JAIST Repository
https://dspace.jaist.ac.jp/

Title	Explainable and transferable deep reinforcement learning for adaptive patrol of rail-guided robot system
Author(s)	Lee, Hosun; Kwon, Jaesung; Chong, Nak Young; Yang, Woosung
Citation	PeerJ Computer Science, 12: e3722
Issue Date	2026-04-14
Type	Journal Article
Text version	publisher
URL	https://hdl.handle.net/10119/20347
Rights	Copyright (c) 2026 Authors. Hosun Lee, Jaesung Kwon, Nak Young Chong and Woosung Yang. PeerJ Computer Science 12:e3722. This is an Open Access article distributed under the terms of Creative Commons Licence CC-BY [https://creativecommons.org/licenses/by/4.0/]. Original publication is available on PeerJ via https://doi.org/10.7717/peerj-cs.3722.
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

PeerJ Computer Science

Explainable and transferable deep reinforcement learning for adaptive patrol of rail-guided robot system

Hosun Lee[1], Jaesung Kwon[2], Nak Young Chong[1] and Woosung Yang[2]

[1] Japan Advanced Institute of Science and Technology, Ishikawa, Japan
[2] Kwangwoon University, Seoul, Republic of Korea

ABSTRACT

Intelligent facility management systems can reduce the workload of human operators by enabling autonomous operation. However, the lack of transparency in existing machine learning-based systems often hinders user trust, especially in safety-critical environments such as industrial and public facilities. To ensure reliability and accountability, autonomous systems must not only perform effectively but also provide human-understandable explanations for their actions. This article presents an explainable deep reinforcement learning framework for a rail-guided patrol robot that adaptively controls its speed based on the visual complexity of its surroundings. The proposed system employs the Deep Deterministic Policy Gradient (DDPG) algorithm to learn a continuous speed-control policy directly from image-based observations. To enhance transparency, Gradient-weighted Class Activation Mapping (Grad-CAM) is integrated into the actor network to visualize which spatial regions of the input most strongly influence speed decisions, providing *post hoc* explanations of the model's decisions. To support real-world deployment, we incorporate a Cycle-Consistent Generative Adversarial Network (CycleGAN)-based domain adaptation module that transforms real camera images into a simulation-compatible visual style, enabling the trained policy to operate without additional retraining. Grad-CAM is also used to assess the semantic consistency of translated images and verify that domain adaptation preserves task-relevant visual cues. Because the proposed framework is designed around lightweight visual inputs and compact neural networks, its computational demand remains modest and suitable for embedded execution. Grad-CAM analysis is used for explainability rather than for action generation, and its computation does not affect the timing of the control loop. The framework is evaluated through extensive experiments in both simulation and a physical testbed environment. Results demonstrate that the robot successfully adjusts its patrol speed in response to scene complexity and that the learned policy provides coherent and meaningful visual explanations. These findings highlight the potential of combining deep reinforcement learning, visual domain adaptation, and explainable AI to realize trustworthy and adaptable autonomous patrol systems.

Submitted 28 August 2025
Accepted 2 February 2026
Published 14 April 2026

Corresponding author
Woosung Yang, dreamrize@kw.ac.kr

Academic editor
Paulo Jorge Coelho

Additional Information and Declarations can be found on page 30

DOI **10.7717/peerj-cs.3722**

Copyright
2026 Lee et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

Subjects Autonomous Systems, Data Mining and Machine Learning, Robotics
Keywords Rail-guided patrol robot, Adaptive speed control, Deep reinforcement learning (DRL), DDPG-based robot control, Simulation-to-real transfer, CycleGAN-based domain adaptation, Explainable artificial intelligence (XAI), Grad-CAM visualization, Facility monitoring automation

How to cite this article Lee H, Kwon J, Chong NY, Yang W. 2026. Explainable and transferable deep reinforcement learning for adaptive patrol of rail-guided robot system. **PeerJ Comput. Sci. 12:e3722 DOI 10.7717/peerj-cs.3722**

INTRODUCTION

There has been an increase in the demand for intelligent systems that perform monitoring and surveillance of multi-use facilities such as wholesale markets, exhibition halls, and terminals. These facilities often require continuous monitoring for repetitive and routine tasks, which can be burdensome for human operators. The current approach involves installing a large number of fixed monitoring devices such as CCTV cameras, sensors, and alarms. While these systems are easy to deploy and operate, they offer limited field coverage, lack mobility, and are insufficient in responding effectively to unexpected on-site events.

Over the past decade, robotic systems have attracted growing attention for their potential to provide scalable, flexible, and efficient automation solutions for large-scale facility monitoring. To overcome the limitations of fixed devices, robot-assisted facility management approaches have been explored, aiming to improve situational awareness and rapid response capabilities (*Halder & Afsari, 2023*; *Rakha & Gorodetsky, 2018*; *Tavakoli et al., 2018*; *Carvalho et al., 2017*; *Lee et al., 2023*; *Macaulay & Shafiee, 2022*).

Mobile platforms such as unmanned aerial vehicles (UAVs) and automated guided vehicles (AGVs) have been studied for patrol tasks in complex environments, but they often require high-level position control and collision avoidance (*Rakha & Gorodetsky, 2018*). Rail-guided systems offer an attractive alternative due to their simplified localization and safe separation from people and equipment (*Bengel et al., 2009*; *Christensen et al., 2011*; *Kim et al., 2012*; *Tavakoli et al., 2018*). Rails installed along ceilings or walls ensure dedicated patrol paths, enabling robots to follow scheduled monitoring routes reliably and without interference. In emergencies, the robot can quickly reach the designated site by bypassing obstacles on the ground. By automating repetitive monitoring tasks and embedding incident detection functions, managers can focus on responding to critical alarms and improve operational efficiency (*Macaulay & Shafiee, 2022*). These structural characteristics provide repeatable and constrained motion patterns that simplify system operation and learning, but do not imply explicit disturbance-rejection or robust control guarantees.

Various studies have addressed facility monitoring using rail-guided robots (*Galassi et al., 2014*; *Xu et al., 2018*). This research aims to develop an intelligent patrol system (see Fig. 1) that integrates autonomous navigation and multiple sensors to detect and respond to conflagrations, accidents, and hazardous conditions throughout a facility. To achieve this, a rail-guided mobile platform is designed to provide safe and efficient monitoring coverage. The system further integrates image-based and deep learning-based detection functions to assist human operators and enhance proactive safety management.

However, despite their mechanical stability, most existing rail-guided systems still rely on fixed-gain or fixed-speed control. Such controllers implicitly assume a consistent visual environment, yet indoor facilities exhibit large variations due to human congestion, object movement, and illumination changes. These nonlinear and scene-dependent factors make it difficult to analytically determine an optimal patrol speed, causing fixed-speed control to either overshoot in complex areas or move too slowly in simple segments.

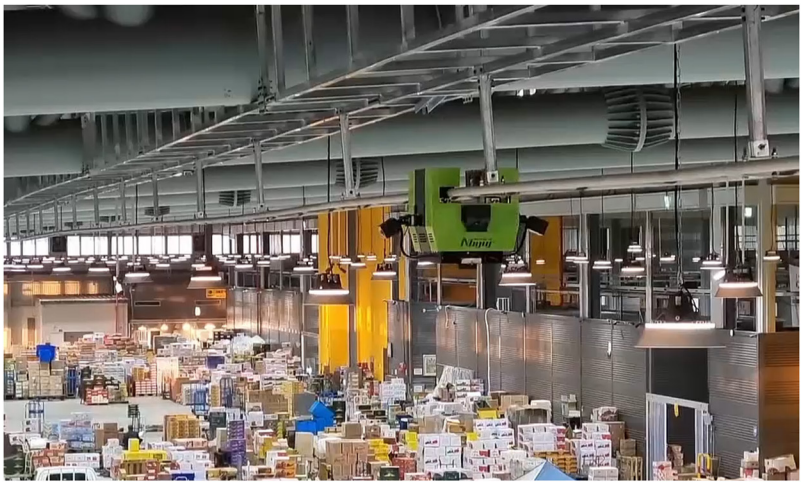


Figure 1 **Rail-guided smart patrol robot on an agricultural wholesale market.**
Full-size DOI: 10.7717/peerj-cs.3722/fig-1

Similar limitations have been reported in broader mobile robot research, where classical PID or fuzzy logic controllers degrade under dynamic or visually cluttered conditions (*Hentout, Maoudj & Kouider, 2024*). Recent studies further show that machine learning–based gain optimization is often required even for humanoid or assistive robots, particularly when digital twins are used to capture real-world variability (*Mateescu et al., 2025*). Motivated by these findings, we also evaluate fixed-speed baselines to highlight their inability to ensure consistent information acquisition across heterogeneous scenes.

Learning-based control provides a more suitable alternative because a policy can directly interpret raw visual input and adapt online without explicit modeling. Reinforcement learning enables behavior that adjusts to scene complexity, human activity, and other emergent visual cues—an essential capability for real-world deployment where environmental variations are unavoidable.

While our previous work focused on developing the robotic platform (*Lee et al., 2023*) and implementing a Deterministic Policy Gradient (DDPG)-based speed control policy in simulation (*Lee et al., 2024*), this article advances the framework by adding explainable AI capabilities and sim-to-real deployment strategies for real-world facility monitoring.

The remainder of this article is organized as follows. 'Preliminaries' reviews the background on facility monitoring, deep reinforcement learning, explainable AI, and simulation-to-real transfer. 'Rail-Guided Smart Patrol Robot for Facility Safety Management' introduces the rail-guided smart patrol robot and its operational architecture, which provides the physical platform for adaptive speed control. 'Adaptive Patrol Speed Control Scheme Based on DDPG Algorithm' presents the proposed adaptive patrol-speed framework, including the DDPG controller, reward design, Gradient-weighted Class Activation Mapping (Grad-CAM)-based explainability, and Cycle-Consistent Generative Adversarial Network (CycleGAN)-enabled sim-to-real deployment. 'Evaluation' reports experimental results in both simulation and real-world environments, including baseline comparisons, visual explanations, and quantitative

evaluation of domain adaptation. Finally, 'Conclusion' concludes the article and discusses limitations and future research directions.

PRELIMINARIES

Smart patrol systems for facility monitoring

Portions of this text were previously published as part of a preprint (https://doi.org/10.1109/SII55687.2023.10039462); see also (*Lee et al., 2023*). Smart patrol systems are designed to autonomously monitor environments such as industrial plants, utility tunnels, and indoor facilities. These systems commonly utilize mobile platforms equipped with sensors, including cameras, LiDAR, and thermal detectors, to collect environmental data (*Halder & Afsari, 2023*; *Rakha & Gorodetsky, 2018*). Traditional patrol robots typically follow pre-defined routes at constant speeds, which limits their ability to respond to variable risk levels or dynamically changing surroundings.

Compared to fixed monitoring devices such as CCTV and environmental sensors, smart patrol robots can provide mobile coverage and targeted inspection based on real-time sensor feedback. Rail-guided robots, in particular, simplify the navigation problem by constraining the robot's path, enabling precise control while maintaining safety in shared environments (*Bengel et al., 2009*; *Kim et al., 2012*; *Tavakoli et al., 2018*). However, for such systems to be truly intelligent and autonomous, adaptive decision-making and human-understandable reasoning are essential—both of which are the focus of this work.

Deep reinforcement learning

Deep Reinforcement Learning (DRL) combines reinforcement learning (RL) with deep neural networks to enable agents to learn complex policies directly from high-dimensional input data such as images (*Lillicrap et al., 2015*). An agent learns to interact with an environment by observing states, taking actions, and receiving rewards that guide the learning process.

In this study, we adopt the DDPG algorithm, a model-free, off-policy actor-critic method that is particularly well-suited for continuous action spaces. DDPG consists of two neural networks: (1) an *actor* that predicts the best continuous action (*e.g.*, patrol speed) given the current state (*e.g.*, camera image), and (2) a *critic* that evaluates the predicted action by estimating its expected return. The DDPG algorithm enables the patrol robot to learn how to adjust its movement speed based on environmental complexity inferred from visual inputs, without relying on manually defined heuristics.

Our earlier work on rail-guided patrol robot development (*Lee et al., 2023*) established the hardware and software infrastructure for autonomous monitoring. Building upon this platform, a follow-up study (*Lee et al., 2024*) proposed a DDPG-based patrol speed controller that adaptively adjusted the robot's speed based on visual complexity in a simulated environment. While this prior approach demonstrated the potential of deep reinforcement learning for intelligent speed control, it lacked support for explainability and real-world applicability. In this article, we extend the framework by incorporating Grad-CAM for *post hoc* explanation and leveraging CycleGAN-based domain adaptation to enable sim-to-real deployment in physical environments.

Explainable artificial intelligence (XAI)

As AI systems grow in complexity, their internal decision-making becomes increasingly opaque—a phenomenon often referred to as the "black-box problem." Explainable Artificial Intelligence (XAI) aims to provide *post hoc* reasoning tools that help humans understand, trust, and manage the outputs of such models (*Vilone & Longo, 2020*, *2021*).

It is important to distinguish *explainability* from *interpretability*: interpretability refers to how inherently understandable a model is (*e.g.*, linear regression), while explainability focuses on producing understandable outputs or explanations for models that are not interpretable by default, such as deep neural networks.

