different error components. Specifically, λ_1 controls the penalty for lateral offset in the x-direction—that is, how far the robot is from being centered with respect to the docking target—while λ_2 adjusts the penalty for angular misalignment, as inferred from the difference in LiDAR distances on the left and right sides. A larger value of λ_1 strengthens the robot's tendency to stay aligned with the docking centerline, while the moderate value of λ_2 avoids overreacting to small orientation noise, thus promoting smoother trajectory convergence. This configuration has been shown to significantly reduce lateral oscillations and improve docking success rates in both simulation and real-world experiments. z_{define} is the defined distance between the robot and the docking AprilTag. In (2), three components are combined with appropriate weights into a negative reward signal, penalizing deviations from the desired state. This formulation ensures that the agent receives meaningful feedback even when docking is not yet completed, thereby accelerating policy learning, improving convergence, and enhancing the overall stability of the training process.

In (1), the reward term r_t^{bonus} represents positive incentives that encourage desirable behavior beyond simple error minimization. These bonus terms typically provide small rewards when the robot exhibits progress toward successful docking. r_t^{bonus} is designed as (3).

$$r_t^{bonus} = \begin{cases} 1.0, if\, x_t < 0.03 (\textit{robot is well centered to the AprilTag.}) \\ 0.5, if\, |z_t| < 1.5 (\textit{robot is getting close to the AprilTag.}) \\ 0, \qquad\qquad\qquad\qquad\qquad\qquad \textit{otherwise} \end{cases} \tag{3}$$

These bonuses do not penalize errors, but instead reward intermediate achievements to guide the PPO agent's learning, such as: moving forward toward the docking target and maintaining a well-centered alignment.

The reward term $r_t^{success}$ refers to the terminal reward granted when the robot successfully completes the docking task. It is designed to strongly reinforce the final goal of the agent and encourage it to learn policies that consistently lead to success. It is defined as (4).

$$r_t^{success} = \begin{cases} 30, & \textit{if docking is successful at time } t. \\ 0, & \textit{otherwise} \end{cases} \tag{4}$$

The success reward is only given when the forward distance $x_t < 1.0$, lateral offset in the x-direction $|z_t| < 0.05$, and the angle difference $\mid d_{t1} - d_{t2} \mid < 0.05$.

The reward term $r_t^{penalty}$ refers to the negative reward applied when the agent exhibits undesirable behaviors during docking. These penalties are crucial for discouraging unsafe or ineffective actions and guiding the robot toward robust and safe docking policies. It is defined as (5).

$$r_t^{penalty} = \begin{cases} -10, & \textit{if collision is detected based on LiDAR sensor data.} \\ -5, & \textit{if the April Tag marker is lost for defined time.} (\textit{timeout}) \\ -2, & \textit{if episode timeout is reached without docking.} \\ 0, & \textit{otherwise} \end{cases} \tag{5}$$

Collision penalty prevents risky behavior that might damage the robot or environment. AprilTag marker timeout penalty encourages the agent to stay visually aware and maintain detection of the docking target. Episode timeout penalty discourages inefficient wandering or overly long attempts.

Integrating the four reward terms (2)–(5), the reward r_t is calculated as (1). It balances progressive rewards with strict penalties, enabling the PPO agent to optimize its policy for both effectiveness and safety in the autonomous docking task.

In RL, an agent interacts with the environment to accumulate experience and learn how to take optimal actions that maximize long-term rewards. PPO is an efficient and stable policy optimization algorithm that aims to maximize the expected cumulative discounted return as (6):

$$\pi^* = \underset{\pi}{argmax} \mathbb{E}\left[\sum_t \gamma^t r_t\right] \tag{6}$$

where π denotes the policy function that determines the probability of taking an action given a state, r_t is the immediate reward at time step t, and $\gamma \in [0, 1]$ is the discount factor that balances short-term and long-term gains. During each training episode, defined by successful docking, collision, or timeout, the agent collects transition tuples (s_t, a_t, r_t, s_{t+1}), where s_t represents the current state, a_t is the action taken, r_t is the reward received, and s_{t+1} is the next state. PPO uses these experiences to compute advantage estimates and update the policy via a clipped surrogate objective function, which constrains policy updates to prevent instability. This process enables the agent to iteratively refine its policy through

gradient descent, gradually learning to produce smooth and reliable docking maneuvers. By optimizing the policy within a constrained update range and incorporating both positive reinforcement and penalties, PPO achieves a balance between exploration and exploitation, ultimately improving docking accuracy and robustness in complex environments.

2.2. Sim-to-Real Transfer System

To enable safe and effective policy training for autonomous docking, we design a Sim-to-Real transfer system that bridges virtual and real-world environments using a digital twin approach. As the system architecture shown in Figure 2, it is composed of three layers: Gazebo Simulator, ROS Middleware, and the Real-World Deployment.

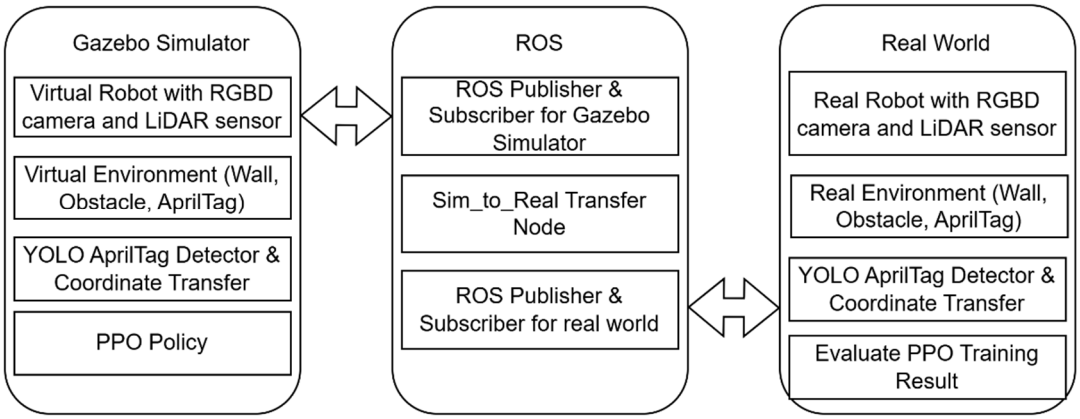


Figure 2. Sim-to-Real System Architecture.

