Doctoral Dissertation

An Explainable Deep Reinforcement Learning Framework for Patrol Speed Control and Sim-to-Real Deployment of Rail-Guided Robot System

Hosun Lee

Supervisor: **Nak Young Chong**

Division of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
[Information science]

March, 2026

Abstract

Intelligent facility monitoring increasingly relies on autonomous robotic systems, yet the deployment of learning-based control policies in real environments remains challenging due to their black-box nature and the discrepancy between simulation and real-world conditions. Rail-guided patrol robots, while mechanically stable and suitable for large indoor facilities, commonly operate with fixed-speed or rule-based controllers that do not adapt to variations in visual scene complexity. This dissertation presents a deep reinforcement learning–based framework for adaptive patrol speed control of a rail-guided robot system. Patrol speed control is formulated as a continuous decision-making problem in which the robot observes forward-facing RGB images and outputs a continuous velocity command. The Deep Deterministic Policy Gradient (DDPG) algorithm is employed to learn context-dependent speed modulation along a predefined rail trajectory. To encourage adaptation to visual conditions, an entropy-driven reward function is adopted. Image entropy is used as a proxy for visual information density, allowing the learned policy to adjust patrol speed in response to changes in visual scene complexity without relying on manually designed heuristics. This formulation enables perception-driven patrol behavior to emerge from visual statistics. To facilitate analysis of the learned policy, gradient-based visual interpretation is incorporated into the framework. Grad-CAM is applied to the actor network to generate action-specific attention maps that highlight image regions associated with speed decisions. These visualizations provide interpretable evidence of how visual inputs are reflected in control outputs and support qualitative inspection of internal policy behavior. For deployment, the visual discrepancy between simulation and real environments is addressed using CycleGAN-based unpaired image-to-image translation. Real camera images are translated into simulation-style representations, enabling zero-shot execution of the trained policy. The consistency of visual attention patterns before and after translation is examined through Grad-CAM-based analysis. Beyond algorithmic development, a complete rail-guided patrol robot system is implemented and deployed in an agricultural

wholesale market on a ceiling-mounted rail loop of approximately 220 meters to verify system-level integration and operational feasibility. Experiments conducted in simulation and controlled testbeds involve the quantitative analysis of patrol behavior and visual attention patterns, and are used to examine behavioral consistency across environments, rather than to provide full validation of real-world performance.

This dissertation contributes to deep reinforcement learning and robotic perception by addressing adaptive patrol speed control under visually heterogeneous environments with a particular focus on policy interpretability. While prior work on learning-based control primarily emphasizes performance, it often provides limited insight into how perceptual information influences continuous control decisions. This research instead emphasizes interpretability as a means of analyzing and understanding learned behavior. A central contribution of this work is the formulation of patrol speed control as a perception-driven continuous control problem, in which visual scene complexity directly influences the learned policy. Image entropy is employed as a reward signal to reflect variations in visual information density, providing a practical mechanism for linking perception and control without relying on task-specific heuristics. Another key contribution is the application of Grad-CAM to the actor network of a continuous-action deep reinforcement learning policy. By producing action-specific attention maps, the proposed framework provides interpretable visual evidence of image regions associated with speed decisions. These visualizations are treated as correlational indicators rather than causal explanations, and their limitations are explicitly acknowledged. The integration of interpretability with sim-to-real transfer further strengthens the contribution of this research. Instead of evaluating domain adaptation solely through control performance, this dissertation examines whether interpretable visual cues remain consistent after visual domain translation. The analysis of Grad-CAM attention before and after CycleGAN-based translation provides empirical insight into the stability of interpretability across simulated and real environments. Overall, this dissertation does not claim the introduction of new learning algorithms or theoretical models. Its contribution lies in the structured integration of adaptive deep reinforcement learning, visual interpretability, and sim-to-real deployment within a unified patrol framework, offering methodological insights applicable to a broad range of robotic monitoring and inspection systems.

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

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The demand for intelligent monitoring and safety management systems has significantly increased in large-scale multi-use facilities such as wholesale markets, exhibition halls, and transportation terminals. These environments often require continuous surveillance of repetitive and routine tasks, which impose a high workload on human operators. Conventional approaches primarily rely on fixed monitoring devices such as CCTV cameras, fire alarms, and environmental sensors. Although such devices are easy to deploy and maintain, their functionality is limited to static field coverage and reactive responses to detected events [1, 2].

At the international level, facility safety management and disaster risk reduction (DRR) have been recognized as critical global agendas. The UNDRR Disaster Resilience Scorecard for Cities provides a standardized framework to evaluate resilience and safety practices in urban environments, emphasizing the importance of systematic monitoring and preparedness at the facility level [3]. Similarly, the UNDP–UNDRR Data and Digital Maturity framework highlights that digital technologies and data-driven approaches are becoming central to enhancing resilience and enabling proactive disaster risk management [4]. In parallel, the UNDRR Disaster Data Portal underscores the global effort to collect, manage, and utilize disaster-related data to support evidence-based decision-making [5]. These initiatives collectively stress that safety management is no longer limited to reactive systems but increasingly depends on continuous, intelligent, and adaptive

monitoring.

Over the past decade, robotic systems have emerged as promising alternatives for facility monitoring, offering scalability, mobility, and adaptive decision-making. Various approaches using unmanned aerial vehicles (UAVs) and automated guided vehicles (AGVs) have been investigated to enhance facility patrol and inspection, but these platforms require advanced localization, collision avoidance, and complex control mechanisms in cluttered environments [6]. In contrast, rail-guided robots have been proposed as an effective solution due to their high localization accuracy, safe separation from humans, and simplified navigation [7–9]. By installing rails along ceilings or walls, the patrol robot can follow predesigned routes with minimal interference, enabling both efficient routine surveillance and rapid access to emergency sites.

As shown in Fig. 1.1, the developed rail-guided robot has been installed and operated in an agricultural wholesale market, which motivates the need for adaptive speed control and operator-facing explanations under diverse visual complexity.

1.2 Problem Statement

Despite the growing international emphasis on resilience, data-driven monitoring, and digital maturity in disaster risk reduction [3–5], several challenges remain in implementing effective safety management within large-scale facilities.

First, current facility monitoring systems often rely on static patrol strategies, such as fixed-speed navigation or predefined routes. While these approaches ensure predictable operation, they are insufficient for environments where the amount of useful information varies significantly across regions. In areas of higher environmental complexity, fixed patrol plans may fail to capture critical information, whereas in less informative regions, they may generate redundant data [2]. Thus, there is a pressing need for adaptive patrol strategies that adjust operational parameters, such as speed, according to situational demands.

In particular, most existing rail-guided patrol systems, despite their mechanical stability, continue to rely on fixed-gain or fixed-speed control schemes. Such approaches implicitly assume a relatively consistent visual environment. However, indoor facilities exhibit substantial visual variability caused by human congestion, object movement, and

illumination changes. These nonlinear and scene-dependent factors make it difficult to analytically determine an optimal patrol speed, often leading fixed-speed controllers to either overshoot in visually complex areas or operate inefficiently in simpler regions.

Similar limitations have been reported in broader mobile robot research, where classical PID or fuzzy logic controllers degrade under dynamic or visually cluttered conditions [10]. Recent studies further indicate that learning-based gain optimization becomes necessary even for humanoid or assistive robots, particularly when digital twins are employed to capture real-world variability [11]. These findings suggest that static control strategies are fundamentally inadequate for ensuring consistent information acquisition across heterogeneous environments.

Second, although deep learning-based perception methods have improved hazard and anomaly detection capabilities in facility monitoring [1], the decision-making process of reinforcement learning models typically operates as a "black box". This lack of transparency hinders operator trust in safety-critical scenarios, where human supervisors must be able to understand and validate the rationale behind robotic decisions. Without explainability, even highly accurate models risk limited adoption in real-world facility management.

Finally, deploying learning-based patrol strategies faces the well-known sim-to-real gap. Policies trained exclusively in simulation environments often degrade when transferred to real facilities due to discrepancies in lighting, environmental noise, and structural variation. Bridging this gap remains a significant obstacle to the practical implementation of reinforcement learning for safety monitoring applications.

1.3 Research Objective

The primary objective of this dissertation is to develop an Explainable Deep Reinforcement Learning (XDRL) framework for rail-guided patrol robots that ensures adaptive, trustworthy, and transferable patrol operations in facility environments. The specific objectives are as follows:

1. To design a rail-guided patrol platform integrated with perception modules for environmental monitoring and hazard detection.

2. To develop a reinforcement learning-based adaptive speed control framework that dynamically adjusts the patrol plan according to environmental complexity.

3. To integrate explainable AI methods to visualize and communicate the environmental features influencing control decisions.

4. To investigate sim-to-real transfer strategies for deploying policies trained in simulation to real-world facilities.

1.4 Contributions

The main contributions of this dissertation are summarized as follows:

- Development of a rail-guided patrol robot platform tailored for large-scale indoor monitoring with safe, efficient, and scalable operation.
- Proposal of a reinforcement learning-based patrol speed control framework that adjusts robot behavior in response to environmental information.
- Introduction of an XAI-enabled decision-making system that provides visual explanations for key control values, improving operator trust and system transparency.
- Implementation of domain adaptation techniques and transfer learning methods to bridge the gap between simulation-trained models and real-world applications.
- Validation of the proposed framework through simulation studies and real-world experiments, demonstrating its effectiveness in adaptability, explainability, and operational safety.

1.5 Thesis Structure

The rest of this dissertation is organized as follows.

Chapter 2 presents technical preliminaries on smart patrol systems for facility safety monitoring, deep reinforcement learning for continuous control, post hoc visual explainability, and simulation-to-real transfer.

Chapter 3 surveys related works on rail-guided patrol systems, DRL for robotic control, XAI in robotics, and sim-to-real techniques, and identifies the research gap.

Chapter 4 describes the rail-guided smart patrol robot and the facility monitoring system, including mechanical design, sensing/communication modules, operation modes, and driving dynamics.

Chapter 5 develops the proposed framework: the patrol problem and entropy-driven reward, the DDPG actor–critic, Grad-CAM integration, and CycleGAN-based visual domain adaptation for zero-shot sim-to-real. It also outlines the evaluation protocol.

Chapter 6 reports experiments in simulation and a real-world testbed, analyzing learning dynamics, adaptive speed behavior under visual complexity, quality of explanations, and sim-to-real performance (including attention alignment).

Chapter 7 concludes with findings, limitations, and future directions such as quantitative XAI evaluation, multi-robot extensions, and on-device adaptation.

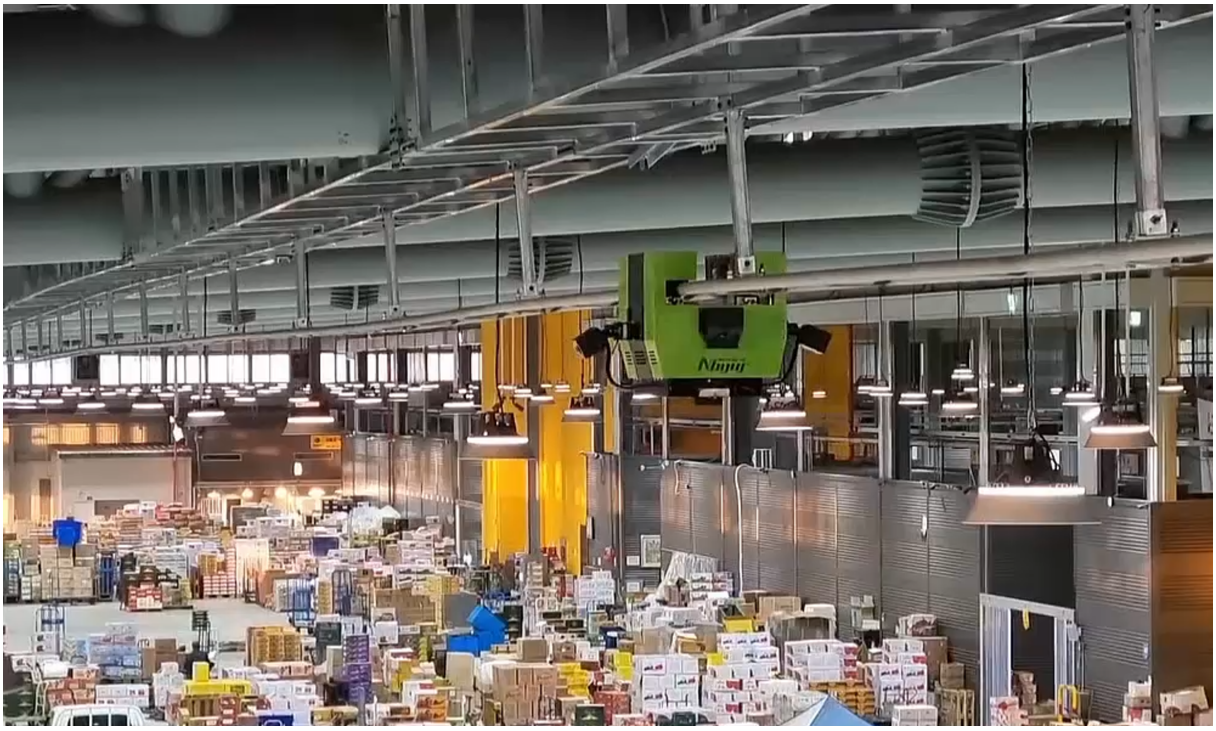


Figure 1.1: Deployed rail-guided smart patrol robot operating in an agricultural wholesale market. The ceiling-mounted dual-rail track, side pan-tilt camera modules, and localization bar-codes enable safe, repeatable patrol and continuous data streaming to the remote control room.