In our work, the DDPG-based control policy is non-interpretable due to its use of deep neural networks. To address this, we employ Grad-CAM, a widely used *post hoc* explainability technique, to generate visual explanations of the robot's decisions (*Selvaraju et al., 2017*). Grad-CAM highlights the spatial regions of the input image that contributed most to the output action, offering insights into what visual features the model "attended to." This improves transparency, supports operator trust, and assists in diagnosing unintended behaviors in autonomous patrol systems.

Simulation-to-real transfer *via* domain adaptation

Training DRL models directly in real-world environments is often impractical due to safety concerns, limited access, and high operational costs. Consequently, DRL agents are typically trained in simulation environments where exploration is cheaper, faster, and safer.

However, a key challenge in deploying these trained policies in real-world applications is the *simulation-to-reality gap* (or sim-to-real gap)—a performance degradation that arises from discrepancies between the simulated training environment and the real-world deployment setting. These discrepancies may stem from visual differences, sensor noise, or imperfect physical modeling (*James et al., 2019*; *Tobin et al., 2017*).

To overcome this challenge, domain adaptation methods are employed to reduce the mismatch between simulated and real-world inputs. In this study, we adopt CycleGAN (*Zhu et al., 2017*), an unpaired image-to-image translation method, to transform real-world camera inputs into the visual style of the simulation environment. This transformation allows the trained policy to process real-world observations in a consistent format, thereby enabling zero-shot transfer without requiring additional fine-tuning.

We further use Grad-CAM visualizations to evaluate the semantic alignment of the translated images, ensuring that CycleGAN preserves task-relevant visual features during domain adaptation. This combination of domain adaptation and visual explanation supports robust and explainable real-world deployment. The notation used throughout this article is summarized in Table 1.

RAIL-GUIDED SMART PATROL ROBOT FOR FACILITY SAFETY MANAGEMENT

Figure 2 shows detailed configurations of the proposed rail-guided robot-based facility monitoring system. To accomplish the development of the target system, the automated

Table 1 Notation used in this paper. Bold letters denote vectors or matrices.

Symbol	Description
$\mathbf{s}_t$	State vector at time t
a_t	Patrol speed (scalar action)
r_t	Reward (scalar)
$\mu_{\theta^\mu}(\mathbf{s}_t)$	Deterministic policy (actor)
$Q_{\theta^Q}(\mathbf{s}_t, a_t)$	Action-value function (critic)
$\mathscr{D}$	Replay buffer (set of transitions)
$\mathbf{A}^k$	k-th CNN feature map (matrix)
$\mathbf{L}^c$	Grad-CAM heatmap for class/action c (matrix)

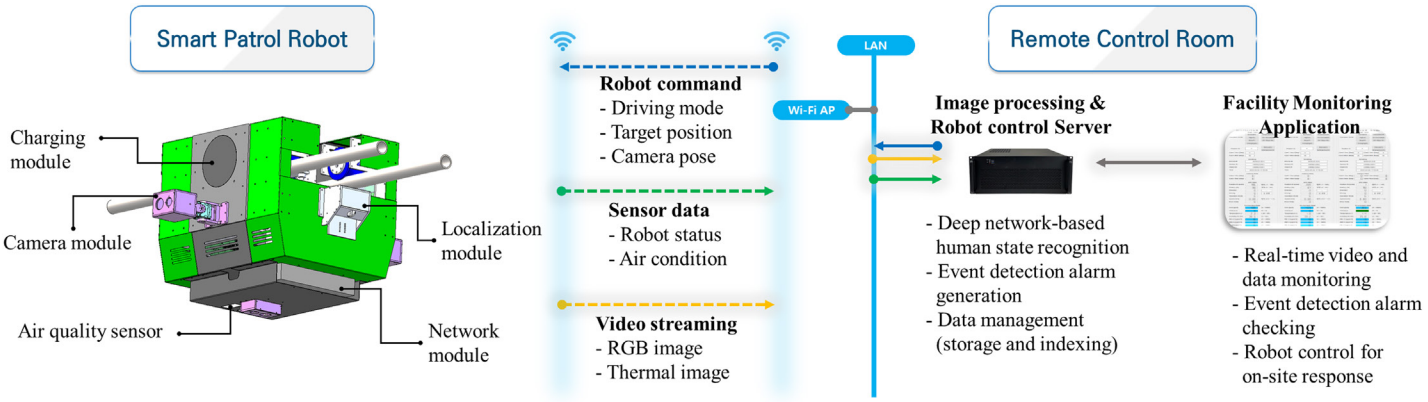


Figure 2 System overview: configuration of Smart patrol robot, the definition of data, and services of Remote control room.
Full-size DOI: 10.7717/peerj-cs.3722/fig-2

patrol robot and the data processing server in the remote control room are required to embed the next five functions.

- **24-h mobile patrol**: The smart patrol robot automatically drives according to the plan to check the status of the facility, and it is possible to return to the charging station after completion. Considering the monitoring viewing angle and distance, the robot is required to place within ± 0.2 m position error. The docking range of the wireless charging station is within ± 0.05 m. Through precise position control using the brushless direct current (BLDC) motor, the driving unit is possible to achieve with ± 0.01 m error by the encoder pulse computation. Additionally, the absolute position is collected using the barcode reader of the localization module by reading the barcode attached to every 1 m of the rail path. To prevent collisions while driving, ultrasonic sensors are installed in the front and rear to perform emergency stop operations to facilitate automatic driving.
- **Active camera motion**: The camera module attached to the side includes a pan-tilt actuator to move the direction of the cameras up, down, left, and right, so the operator can check the situation in the facility where detailed observation is required. A full-HD RGB camera with 60° Field Of View (FOV) and a thermal camera with a 320 × 240

pixels sensor, 60 ° FOV are selected to cover a cell of stores horizontally and floor to top of the store vertically. Real-Time Streaming Protocol (RTSP) servers are created for each camera to stream videos over a connection from the server.

- **Environmental monitoring**: The environmental condition in the facility is monitored by measuring the air quality at each location during patrol. The air quality sensor is selected to measure the following seven types of air quality; temperature, humidity, CO_2, Volatile Organic Compounds (VOC), Particulate Matter ($PM1.0$, $PM2.5$, $PM10$). The measured data is converted into a message form and transmitted to the server every 1 s.
- **Fire detection and response**: All data acquired while the robot is patrolling in the facility is collected and stored in the server of the remote control room in real time. The video streams of the camera module are transferred to the server by connecting to the RTSP server running on the robot. The maximum temperature in the thermal image is detected and classified into a state of interest/caution/alert/danger according to the temperature level. The RGB image confirms whether a fall accident has occurred through human detection and posture recognition. To collect data, the server creates a Message Queuing Telemetry Transport (MQTT) message broker to collect and manage all data transmitted by the robot. Changes in air quality data are monitored and an environmental anomaly detection alarm is generated.
- **Dissemination of on-site situation and countermeasures**: In addition to the MQTT message broker for sensor data, an MQTT message broker for robot driving system control and camera posture control is also created to convert commands from the dashboard into messages. The message stored in the broker is checked in real time by the robot, and the robot operates according to the command. Therefore, the operator can check the data collected from the server and the generated alarm information, and remotely control the robot for further confirmation and action on the on-site situation.

Rail-guided patrol robot

Figure 3 shows detailed mechanical configurations of the driving system of the developed smart patrol robot. The proposed patrol robot is designed to run on a 2-track rail structure with four wheels. The rail structure is supported by square frames installed on the ceiling. The robot is driven by controlling the BLDC motor placed in the robot body. The motor shaft is connected to the differential gear and the gear output shaft is connected to the left and the right pulley and belt respectively. After the pulley and belt components, the driving wheels are connected. The circular pipe is gripped with a U-shaped wheel, and a damper pushes the pipe to maintain the contact condition between the robot and the rail.

Figure 4 shows the relationship between the robot motion, $x, \dot{x}, \ddot{x}$ and the output torque of the motor, τ_m, and it can be defined as follows:

$$m\ddot{x} + b\dot{x} + f_F = f_A \tag{1}$$

where, x is the position of the robot according to the rail path. m is the total mass of the robot, and b is the damping coefficient at the wheel. The driving force, f_A can be driven

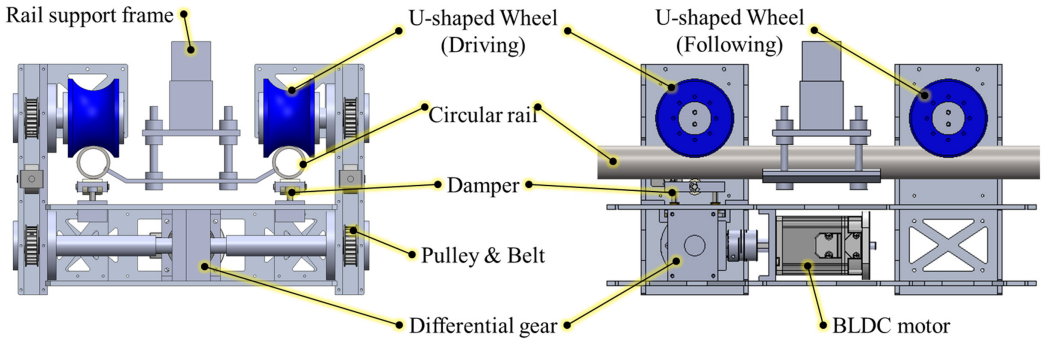


Figure 3 **Configuration of driving system.** Full-size DOI: 10.7717/peerj-cs.3722/fig-3

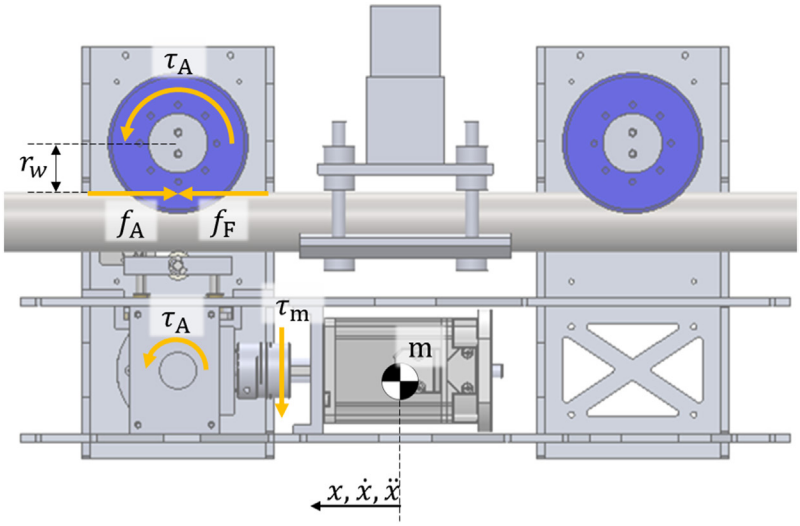


Figure 4 **Dynamic model of the smart patrol robot.** Full-size DOI: 10.7717/peerj-cs.3722/fig-4

with the driving torque, τ_A, and the radius of the wheel, r_w. The driving torque, τ_A is driven with the motor torque, τ_m, and the gear ratio, n.

$$f_A = \tau_A / r_w. \tag{2}$$

$$\tau_A = n\tau_m. \tag{3}$$

The following equation is the model of the friction force.

$$f_F = \mu mg \; sign(\dot{x}) \tag{4}$$

where, μ and g are the friction coefficient of the contact surface and the gravity acceleration respectively. The signum function is used to define the direction of the friction force. Using the above dynamic model, the motor selection and damper internal spring can be selected according to the target performance of the driving system. In this research, a 200 W BLDC motor with 3,000 rpm speed and 0.635 Nm rated torque is selected to achieve 1.8 m/s of the peak speed.