To enable safe, repeatable, and accelerated training of the docking policy, we construct a high-fidelity simulation environment using Gazebo Simulator, which acts as the digital twin of the real robot system. The virtual environment is designed to closely match the physical world in both robot configuration and sensor placement. The robot model is defined using a URDF/Xacro description, equipped with a RGBD camera (Orbbec Dabai depth camera), 2D LiDAR sensor (EAI T-mini Pro LiDAR), and differential drive controller. The simulated environment includes various elements such as walls, obstacles, and AprilTag markers, which are commonly found in real-world warehouse or factory docking scenarios. These elements are dynamically configurable, allowing us to simulate a wide range of environmental conditions such as narrow corridors, lighting variations, and partial occlusions. For perception, we implement a virtual YOLO-based AprilTag detector that processes RGB images to estimate the center position of the docking marker in pixel coordinates. This information is combined with depth data from the camera to compute the relative 3D position of the marker in the camera frame. Additionally, LiDAR scan data is used to estimate angular misalignment by comparing distance readings on the left and right sides of the robot. These serve as state inputs to a PPO policy trained in simulation. This environment is wrapped using the Gym API, making it compatible with standard reinforcement learning frameworks. The robot's action space is defined as continuous velocity commands (v, ω), which directly control linear and angular movement. A carefully designed reward function penalizes misalignment, lateral drift, and collisions, while rewarding smooth and accurate docking. The Gazebo-based simulator supports rapid training iterations without the risk of damaging hardware. It also allows domain randomization, such as marker position jitter, sensor noise, and occlusions, which is essential for improving policy robustness and enabling successful sim-to-real transfer.

The Robot Operating System (ROS) serves as the middleware layer that connects the simulated environment in Gazebo with the real-world robotic platform. It plays a

crucial role in enabling seamless data exchange, synchronization of sensor streams, and deployment of trained policies across both domains. ROS provides publisher and subscriber nodes for acquiring sensor data, including RGB images, Depth data, and LiDAR scans, as well as for issuing velocity control commands. These nodes ensure that both the simulation and real-world environments expose a consistent set of ROS topics allowing the reinforcement learning framework to remain agnostic to whether the data is coming from Gazebo or physical sensors. We developed a ROS node named sim_real_transfer to enable bidirectional control synchronization between the Gazebo simulation environment and the physical robot. This node can listen to RGB data, depth data, LiDAR data, or odometry data from both systems and, based on the control mode specified via the/switch_control topic ("sim" or "real"), forwards velocity commands from the simulation to the real robot or vice versa. This mechanism allows trained policies in simulation to be seamlessly deployed on real hardware and also enables the real robot to drive the simulation for debugging and analysis. As a central bridge in our sim-to-real architecture, this node ensures consistency in perception, state generation, and control across domains, providing a robust foundation for policy transfer and real-world validation.

In the real world, the physical robot mirrors the sensor configuration and perception stack of the simulator, ensuring consistency between training and deployment environments. The trained PPO policy is directly transferred to the real robot for evaluation, allowing us to test and validate docking behaviors in real-world conditions, such as varying lighting, floor textures, and sensor noise, without requiring policy re-training. This sim-to-real architecture not only ensures reproducibility and robustness of the learned behaviors, but also minimizes physical wear, reduces safety risks, and enables rapid iteration between simulation and deployment stages. By maintaining a consistent interface and data flow across both domains, our system achieves reliable performance in practical autonomous docking tasks.

3. Simulation and Experiment Results

3.1. PPO Based Docking Simulation

In the study, we adopt the Proximal Policy Optimization (PPO) algorithm implemented in the Stable-Baselines3 framework to train a policy for autonomous docking. The training is conducted in our custom DockingEnv, which follows the OpenAI Gym interface and interacts with a Gazebo-based digital twin via ROS. The PPO agent uses a multilayer perceptron (MLP) policy and value network, with key hyperparameters set as follows: 2048 steps per rollout, batch size of 64, discount factor $\gamma = 0.99$, GAE lambda = 0.95, learning rate = 3×10^{-4}, clipping range of 0.2, and 10 epochs per update. The reward function, designed in DockingEnv, encourages the agent to align with the docking station, reduce lateral error, and avoid collisions. The final trained model is saved as final_ppo_docking.zip for deployment to a real robot using the sim-to-real transfer framework. To facilitate monitoring and reproducibility, we log training data using TensorBoard and employ a checkpoint callback that saves the model every 10,000 steps. The total number of training timesteps is set to 100,000. The final trained policy is stored for deployment on a real robot using a sim-to-real transfer strategy. This configuration balances learning stability and sample efficiency, making it well-suited for the continuous control demands of docking with multi-sensor fusion.

The training process was monitored through key metrics logged in TensorBoard. As shown in Figure 3, the curve starts from a low reward (around -150), indicating poor initial performance with frequent collisions or misalignment. Over time, the average reward steadily increases and is over than 0, reflecting the agent's improved docking behavior. The

consistently rising episode reward suggests that the PPO agent is learning more effective and stable docking strategies as training progresses.

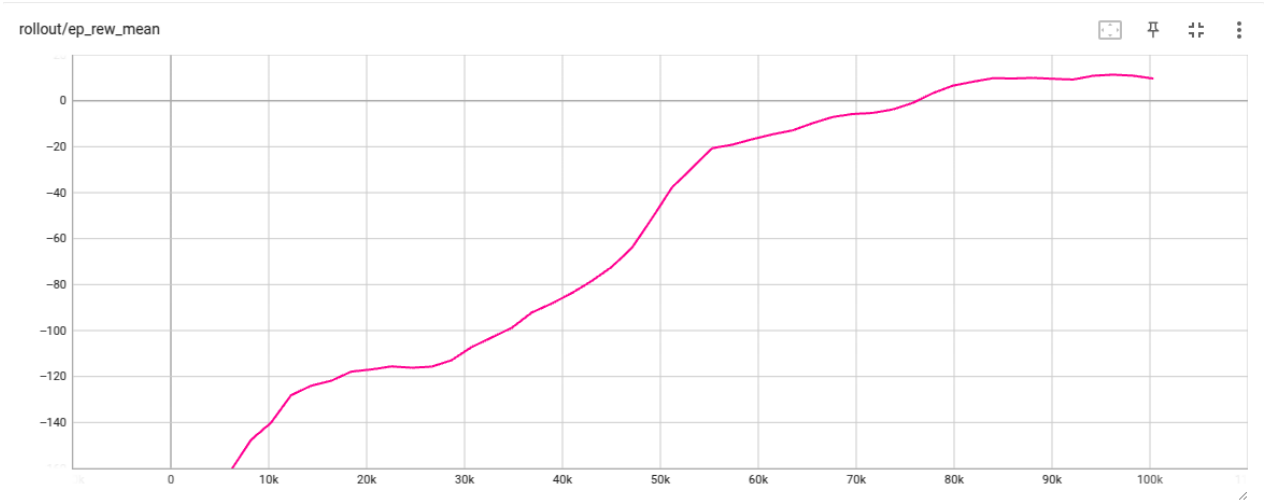


Figure 3. Mean Episode Reward.