Chapter 2

Preliminaries

2.1 Smart Patrol Systems for Facility Safety Monitoring

Facility safety monitoring systems have evolved through several technological generations, driven by advances in sensing, networking, and artificial intelligence. Early approaches relied primarily on human patrol and simple sensor devices, while recent developments incorporate robotics and intelligent decision making. This section outlines the generational evolution of facility safety monitoring, from manual inspection to autonomous and intelligent systems.

2.1.1 First Generation: Manual Monitoring

The first generation of facility safety monitoring was based on human patrols supported by analog CCTV cameras and basic fire or gas detectors. Alarm devices such as sirens and warning lights were used to alert facility managers, who were required to respond directly on-site. While effective in small-scale environments, these systems were inefficient for large facilities and introduced delays in emergency response.

2.1.2 Second Generation: Automated and Integrated Monitoring

The second generation introduced digitalization and partial automation. Digital CCTV and IoT-based environmental sensors (temperature, humidity, vibration, gas) enabled real-time data acquisition. Centralized monitoring platforms integrated multimodal sensor data, while event-driven systems triggered automated responses such as sprinklers or ventilation. Building automation systems (BAS) and building energy management systems (BEMS) were also widely adopted, contributing to improved efficiency and reliability in facility management. Similar technological trends have been observed in supervisory control and data acquisition (SCADA) systems, which transitioned from isolated analog devices to networked and integrated digital architectures [12, 13].

2.1.3 Third Generation: Intelligent Analysis with AI

The third generation leveraged artificial intelligence for intelligent analysis and predictive safety management. Video analytics enabled advanced functions such as intrusion detection, smoke recognition, and object fall detection. Big data techniques supported predictive maintenance, while digital twin models provided virtual simulations of safety scenarios. Importantly, explainable AI (XAI) methods began to emerge, allowing safety managers to interpret AI-driven alerts and enhance decision trustworthiness. These developments reflect the broader adoption of machine learning techniques in reliability engineering and safety applications [14]. Moreover, the concept of "Safety 4.0" emphasizes the integration of AI, IoT, and predictive analytics to create a proactive safety culture in industrial environments [15].

2.1.4 Fourth Generation: Autonomous and Integrated Safety Intelligence

The fourth generation integrates robotics, AI, and cyber-physical systems into a unified and autonomous safety monitoring framework. Patrol robots and drones are deployed for real-time inspection and emergency response, especially in hazardous or hard-to-reach areas. Reinforcement learning (RL) enables adaptive speed and path control, while

cloud and mobile platforms facilitate remote management and maintenance. Furthermore, integration with smart city infrastructures allows cross-facility safety management and multi-system interoperability. Cyber-physical security (CPS) mechanisms ensure resilience against both digital and physical threats. This stage represents the realization of a fully autonomous and intelligent safety ecosystem, aligning with the trajectory of industrial digital transformation and intelligent safety management [14, 15].

2.1.5 Summary of Generational Road-map

Table 2.1 summarizes the generational road-map of facility safety monitoring systems.

Table 2.1: Generational Roadmap of Facility Monitoring Systems (Transposed)

Aspect	**1st: Manual**	**2nd: Automated**	**3rd: Intelligent**	**4th: Autonomous**
Key Concept	Human-centered basic monitoring	Digitalization and integration	AI-based risk recognition and prediction	Self-adaptive patrol and intelligent operation
Technologies	Analog CCTV, fire/gas sensors, simple alarms, manual response	Digital CCTV, IoT sensors, integrated monitoring platforms, BAS/BEMS, automated alarms	Video analytics (intrusion, smoke detection), predictive maintenance, digital twin, XAI explanations	Patrol robots/drones, RL-based adaptive control, smart city integration, CPS security, cloud/mobile maintenance
Limitations	Inefficient for large-scale facilities; delayed emergency response	Still reactive, fixed coverage	Complexity of AI models; need for interpretability	Challenges in sim-to-real transfer; system integration risks
Outcomes	Basic safety assurance at small scale	Improved efficiency and real-time monitoring	Preventive safety management; improved trust in decision-making	Towards fully autonomous safety management; enhanced cost efficiency and robustness

2.2 Deep Reinforcement Learning

2.2.1 Fundamentals of Reinforcement Learning

Reinforcement Learning (RL) provides a mathematical framework in which an agent learns to make sequential decisions by interacting with an environment. Formally, RL problems are modeled as a Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s, a)$ is the transition probability from state s to s' given action a, $r(s, a)$ is the reward, and $\gamma \in [0, 1)$ is the discount factor. The objective of the agent is to maximize the expected return

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}.$$

In the context of robotics, the MDP formulation allows modeling of complex sequential tasks such as trajectory tracking, adaptive speed control, and safe obstacle avoidance, where the agent continuously refines its strategy through trial-and-error interaction with the environment.

2.2.2 Deep Reinforcement Learning Framework

Classical RL methods are limited when applied to robotic domains, as robotic systems often involve high-dimensional sensory inputs (e.g., camera images, LiDAR scans) and continuous action spaces (e.g., motor torques or velocities). Deep Reinforcement Learning (DRL) addresses these challenges by incorporating deep neural networks as function approximators, enabling policies and value functions to generalize across complex state spaces. A DRL-based robotic agent observes raw sensor data, processes it through convolutional or recurrent networks, and selects control actions such as speed commands or joint movements. This approach has enabled significant progress in real-world robotic tasks, including robotic manipulation, mobile robot navigation, and autonomous vehicle control.

2.2.3 Value-based and Policy-based Methods

DRL algorithms are typically divided into value-based and policy-based approaches.

Value-based methods approximate the action-value function $Q(s, a)$, selecting the action with the highest estimated return. The Deep Q-Network (DQN) is a representative method that achieved human-level performance in video games and has been adapted to discrete robotic control tasks, such as path planning on grid-based maps. However, value-based methods struggle in robotics applications with continuous action spaces, since the discretization of actions (e.g., motor torques) is impractical.

Policy-based methods directly parameterize the policy $\pi_\theta(a|s)$, making them more suitable for robotic applications where actions are inherently continuous. Algorithms such as REINFORCE have been applied to robotic locomotion and manipulation, where the policy outputs continuous control commands. Despite their suitability for continuous control, policy-based methods often suffer from high variance in gradient estimation, leading to instability during training.

2.2.4 Actor–Critic Methods

Actor–Critic methods combine the benefits of value-based and policy-based learning, making them particularly attractive for robotics. The actor produces actions based on sensor-derived states, while the critic evaluates these actions by estimating value functions. Through this feedback loop, the actor learns to refine its control strategy based on both direct experience and value-based evaluation. Notable methods such as Advantage Actor–Critic (A2C) and Asynchronous Advantage Actor–Critic (A3C) have been applied to robotic applications, including real-time motion planning and multi-robot coordination, due to their improved stability and faster convergence.

2.2.5 Deterministic Policy Gradient (DPG)

Many robotic tasks, such as velocity regulation, force control, or path following, require continuous control signals. Deterministic Policy Gradient (DPG) methods address this by introducing a deterministic policy $\mu_\theta(s)$ that maps states directly to actions without sampling from probability distributions. The deterministic policy gradient theorem is given by:

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)\big|_{a=\mu_\theta(s)}\right],$$

where ρ^{μ} is the state distribution induced by the policy. This formulation reduces variance in gradient estimation and is well-suited for robotic domains, where stable and precise control signals are critical.

2.2.6 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) extends DPG by integrating deep neural networks, enabling scalable learning in high-dimensional continuous domains. In robotics, DDPG has been widely adopted for tasks such as robotic arm manipulation, UAV trajectory control, and autonomous navigation, where actions such as joint torques or wheel velocities must be continuously adjusted. The algorithm consists of two key components:

- **Actor network**: generates continuous control commands (e.g., patrol speed, joint torque) given the current state.
- **Critic network**: evaluates the quality of state–action pairs by estimating the action-value function $Q(s, a)$.

To enhance training stability, DDPG employs a **replay buffer**, which stores past robotic experiences (s, a, r, s') and enables random sampling to break correlations in sequential data, and **target networks**, which are slowly updated to stabilize learning. These features make DDPG highly effective for robotic systems that require adaptive and precise control in dynamic environments. Nevertheless, the use of deep networks makes DDPG policies difficult to interpret, motivating the integration of explainability techniques, which will be discussed in Section 2.3.

2.3 Explainable Artificial Intelligence (XAI)

As AI systems grow in capability and adoption across safety-critical domains, their opacity raises concerns about trust, accountability, and safe operation [16, 17]. Deep models often function as *black boxes* whose internal decision-making is not readily understandable to humans, which is problematic in applications where erroneous predictions can impose significant risks. Accordingly, *Explainable Artificial Intelligence (XAI)* has emerged to provide human-understandable justifications for model outputs while preserving predictive performance [16, 17].

In the context of this dissertation, the patrol robot's control policy is realized by a deep neural network, which is inherently non-interpretable. To support operator trust and facilitate diagnosis during deployment, post hoc explanations are required to reveal which aspects of the visual input most influence speed decisions.

2.3.1 Methods for Explainability

XAI approaches can be broadly grouped as follows:

- **Post-hoc explanation methods.** These analyze trained black-box models to produce explanatory artifacts. For vision models, gradient-based visualizations such as saliency maps and **Grad-CAM** highlight spatial regions that contribute most to a particular output [18]. Perturbation-based methods approximate local decision boundaries by probing model responses to input variations, e.g., **LIME** [19] and **SHAP** [20].

- **Inherently interpretable models.** These are transparent by design (e.g., linear regression, decision trees, rule-based systems) and allow direct inspection of decision logic. While highly understandable, they often lack the expressive power needed for high-dimensional perception and continuous control.

Post-hoc methods scale to complex models but provide approximate reasoning; interpretable models provide clarity but may underfit complex tasks. In robotics with high-dimensional sensory inputs (images) and real-time constraints, post-hoc visual/perturbation explanations are particularly attractive [16].

2.3.2 Explainable vs. Interpretable AI

Although used interchangeably, the terms carry distinct meanings [16,17]. *Interpretability* refers to models whose structure/parameters are directly understandable; *explainability* refers to techniques that elucidate black-box behavior. A concise comparison is given in Table 2.2.

Table 2.2: Comparison between Interpretable AI and Explainable AI

Aspect	**Interpretable AI**	**Explainable AI**
Definition	Inherently transparent models	Explanations for black-box models
Examples	Linear regression, Decision trees, Rule-based models	Grad-CAM [18], LIME [19], SHAP [20], Saliency Maps
Strengths	High transparency, easy to trust	Applicable to deep learning models; local attributions possible
Limitations	Limited expressiveness in complex tasks	Explanations are approximate and may not fully reveal reasoning

2.3.3 Explainability in Reinforcement Learning and Robotics

In reinforcement learning (RL), agents learn sequential decision-making policies by interacting with uncertain environments. Understanding *why* an agent selects a specific action at a given state is essential for debugging, safety validation, and operator trust. For vision-driven policies, gradient-based localization methods (e.g., **Grad-CAM**) can highlight regions in the sensory input that most influence the action, yielding an intuitive visual rationale [18]. Perturbation-based methods (e.g., **LIME** [19], **SHAP** [20]) can complement such maps by quantifying feature contributions around a given state–action pair. In robotic settings, these tools help verify that decisions are driven by task-relevant features (e.g., humans, obstacles, boundaries) rather than spurious correlations.

In this dissertation, **Grad-CAM** is adopted for the patrol robot's vision-based policy because it (i) produces spatially localized heatmaps tied to convolutional features, (ii) requires no retraining or architectural changes, and (iii) is fast enough to be generated alongside inference for operator-facing inspection [18]. Concretely, Grad-CAM is integrated with the DDPG *actor* to visualize which image regions most influence the continuous speed output. Formal details of the weighting and heatmap construction are provided later in Chapter 5, where the role of these explanations in assessing attention alignment *before and after* domain adaptation [16, 17].

Limitations and caveats. Post-hoc explanations offer *approximate* insight rather than causal proof; they can be sensitive to layer choice and may miss non-visual factors. Therefore, Grad-CAM is treated as a complementary tool for transparency and debugging, not

as a sole safety guarantee. In this work, it serves to (i) increase operator confidence in adaptive speed decisions and (ii) verify that sim-to-real adaptation preserves semantically meaningful attention.

2.4 Simulation-to-Real Transfer via Domain Adaptation

Training deep reinforcement learning (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 [21, 22].

To overcome this challenge, domain adaptation methods are employed to reduce the mismatch between simulated and real-world inputs. In this study, an unpaired image-to-image translation method, CycleGAN [23], is employed 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.

Grad-CAM visualizations are further used to evaluate the semantic alignment of the translated images, in order to assess whether CycleGAN preserves task-relevant visual features during domain adaptation. This combination of domain adaptation and visual explanation supports robust and explainable real-world deployment.

Chapter 3

Related Works

This chapter reviews prior work in four areas that are central to this dissertation: (i) rail-guided patrol robot systems, (ii) deep reinforcement learning for robotic control, (iii) explainable AI (XAI) in robotics, and (iv) simulation-to-real (sim-to-real) transfer. It is concluded with a summary that positions the research gap addressed in this dissertation.