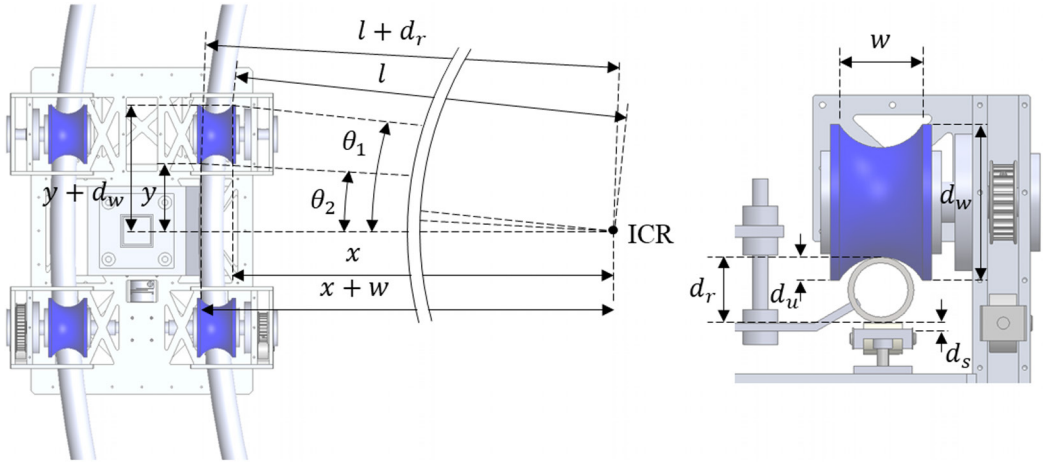


Figure 5 **Kinematical descriptions of the rail structure and the smart patrol robot.**
Full-size DOI: 10.7717/peerj-cs.3722/fig-5

Figure 5 shows the state of the minimum radius of rotation that does not deviate from the groove of the wheel. The robot can be driven by constraining it to a circular rail pipe using a U-shaped wheel and damper when the maximum stroke of the damper, d_s is shorter than the depth of the groove of the wheel, d_u. Therefore, the radius of rotation of the rail is limited according to the design parameters of the robot. The equation expresses the distances from the ICR to the inner contact point and the outer contact point of the wheel and the rail.

$$l \sin \theta_1 = y + d_w \tag{5}$$
$$l \cos \theta_1 = x \tag{6}$$
$$(l + d_r) \sin \theta_2 = y \tag{7}$$
$$(l + d_r) \cos \theta_2 = x + w \tag{8}$$

where, the radius of the rotational path, l, and the distance between ICR and wheel, x, can be found by determining the design parameters of the robot and the rail; the distance between the front wheels and the rear wheels, y, the wheel diameter, d_w, the rail diameter, d_r, the groove width of the wheel, w. Using trigonometric functions and rearranging the equation, the expression for l can be obtained as follows:

$$l = \sqrt{(y + d_w)^2 + x^2} \tag{9}$$

where,

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}. \tag{10}$$
$$a = \left(\frac{w}{d_r}\right)^2 - 1. \tag{11}$$
$$b = \frac{w(y^2 + w^2 - (y + d_w)^2 - d_r^2)}{d_r^2}. \tag{12}$$

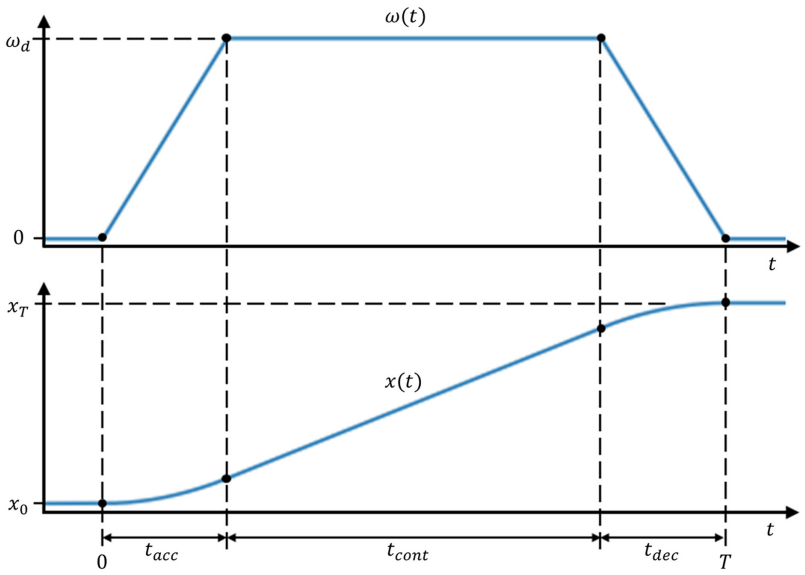


Figure 6 Speed reference model for driving control. Full-size DOI: 10.7717/peerj-cs.3722/fig-6

$$c = \left(\frac{y^2 + w^2 - (y + d_w)^2 - d_r^2}{2d_r}\right)^2 - (y + d_w)^2. \tag{13}$$

Figure 6 is a reference model of the speed controller used for position control of the robot. The acceleration time and deceleration time are experimentally decided in the range where vibration does not occur during acceleration and deceleration. When the target distance and constant velocity are determined, the position of the robot can be controlled with constant movement time. Also, it is possible to correct errors that occur due to the acceleration/deceleration of the robot and the slip of the wheels.

Management system

To manage the facility with the configured robot-server system, it is necessary to classify the necessary tasks and define the appropriate operation commands. The tasks are largely divided into a task that collaborates with the operator and a planned task without the operator's intervention, and the actions are defined as the task for movement and the task for data acquisition.

- **Normal-speed patrol**: In order to monitor the status of the facility without the intervention of an operator, the robot carries out the patrol task according to a planned schedule, acquires environmental data, and performs video streaming. In a situation where the facility is in a normal state, it is driven at a low speed (0.6 m/s) to periodically check changes in the environmental condition to acquire as much data as possible about the facility. The angles of the camera module are fixed in advance while performing patrol operations. The alarm information is generated from the detection function and provided to the operator. This operation can reduce the workload by acting on behalf of operators on repetitive and tedious tasks.

- **High-speed patrol**: At the request of the manager, it moves at a faster speed than normal speed patrols to quickly monitor the general management area of the facility remotely. both environmental data acquisition and video streaming are performed in the same way as normal speed patrols. The angles of the camera module are fixed in advance while performing patrol operations. The maximum speed (1.2 m/s) is determined as within the range where video streaming is possible. When driving at high speed, it is difficult to obtain enough image data for the detection function, so the alarm information cannot be provided. This operation can increase work efficiency by assisting operators with repetitive and tedious tasks faster.
- **Manual driving**: This operation mode is for the operator to remotely monitor a specific location of the facility from the control room. The operator's decision is the first prior and allows a high degree of freedom. The robot receives the location input from the operator and moves at high speed (1.0 m/s). While driving, the operator can manually control the camera angle remotely. However, the detection function is not supported until arriving at the target point. This operation can increase work performance by extending the operators' task area.
- **Emergency**: When an emergency alarm is received from the linked external system, the robot operation task can be immediately confirmed with this operation command. The robot moves at the maximum speed (1.5 m/s) according to the received emergency information and performs the detection and monitoring functions after arriving at the site. This operation provides the highest level of automation, however, it is important to link it with highly reliable information.

Figure 7 is a flowchart for the transition of operation mode for task execution. In addition to operation modes, there are actions required to initialize the system or switch to another mode after stopping or completing each operation mode. Stop mode is a command to pause all work, and it is possible to resume or change work after grasping the situation. In charging mode, the battery status is checked to recharge the consumed power after performing all tasks and returns to the charger when the battery is low. Robot control commands are transmitted to the robot in the form of messages through the server's MQTT broker.

The facility monitoring application can monitor all information about the robot status, environment state, and detection alarm by subscribing to three message brokers; robot data, sensor data, and alarm data. And robot control is also possible by publishing a robot command message to the robot command MQTT broker.

ADAPTIVE PATROL SPEED CONTROL SCHEME BASED ON DDPG ALGORITHM

In order to effectively monitor the situation of the target facility, we propose a control model based on the DDPG algorithm as a method of training the optimum robot patrol speed based on the amount of information on the site. DDPG is an actor-critic algorithm-based reinforcement learning algorithm that applies Deep Neural Network techniques to the Deterministic Policy Gradient (DPG) algorithm (*Lillicrap et al., 2015*).

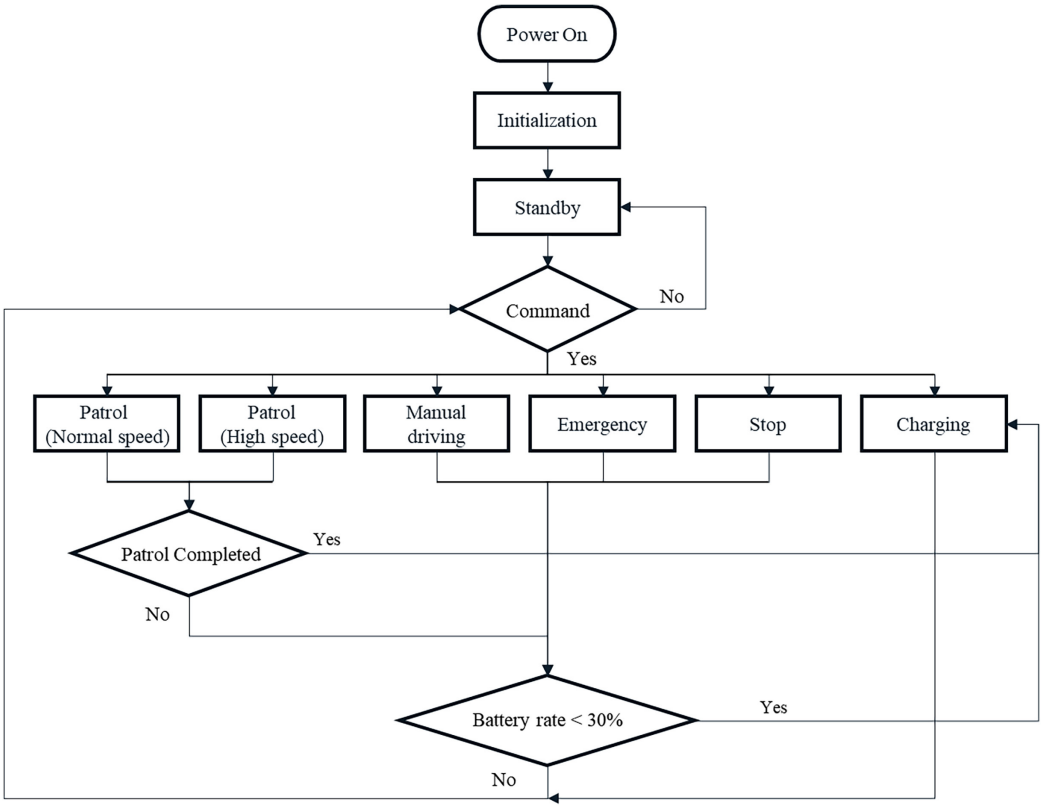


Figure 7 **Flow chart of smart patrol robot operation.** Full-size DOI: 10.7717/peerj-cs.3722/fig-7

The existing policy-based reinforcement learning algorithm can only handle discrete actions where the policy outputs the probability of taking the possible actions. However, the DPG algorithm deterministically determines and outputs the action value among continuous values without using the probability distribution of the policy to determine the action. Therefore, it is possible to output the action value in the real number range without making a choice.

Problem definition

The patrol speed control problem in facility monitoring is formulated as a continuous control task within a reinforcement learning framework. The robot should adjust its patrol speed adaptively based on the complexity of the observed environment, as interpreted from onboard image data.

Let $\mathbf{s}_t$ be the observed state at time t, which consists of the visual input and optional sensor data. The action $a_t \in \mathbb{R}$ corresponds to the patrol speed. The objective is to learn a deterministic policy $\pi : \mathbf{s}_t \rightarrow a_t$ that maximizes the expected cumulative reward:

$$J = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{14}$$

where γ is the discount factor, and r_t is the reward signal at time t. To define the reward, the amount of information in the observed image is estimated using entropy. The robot is

expected to move slowly in high-entropy (information-rich) areas and quickly in low-entropy (redundant) areas to maximize monitoring efficiency.

Specifically, the reward function is designed to minimize the difference between the desired information acquisition, E_d, and the actual information acquired from the newly observed area, calculated based on the image entropy. The amount of obtained information is defined as the entropy equal to the ratio of the newly added area among the entropy of the current frame image. The ratio of the newly added area is defined by the distance the robot travels between two frames, v/fps, over the width of the field of view, w_{FOV}. Therefore, the reward function is expressed as:

$$reward = -\left|E_d - E_f\left(\frac{v}{fps \times w_{FOV}}\right)\right|. \tag{15}$$

Although entropy is a widely used proxy for visual information, it also has inherent limitations. High entropy can be produced by texturally complex but semantically insignificant regions, such as noisy backgrounds or high-frequency patterns without structural meaning. Therefore, entropy does not fully capture task-relevant semantic importance. In our setting, however, entropy remains effective because the rail-guided patrol scenario contains consistent structural cues whose entropy variations correlate strongly with changes in visual complexity. Moreover, entropy is computationally lightweight and can be evaluated densely at every time step, providing a reliable signal for real-time reinforcement learning. This design encourages the robot to adapt its speed to the amount of new information obtained during movement.

Although sparse rewards are a common challenge in actor–critic reinforcement learning, the proposed patrol-speed control task does not suffer from sparsity. The reward is computed at every time step based on the deviation between the observed entropy and the target entropy profile, resulting in a dense and continuous feedback signal. This design ensures that the agent receives informative gradients throughout the trajectory, avoiding long-horizon credit assignment issues typically associated with sparse-reward problems.

To solve this continuous control problem, we adopt the DDPG algorithm, which is well-suited for learning deterministic policies in high-dimensional action spaces. It uses an actor network to generate the patrol speed based on the current observation, and a critic network to estimate the expected return of the actor's decision. This architecture enables learning of robust and adaptive patrol behaviors tailored to dynamic and visually complex facility environments.

Optimization-based interpretation

The patrol-speed control task can be expressed as a reward maximization problem in which the deterministic policy $\mu_{\theta^\mu}(s)$ is trained to maximize the discounted return

$$\max_{\theta^\mu} J(\theta^\mu) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{16}$$

In the stochastic-policy setting, this objective is optimized through the policy-gradient theorem

Table 2 Hyperparameters of the DDPG algorithm and tested stability ranges.