In Figure 4, initially, the robot takes a long time to complete each episode (~260 steps), showing inefficiency. As training continues, the episode length decreases significantly to around 160 steps, meaning the robot docks more quickly and efficiently. The decreasing episode length indicates that the agent is learning to complete the docking task in fewer steps, suggesting better control and decision-making efficiency.

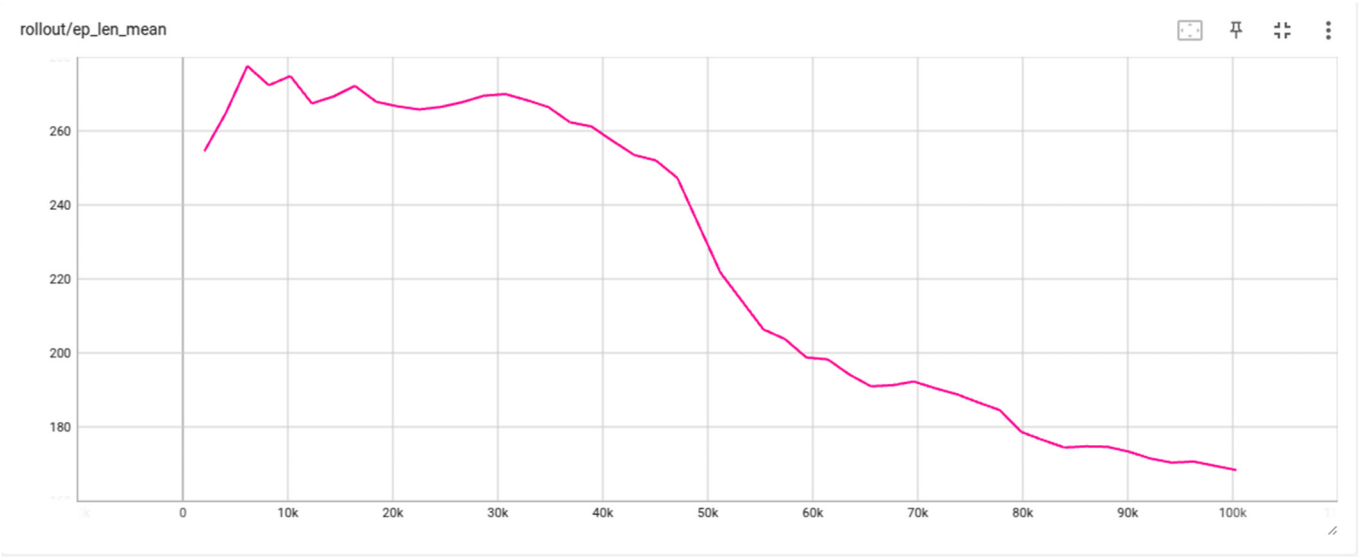


Figure 4. Mean Episode Length.

As shown in Figure 5, initially, the explained variance rises is lower than 0.5. Then it rises rapidly after 10 k steps and stays consistently above 0.95, which is a strong sign of accurate value estimation. A high explained variance (close to 1.0) indicates that the value function accurately predicts rewards, supporting stable and effective policy updates.