3.1 Rail-Guided Patrol Robot Systems

Rail-guided robots have been explored as practical platforms for large-area monitoring in industrial and public facilities due to their predictable motion, simplified localization, and safe separation from pedestrians and equipment [7,24,25]. They have been applied to tasks such as environmental sensing, thermal inspection, and security surveillance, where scheduled patrolling and remote observation are required [2, 26–28]. Typical systems integrate RGB/thermal cameras and environmental sensors, stream data to a remote control room, and follow pre-defined routes at one or a few fixed speeds.

Despite demonstrated practicality, two limitations recur in the literature. First, *non-adaptive speed control*: patrol velocity is typically constant or manually selected, which can be inefficient when environmental complexity varies along the route. Second, *limited transparency*: when ML-based components are used (e.g., detection or triage), operator-facing explanations are scarce, which may hinder trust and timely intervention in safety-critical settings. These limitations motivate adaptive and explainable control strategies on rail-guided platforms.

3.2 Deep Reinforcement Learning for Robotic Control

Deep reinforcement learning (DRL) has enabled learning-based controllers that map high-dimensional observations (e.g., images) to continuous control actions without manual heuristics. The Deep Deterministic Policy Gradient (DDPG) algorithm [29] is a model-free, off-policy actor–critic method tailored to continuous action spaces and has been widely used in manipulation and locomotion. For patrol speed control, DDPG is attractive because it (i) outputs continuous velocities, (ii) learns directly from visual features, and (iii) can leverage experience replay for sample-efficient training.

Prior DRL-based patrol or navigation studies typically focus on free-moving mobile robots, where localization and collision avoidance dominate the problem. In contrast, rail-guided robots constrain the path and simplify safety, shifting the challenge toward *context-aware speed modulation* along a known trajectory. Existing rail-guided systems seldom employ DRL for this purpose; when learning-based methods appear, they often lack explicit mechanisms for operator interpretability. This dissertation therefore leverages DDPG to learn adaptive speed policies from images while preparing for explainability and deployment.

3.3 Explainable AI in Robotics

As deep policies become more capable, their opacity raises concerns about safety, accountability, and operator trust. Explainable AI (XAI) provides post hoc tools to clarify model behavior [16, 17]. For vision-driven policies, gradient-based localization methods such as Grad-CAM [18] highlight spatial regions that most influence a particular output. Grad-CAM is appealing in robotic perception–control pipelines because it (i) is architecture-agnostic at the last convolutional layer, (ii) requires no retraining, and (iii) produces visual overlays that operators can readily interpret.

In robotics, XAI has been applied to inspect perception modules (e.g., object detection) and to analyze failure modes in end-to-end policies. However, in patrol settings, there remains a gap between *adaptive control* and *operator-facing explanations*. Few studies close this gap by integrating a learned continuous-speed policy with real-time visual

rationales on where and why the robot slows down or speeds up. This work incorporates Grad-CAM into the actor network to surface action-specific attention maps for speed decisions.

3.4 Sim-to-Real Transfer Techniques

Policies trained in simulation often face degradation when deployed in the real world due to appearance and dynamics mismatches—the sim-to-real gap [21]. Two major approaches exist: *domain randomization* perturbs simulation textures, lighting, and geometry during training so that the policy generalizes to reality [22]; *domain adaptation* transforms observations across domains to reduce distribution shift at deployment. CycleGAN [23] enables unpaired image-to-image translation that preserves semantic structure while matching appearance statistics across domains. For vision-driven control, translating real images into the simulation style can make a simulation-trained policy immediately applicable without finetuning.

Most sim-to-real studies evaluate task success but do not examine whether *explanations* remain meaningful after adaptation. In patrol speed control, this research is interested in both (i) preserving performance under visual shifts and (ii) maintaining interpretable attention (e.g., focusing on crowded or visually complex areas). This dissertation, therefore, combines CycleGAN-based adaptation with Grad-CAM analysis to verify the semantic alignment of attention before and after translation.

3.5 Summary and Research Gap

Prior work establishes (i) rail-guided robots as viable patrol platforms [7, 24–28], (ii) DRL—particularly DDPG—as a strong candidate for continuous control [29], (iii) Grad-CAM as a practical post hoc explainer for vision-based models [16–18], and (iv) domain adaptation as an effective means for sim-to-real transfer [21–23]. However, an integrated framework that *simultaneously* delivers (a) adaptive patrol speed on a rail-guided platform, (b) operator-facing visual explanations of speed decisions, and (c) zero-shot sim-to-real deployment with preserved explainability, remains underexplored.

Table 3.1 summarizes this gap. Addressing it requires a unified pipeline that (1)

learns a continuous speed policy from images, (2) explains decisions via attention maps, and (3) deploys zero-shot to the real world using domain adaptation while validating the persistence of meaningful attention. The remainder of this dissertation develops and evaluates such a framework.

Table 3.1: Positioning of this dissertation relative to representative directions in prior work.

Direction	**Rail-guided**	**Adaptive speed**	**Explainability**	**Sim-to-real**
Rail-guided patrol systems [26–28]	✓	×	×	△
DRL for control (DDPG) [29]	△	✓	×	△
XAI for vision policies [16, 18]	△	△	✓	△
Domain adaptation [22, 23]	△	△	×	✓
This dissertation	✓	✓	✓	✓

Legend: ✓ explicitly addressed; △ partially or implicitly; × not addressed.

Chapter 4

Rail-guided Smart Patrol Robot for Facility Monitoring

Unlike previous works that have explored rail-guided robots for facility monitoring [26–28], the rail-guided smart patrol robot addressed in this dissertation is a novel system that was designed and implemented as part of this doctoral research. It was designed and built from scratch, integrating BLDC-driven driving system, multi-sensor modules (RGB and thermal cameras, air quality sensors), and a real-time remote management server. As illustrated in Figure 4.1, the platform is capable of autonomous patrolling, environmental monitoring, fire and accident detection, and remote operator collaboration. To develop the target system, the automated patrol robot and the data processing server in the remote control room must be equipped with the next five functions.

- **24-hour 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 $\pm 0.2m$ position error. The docking range of the wireless charging station is within $\pm 0.05m$. Through precise position control using the BLDC motor, the driving unit is possible to achieve with $\pm 0.01m$ 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 $1m$ 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 320x240 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 7 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 second.

- **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.

Table 4.1: Specifications of the developed rail-guided smart patrol robot

Item	**Specification**
Size, Weight	$400 \times 470 \times 430$ mm, 18 kg
Driving speed and time	0.6 / 1.2 / 1.5 m/s, max 1 hour
Environmental sensors	Temperature, Humidity, CO_2, VOC, PM(1.0, 2.5, 10)
RGB camera	1/2.9" CMOS, 1465×1088, FOV 60°
Thermal camera	320×240, $-40 \sim 330$°C, FOV 60°
Battery	17 Ah, 0.75 C

The main specifications of the developed robot are summarized in Table 4.1. This hardware platform provides the **experimental foundation** for the research in this dissertation. Not only does it support the implementation of adaptive patrol speed control, but it also enables real-world validation of explainability methods (Grad-CAM) and sim-to-real deployment (CycleGAN). In the following sections, the detailed mechanical design, the driving system, and the management architecture of the developed smart patrol robot are described.

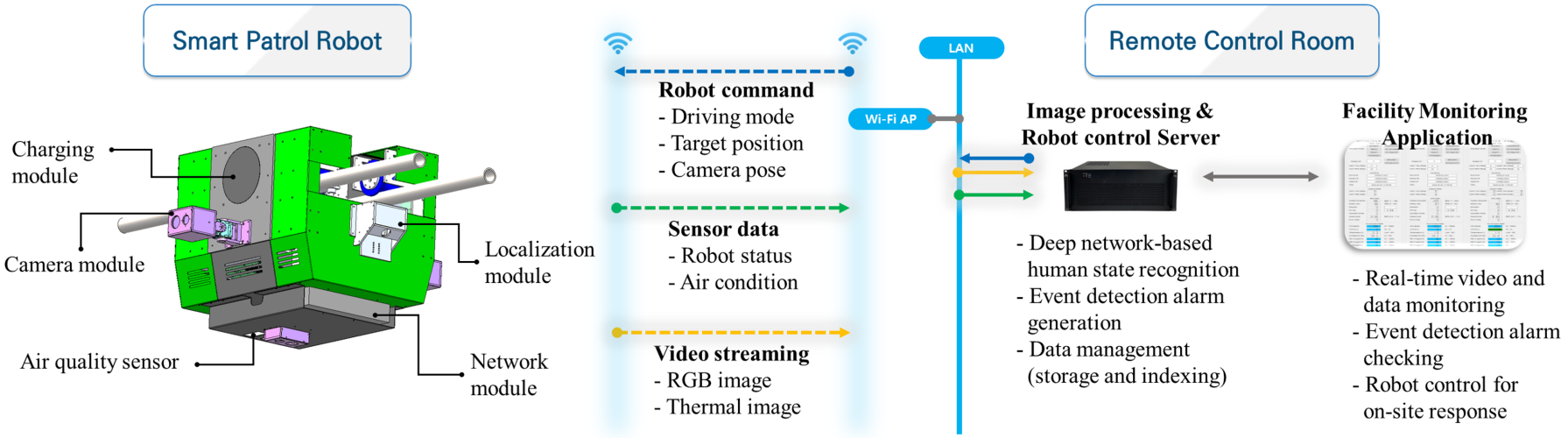


Figure 4.1: System overview: configuration of Smart patrol robot, the definition of data, and services of Remote control room

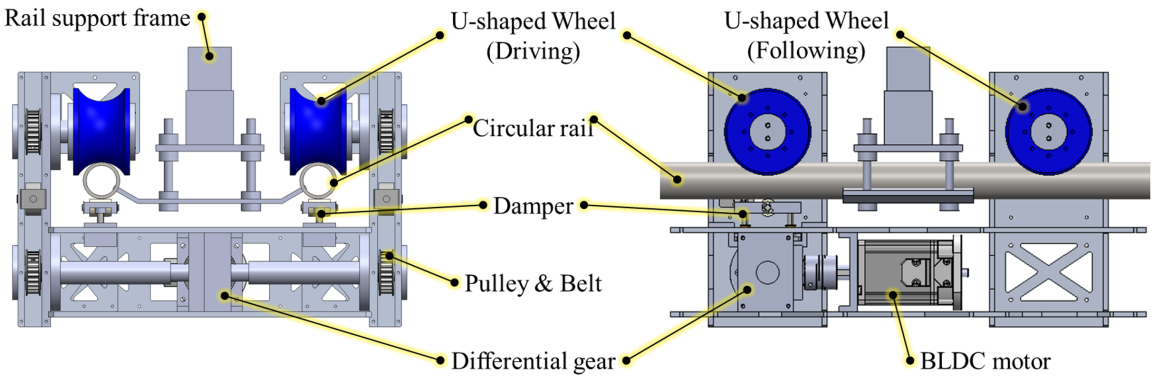


Figure 4.2: Configuration of driving system

4.1 Rail-guided patrol robot

Figure 4.2 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 4 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 4.3 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:

$$m\ddot{x} + b\dot{x} + f_F = f_A \tag{4.1}$$

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 with the driving torque, τ_A, and the radius of the wheel, r_w. The driving torque, τ_A is

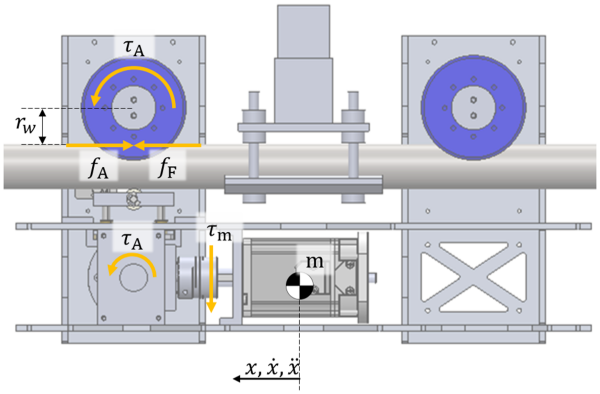


Figure 4.3: Dynamic model of the smart patrol robot

driven with the motor torque, τ_m, and the gear ratio, n.

$$f_A = \tau_A / r_w \tag{4.2}$$

$$\tau_A = n\tau_m \tag{4.3}$$

The following equation is the model of the friction force.

$$f_F = \mu mg \; sign(\dot{x}) \tag{4.4}$$

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 3000 rpm speed and 0.635 Nm rated torque is selected to achieve 1.8 m/s of the peak speed.

Figure 4.4 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.

$$l \sin \theta_1 = y + d_w \tag{4.5}$$

$$l \cos \theta_1 = x \tag{4.6}$$

$$(l + d_r) \sin \theta_2 = y \tag{4.7}$$

$$(l + d_r) \cos \theta_2 = x + w \tag{4.8}$$

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:

$$l = \sqrt{(y + d_w)^2 + x^2} \tag{4.9}$$

where,

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{4.10}$$

$$a = \left(\frac{w}{d_r}\right)^2 - 1 \tag{4.11}$$

$$b = \frac{w(y^2 + w^2 - (y + d_w)^2 - d_r^2)}{d_r^2} \tag{4.12}$$

$$c = \left(\frac{y^2 + w^2 - (y + d_w)^2 - d_r^2}{2d_r}\right)^2 - (y + d_w)^2 \tag{4.13}$$

Figure 4.5 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.