Parameter	Final value	Range tested	Rationale
actor_LR	0.0001	1×10^{-4}–1×10^{-3}	Stable convergence
critic_LR	0.0005	5×10^{-4}–1×10^{-3}	Avoid Q oscillation
mem_max_size	30,000	3×10^{4}–1×10^{5}	No benefit beyond
batch_size	128	64–256	Balanced stability
dc_factor (γ)	0.9	0.8–0.99	Standard RL range
τ	0.001	0.001–0.01	Prevent divergence

$$\nabla_\theta J = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)\, Q^\pi(s,a)], \tag{17}$$

but DDPG replaces the stochastic policy with a deterministic one, yielding the deterministic policy gradient

$$\nabla_{\theta^\mu} J = \mathbb{E}\left[\nabla_a Q_{\theta^Q}(s,a) \nabla_{\theta^\mu} \mu_{\theta^\mu}(s)\right], \tag{18}$$

which corresponds to the update rule in Eq. (21). Here, the critic $Q_{\theta^Q}(s,a)$ provides a differentiable approximation to the action-value function, enabling gradient-based policy improvement. The hyperparameters in Table 2 (*e.g.*, learning rates, batch size, replay buffer size, and soft update factor τ) directly influence the stability of this optimization process by controlling the smoothness of target updates and the variance of gradient estimates.

This explanation clarifies that the proposed controller solves the patrol-speed problem as a continuous optimization task, in contrast to fixed-speed heuristics that lack the ability to adapt to visually heterogeneous facility environments.

The proposed framework also improves data efficiency through its neural-network–based design. First, because DDPG is an off-policy actor–critic algorithm, past transitions stored in the replay buffer are reused multiple times during training. Let $\mathscr{D} = (\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ denote the replay buffer; each element contributes to multiple gradient updates of both the actor and critic, effectively increasing the useful sample count without requiring additional environment interaction. This contrasts with on-policy methods, where each sample can be used only once.

Second, the CNN encoder provides a compact visual representation $\phi(\mathbf{s}_t)$ that captures salient spatial structures. By learning from these shared representations, the actor and critic optimize their parameters with fewer input samples, improving the overall sample efficiency of policy learning.

Third, the CycleGAN-based domain translation contributes to data efficiency by reducing the amount of real-world interaction needed for deployment. Instead of collecting extensive real-world experience to fine-tune the policy, the generator G maps real images into the simulation domain on which the policy was originally trained. Because the observation distribution is aligned in this way, the same policy μ_{θ^μ} can operate in real environments without additional rollouts or fine-tuning, effectively reusing the simulation-trained model across domains.

These properties provide an algorithmic basis for the improved data efficiency observed in our framework.

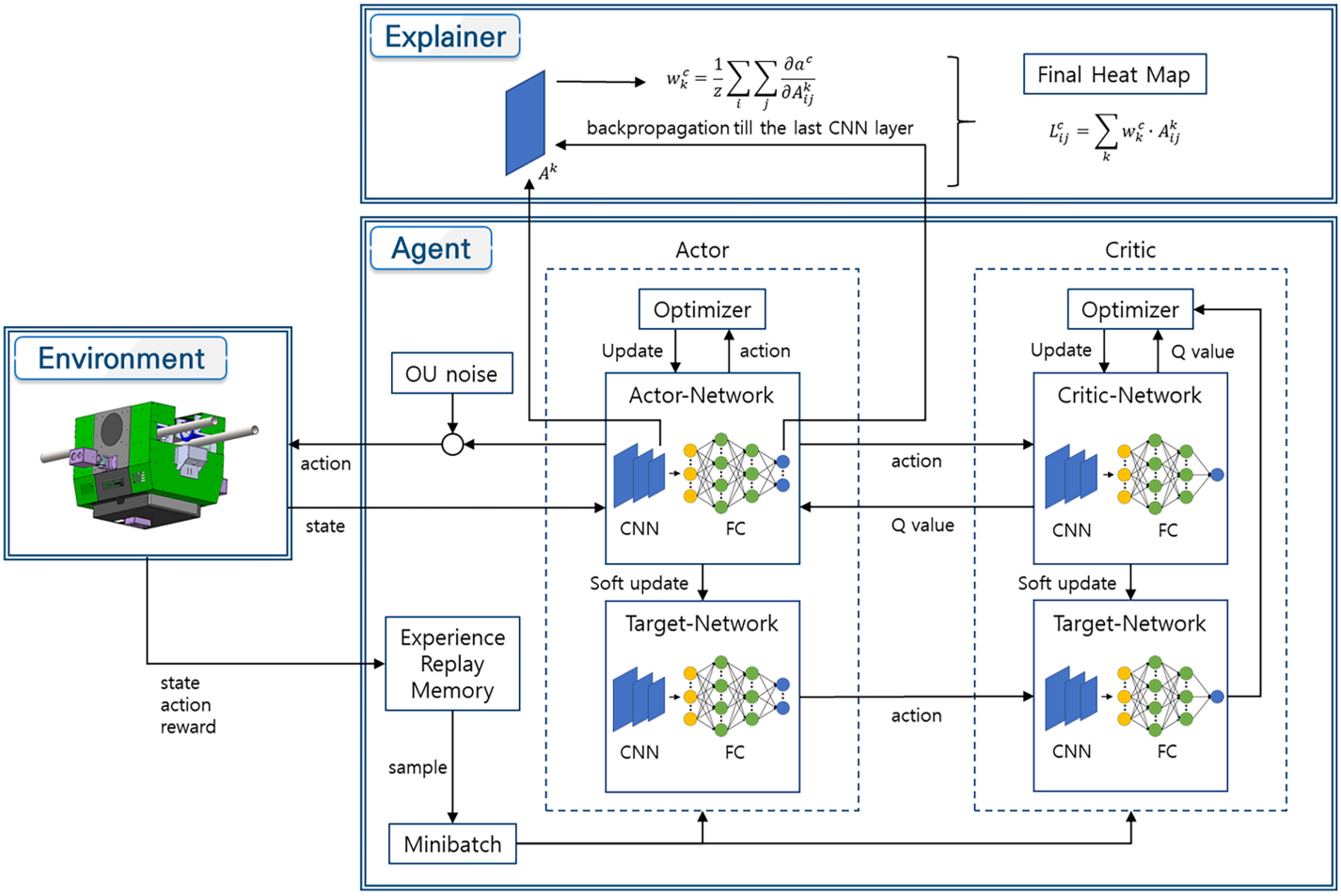


Figure 8 **Patrol speed control model using DDPG algorithm and Grad-CAM algorithm.** Full-size DOI: 10.7717/peerj-cs.3722/fig-8

Model architecture

Regarding the problem of determining the robot's patrol speed according to the site's condition, which is the objective of this study, the DDPG algorithm can be used to design a control model that outputs continuous robot control values with image data of the site as input (Fig. 8).

An experience replay method is used to prevent gradients from being biased due to the temporal correlation of training data. The experience is not directly used for training, but stored in the replay buffer, and N samples are randomly extracted from the buffer.

Next, the target actor neural network and the target critic neural network are created. When the Critic Neural Network is updated, the loss function of Eq. (19) is used, and at this time, the time difference target (TD-Target) of Eq. (20) is affected and changed.

$$L = \frac{1}{N}\sum_t \{Q(\mathbf{s}_t, a_t; \theta)\}^2. \tag{19}$$

$$y = r + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a'; \theta^-). \tag{20}$$

In practice, actor–critic methods are known to suffer from value overestimation or underestimation in the critic network, which can cause instability in the

temporal-difference updates. Although our controller does not incorporate the full twin-critic architecture used in TD3, we employ several stabilization strategies that effectively reduce estimation drift. These include using a small soft-update factor τ for the target networks, moderate learning rates for both the actor and critic, and bounded action outputs, all of which help prevent abrupt oscillations in the Q-value estimates during training. Our empirical examination of hyperparameter configurations further confirms that inappropriate settings lead to oscillatory critic behavior, while the final chosen values maintain smooth and stable TD updates.

When the Actor Neural Network is updated, the gradient value of the policy parameter is used. This gradient function is shown in Eq. (21).

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_t \left[\nabla_a Q(\mathbf{s}, a; \theta^Q)|_{\mathbf{s}=\mathbf{s}_t, a=\mu(\mathbf{s}_t|\theta^\mu)} \nabla_{\theta^\mu} \mu(\mathbf{s}; \theta^\mu)|_{\mathbf{s}=\mathbf{s}_t} \right] \tag{21}$$

θ^Q notes the parameters of the Critic Network and θ^μ notes the parameters of the Actor Network. In this case, the target keeps changing, making stable learning impossible. Therefore, the target actor neural network and the target critic neural network are maintained separately so that they slowly track the parameters of the main networks (soft target update).

The parameters $\theta^{\mu-}$ and θ^{Q-} of the Target Actor Network and the Target Critic Network are updated by the update rate τ as follows:

$$\theta^{Q-} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q-}. \tag{22}$$

$$\theta^{\mu-} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu-}. \tag{23}$$

Finally, noise is applied to the action. The original behavior is random, so the given environment must be properly explored, but DDPG uses a deterministic policy, so the policy can become uniform. To address this, Ornstein-Uhlenbeck (OU) noise is applied during training to encourage exploration.

While the proposed controller is based on the DDPG framework, it shares conceptual similarities with TD3 in that both employ a deterministic policy for continuous control, which is advantageous for real-time deployment on a rail-guided robot with strict latency constraints. In contrast, SAC introduces a stochastic policy and entropy maximization, offering strong exploration but also higher computational cost and increased inference latency, which are undesirable for our on-board processor. Our design therefore adopts a deterministic actor–critic architecture with a single critic network to balance control performance, computational efficiency, and real-time feasibility. A full integration of twin-critic updates (as in TD3) or entropy-regularized stochastic policies (as in SAC) is left as an important direction for future extensions.

Integration of Grad-CAM algorithm with the DDPG model

To enhance the explainability of the trained deep reinforcement learning model, we integrate a visual explanation module based on Grad-CAM (Gradient-weighted Class Activation Mapping), a widely adopted *post hoc* technique in the field of Explainable Artificial Intelligence (XAI). This addition enables system administrators and end-users to

understand the reasons behind the robot's decisions, particularly in determining patrol speed, without requiring access to or full comprehension of the model's internal architecture. Such explainability is essential in safety-critical applications, where transparency and human trust are of paramount importance.

We adopt Grad-CAM for policy explanation because its gradient-based activation maps naturally extend to continuous control outputs, including our regression-style patrol-speed action. Unlike perturbation-based XAI methods such as Local Interpretable Model-agnostic Explanations (LIME) or Shapley Additive exPlanations (SHAP), which require numerous input perturbations and incur substantial computational cost (*Ribeiro, Singh & Guestrin, 2016*; *Lundberg & Lee, 2017*), Grad-CAM produces a spatial relevance map with a single backward pass, making it suitable for real-time deployment on embedded hardware. Furthermore, the spatial consistency of Grad-CAM aligns with our objective of identifying visually complex regions that influence the adaptive-speed policy, providing an explainable link between scene structure and the resulting control behavior.

The proposed DDPG-based controller employs a CNN to extract spatial features from image inputs. However, due to the deep nature of the network and the abstraction process in fully connected (FC) layers, the model lacks explainability—meaning its internal parameters and representations are not directly understandable to humans. As a result, although the model may exhibit effective behavior, it remains a black box in terms of decision rationale. To address this, Grad-CAM is applied to the actor network to provide visual explanations of the model's outputs. As illustrated in Fig. 8, Grad-CAM generates a class-discriminative localization map by utilizing the gradients of a selected output (corresponding to an action) with respect to the final convolutional feature maps. These gradients capture the influence of different spatial regions in the input image on the model's output. Let $\mathbf{A}^k \in \mathbb{R}^{u \times v}$ be the activation map of the k-th channel in the final convolutional layer, where u and v denote the spatial dimensions. The importance weight w_k^c for action c is calculated as:

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial a^c}{\partial A_{ij}^k}. \tag{24}$$

Here, Z is a normalization term equal to the number of pixels in the feature map, and $\frac{\partial a^c}{\partial A_{ij}^k}$ represents the gradient of the action score with respect to each spatial location in the feature map, obtained *via* backpropagation. Using these weights, the final explanation map L^c is computed by a weighted sum of the activation maps:

$$\mathbf{L}^c = \sum_k w_k^c \cdot \mathbf{A}^k. \tag{25}$$

The resulting heatmap is then upsampled to the original input image size and overlaid to highlight the regions most influential to the action decision. Importantly, this process does not modify the model or require retraining, thus serving as a model-agnostic *post hoc* explanation tool.