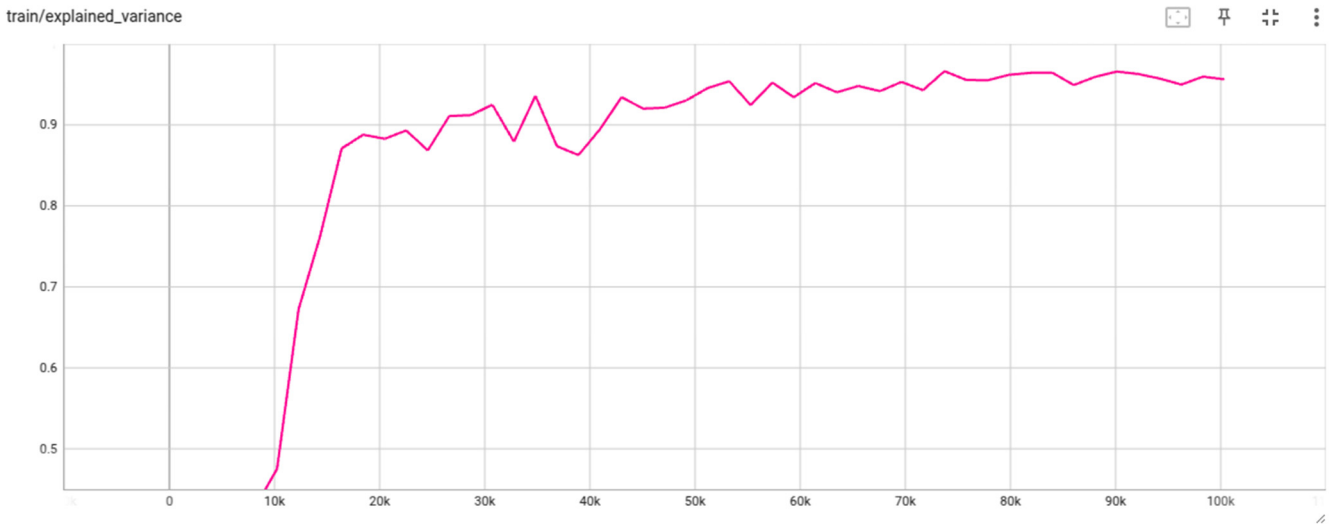


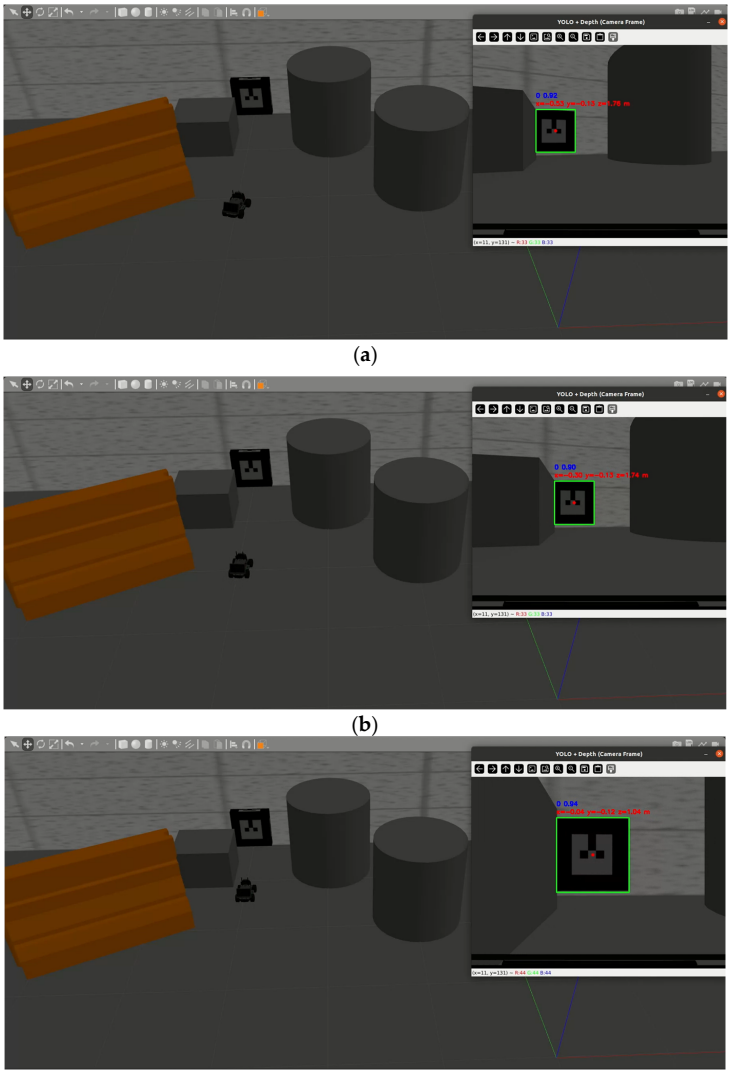
Figure 5. Explained Variance.

Figure 6 shows snapshots from the simulation video (https://www.dropbox.com/scl/fi/bxqokb88uu8xmya992iup/PPO_docking_sim1.mp4?rlkey=wlt965gs8dufe09vkr5uh8s60&st=jugg1gw1&dl=0) (accessed on 25 July 2025), where the PPO-trained autonomous mobile robot performs the docking task, including several obstacles. These snapshots are taken from the deployment of the final trained model and demonstrates the effectiveness of the learned policy. Figure 6a shows the initial stage, where the robot is far from the docking station. Figure 6b shows the approaching stage, where the robot has moved closer and begun alignment. Despite partial occlusion of the AprilTag, the robot was still able to detect the tag and successfully proceed with the docking process. This demonstrates the robustness of the YOLO-based perception system in handling incomplete visual cues. Figure 6c shows the final stage, where the robot has successfully finished docking. Throughout the docking process, the robot was able to detect obstacles in its environment, actively avoid them, and plan an optimal trajectory toward the docking target. This demonstrates the policy's ability to balance collision avoidance with goal-directed behavior. The right panel shows the YOLO + Depth detection results: the green bounding box indicates the detected docking marker, the red text displays the detected 3D position (x, y, z) of the target relative to the camera frame, and the blue text indicates the detection confidence score.

Figure 7 shows the lateral error during the docking process. Initially, the robot exhibits a lateral offset of approximately 0.5 m. As the PPO-based policy takes effect, the error is reduced steadily, converging to near-zero after around 80 timesteps. The remaining fluctuations are minimal, indicating that the retrained model achieves stable trajectory correction without significant overshoot or oscillation.

Figure 8 presents the distance to the marker. The robot approaches the docking target in a smooth and consistent manner, demonstrating reliable forward progression. The continuous decrease from 1.85 m to 1.1 m shows that the PPO policy enables effective distance control during docking.

Figure 9 illustrates the LiDAR distance difference between the left and right side, which reflects the angular misalignment of the robot. Initially, large deviations are observed, indicating significant yaw error. As the episode progresses, the difference gradually diminishes and stabilizes around zero, suggesting that the robot learns to align itself symmetrically with respect to the docking station.



(**a**)

(**b**)

(**c**)

Figure 6. Snapshots of PPO-based docking task at different stages: (**a**) initial stage, (**b**) approaching the docking station stage, (**c**) finishing docking stage.

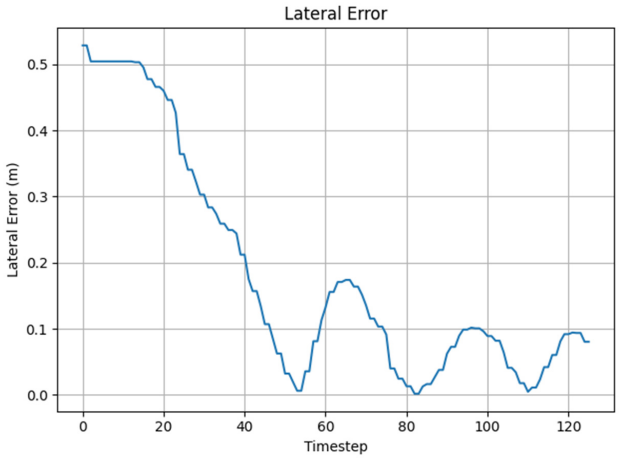


Figure 7. Lateral error during the docking process.