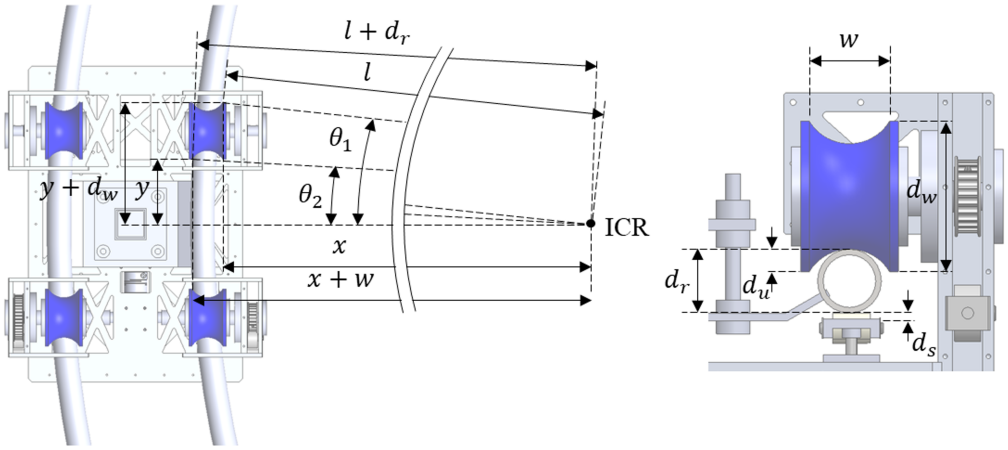


Figure 4.4: Kinematical descriptions of the rail structure and the smart patrol robot

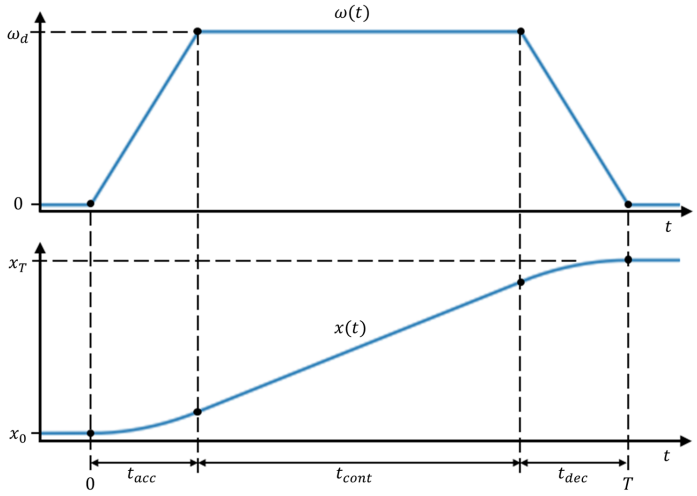


Figure 4.5: Speed reference model for driving control

4.2 Management System

The management system of the developed rail-guided patrol robot is responsible for coordinating all aspects of the facility monitoring operation, from sensor data collection to remote operator interaction. It consists of two tightly integrated components: (1) the onboard robot system that executes patrol missions and acquires sensory data, and (2) the image processing and control server that manages data storage, computationally intensive processing, and communication with the remote control room. Together, these two components form a distributed architecture in which the robot provides mobility and sensing, while the server delivers computational power and connectivity to external monitoring

Table 4.2: Specifications of the image processing and robot control server

Item	**Specification**
Operating System	Linux
CPU	Intel Xeon E5-1630
Memory	16 GB
Storage	2 TB SSD
GPU	NVIDIA RTX 2060
Network	Dual-port 1 Gb Ethernet

platforms.

4.2.1 System Architecture

The on-board robot is equipped with driving units, environmental sensors, an RGB and a thermal camera module, as well as embedded controllers for motor control and low-level sensor fusion. While the robot is capable of basic autonomous functions such as movement along the rail, obstacle detection through ultrasonic sensors, and wireless docking at the charging station, its embedded hardware is intentionally kept lightweight in order to minimize weight and power consumption. This design choice allows the robot to achieve longer operational times while delegating high-load tasks such as image analysis and database management to the external server.

The dedicated image processing and robot control server serves as the computational backbone of the system. It processes incoming visual streams from the robot, performs thermal image analysis for fire detection, manages large-scale data storage for environmental monitoring, and implements real-time communication protocols for robot control. This separation of concerns—mobility and sensing on the robot side, computation and management on the server side—ensures a balanced distribution of functionality and robustness in deployment.

The hardware specifications of the robot and server are summarized in Table 4.1 and Table 4.2, respectively. The robot itself provides the sensing and actuation platform, while the server ensures that the acquired data can be processed, stored, and acted upon efficiently in real time.

By leveraging this server–robot architecture, the patrol system can simultaneously support real-time monitoring tasks (such as streaming video and issuing patrol commands)

and computationally demanding modules (such as running deep reinforcement learning models and explainability algorithms). In practice, this means that the operator in the remote control room can access live camera feeds, environmental sensor data, and automatically generated alarm information through a unified dashboard, while the server ensures that these services are provided reliably without overloading the robot hardware.

4.2.2 Operation Modes and Command Structure

Beyond the hardware and computational infrastructure, the management system defines a structured set of operation modes and commands that allow the robot to adapt its behavior to different monitoring contexts. The operation modes were designed to balance autonomy and operator control, ensuring both efficiency during routine patrols and flexibility in responding to unexpected events. Four primary operation modes are implemented: normal-speed patrol, high-speed patrol, manual driving, and emergency response.

Normal-speed patrol. In this mode, the robot autonomously follows a predefined patrol schedule at a nominal speed of 0.6 m/s. The purpose is to continuously collect environmental data (temperature, humidity, air quality) while streaming RGB and thermal video. The camera module operates at fixed angles, providing periodic observations of the facility. Since the system prioritizes thorough data collection in this mode, it is best suited for routine facility monitoring when no immediate risk is present. The alarm generation module remains active, and any anomaly detected (e.g., sudden temperature increase, abnormal posture recognition) is reported to the operator in real time.

High-speed patrol. When an operator requests a faster inspection, the robot switches to a speed of 1.2 m/s to cover the facility more quickly. Although video and environmental data are still collected, the reduced dwell time at each location means fewer samples are available for anomaly detection. Thus, this mode sacrifices depth of sensing for coverage speed, allowing managers to quickly obtain an overview of facility conditions. It is particularly useful in large environments where multiple patrol rounds are required.

Manual driving. This mode gives full control to the operator in the remote control room. The robot can be directed to any location on the rail at up to 1.0 m/s, with the camera angle adjusted manually via the pan-tilt module. In this mode, the robot func-

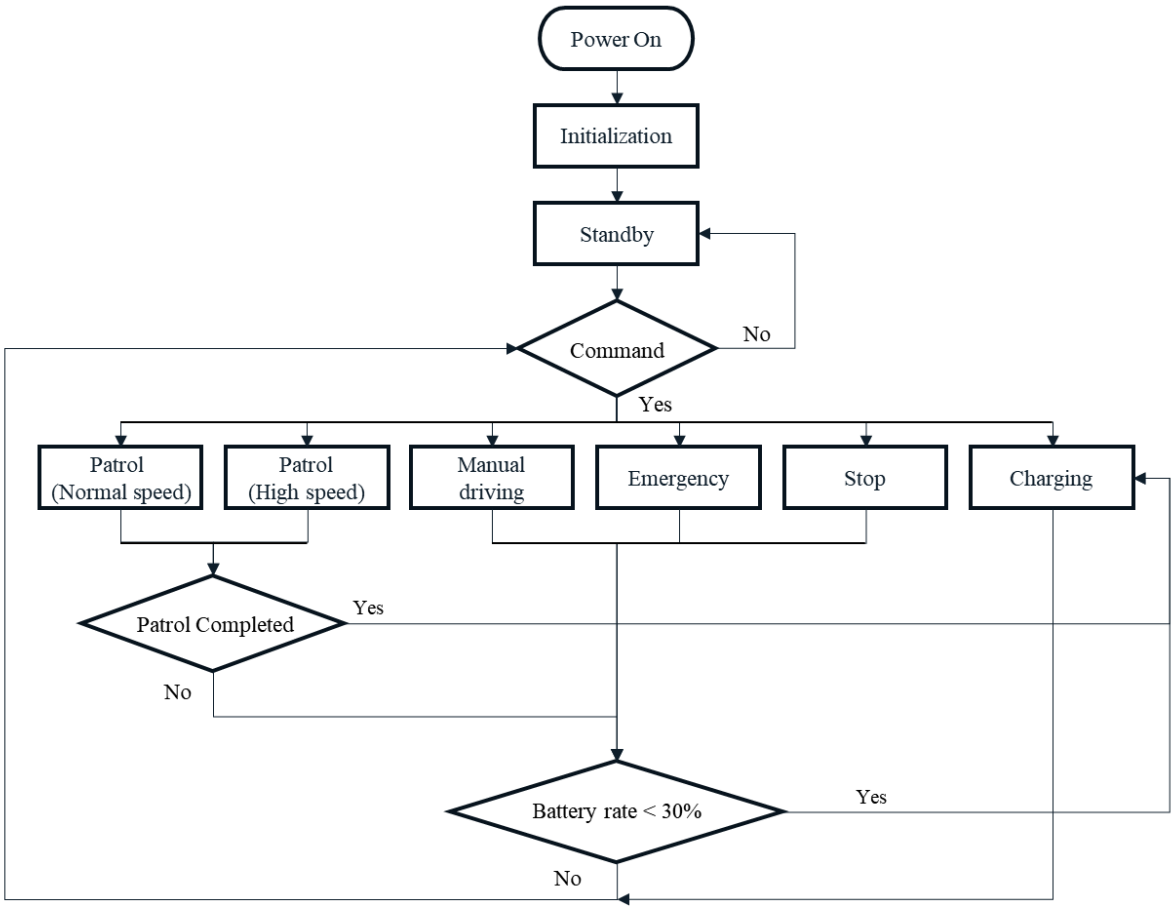


Figure 4.6: Flowchart of operation modes and command transitions in the management system

tions as a telepresence device, enabling human operators to perform targeted inspections. However, automated detection functions are disabled until the robot reaches the target location, as the system prioritizes responsiveness to operator commands over autonomous analysis.

Emergency response. In case of external alarms (e.g., fire detection by other systems), the robot immediately switches to the emergency mode, moving at its maximum speed of 1.5 m/s to the designated location. Once it arrives, all sensing and detection modules are activated at full capacity to assess the situation. This is the most automation-intensive mode, designed to minimize reaction time and support crisis management. Its reliability depends on robust integration with external alarm sources and high confidence in anomaly detection algorithms.

Figure 4.6 illustrates the logical flow of transitions between operation modes. In

addition to these four main modes, auxiliary states such as stop mode and charging mode are also defined. Stop mode allows the operator to pause ongoing tasks for closer inspection or to switch to another mode. Charging mode is automatically triggered when the battery level falls below a threshold, directing the robot to return to the docking station.

By designing this hierarchical command structure, the management system ensures that the patrol robot can flexibly adapt to different operational contexts. Routine monitoring, rapid inspection, targeted manual control, and emergency response are all supported within a unified framework. Importantly, all these operation modes are enabled by the robot–server architecture described in Section 4.2.1, which provides the necessary computational resources, communication backbone, and real-time integration with the remote control room.

4.3 Field Deployment and On-Site Experiments

This section reports the field deployment of the developed rail-guided smart patrol system in an agricultural wholesale market and summarizes the operational results on site.

4.3.1 Site and Rail Layout

The patrol system was installed in a large indoor wholesale market and operated along a ceiling-mounted dual-rail track. Figure 4.7 shows the floor plan of the rail structure. The path was designed as a closed loop to cover both storefronts and passageways efficiently. The total loop length is $220m$. At the nominal patrol speed of 0.6 m/s, a single cycle takes approximately 6.1 min; at 1.2 m/s it takes about 3.1 min.

4.3.2 Experimental setup and results

An end-to-end operation was performed with the following scenarios:

- **Routine patrol (general mode):** closed-loop traversal at 0.6 m/s with continuous acquisition of robot state and environmental data; RGB/thermal video streaming to the server.

Figure 4.7: Design of rail path: straight section (blue) and round section (red)

- **Repositioning:** operator-issued commands for high-speed patrol(1.2 m/s), stop, and manual driving(0.6 m/s or 1.2 m/s).

- **Auto-docking:** return-to-charger and wireless docking at the end of the mission.

The server (Section 4.2) subscribed to MQTT topics for robot/sensor telemetry and ingested RTSP streams for storage and live monitoring. All timestamps were synchronized to the server clock.

Figure **??** summarizes representative time-series results during a complete run: initial low-speed patrol, an operator commands for high-speed patrol, stop, manual driving and auto-docking. Robot position is reported in rail arc-length coordinates; mode transitions are annotated. The environmental sensor data is also transmitted to the server and stored.