By incorporating Grad-CAM, our system supplements the non-explainable DDPG policy with actionable visual explanations, thereby improving operational transparency,

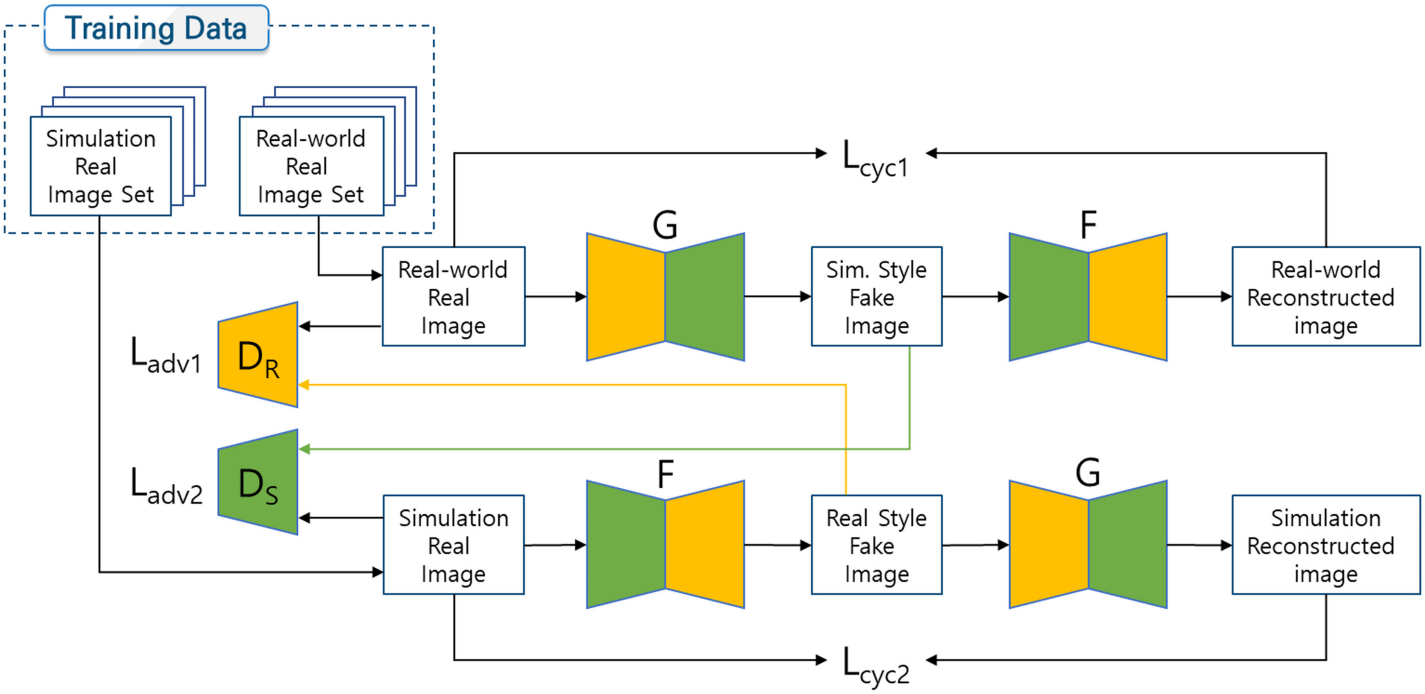


Figure 9 **CycleGAN architecture for visual domain adaptation.** Full-size DOI: 10.7717/peerj-cs.3722/fig-9

supporting trust in autonomous decision-making, and aiding in model debugging during deployment.

Sim-to-real deployment *via* CycleGAN

Despite training the patrol speed control policy entirely within a simulated environment, direct deployment of the learned model in real-world settings is hindered by the significant visual domain gap between simulation-rendered and real-world images. Variations in lighting, texture, background clutter, and sensor noise can severely degrade the performance of a vision-based reinforcement learning policy trained solely on synthetic data. To bridge this sim-to-real gap, we employ a CycleGAN-based visual domain adaptation strategy that enables the trained policy to generalize to real-world conditions without requiring any additional retraining.

CycleGAN (*James et al., 2019*) is a framework for unpaired image-to-image translation. It comprises two generators and two discriminators that are jointly trained using adversarial loss and cycle-consistency loss. Generator G learns to translate real-world images to the simulation style, while generator F performs the inverse transformation. During training, the cycle-consistency constraint ensures that an image translated from one domain to the other and back again retains its semantic structure. Figure 9 illustrates the training architecture of the CycleGAN model used in our framework.

Once training is complete, only the generator G is retained for deployment. As depicted in Fig. 10, real-time camera images captured by the rail-guided robot are passed through G to produce simulation-style representations. These translated images are then fed directly

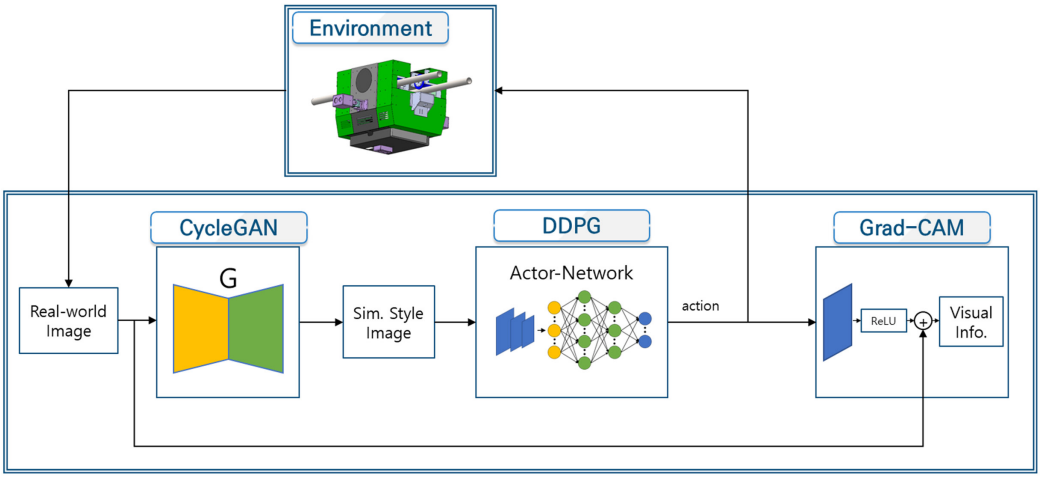


Figure 10 **Real-world deployment using generator *G*, DDPG actor and Grad-CAM.**
Full-size DOI: 10.7717/peerj-cs.3722/fig-10

into the pretrained actor network of the DDPG controller, which computes the appropriate patrol speed. This modular deployment architecture allows the simulation-trained policy to function effectively in the real world, with no additional tuning or retraining. To ensure that the translated images preserve task-relevant semantic content, we additionally apply Grad-CAM to the actor network during evaluation. The resulting attention maps confirm that the model continues to focus on meaningful visual regions even after domain translation, further validating the effectiveness of the CycleGAN-based adaptation.

In the following section, we present evaluation results obtained in both simulation and real-world scenarios to assess the effectiveness of the proposed patrol speed control framework. The evaluation focuses on three key aspects: the learning performance of the DDPG controller, the generalization ability of the trained policy under varying visual conditions, and the explainability of its decisions through Grad-CAM visualizations.

Runtime computational feasibility

The proposed runtime pipeline consists of three stages: (1) transforming the incoming real image into the simulation style using the CycleGAN generator *G*, (2) evaluating the lightweight DDPG actor network to produce the patrol speed, and (3) issuing the motor command. Because all processing operates on $64 \times 64 \times 3$ RGB images, the computational load of both the generator and the actor network is small. A forward pass through a compact CycleGAN generator and the actor network fits comfortably within the 100 ms control-period budget required for 10 Hz operation, even under conservative GPU inference assumptions. Grad-CAM visualization is executed offline and therefore does not add any latency to the control loop. These considerations confirm that the integrated system is compatible with real-time deployment on the rail-guided robot.

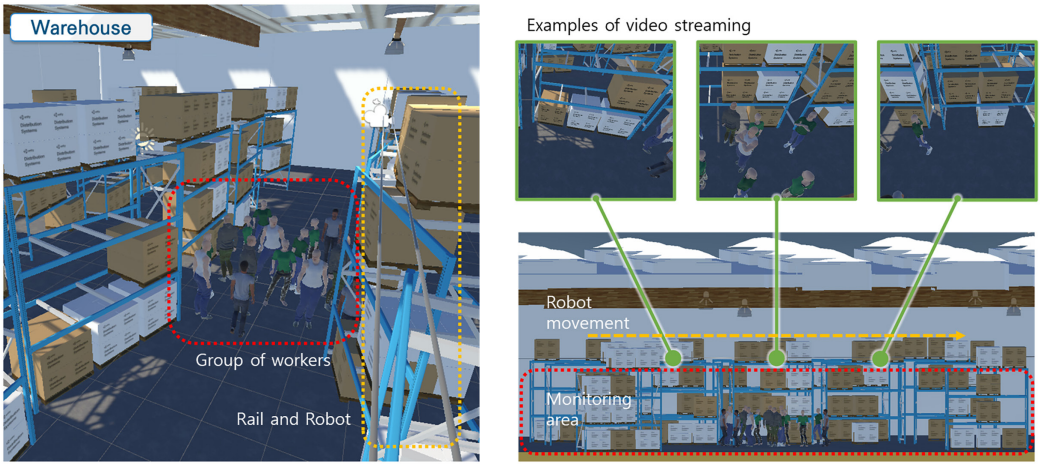


Figure 11 **Simulation environment.** Full-size DOI: 10.7717/peerj-cs.3722/fig-11

EVALUATION

To assess the effectiveness of the proposed patrol speed control framework, we conducted a series of experiments in both simulation and real-world environments. In addition to evaluating learning-based behaviors, we first assessed the structural validity of the ceiling-mounted rail by measuring its roll and pitch angles during robot operation, ensuring that the physical test environment does not introduce unintended bias into the controller's performance. The evaluation therefore focuses on five main aspects: (1) the structural stability of the rail environment based on roll and pitch measurements, (2) the learning performance of the DDPG controller, (3) the adaptability of the trained policy to varying environmental complexity, (4) the explanation of its decisions using Grad-CAM visualizations, and (5) the effectiveness of CycleGAN-based domain adaptation in enabling zero-shot sim-to-real policy transfer. We also validate the effectiveness of CycleGAN-based domain adaptation in transferring the learned policy to real-world scenarios without retraining, using visual explanations as an offline diagnostic tool.

Simulation-based experiments

Simulation environment setup

To train and evaluate the proposed DDPG-based patrol speed controller, we constructed a virtual warehouse environment using Unity, as illustrated in Fig. 11. The simulation mimics the layout of a real facility, including aisles and storefronts, by integrating Unity asset packages such as ML-Agents, Unity Warehouse, and UMA (Unity Multipurpose Avatar) (*Juliani et al., 2020*; *Unity-Technologies, 2022a*, *2022b*).

The patrol robot moves along a straight ceiling-mounted rail located 4 m above the floor, following a predefined linear path. A side-mounted RGB camera, tilted 45 degrees downward, captures visual input of the aisle region beneath the robot. This field of view allows the robot to monitor both occupied and unoccupied areas. To emulate realistic surveillance scenarios, the environment includes two types of regions: (1) empty zones

with no human presence, and (2) populated zones with dense groups of workers or visitors. This spatial variation in visual complexity enables the system to learn to adjust its patrol speed based on perceived environmental entropy.

The simulation environment is controlled using Unity ML-Agents, which enables interaction between the virtual agent and an external deep reinforcement learning pipeline. The ML-Agents framework also allows synchronization of sensor input, action execution, and reward computation, making it suitable for learning high-level control policies. The state vector fed into the actor network consists of 4,097 features: a 4,096-dimensional feature vector extracted from the camera image using a CNN encoder, and one scalar representing the robot's current speed. The actor network is composed of two fully connected layers with 2,049 neurons each, activated by ReLU, followed by a Tanh-activated output layer to constrain the speed within a valid range. The critic network processes the state input through a fully connected layer, then concatenates it with the action input and passes the combined vector through two more fully connected layers. The final layer outputs the estimated Q-value without an activation function.

The values in Table 2 were not only selected based on prior DDPG literature but also verified through a small-scale sensitivity analysis. For each parameter, we examined a range of candidate values (*e.g.*, actor/critic learning rates from 10^{-4} to 10^{-3} and soft update factors τ from 0.001 to 0.01) and confirmed that the chosen configuration consistently produced stable Q-value updates and smooth policy convergence. Parameters outside these ranges often resulted in oscillatory critic behavior or significantly slower learning. Thus, the final values in the table lie well within a stable region for the proposed patrol-speed control task.

All experiments were conducted on a desktop machine with an Intel i7-8700K CPU @ 3.70 GHz, 32 GB RAM, and an NVIDIA RTX 3080Ti GPU.

Learning performance of DDPG controller

To evaluate the training dynamics of the proposed DDPG-based patrol speed controller, we monitored key metrics including actor loss, critic loss, Q-value, and episodic score throughout the learning process. The replay buffer was populated for the initial 10 episodes, after which learning began and continued up to episode 110.

As shown in Fig. 12A, the actor loss started from 0 and quickly decreased, stabilizing around -10 after approximately 50 episodes. This indicates that the actor network successfully converged to a policy that consistently outputs effective patrol speeds. Figure 12B shows the critic loss, which increased rapidly from 0 and converged to approximately 0.25 after 40 episodes. This reflects the stabilization of value function estimation by the critic network. As shown in Fig. 12D, the predicted Q-value increased rapidly from 0 and reached a plateau around 9 by episode 40, demonstrating improved expected return estimation as the policy matured.

Finally, the episodic score, shown in Fig. 12C, gradually improved over the course of training. It started around 1,100 and slowly increased with oscillations, eventually reaching approximately 1,600. This upward trend indicates the increasing effectiveness of the policy in acquiring high-entropy visual information while maintaining efficient patrol movement.