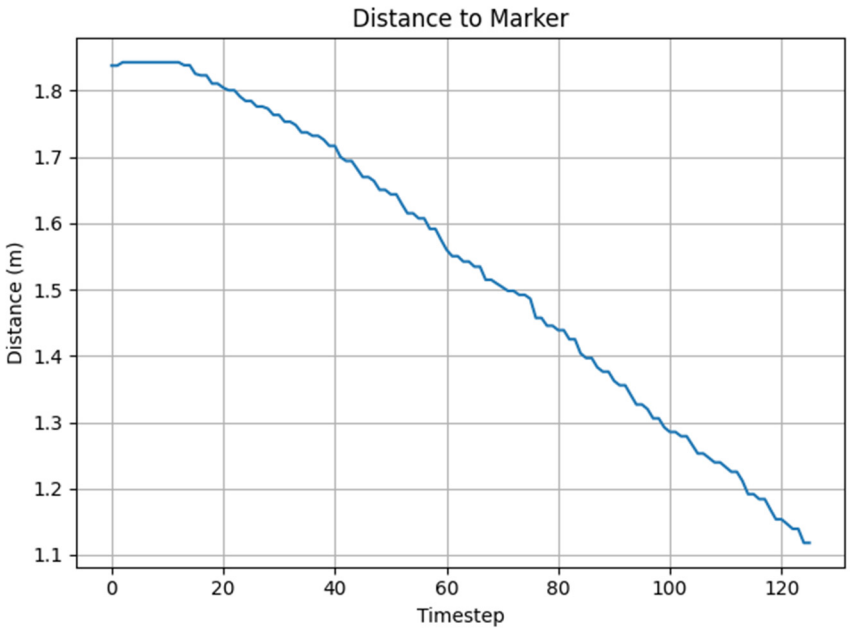


Figure 8. Distance to the marker during the docking process.

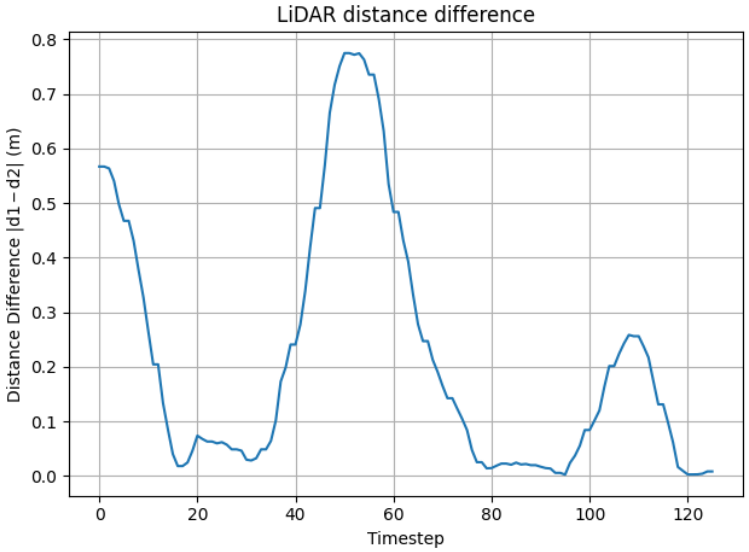
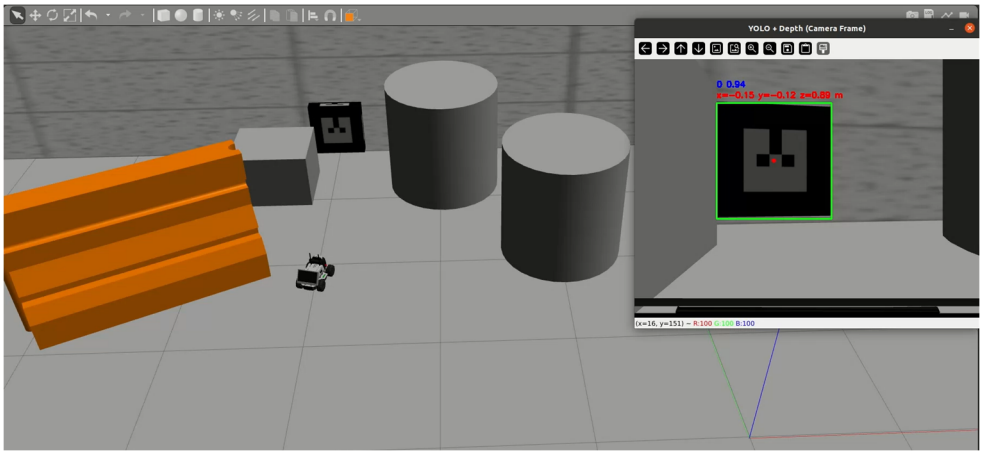
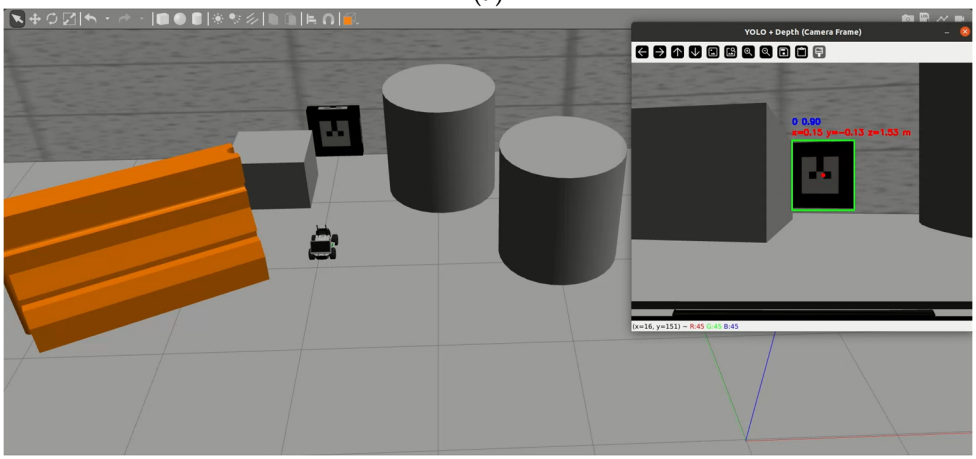


Figure 9. LiDAR distance difference between the left and right side.