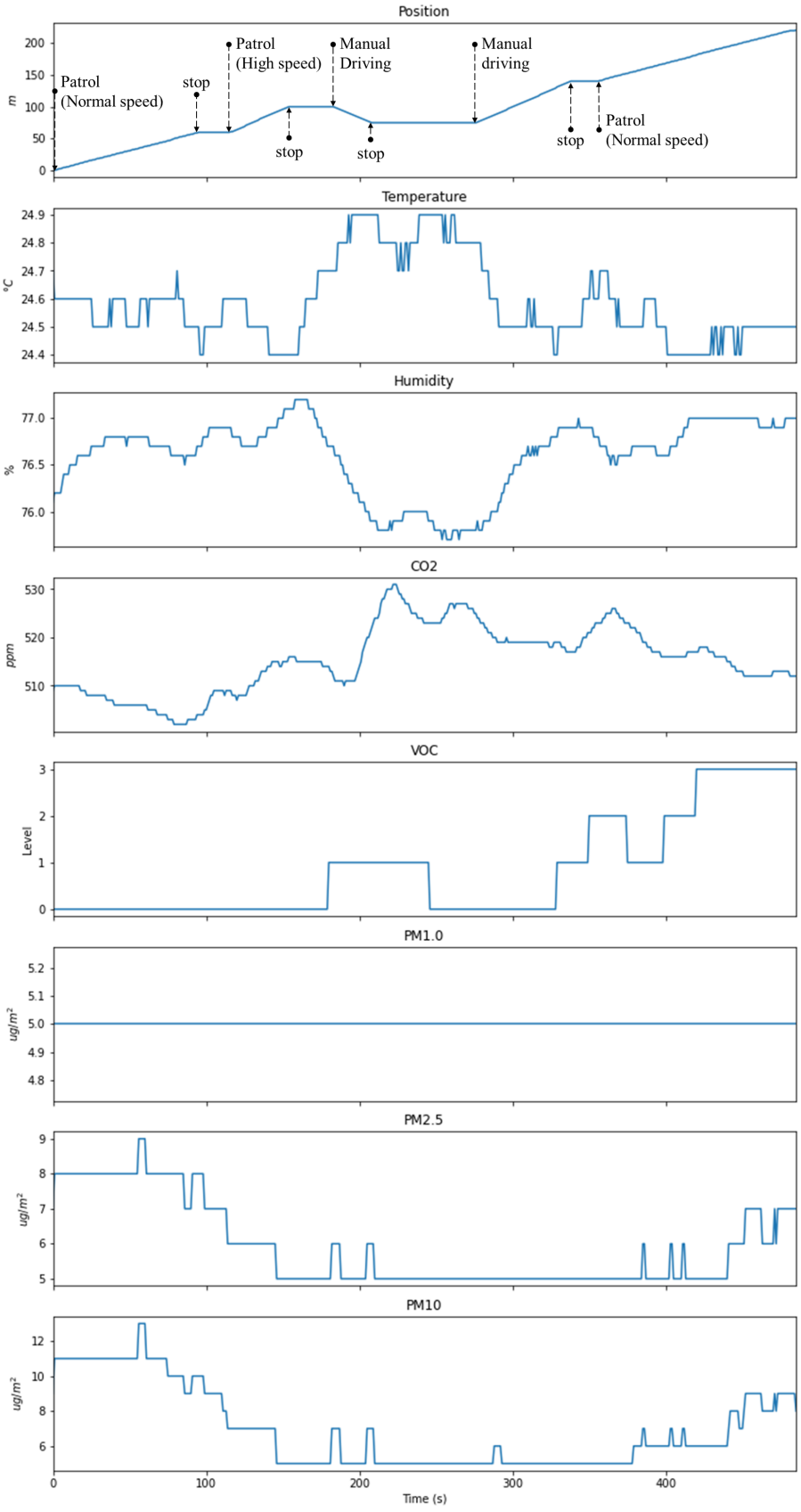


Figure 4.8: Result of system operation: Robot position, Temperature, Humidity, CO2, VOC, PM1.0, PM2.5, PM10

Chapter 5

Adaptive patrol speed control scheme based on DDPG algorithm

In this study, rail-guided patrol robot systems are particularly focused on, which have been practically deployed for large-scale monitoring in industrial and public facilities. Due to their predictable motion, simplified localization, and safe separation from pedestrians and equipment, rail-guided platforms provide a reliable infrastructure for continuous inspection tasks [7, 24, 25].

Typical deployments integrate RGB and thermal cameras with environmental sensors, stream multimodal data to remote monitoring centers, and operate along pre-defined routes at one or a few fixed speeds [2, 26–28]. These systems have demonstrated effectiveness in applications such as environmental sensing, thermal anomaly detection, and security surveillance, where scheduled patrolling and remote situational awareness are essential.

However, two recurring limitations motivate the development of the proposed framework. First, *non-adaptive speed control*: existing systems generally employ constant or manually selected patrol velocities, which may lead to inefficiencies when environmental complexity varies along the route. Second, *limited transparency*: when machine learning components are utilized (e.g., for anomaly detection or event triage), explanations for operator-facing decisions are rarely provided, potentially reducing trust and hindering timely intervention in safety-critical contexts.

To address these limitations, this work introduces an adaptive and explainable control

framework for rail-guided patrol robots, leveraging deep reinforcement learning (DRL) for dynamic speed regulation and explainable AI (XAI) for transparent decision support.

In order to effectively monitor the situation of the target facility, a control model is proposed based on the Deep Deterministic Policy Gradient (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 DPG (Deterministic Policy Gradient) algorithm [29]. 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.

5.1 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:

$$J = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \tag{5.1}$$

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:

$$reward = -\left| E_d - E_f\left(\frac{v}{fps \times w_{FOV}}\right)\right| \tag{5.2}$$

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 this 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, the Deep Deterministic Policy Gradient (DDPG) algorithm is adopted, 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.

5.1.1 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

$$\max_{\theta^\mu} J(\theta^\mu) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{5.3}$$

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

$$\nabla_\theta J = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)\, Q^\pi(s,a)]\,, \tag{5.4}$$

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

$$\nabla_{\theta^\mu} J = \mathbb{E}[\nabla_a Q_{\theta^Q}(s,a) \nabla_{\theta^\mu} \mu_{\theta^\mu}(s)]\,, \tag{5.5}$$

which corresponds to the update rule in Eq. 5.8. 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 6.1 (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-net–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 $\mathcal{D} = (\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ 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 the proposed framework.

5.2 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 (Figure 5.1).

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. 5.6 is used, and at this time, the time difference target (TD-Target) of Eq. 5.7 is affected and changed.

$$L = \frac{1}{N} \sum_{t} \{Q(\mathbf{s}_t, a_t; \theta)\}^2 \tag{5.6}$$

$$y = r + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a'; \theta^-) \tag{5.7}$$

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 the proposed controller does not incorporate the full twin-critic architecture used in TD3, several stabilization strategies are employed to 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. 5.8.

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_t \left[\nabla_a Q(\mathbf{s}, a; \theta^Q)|_{\mathbf{s}=\mathbf{s}_t, a=\mu(\mathbf{s}_t|\theta^\mu)} \nabla_{\theta^\mu} \mu(\mathbf{s}; \theta^\mu)|_{\mathbf{s}=\mathbf{s}_t} \right] \tag{5.8}$$

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:

$$\theta^{Q-} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q-} \tag{5.9}$$

$$\theta^{\mu-} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu-} \tag{5.10}$$

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 resource-constrained on-board processors. 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.

5.3 Integration of Grad-CAM Algorithm with the DDPG Model

5.3.1 Grad-CAM-Based Policy Interpretability

To enhance the explainability of the trained deep reinforcement learning model, a visual explanation module is integrated 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.

Grad-CAM for policy explanation is adopted because its gradient-based activation maps naturally extend to continuous control outputs, including a regression-style patrol-speed action. Unlike perturbation-based XAI methods such as LIME or SHAP, which require numerous input perturbations and incur substantial computational cost [30, 31], 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 the 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 Convolutional Neural Network (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 Figure 5.1, 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:

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial a^c}{\partial A_{ij}^k} \tag{5.11}$$

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:

$$\mathbf{L}^c = \sum_k w_k^c \cdot \mathbf{A}^k \tag{5.12}$$

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, this system supplements the non-explainable DDPG policy with actionable visual explanations, thereby improving operational transparency, supporting trust in autonomous decision-making, and aiding in model debugging during deployment.

5.3.2 Attention-Guided Input Perturbation Analysis

To further examine how the Grad-CAM visualizations relate to the input sensitivity of the learned patrol speed control policy, an attention-guided input perturbation analysis is conducted. The purpose of this analysis is not to validate the causal correctness or faithfulness of the explanations, but to provide qualitative interpretive evidence regarding

whether the image regions emphasized by Grad-CAM correspond to visually influential inputs for speed decision-making.

Given an input image and its corresponding Grad-CAM attention map, four types of perturbed inputs are constructed for comparison. These include the original image, an image in which blurring is applied to regions with high attention scores, an image in which blurring is applied to regions with low attention scores, and an image in which blurring is applied to randomly selected regions of identical spatial extent. The random-region perturbation is introduced as a control condition to distinguish attention-guided effects from arbitrary degradation of visual information. All perturbations are applied in a consistent manner to ensure a fair comparison across different input conditions.

For each perturbed input, the corresponding patrol speed output is generated by the DDPG actor network and compared across conditions. When visually salient regions identified by Grad-CAM are blurred, the resulting speed outputs tend to exhibit larger variations compared to cases in which low-attention regions or randomly selected regions are perturbed. This tendency is observed consistently across representative images collected from both simulation and real-world testbed environments, indicating that the learned policy is relatively more sensitive to perturbations in visually emphasized regions.

These observations suggest a qualitative alignment between the regions highlighted by Grad-CAM and the input sensitivity characteristics of the learned policy. Rather than implying causal attribution or faithful explanation of the internal decision-making mechanism, the perturbation analysis serves as an interpretability-oriented tool that supports the qualitative understanding of how visual information is reflected in patrol speed control decisions. Accordingly, the results of this analysis are intended to complement the Grad-CAM-based policy visualization, and to provide additional interpretive context for the experimental evaluations presented in Chapter 6. Additional implementation details, representative examples of the attention-guided input perturbation procedure, and a supplementary quantitative summary of output variance are provided in Appendix A to support the interpretability analysis.

The interpretability analyses presented in this section focus on understanding how visual inputs influence patrol speed decisions within the learned policy. In the following section, the attention is shifted to sim-to-real deployment, where visual domain discrep-

Table 5.1: Example patrol speed outputs generated by the DDPG actor network under different input perturbation conditions. The values are shown for illustrative purposes to highlight relative variations in policy response, rather than to provide a statistical or performance-based comparison.

Input condition	**Output speed (m/s)**
Original image	0.598
High-attention region blur	0.604
Low-attention region blur	0.562
Random-region blur	0.588

ancies between simulation and real environments are addressed through CycleGAN-based image translation.

5.4 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, a CycleGAN-based visual domain adaptation strategy is employed to enable the trained policy to generalize to real-world conditions without requiring any additional retraining.

To bridge this gap, CycleGAN [23] is adopted for unpaired image-to-image translation. A broader view of sim-to-real challenges is provided in [21]. 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 5.3 illustrates the training architecture of the CycleGAN model used in this framework.

Once training is complete, only the generator G is retained for deployment. As depicted in Figure 5.4, 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 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, Grad-CAM is additionally applied 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.

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.

In the following chapter, the evaluation results obtained in both simulation and real-world scenarios are presented 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.

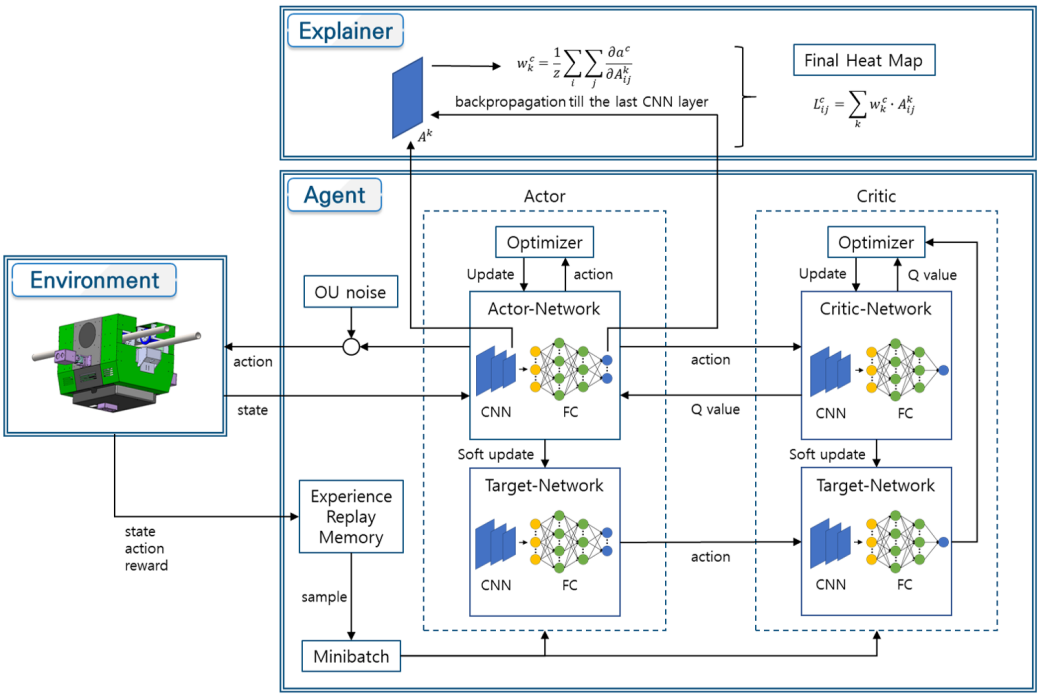


Figure 5.1: Patrol speed control model using DDPG algorithm and Grad-CAM algorithm

(a) (b) (c) (d) (e)

Figure 5.2: Examples of attention-guided input perturbations used for interpretability analysis. From left to right: (a) the original input image, (b) the corresponding Grad-CAM attention map, (c) the image with blurring applied to high-attention regions, (d) the image with blurring applied to low-attention regions, and (e) the image with blurring applied to randomly selected regions of identical spatial extent. These examples are presented to illustrate the perturbation design and do not represent a performance evaluation.