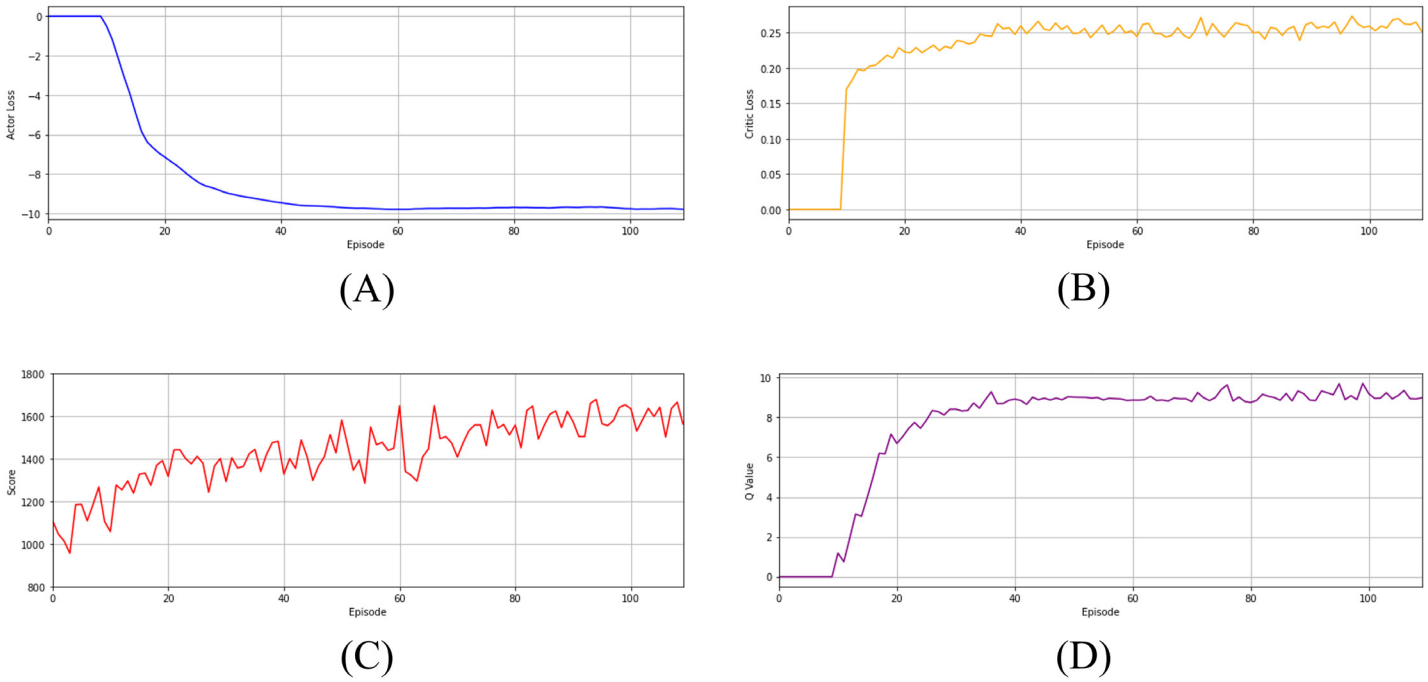


Figure 12 Learning performance of DDPG. (A) Actor loss. (B) Critic loss. (C) Score. (D) Q-value.
Full-size DOI: 10.7717/peerj-cs.3722/fig-12

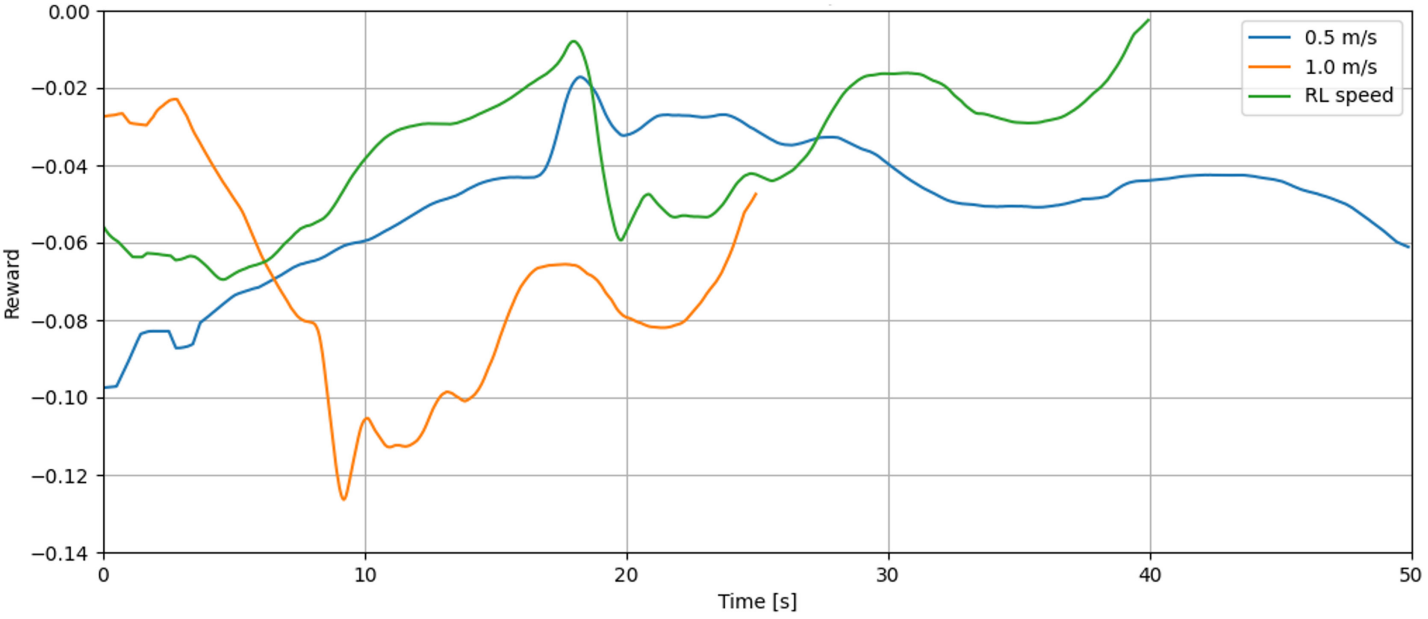
Overall, these results confirm that the DDPG framework successfully learned a stable and performant control policy within 110 episodes.

Baseline comparison with fixed-speed driving

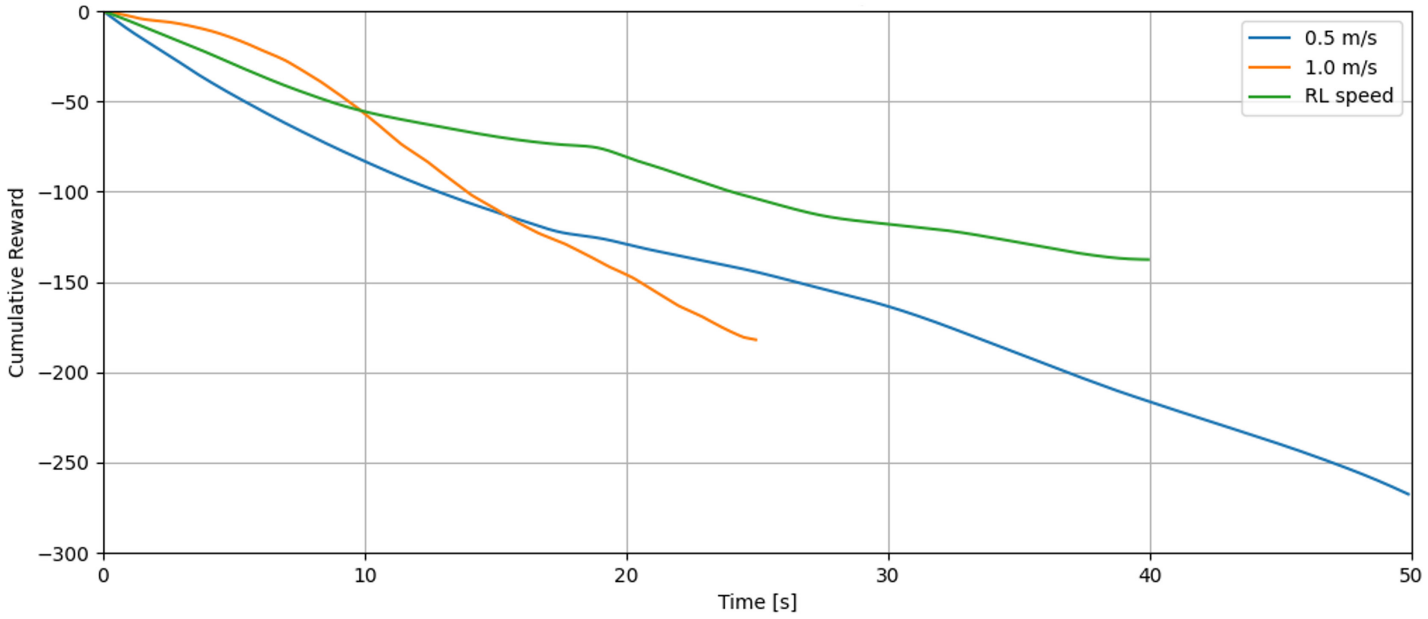
To quantitatively validate the benefit of adaptive speed control, we compared the proposed DDPG policy with fixed-speed baselines using the same entropy-based reward function. Three policies were evaluated: constant 0.5 m/s, constant 1.0 m/s, and the speed profile produced by the trained DDPG controller. Since the reward depends only on the captured image and the robot's instantaneous speed, this setup provides a fair and direct comparison of information-acquisition performance across different policies.

Figure 13A shows the per-step reward over time for the three speed profiles. The fixed-speed trajectories exhibit larger deviations from the desired entropy target. The 0.5 m/s case yields consistently low reward due to insufficient newly observed area per frame, while the 1.0 m/s case frequently overshoots the target because of excessive frame-to-frame motion. In contrast, the DDPG-controlled trajectory maintains reward values closer to zero for most of the path, indicating that the learned policy more effectively matches the desired information-acquisition rate by adjusting speed based on visual complexity.

Figure 13B presents the cumulative reward comparison. Both fixed-speed baselines accumulate substantial negative reward as the patrol progresses. The 0.5 m/s baseline accumulates the largest penalty from consistently low information gain, while the 1.0 m/s baseline exhibits substantial cumulative penalty from excessive movement. On the other



(A)



(B)

Figure 13 Comparison of fixed-speed (0.5 and 1.0 m/s) and DDPG-based adaptive-speed policies. (A) Per-step reward curves. (B) Cumulative reward curves.
Full-size DOI: 10.7717/peerj-cs.3722/fig-13

hand, the trained DDPG policy shows a much slower decline in cumulative reward, demonstrating superior long-term performance in minimizing the mismatch between desired and actual information acquisition.

These results confirm that the proposed adaptive-speed controller outperforms fixed-speed driving in both instantaneous and cumulative reward. The learned policy effectively balances motion and information gain, validating the design of the entropy-based reward function and demonstrating the advantage of learning-based adaptive patrol speed control.

Simulation-based evaluation with visual explanation

To evaluate the learned patrol speed control policy in a controlled setting, we first conducted simulation-based tests using the same Unity environment in which the model was trained. This allows us to analyze the behavior of the trained DDPG controller in a noise-free and consistent context, focusing on how well it adapts its patrol speed to varying visual complexity. We also examine whether the model's decision-making process can be explained visually using Grad-CAM, thereby validating the explainability of the learned policy before deployment.

Figure 14 shows the simulation result of the trained controller. The trained controller takes the image captured by the robot from the environment as input. The patrol speed of the robot was controlled according to the output value from the trained controller. The comparison result of the speed at seven locations in the monitoring area with sample images shows the robot patrols at the highest speed in the empty space area and moved at the lowest speed in the area with crowded workers and visitors. By calculating the entropy of each sample image, it was confirmed that the amount of information in the environment is represented. The images in the bottom row are Grad-CAM result images. It expresses the degree of influence in determining the patrol speed of the robot on the input image. It shows the influence is significant in the 2nd and 3rd sections, where the speed continues to change.

The trained policy exhibits consistent behavior by increasing patrol speed in low-entropy areas and reducing it in visually complex or crowded regions. Such behavior aligns with the intended adaptive strategy, where slower movement enables detailed monitoring in high-risk zones. These findings validate the adaptive behavior of the policy in simulation and establish a solid foundation for its deployment in real-world environments, which we evaluate next.