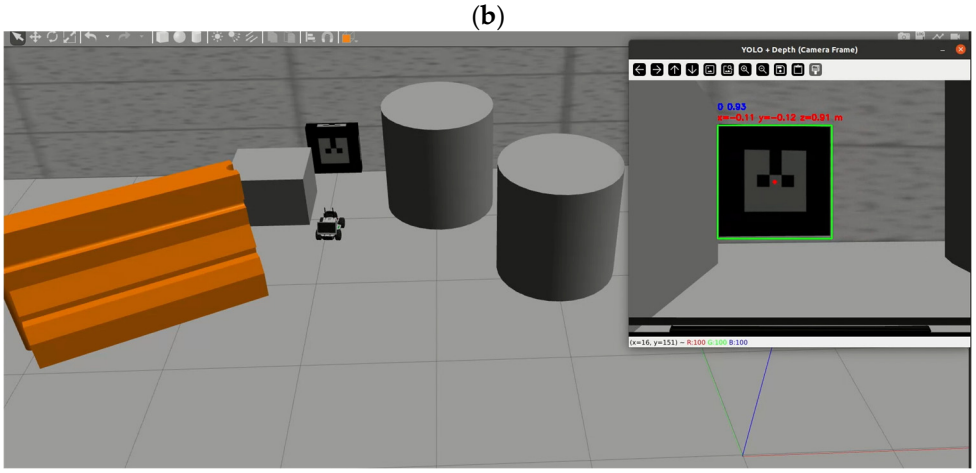
To demonstrate the robustness and generalization capability of the trained PPO policy, we conducted docking validation under varying illumination condition. Despite changes in ambient lighting, the robot consistently succeeded in completing the docking task without failure. Please refer to the video (https://www.dropbox.com/scl/fi/f52yv7se0ro88k51iht15/PPO_docking_sim2.mp4?rlkey=7zdr6k1morhx2f7mcpqdmlupt&st=zw0ni76o&dl=0) (accessed on 25 July 2025) for full trial footage. A representative snapshot from one of these trials is shown in Figure 10. Figure 10a shows the initial stage. Figure 10b shows the approaching stage and Figure 10c shows the final stage, where the robot has successfully finished docking.



(**a**)



(**b**)



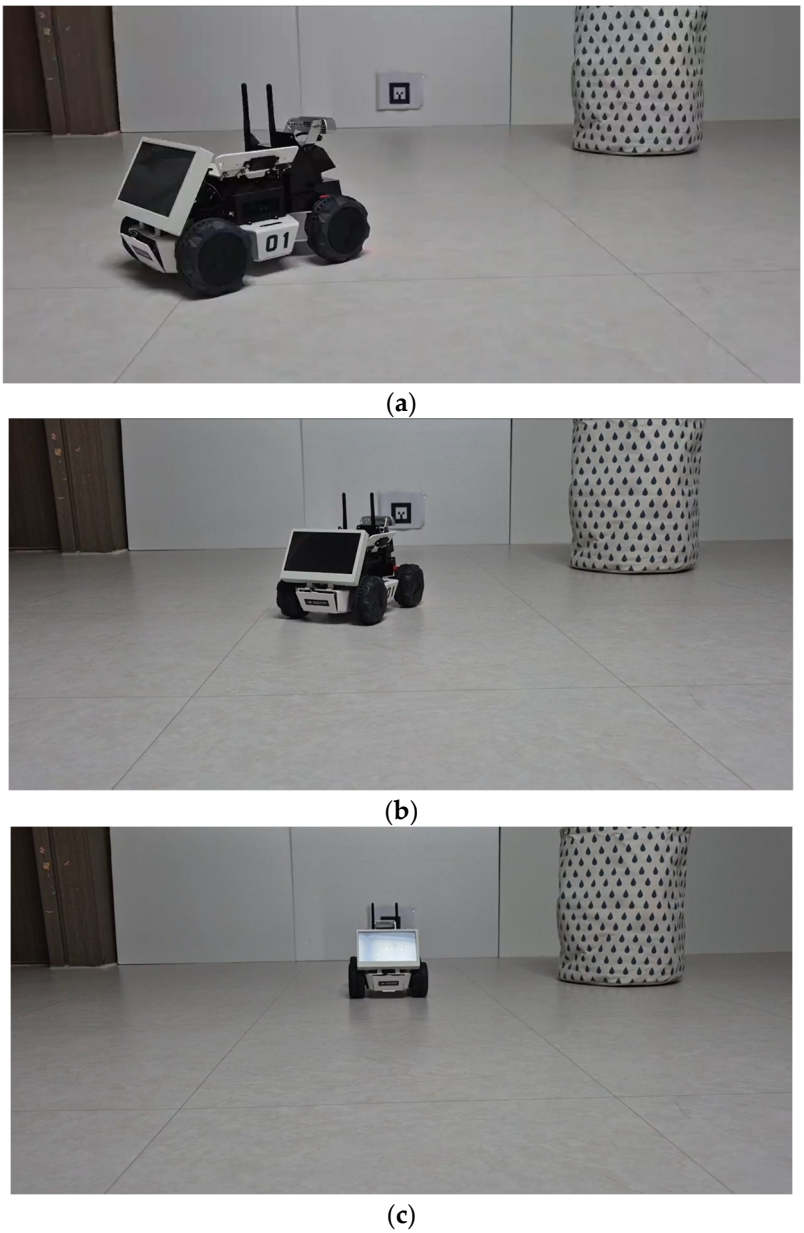
(**c**)

Figure 10. Snapshots of PPO-based docking task under strong illumination at different stages: (**a**) initial stage, (**b**) approaching the docking station stage, (**c**) finishing docking stage.

3.2. Sim-to-Real-Based Docking Deployment

To further validate the effectiveness of the trained policy in real-world scenarios, we deploy the PPO agent on a physical mobile robot using the sim-to-real transfer strategy. Snapshots from the real-world experiment video (https://www.dropbox.com/scl/fi/i0iixy81v3aobrfe9h8nf/Real_Experiment.mp4?rlkey=zlrrfwpturi5k1wvll2vuq0jn&st=ok7hxgi&dl=0) (accessed on 25 July 2025) are presented in Figure 11, demonstrating the docking behavior at three key stages. Figure 11a shows the robot in the initial stage, positioned away from the docking station. Figure 11b depicts the approaching phase, where the robot

has aligned itself with the marker and is moving closer. Figure 11c illustrates the final docking phase, where the robot has successfully positioned itself in front of the target marker. These results confirm that the policy trained entirely in simulation generalizes well to real-world environments with visual and depth sensing noise, successfully performing robust and accurate docking maneuvers.



(**a**)

(**b**)

(**c**)

Figure 11. Snapshots from the real-world deployment of the PPO-trained policy: (**a**) initial stage, (**b**) approaching the docking station, and (**c**) final docking stage.

Figure 12 shows the robot's movement trajectory during real-world docking. The trajectory begins at approximately (–3.0, 1.8) meters and proceeds along the Y-axis toward the docking station. Throughout the trajectory, the robot maintains a relatively straight path while performing minor lateral adjustments to reduce alignment error. This behavior reflects the effectiveness of the PPO-trained policy in controlling both linear and angular velocities for smooth and accurate docking. The small deviations in the X direction indicate that the agent successfully corrected misalignment using sensor feedback during the approach.