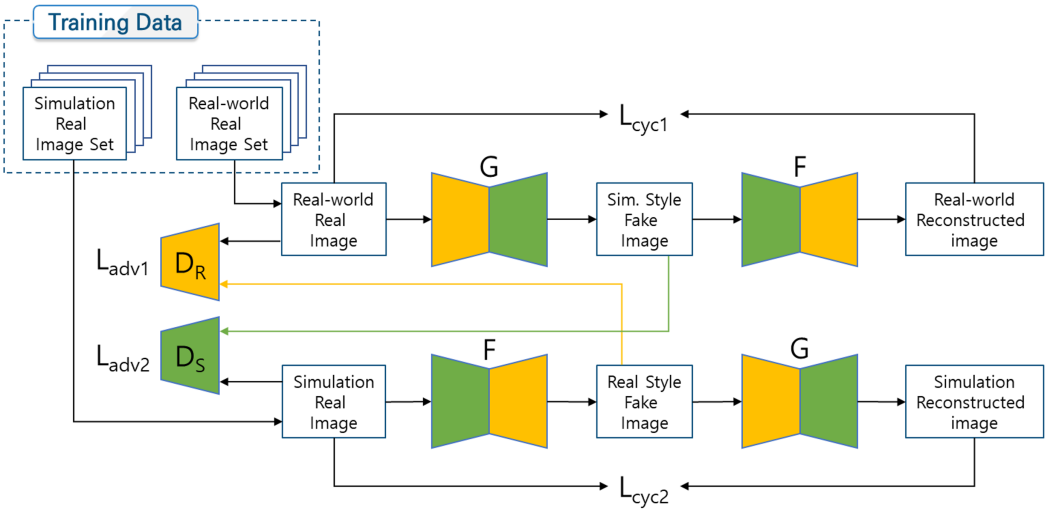


Figure 5.3: CycleGAN architecture for visual domain adaptation.

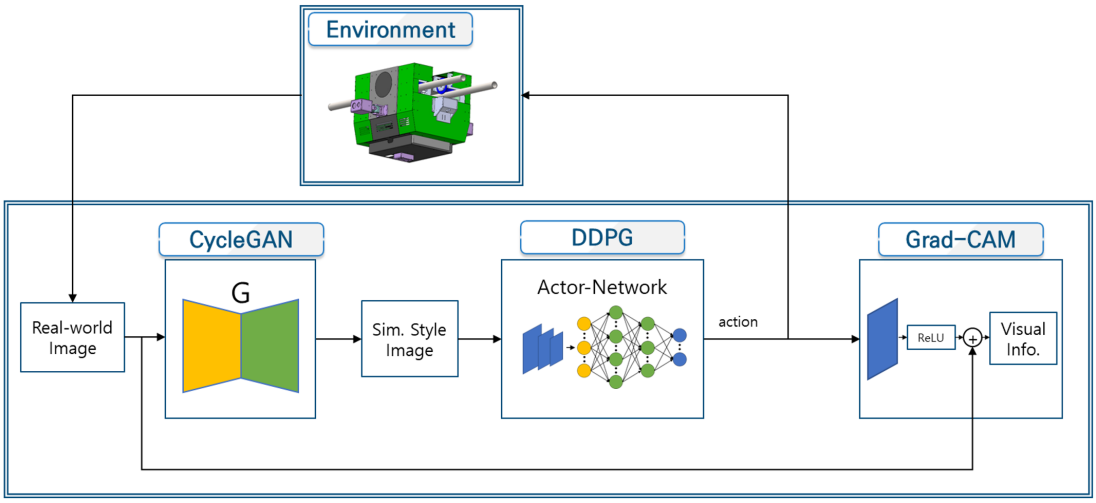


Figure 5.4: Real-world deployment using generator G, DDPG actor and Grad-CAM.

Chapter 6

Evaluation

To assess the effectiveness of the proposed patrol speed control framework, a series of experiments is conducted in both simulation and real-world environments. In addition to evaluating learning-based behaviors, the structural validity of the ceiling-mounted rail was first assessed 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. The effectiveness of CycleGAN-based domain adaptation for transferring the learned policy to real-world scenarios without retraining is assessed through the use of visual explanations as an offline diagnostic tool.

6.1 Simulation-Based Experiments

6.1.1 Simulation Environment Setup

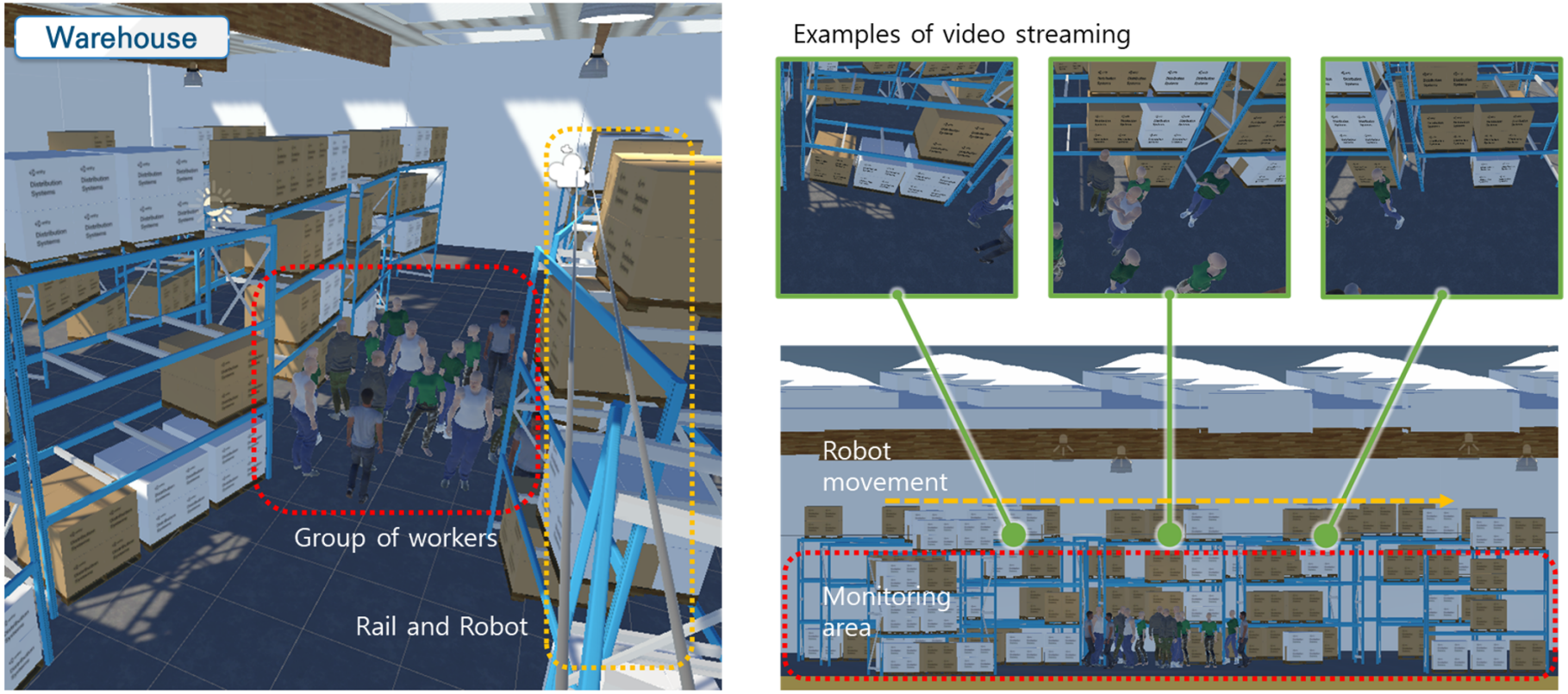


Figure 6.1: Simulation environment.

To train and evaluate the proposed DDPG-based patrol speed controller, a virtual warehouse environment is constructed using Unity, as illustrated in Figure 6.1. 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) [32–34].

The patrol robot moves along a straight ceiling-mounted rail located 4 meters 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 4097 features: a 4096-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 2049 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 6.1 were initially chosen based on commonly adopted DDPG configurations and further verified through a small-scale sensitivity analysis. For each parameter, a range of candidate values was examined (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.

Table 6.1: 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

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

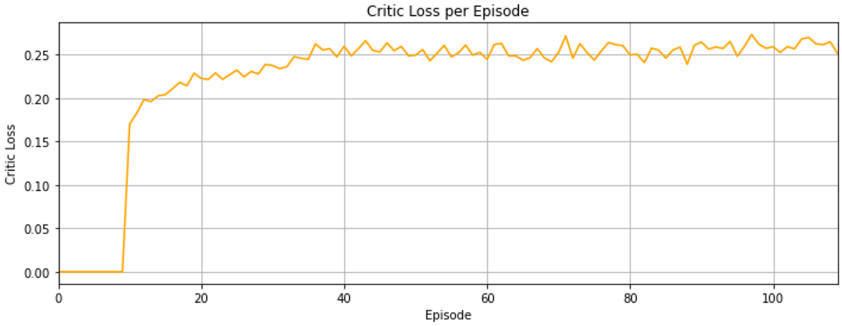
6.1.2 Learning Performance of DDPG Controller

To evaluate the training dynamics of the proposed DDPG-based patrol speed controller, key metrics are monitored 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.

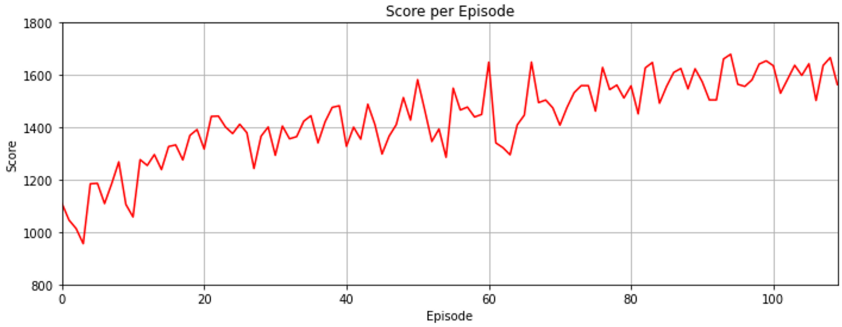
As shown in Figure 6.2a, the actor loss started from 0 and quickly decreased, stabilizing around -10 after approximately 50 episodes. This indicates that the actor network successfully converged to a policy that consistently outputs effective patrol speeds. Figure 6.2b shows the critic loss, which increased 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 Figure 6.2d, 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 Figure 6.2c, gradually improved over the course of training. It started around 1100 and slowly increased with oscillations, eventually reaching approximately 1600. This upward trend indicates the increasing effectiveness of the policy in acquiring high-entropy visual information while maintaining efficient patrol movement. Overall, these results confirm that the DDPG framework successfully learned a stable and performant control policy within 110 episodes.

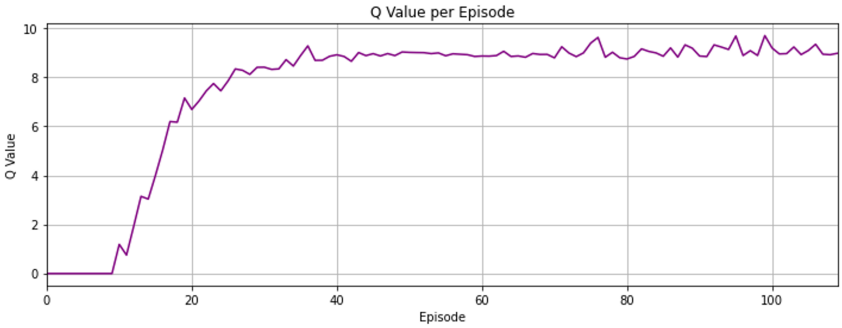
(a) Actor loss



(b) Critic loss



(c) Score



(d) Q-value

Figure 6.2: Learning performance of DDPG: Actor/Critic loss, Score, Q-value

6.1.3 Baseline Comparison with Fixed-Speed Driving

The benefit of adaptive speed control is quantitatively evaluated by comparing the proposed DDPG policy with fixed-speed baselines under an identical 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.

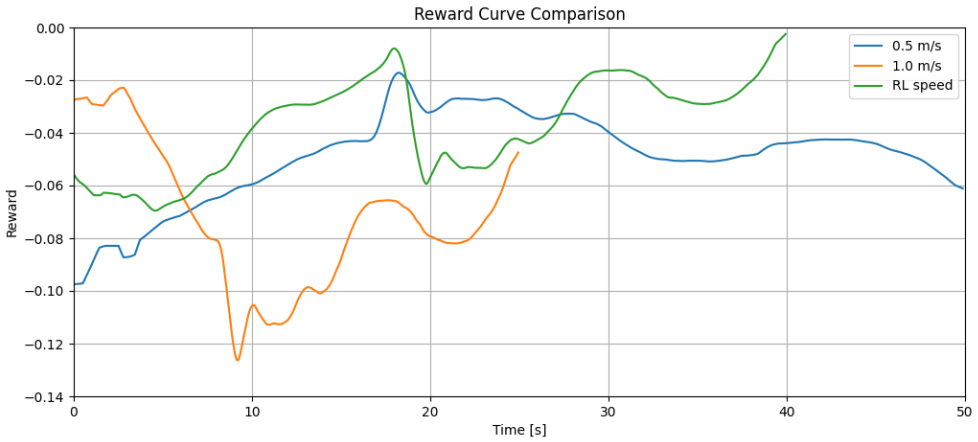
Figure 6.3a shows the per-step reward over time for the three speed profiles. The fixed-speed trajectories exhibit larger deviations from the desired entropy target. The 0.5 m/s case yields consistently low reward due to insufficient newly observed area per frame, while the 1.0 m/s case frequently overshoots 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 6.3b 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 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.