Real-world experiments

Rail inclination analysis

Before conducting adaptive speed control experiments, we evaluated the inclination of the ceiling-mounted rail to ensure that structural misalignment would not affect the robot's movement or visual perception. Figure 15 shows the roll and pitch angles measured while the robot traveled along the rail at a constant speed of 0.4 m/s. The roll angle (ANGLE_X) remained within +0.70° to −2.20°, indicating that the left–right deviation of the rail stayed within the acceptable tolerance of ±2.5°. The pitch angle (ANGLE_Y) varied between +1.15° and −2.56°, demonstrating that the forward–backward inclination also remained within the normal operating range. These measurements verify that the rail environment is structurally stable and suitable for reliable evaluation, ensuring that the subsequent

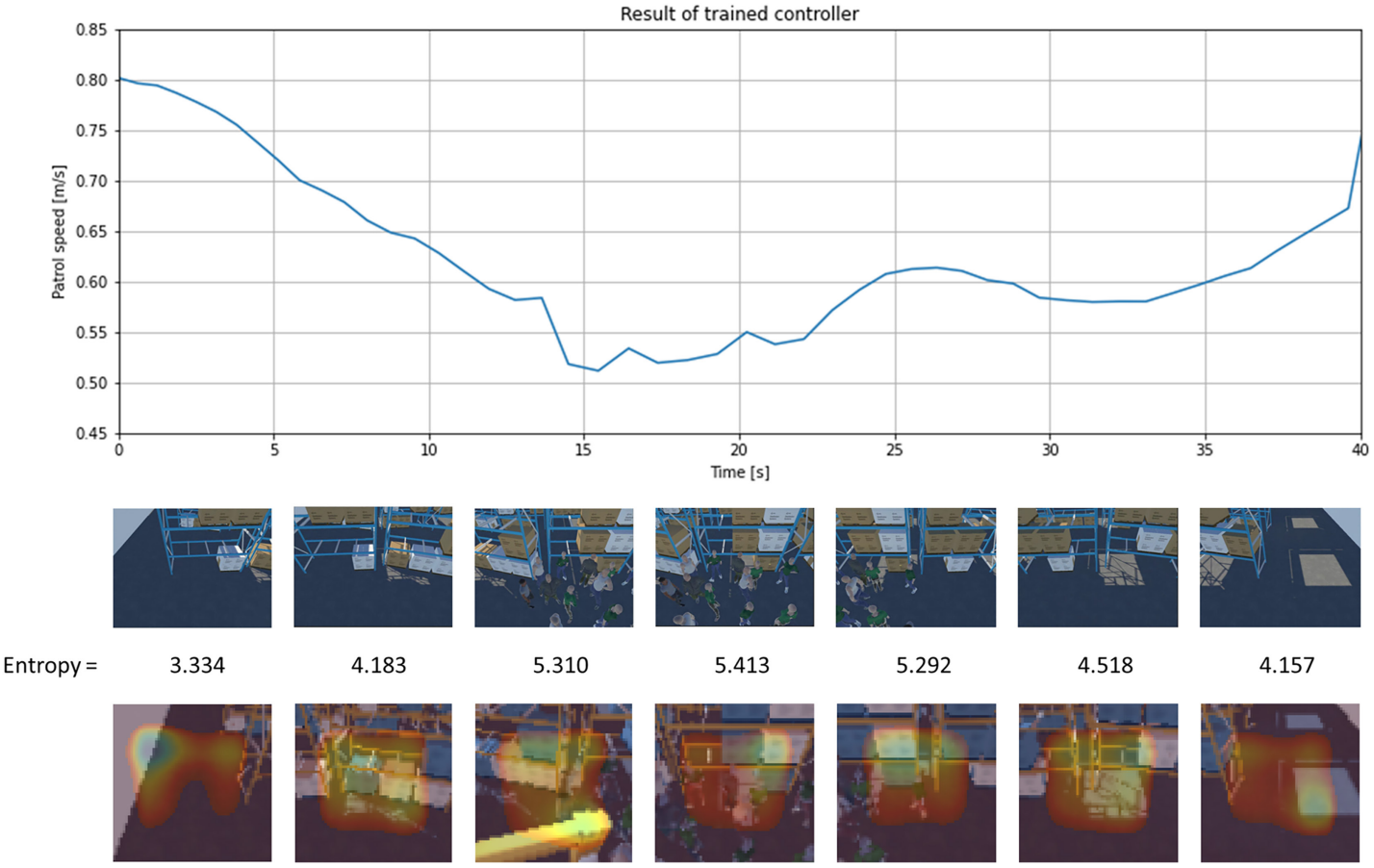


Figure 14 Simulation results of the trained patrol speed controller. (Top) input images and corresponding speeds at seven locations. (Bottom) Grad-CAM visualizations.
Full-size DOI: 10.7717/peerj-cs.3722/fig-14

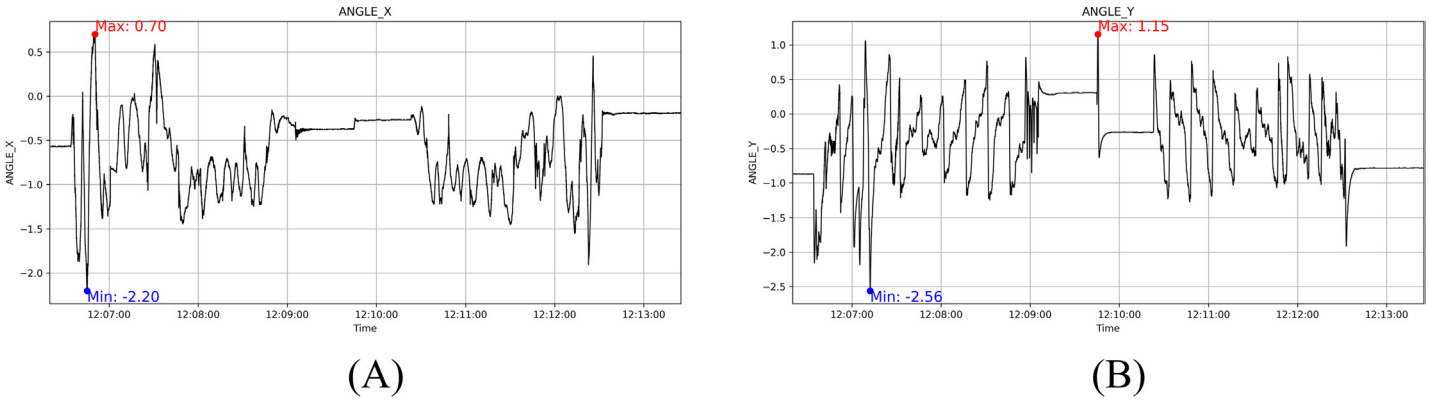


Figure 15 Rail inclination measurement results during constant-speed driving. (A) Roll angle (ANGLE_X). (B) Pitch angle (ANGLE_Y).
Full-size DOI: 10.7717/peerj-cs.3722/fig-15

analysis of the DDPG controller is not confounded by unintended geometric bias or irregularities in the rail itself.

Real-world test environment setup

To evaluate the proposed patrol speed control framework under real-world conditions, we established a physical test environment within our laboratory. A ceiling-mounted rail structure, installed 3 m above the floor and extending 8 m in length, was used to guide the robot along a fixed linear path. The robot was designed to patrol this path autonomously while capturing images of the monitored area located below the rail. The patrol scenario involved a dynamic workspace where workers were moving around during the robot's operation. The monitored area included various industrial elements such as storage tables, tools, and equipment, thereby simulating a realistic surveillance setting with visual clutter and human activity.

During each patrol run, the robot's onboard camera continuously recorded images of the environment. Each run lasted 9 s, and a total of three runs were conducted to collect 810 real-world images. These images were used not only for performance evaluation but also as training data for the CycleGAN model used in domain adaptation. For this purpose, the real-world images were paired with 1,000 simulated images captured in the Unity environment, which represent similar patrol scenarios rendered under controlled conditions. This environment provided a practical basis for validating the trained DDPG policy's transferability from simulation to a real-world system, with visual domain adaptation enabled by CycleGAN.

Real-world deployment results with domain adaptation

With the real-world test environment and image dataset in place, we deployed the trained DDPG controller in combination with the CycleGAN-based domain adaptation module to evaluate the system's performance in a real deployment scenario. The goal was to verify whether the simulation-trained policy, when provided with translated inputs, could maintain effective and explainable patrol behavior in the physical environment.

Figure 16 presents the real-world experiment results, including patrol speed variation across time, sample images from seven locations, entropy values of the input images, and the corresponding Grad-CAM visualizations. These results are structured similarly to the simulation-based evaluation for consistent comparison.

In the first and second sections of the patrol path, the robot maintained a low patrol speed. The corresponding Grad-CAM maps show strong activation around workers and visually cluttered regions in the background, indicating that the model correctly identified areas of high surveillance priority. In sections three and four, the patrol speed increased. Interestingly, Grad-CAM highlighted broader regions with intensified activation, suggesting that the model recognized a moderate level of complexity but deemed it safe to patrol faster. Section five exhibited the highest patrol speed, with Grad-CAM focusing on a narrow region, likely due to minimal visual information and the absence of dynamic elements. In the latter part of the path, the patrol speed varied again—slowing down and speeding up in response to environmental changes. Grad-CAM attention maps in these

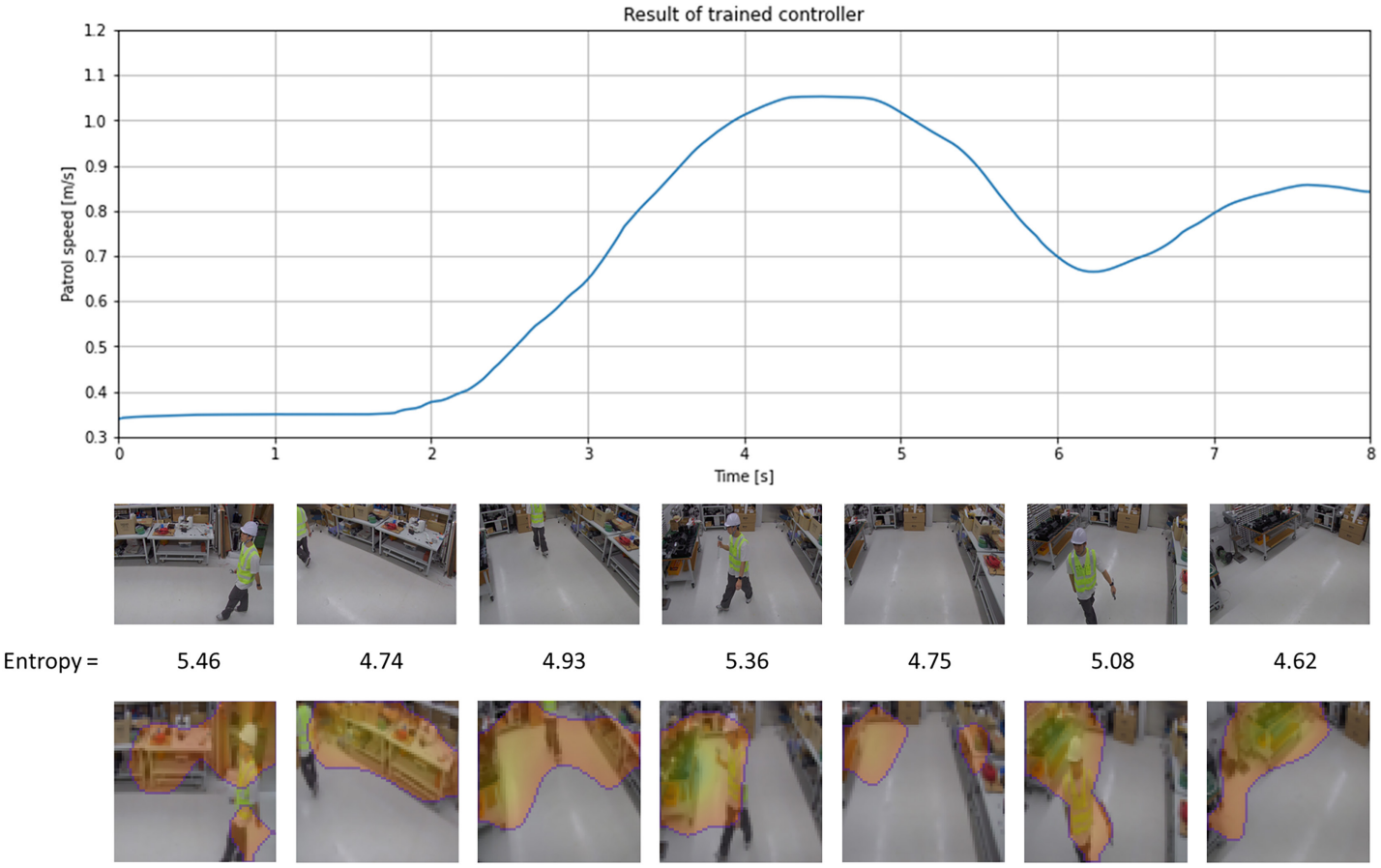


Figure 16 Experimental results of the trained patrol speed controller. (Top) input images and corresponding speeds at seven locations. (Bottom) Grad-CAM visualizations. Full-size DOI: 10.7717/peerj-cs.3722/fig-16

segments revealed more widely activated areas, reaffirming the model's ability to dynamically adapt its attention based on the complexity of the scene.

To investigate the effect of CycleGAN-based domain adaptation on the explainability of the trained policy, we compared Grad-CAM visualizations using real-world and translated simulation-style images as inputs. As shown in Fig. 17, the Grad-CAM result on the real-world image (C) is diffuse and poorly aligned with key semantic elements, whereas the result on the Simulation-style image (D) highlights the human and relevant working area more clearly. This demonstrates that domain-adapted inputs preserve semantically meaningful features for policy explanation and even enhance the alignment between input content and the model's attention.