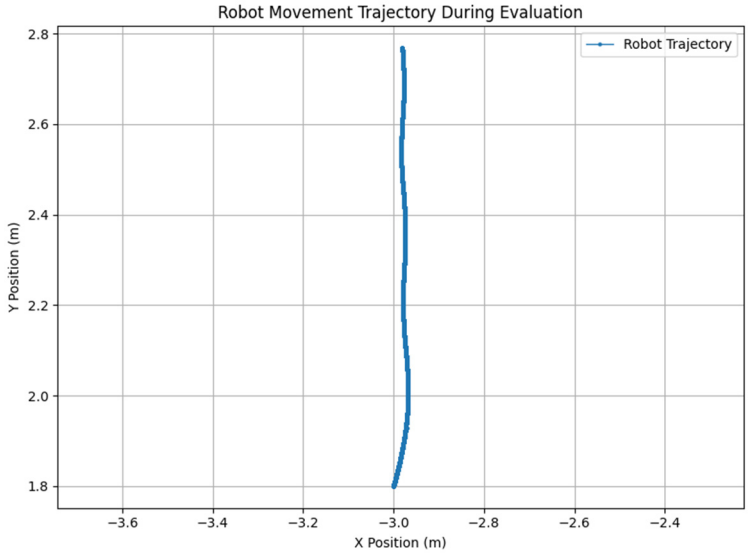


Figure 12. Robot's movement trajectory during real-world docking.

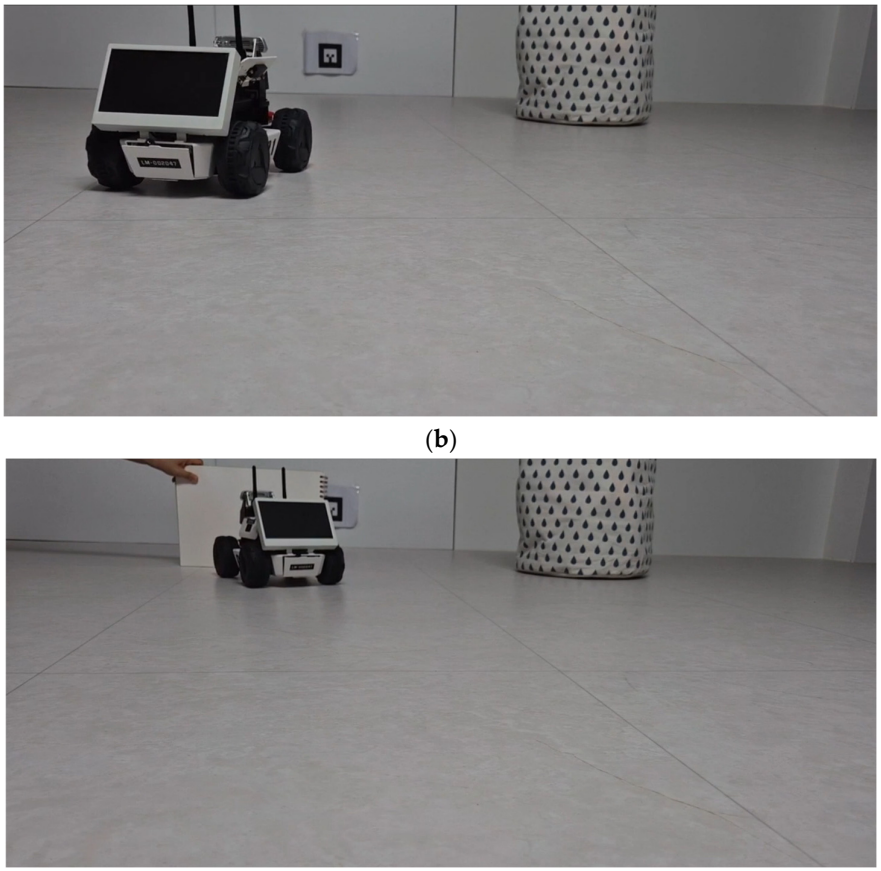
We conducted 50 real-world docking trials using the trained PPO policy on our AMR platform. The robot successfully completed 45 out of 50 docking attempts, resulting in a 90% success rate. The five failures were primarily attributed to slip errors caused by wheel-ground interaction, which led to inaccurate rotation during the yaw alignment phase. Despite these challenges, the average docking time across successful trials was approximately 18 s.

To enhance operational safety, we implemented an emergency stop mechanism. In real-world deployment, if the robot's LiDAR sensor detects an object within 0.1 m, the robot will automatically stop. As demonstrated in the video (https://www.dropbox.com/scl/fi/6tuxqdf1lja4fhl5qjv9g/Real_Experiment_Emergency_Stop.mp4?rlkey=473k187avf25zcd03m34m5cfw&st=j06vg21b&dl=0) (accessed on 25 July 2025) and shown in Figure 13, the robot performs this safety behavior in practice: Figure 13a the robot starts from its initial pose; Figure 13b it proceeds toward the docking station; Figure 13c upon detecting a white barrier closer than 0.1 m, the robot triggers the emergency stop and stops immediately.



(**a**)

Figure 13. *Cont.*



(**b**)

(**c**)

Figure 13. Snapshots from the real-world deployment with emergency stop: (**a**) initial stage, (**b**) approaching the docking station, and (**c**) emergency stop.

4. Conclusions

This paper presents a deep reinforcement learning-based autonomous docking framework for mobile robots, leveraging Proximal Policy Optimization (PPO) and multi-sensor fusion. By integrating YOLO-based visual detection, depth estimation, and LiDAR-based orientation correction, the system generates a robust and compact state representation that enables reliable docking even under partial occlusion, lighting variation, and cluttered environments. A Gym-compatible training environment is constructed in Gazebo as a high-fidelity digital twin, allowing safe, repeatable, and efficient policy training without hardware risks. The PPO agent outputs continuous linear and angular velocity commands, guided by a custom reward function that balances lateral error, angular misalignment, smoothness, and safety. The trained policy is deployed on a real AMR platform via a sim-to-real transfer strategy, achieving a 90% success rate in 50 real-world trials with average docking time of 18 s. Failures were primarily caused by wheel slip, leading to rotational deviations.