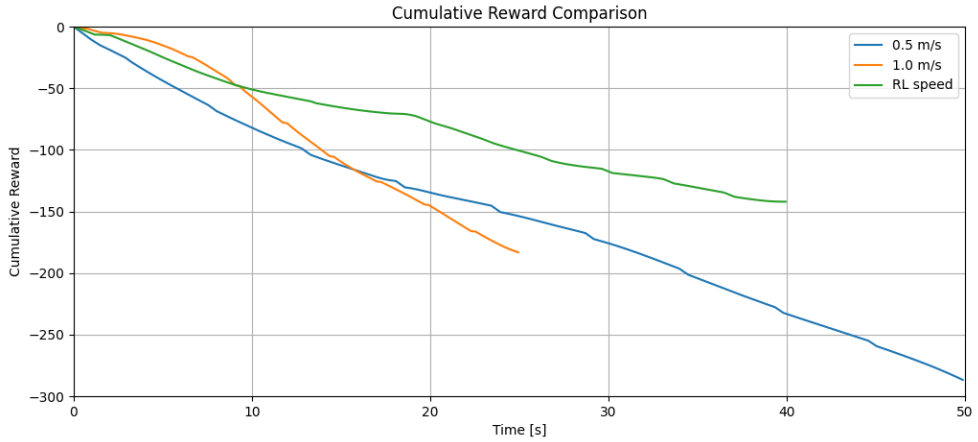
These results confirm that the proposed adaptive-speed controller outperforms fixed-speed driving in both instantaneous and cumulative reward. The learned policy effectively balances motion and information gain, validating the design of the entropy-based reward function and demonstrating the advantage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

6.1.4 Simulation-Based Evaluation with Visual Explanation

To evaluate the learned patrol speed control policy in a controlled setting, simulation-based tests are first conducted 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



(a) Per-step reward comparison



(b) Cumulative reward comparison

Figure 6.3: Comparison of fixed-speed (0.5 m/s and 1.0 m/s) and DDPG-based adaptive-speed policies. (a) Per-step reward curves. (b) Cumulative reward curves.

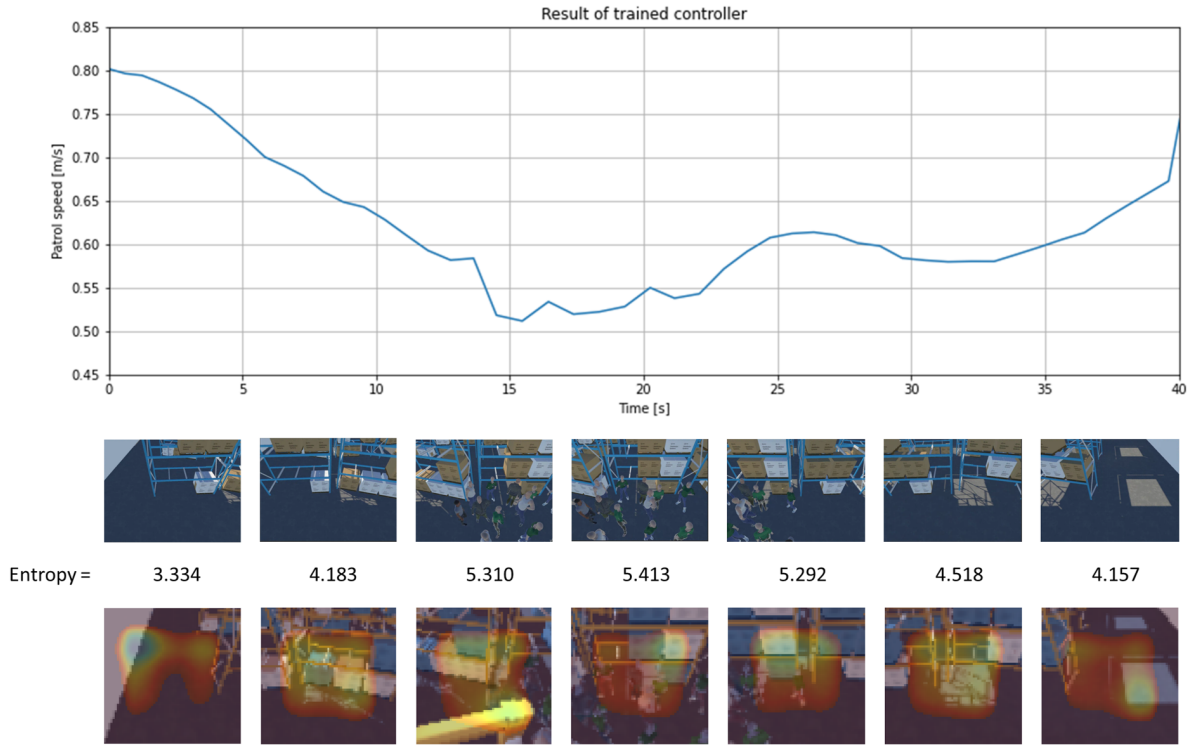
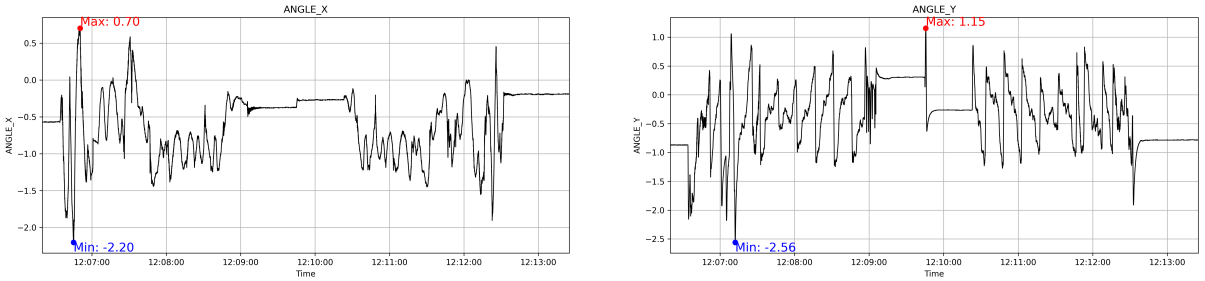


Figure 6.4: Simulation results of the trained patrol speed controller. (Top) input images and corresponding speeds at seven locations. (Bottom) Grad-CAM visualizations

to varying visual complexity. It is also examined 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 6.4 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 7 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-



(a) Roll angle (ANGLE_X) (b) Pitch angle (ANGLE_Y)

Figure 6.5: Rail inclination measurement results during constant-speed driving. (a) Roll angle; (b) Pitch angle.

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 are evaluated next.

6.2 Real-World Experiments

Rail Inclination Analysis

Before conducting adaptive speed control experiments, the inclination of the ceiling-mounted rail was evaluated to verify that structural misalignment would not affect the robot's movement or visual perception. Figure 6.5 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 analysis of the DDPG controller is not confounded by unintended geometric bias or irregularities in the rail itself.

6.2.1 Real-World Test Environment Setup

To evaluate the proposed patrol speed control framework under real-world conditions, a physical test environment is established within a laboratory testbed. A ceiling-mounted rail structure, installed 3 meters above the floor and extending 8 meters 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. Detailed descriptions of the robot platforms, sensing configurations, and deployment environment are provided in Appendix B.

During each patrol run, the robot's onboard camera continuously recorded images of the environment. Each run lasted 9 seconds, 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.

6.2.2 Real-World Deployment Results with Domain Adaptation

With the real-world test environment and image dataset in place, the trained DDPG controller is deployed in combination with the CycleGAN-based domain adaptation module to evaluate the system's performance in a real-world 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 6.6 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.

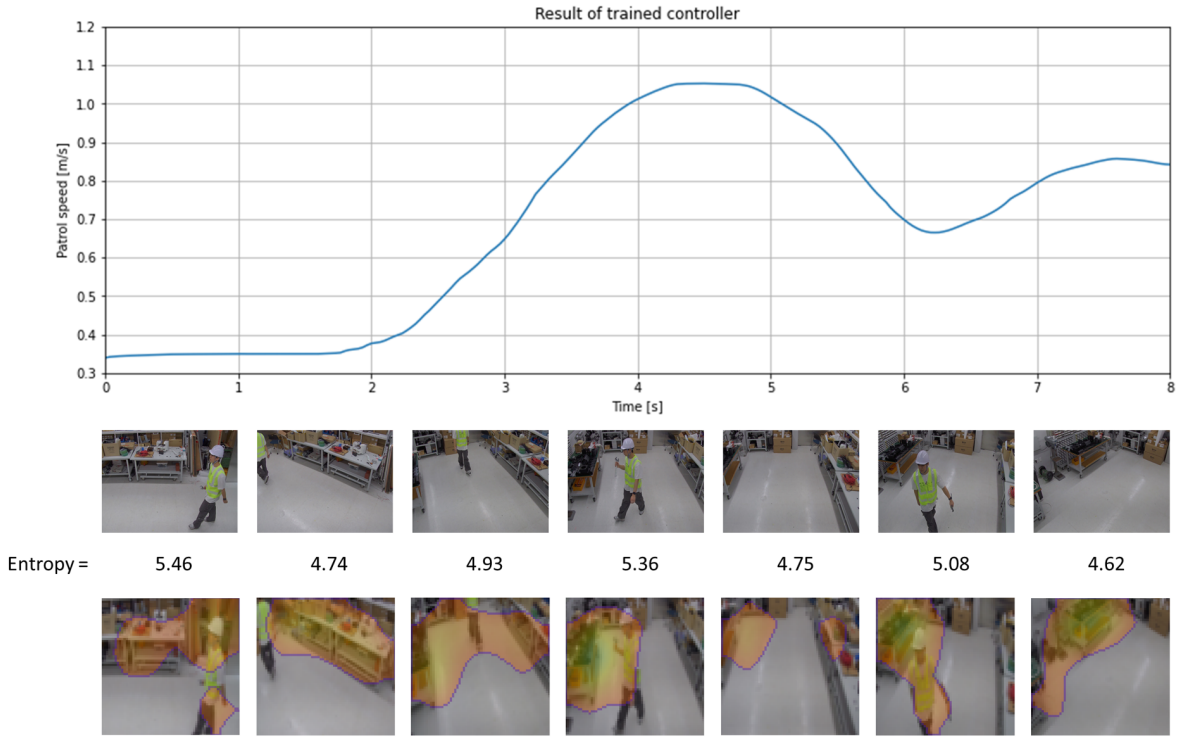


Figure 6.6: Experimental results of the trained patrol speed controller. (Top) input images and corresponding speeds at seven locations. (Bottom) Grad-CAM visualizations

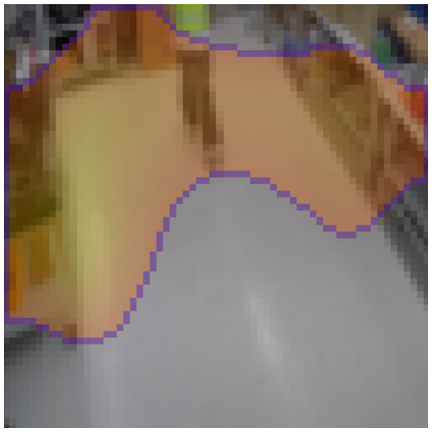
In the first and second sections of the patrol path, the robot maintained a low patrol speed. The corresponding Grad-CAM maps show strong activation around workers and visually cluttered regions in the background, indicating that the model correctly identified areas of high surveillance priority. In 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 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, Grad-CAM visualizations are compared using real-world and translated simulation-style images as inputs. As shown in Figure 6.7, the Grad-CAM result on the real-world image (c) is diffuse and poorly aligned with key semantic elements, whereas

(a) Real-world image

(b) Sim.-style image

(c) Grad-CAM on (a)



(d) Grad-CAM on (b)

Figure 6.7: 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.

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

6.3 Quantitative Evaluation of CycleGAN-based Sim-to-Real Translation

To quantify the effectiveness of CycleGAN in reducing the visual domain gap, two metrics relevant to downstream reinforcement learning—Kernel Inception Distance (KID) and Structural Similarity Index (SSIM)—were evaluated.

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.

As summarized in Figure 6.8, CycleGAN consistently improves both structural similarity and feature-level alignment compared to the direct simulation-to-real domain.