Overall, the real-world results demonstrate that the proposed framework, enhanced with CycleGAN-based domain adaptation, enables the simulation-trained DDPG controller to perform robustly in physical environments. The Grad-CAM visualizations further validate the explainability of the decision-making process, showing consistent alignment between attention focus and environmental cues. These results confirm that the

(A) (B) (C) (D)

Figure 17 Comparison of Grad-CAM visualizations using real and CycleGAN-translated images as inputs to the trained DDPG controller. (A) Real-world input image. (B) CycleGAN-translated simulation-style image. (C) Grad-CAM result with real image input. (D) Grad-CAM result with translated image input. Full-size DOI: 10.7717/peerj-cs.3722/fig-17

simulation-trained policy, combined with CycleGAN-based domain adaptation, can be effectively and explainably deployed in real-world patrol scenarios.

Quantitative evaluation of CycleGAN-based sim-to-real translation

To quantify the effectiveness of CycleGAN in reducing the visual domain gap, we evaluated two metrics relevant to downstream reinforcement learning: Kernel Inception Distance (KID) and Structural Similarity Index Measure (SSIM). These metrics capture feature-level and structural alignment rather than photorealism alone.

KID. KID measures distributional similarity between feature embeddings. The KID score decreased from 0.357 (simulation *vs.* real) to 0.261 (CycleGAN *vs.* real), indicating improved feature-level consistency and reduced domain discrepancy.

SSIM. SSIM evaluates luminance, contrast, and structural similarity. The SSIM increased from 0.276 to 0.423 after CycleGAN translation, showing enhanced preservation of structural cues such as edges and spatial layout.

Implications. These improvements align with the real-world control results: the DDPG policy shows more stable adaptive-speed behavior, and Grad-CAM visualizations highlight more semantically meaningful regions. Thus, CycleGAN provides task-relevant domain alignment that supports both policy generalization and explainability.

Although a full ablation study is not included, the individual contributions of the main components can be interpreted from the presented analyses. First, the effectiveness of the DDPG controller is isolated through the fixed-speed baseline comparison, which shows clear performance gaps in both instantaneous and cumulative reward. Second, the impact of CycleGAN is quantified using KID and SSIM metrics, demonstrating that domain shift is significantly reduced even before policy deployment. This analysis serves as an implicit ablation, as it measures how much the simulation–real discrepancy is reduced by CycleGAN without requiring a separate full training run. Finally, Grad-CAM does not affect control performance directly but provides explainability, and therefore is not part of the control optimization pipeline that would benefit from ablation.

It is also worth noting that the CycleGAN module contributes to data efficiency by enabling zero-shot policy transfer: the DDPG controller trained entirely in simulation operates effectively on real images after translation, without additional real-world data collection. As shown in Fig. 17, using real images without CycleGAN yields weaker semantic alignment in Grad-CAM maps, whereas translated images maintain task-relevant structure. This comparison implicitly demonstrates that the NN-based components improve sample efficiency, as no further data gathering or on-site retraining was required to achieve robust real-world behavior.

Evaluation summary

The experimental results from both simulation and real-world environments confirm the effectiveness of the proposed patrol speed control framework. The DDPG controller successfully adapted the robot's speed based on visual complexity, while Grad-CAM visualizations provided meaningful insight into the decision-making process. Furthermore, the use of CycleGAN for domain adaptation enabled reliable policy deployment without retraining, preserving both performance and explainability. These findings demonstrate the practical viability of our approach in dynamic monitoring scenarios, setting the foundation for broader deployment in real-world robotic applications.

CONCLUSION

This article presents a rail-guided smart patrol system designed for autonomous facility monitoring. The system aims to reduce human workload and improve safety by enabling the robot to patrol without human intervention. The robot moves along a fixed rail structure to ensure stable navigation in structured environments. It is equipped with a camera module and an air quality sensor to continuously monitor visual and environmental conditions. A real-time communication module transmits sensor data and robot commands to a centralized monitoring platform.

To enable context-aware decision making, we apply the Deep Deterministic Policy Gradient (DDPG) algorithm for adaptive control of the robot's patrol speed. The continuous control capability of DDPG allows the robot to fine-tune its movement speed in response to varying environmental complexity. The reward function is defined using the entropy of observed data to encourage information-rich and diversified behavior. A virtual patrol environment is built using Unity, and ML-Agents is used to connect the environment with the external reinforcement learning framework. Simulation results show that the robot successfully learns to slow down in visually complex or hazardous areas and accelerate in low-risk zones.

In addition to adaptive behavior, explainability is addressed by integrating a Grad-CAM-based explainable AI (XAI) module into the control system. This module generates heatmaps that highlight the visual input regions most influential in the robot's decisions. The use of XAI supports user trust, facilitates model debugging, and improves system transparency. The combined DDPG and Grad-CAM framework enables both effective control and explainable decision-making. We validate the proposed system

through simulation and on-site deployment in an agricultural market setting. The field test demonstrates the practical feasibility and robustness of the system in real-world conditions.

Although the proposed pipeline performs reliably within the tested scenarios, vision-based control remains inherently sensitive to variations in input quality. Abrupt illumination changes, motion blur from rapid movement, camera noise, and unexpected occlusions may degrade the reliability of both the CycleGAN translation and the policy inference. Likewise, if the real-world environment significantly deviates from the simulated training domains, the generator may produce suboptimal translations, potentially affecting downstream control accuracy. Addressing these limitations would require incorporating more diverse visual data, domain-randomization strategies, or robustness-enhancing perception modules, which we identify as future directions. Future work also includes incorporating explicit robust-control structures, such as data-driven sliding-mode formulations, to provide formal guarantees on stability and disturbance rejection beyond the bounded rail disturbances observed in this study. Furthermore, extending the controller with TD3-style twin-critic mechanisms offers a promising direction for enhancing critic stability and reducing value-estimation bias, which we identify as an important opportunity for future refinement.

The framework's modular design allows for future extension to include anomaly detection, obstacle avoidance, and additional robotic behaviors. Integration with more diverse sensor modalities is also possible to enhance situational awareness. This research highlights the synergy between deep reinforcement learning and explainable AI for intelligent robot control. The proposed system offers a scalable and trustworthy solution for long-term autonomous facility management. Future work will focus on quantitative evaluation of XAI performance, multi-agent coordination, and real-time adaptive learning on edge devices.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the National Research Foundation of Korea (NRF) under grant NRF-2022M3C1A3099340, and The present Research has been conducted by the Research Grant of Kwangwoon University in 2024. There was no additional external funding received for this study. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
National Research Foundation of Korea (NRF): NRF-2022M3C1A3099340.
Kwangwoon University.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- Hosun Lee conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Jaesung Kwon performed the experiments, prepared figures and/or tables, and approved the final draft.
- Nak Young Chong conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.
- Woosung Yang conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

All raw data used to produce the figures, tables, and quantitative analyses are available at Zenodo: Lee, H., Kwon, J., Chong, N. Y., & Yang, W. (2025). Dataset for "Explainable Deep Reinforcement Learning for Adaptive Patrol Speed Control of a Rail-Guided Robot System" [Data set]. Zenodo. https://doi.org/10.5281/zenodo.17766302.

The dataset does not include the simulation environment or code for retraining the model. As this is a deep reinforcement learning study, no static dataset is used. Instead, we have provided the full implementation code and environment setup to support reproducibility.

The simulation environment is based on publicly available examples from Unity ML-Agents and Unity Robotics Hub at GitHub:

- https://github.com/Unity-Technologies/ml-agents.
- https://github.com/Unity-Technologies/Unity-Robotics-Hub.

The implementation of the DDPG algorithm and the Grad-CAM visualization method, along with instructions on how to run the trained models within the main loop; and the output results from the trained controller are available in the Supplemental File.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.3722#supplemental-information.

REFERENCES

Bengel M, Pfeiffer K, Graf B, Bubeck A, Verl A. 2009. Mobile robots for offshore inspection and manipulation. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Piscataway: IEEE, 3317–3322.

Carvalho GP, Xaud MF, Marcovistz I, Neves AF, Costa RR. 2017. The DORIS offshore robot: recent developments and real-world demonstration results. *IFAC-PapersOnLine* **50(1)**:11215–11220 DOI 10.1016/j.ifacol.2017.08.2101.

Christensen L, Fischer N, Kroffke S, Lemburg J, Ahlers R. 2011. Cost-effective autonomous robots for ballast water tank inspection. *Journal of Ship Production and Design* **27(3)**:127–136.

Galassi M, Røyrøy A, Carvalho G, Freitas G, From PJ, Costa RR, Lizarralde F, Hsu L, de Carvalho G, de Oliveira JF, de Lima AA, de M, Prego TL, Netto S, AB da Silva E. 2014.

DORIS—a mobile robot for inspection and monitoring of offshore facilities. In: *Proceedings of the Congresso Brasileiro de Automática, Belo Horizonte, Brazil*, 20–24.

Halder S, Afsari K. 2023. Robots in inspection and monitoring of buildings and infrastructure: a systematic review. *Applied Sciences* **13(4)**:2304 DOI 10.3390/app13042304.

Hentout A, Maoudj A, Kouider A. 2024. Shortest path planning and efficient fuzzy logic control of mobile robots in indoor static and dynamic environments. *Science and Technology* **27(1)**:21–36 DOI 10.21203/rs.3.rs-3182222/v1.

James S, Wohlhart P, Kalakrishnan M, Kalashnikov D, Irpan A, Ibarz J, Levine S, Hadsell R, Bousmalis K. 2019. Sim-to-real via sim-to-sim: data-efficient robotic grasping via randomized-to-canonical adaptation networks. 1–14 DOI 10.48550/arXiv.1812.07252.

Juliani A, Berges V-P, Teng E, Cohen A, Harper J, Elion C, Goy C, Gao Y, Henry H, Mattar M, Lange D. 2020. Unity: a general platform for intelligent agents. ArXiv DOI 10.48550/arXiv.1809.02627.

Kim D-H, Lee S, Kang M-S, Chun B-I, Han C-S. 2012. Proposal of built-in-guide-rail type building façade cleaning robot and its motion planning algorithm. In: *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. Piscataway: IEEE, 1004–1009.

Lee H, Kwon J, Lee S, Chong NY, Yang W. 2024. Information-based patrol speed control method for rail-guided robot system using deep deterministic policy gradient algorithm. In: *International Conference on Intelligent Autonomous Systems*. Cham: Springer, 207–214.

Lee H, Kwon J, Shin M, Lee S, Chong NY, Yang W. 2023. Development of rail-guided smart patrol system for surveillance and monitoring of facilities safety. In: *2023 IEEE/SICE International Symposium on System Integration (SII)*. Piscataway: IEEE, 1–6.

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. 2015. Continuous control with deep reinforcement learning. ArXiv DOI 10.48550/arXiv.1509.02971.

Lundberg SM, Lee S-I. 2017. A unified approach to interpreting model predictions. 1–10 DOI 10.48550/arXiv.1705.07874.

Macaulay MO, Shafiee M. 2022. Machine learning techniques for robotic and autonomous inspection of mechanical systems and civil infrastructure. *Autonomous Intelligent Systems* **2(1)**:8 DOI 10.1007/s43684-022-00025-3.

Mateescu A, Popescu DC, Stefan IL, Vlasceanu IM, Petrescu-nita AC, Sacala I, Dumitrache I. 2025. Machine learning control for assistive humanoid robots using blackbox optimization of PID loops through digital twins. *Science and Technology* **28(1)**:63–76 DOI 10.59277/romjist.2025.1.06.

Rakha T, Gorodetsky A. 2018. Review of unmanned aerial system (UAS) applications in the built environment: towards automated building inspection procedures using drones. *Automation in Construction* **93(18)**:252–264 DOI 10.1016/j.autcon.2018.05.002.

Ribeiro MT, Singh S, Guestrin C. 2016. "why should i trust you?": explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. 2017. GRAD-CAM: visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE International Conference on Computer Vision*, 618–626.

Tavakoli M, Viegas C, Sgrigna L, De Almeida AT. 2018. SCALA: scalable modular rail based multi-agent robotic system for fine manipulation over large workspaces. *Journal of Intelligent & Robotic Systems* **89(3)**:421–438 DOI 10.1007/s10846-017-0560-3.

Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Piscataway: IEEE, 23–30.

Unity Technologies. 2022b. Uma2-unity multipurpose avatar. *Available at https://assetstore.unity.com/packages/3d/characters/uma-2-unity-multipurpose-avatar-35611* (accessed 2 March 2023).

Unity-Technologies. 2022a. Robotics-warehouse. *Available at https://github.com/Unity-Technologies/Robotics-Warehouse* (accessed 2 March 2023).

Vilone G, Longo L. 2020. Explainable artificial intelligence: a systematic review. ArXiv DOI 10.48550/arXiv.2006.00093.

Vilone G, Longo L. 2021. Classification of explainable artificial intelligence methods through their output formats. *Machine Learning and Knowledge Extraction* **3(3)**:615–661 DOI 10.3390/make3030032.

Xu J, Liu J, Sheng J, Liu J. 2018. Arc path tracking algorithm of dual differential driving automated guided vehicle. In: *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. Piscataway: IEEE, 1–7.

Zhu J-Y, Park T, Isola P, Efros AA. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.