While these results confirm the feasibility and robustness of the proposed method, further large-scale quantitative trials (e.g., 100+ docking attempts under diverse conditions) are required to statistically validate its performance. Moreover, ablation studies on reward components and sensor modalities would clarify the contribution of each module, and benchmarking against classical controllers (e.g., PID, MPC) would strengthen comparative claims. Future work will focus on incorporating predictive safety layers for formal collision avoidance, extending the framework to dynamic obstacle scenarios, and testing on challenging terrains to address slip-induced errors.

Author Contributions: Conceptualization, Y.D.; methodology, Y.D.; software, Y.D.; validation, Y.D.; formal analysis, Y.D.; investigation, Y.D.; resources, Y.D.; data curation, Y.D.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D. and K.L.; visualization, Y.D.; supervision, K.L.; project administration, K.L.; funding acquisition, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Gyeongsangbuk-do RISE (Regional Innovation System & Education) project.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMRs	Autonomous mobile robots
PPO	Proximal policy optimization
RL	Reinforcement learning
ROS	Robot operating system

References

1. Vongbunyong, S.; Thamrongaphichartkul, K.; Worrasittichai, N.; Takutruea, A. Automatic precision docking for autonomous mobile robot in hospital logistics-case-study: Battery charging. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2021; p. 012060. [CrossRef]
2. Jia, F.; Afaq, M.; Ripka, B.; Huda, Q.; Ahmad, R. Vision- and Lidar-Based Autonomous Docking and Recharging of a Mobile Robot for Machine Tending in Autonomous Manufacturing Environments. *Appl. Sci.* **2023**, *13*, 10675. [CrossRef]
3. Hercik, R.; Byrtus, R.; Jaros, R.; Koziorek, J. Implementation of Autonomous Mobile Robot in SmartFactory. *Appl. Sci.* **2022**, *12*, 8912. [CrossRef]
4. Yilmaz, A.; Temeltas, H. A multi-stage localization framework for accurate and precise docking of autonomous mobile robots (AMRs). *Robotica* **2024**, *42*, 1885–1908. [CrossRef]
5. Fragapane, G.; de Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
6. Zeng, P.; Huang, Y.; Huber, S.; Coros, S. Budget-optimal multi-robot layout design for box sorting. *arXiv* **2025**, arXiv:22412.11281. [CrossRef]
7. Lai, D.; Zhang, Y.; Liu, Y.; Li, C.; Mo, H. Deep Learning-Based Multi-Modal Fusion for Robust Robot Perception and Navigation. *arXiv* **2025**, arXiv:2504.19002. https://arxiv.org/abs/2504.19002.
8. Lbiyemi, M.O.; Olutimehin, D.O. Revolutionizing logistics: The impact of autonomous vehicles on supply chain efficiency. *Int. J. Sci. Res. Updates* **2024**, *8*, 9–26. [CrossRef]
9. Kosari, A.; Jahanshahi, H.; Razavi, S.A. An optimal fuzzy PID control approach for docking maneuver of two spacecraft: Orientational motion. *Eng. Sci. Technol. Int. J.* **2017**, *20*, 293–309. [CrossRef]
10. Simon, J.N.; Lexau, A.; Lekkas, M.; Breivik, M. Nonlinear PID Control for Automatic Docking of a Large Container Ship in Confined Waters Under the Influence of Wind and Currents. *IFAC-PapersOnLine* **2024**, *58*, 265–272. [CrossRef]
11. Tian, C.; Liu, Z.; Chen, H.; Dong, F.; Liu, X.; Lin, C. A Lightweight Model for Shine Muscat Grape Detection in Complex Environments Based on the YOLOv8 Architecture. *Agronomy* **2025**, *15*, 174. [CrossRef]
12. Huang, X.; Yang, S.; Xiong, A.; Yang, Y. Enhanced YOLOv8 With VTR Integration: A Robust Solution for Automated Recognition of OCS Mast Number Plates. *IEEE Access* **2024**, *12*, 179648–179663. [CrossRef]
13. Telepov, A.; Tsypin, A.; Khrabrov, K.; Yakukhnov, S.; Strashnov, P.; Zhilyaev, P.; Rumiantsev, E.; Ezhov, D.; Avetisian, M.; Popova, O.; et al. FREED++: Improving RL Agents for Fragment-Based Molecule Generation by Thorough Reproduction. *arXiv* **2024**, arXiv:2401.09840. https://arxiv.org/abs/2401.09840.
14. Gottschalk, S.; Lanza, L.; Worthmann, K.; Lux-Gottschalk, K. Reinforcement Learning for Docking Maneuvers with Prescribed Performance. *IFAC-PapersOnLine* **2024**, *58*, 196–201. [CrossRef]
15. Wang, W.; Luo, X. Autonomous Docking of the USV using Deep Reinforcement Learning Combine with Observation Enhanced. In Proceedings of the 2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 27–28 August 2021; pp. 992–996. [CrossRef]

16. Lai, C.-C.; Yang, B.-J.; Lin, C.-J. Applying Reinforcement Learning for AMR's Docking and Obstacle Avoidance Behavior Control. *Appl. Sci.* **2025**, *15*, 3773. [CrossRef]
17. Anderlini, E.; Parker, G.G.; Thomas, G. Docking Control of an Autonomous Underwater Vehicle Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3456. [CrossRef]
18. Liu, S. An Evaluation of DDPG, TD3, SAC, and PPO: Deep Reinforcement Learning Algorithms for Controlling Continuous System. In Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023), Shanghai, China, 24–26 November 2024; pp. 15–24. [CrossRef]
19. Mock, J.; Muknahallipatna, S. A Comparison of PPO, TD3 and SAC Reinforcement Algorithms for Quadruped Walking Gait Generation. *J. Intell. Learn. Syst. Appl.* **2023**, *15*, 36–56. [CrossRef]
20. Stączek, P.; Pizoń, J.; Danilczuk, W.; Gola, A. A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment—A Case Study. *Sensors* **2021**, *21*, 7830. [CrossRef] [PubMed]
21. Liang, X.; Xiao, R.; Zhang, J. A Review on Digital Twin for Robotics in Smart Manufacturing. In Proceedings of the 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA), Chengdu, China, 16–19 December 2022; pp. 1510–1515. [CrossRef]
22. Szybicki, D.; Pietruś, P.; Burghardt, A.; Kurc, K.; Muszyńska, M. Application of Digital Twins in Designing Safety Systems for Robotic Stations. *Electronics* **2024**, *13*, 4179. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.