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

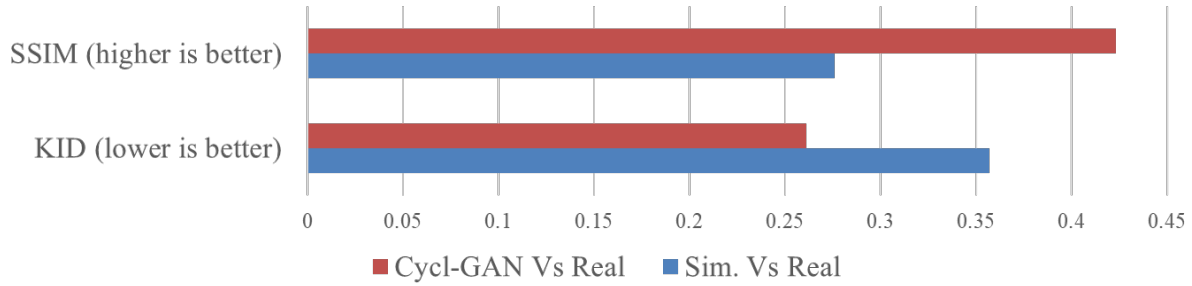


Figure 6.8: Quantitative comparison of visual domain alignment between simulation and real images. CycleGAN translation improves structural similarity (SSIM) and reduces distributional discrepancy (KID) compared to direct simulation-to-real comparison.

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 improved data efficiency by enabling zero-shot policy transfer, in the sense that the DDPG controller trained entirely in simulation can be executed on real images after translation without additional real-world data collection. As shown in Figure 6.7, directly applying the policy to real images without CycleGAN results in weaker semantic alignment in the Grad-CAM maps, whereas translated images preserve task-relevant visual structures. This comparison suggests that the integration of neural network–based visual translation facilitates sample-efficient deployment by reducing the need for on-site data gathering or retraining. The detailed computation procedures for the KID and SSIM metrics used in this evaluation are described in Appendix A.

6.4 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 the proposed approach in dynamic monitoring scenarios, setting the foundation for broader deployment in real-world robotic applications.

Chapter 7

Conclusion

This dissertation 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, the Deep Deterministic Policy Gradient (DDPG) algorithm is applied 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. The proposed system is validated 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 are identified 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 is identified 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.

Appendix A

Additional Analysis and Implementation Details

This appendix provides supplementary analyses and implementation-level details to support the interpretability discussion presented in Chapter 5. The materials included here are intended to improve transparency and reproducibility, without introducing new experimental claims or performance evaluations beyond those discussed in the main chapters.

A.1 Details of Attention-Guided Input Perturbation

This section describes the procedure used to generate attention-guided input perturbations for the interpretability analysis presented in Chapter 5. The purpose of this description is to clarify how the perturbed inputs were constructed and to support reproducibility, rather than to introduce additional experimental results.

Given an input RGB image and its corresponding Grad-CAM attention map obtained from the trained DDPG actor network, perturbations were applied by selectively blurring specific image regions. The Grad-CAM heatmap was first normalized to the range $[0, 1]$ and resized to match the spatial resolution of the input image.

Based on the normalized attention values, two complementary spatial masks were constructed. The *high-attention mask* was defined as the set of pixels whose Grad-CAM values exceeded a predefined percentile threshold, representing regions strongly associated with the patrol speed output according to the Grad-CAM visualization. Conversely,

the *low-attention mask* was defined by applying the same thresholding procedure to the lower tail of the attention value distribution, capturing regions with minimal Grad-CAM activation.

To provide a control condition, a *random mask* was generated for each image. The random mask was constructed by randomly selecting pixel locations such that the total masked area matched that of the corresponding high-attention mask. This design ensures that differences in policy response cannot be attributed solely to the size of the perturbed region.

For each mask type, a blurring operation was applied to the selected regions while leaving the remaining pixels unchanged. Gaussian blur was used as the perturbation operator to attenuate high-frequency visual information without introducing artificial structures. The same blur parameters were applied consistently across all images and perturbation conditions.

Throughout the perturbation analysis, the current speed input to the DDPG actor network was fixed to a nominal reference value. This design choice isolates the effect of visual input perturbations on the patrol speed output and avoids confounding influences from the speed state variable.

The perturbation procedure described above was applied consistently to both simulation images and real-world testbed images. Representative examples of the resulting perturbed inputs are shown in Figures A.1 and A.2, respectively.

A.2 Variance Analysis under Attention-Guided Input Perturbations

This section presents an auxiliary variance analysis of patrol speed outputs under different input perturbation conditions. The analysis is provided to complement the interpretability-oriented discussion in Chapter 5 by offering quantitative context regarding output sensitivity. The reported variance values are not used as performance metrics and do not constitute an evaluation of control quality.

Figure A.1: Attention-guided input perturbation examples for simulation images. For each row, (a) the original input image, (b) the corresponding Grad-CAM attention map, (c) the image with blurring applied to high-attention regions, (d) the image with blurring applied to low-attention regions, and (e) the image with blurring applied to randomly selected regions of identical spatial extent are shown. This figure illustrates the perturbation design used for the interpretability analysis.

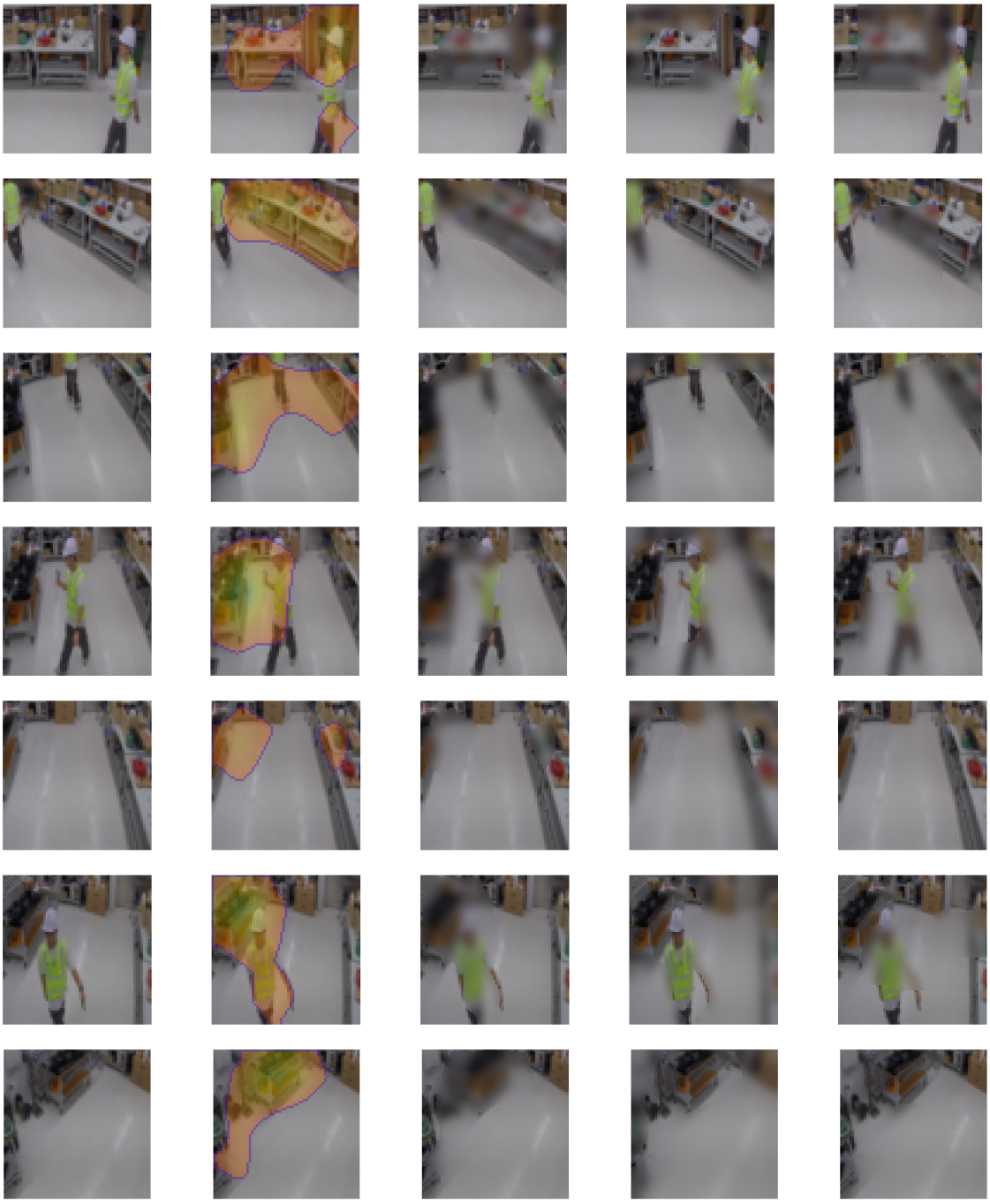


Figure A.2: Attention-guided input perturbation examples for real-world testbed images. For each row, (a) the original input image, (b) the Grad-CAM attention map, (c) the high-attention region blur, (d) the low-attention region blur, and (e) the random-region blur are shown in the same order as the simulation examples. This figure demonstrates the application of the same perturbation procedure to real-world visual inputs.

Table A.1: Output variance of patrol speed predictions under different input perturbation conditions for simulation images. The variance was computed across seven images for each condition and is reported to illustrate relative differences in output sensitivity.

Input condition	**Output variance**
Original image	0.00806
High-attention region blur	0.00746
Low-attention region blur	0.01014
Random-region blur	0.00785

A.2.1 Simulation Images

For the simulation environment, seven representative images were selected. For each image, four input conditions were constructed: the original image, high-attention region blur, low-attention region blur, and random-region blur. The current speed input to the DDPG actor network was fixed across all conditions.

Patrol speed outputs were obtained for each perturbation condition, and the variance of the outputs was computed across the set of seven images. The variance values were calculated as sample variances, normalized by $(n-1)$. Table A.1 summarizes the resulting output variances for the simulation images.

As shown in Table A.1, the output variance differs depending on which image regions are perturbed. Perturbations applied to low-attention regions tend to result in higher variance, whereas perturbations applied to high-attention regions tend to reduce the variance. The variance observed under random-region perturbations remains comparable to that of the original inputs, suggesting that the observed differences are not solely attributable to generic image degradation.

A.2.2 Real-World Testbed Images

A corresponding analysis was conducted using seven images captured from the real-world evaluation testbeds. The same perturbation procedure and fixed speed input condition were applied to ensure consistency with the simulation analysis. Due to environmental factors such as illumination changes and scene variability, the absolute range of patrol speed outputs in the testbed images is broader than that observed in simulation.

Table A.2 reports the output variance values computed across the seven real-world images for each perturbation condition.

Table A.2: Output variance of patrol speed predictions under different input perturbation conditions for real-world testbed images. The variance was computed across seven images for each condition and is reported to illustrate relative differences in output sensitivity.

Input condition	**Output variance**
Original image	0.01769
High-attention region blur	0.01367
Low-attention region blur	0.02775
Random-region blur	0.01852

Similar to the simulation results, perturbations applied to low-attention regions in real-world images result in higher output variance, while perturbations applied to high-attention regions tend to reduce the variance. Although the absolute variance values are larger than those observed in simulation, likely due to increased environmental variability, the relative sensitivity pattern remains consistent across both environments.

Overall, the auxiliary analyses presented in this appendix indicate a qualitative alignment between the Grad-CAM-identified attention regions and the output sensitivity characteristics of the learned patrol speed policy. These results support the interpretability analysis in Chapter 5 by providing additional quantitative context, without implying causal attribution, faithful explanation, or performance evaluation.

A.3 Computation of Quantitative Measures for Sim-to-Real Translation

This appendix describes the computation procedures for the quantitative measures used to compare visual similarity between image domains in the CycleGAN-based sim-to-real translation experiments. All computations follow the implementation used in our evaluation scripts.

A.3.1 Kernel Inception Distance (KID)

Kernel Inception Distance (KID) is computed as the squared Maximum Mean Discrepancy (MMD) between two sets of images in a deep feature space. Given two image sets $\mathcal{X} = \{x_i\}_{i=1}^{m}$ and $\mathcal{Y} = \{y_j\}_{j=1}^{n}$, each image is mapped to a feature vector using a pretrained

Inception-v3 network:

$$f(x) \in \mathbb{R}^d, \quad f(y) \in \mathbb{R}^d. \tag{A.1}$$

In our implementation, Inception-v3 pretrained on ImageNet is used as a feature extractor by removing the final classifier layer (i.e., replacing the fully-connected layer with an identity mapping), and the resulting penultimate features are used (dimension d corresponds to the Inception-v3 feature dimension).

Preprocessing and feature extraction. Each RGB image is resized to 299×299 and converted to a tensor before being forwarded through the Inception-v3 feature extractor. No additional normalization (e.g., mean/standard deviation normalization or input rescaling beyond tensor conversion) is applied. Let $F_X \in \mathbb{R}^{m \times d}$ and $F_Y \in \mathbb{R}^{n \times d}$ denote the resulting feature matrices.

Polynomial-kernel MMD. KID is computed using a third-degree polynomial kernel:

$$k(a, b) = (\gamma\, a^\top b + c_0)^p, \tag{A.2}$$

where $p = 3$, $c_0 = 1$, and $\gamma = 1/d$.

With this kernel, the squared MMD is estimated as:

$$\widehat{\mathrm{MMD}}^2(F_X, F_Y) = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{i'=1}^{m} k(f(x_i), f(x_{i'})) + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{j'=1}^{n} k(f(y_j), f(y_{j'})) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(f(x_i), f(y_j)). \tag{A.3}$$

Equation (A.3) corresponds to the mean-of-kernel-matrices form used in the implementation, i.e., it includes diagonal terms in the within-domain sums. Unlike common KID practice that averages an unbiased MMD estimator over multiple random subsets, our implementation computes the above estimate once using all available features in each folder.

Evaluation pairs. KID is reported for two comparisons: (i) raw simulation images vs. real images, and (ii) CycleGAN-translated images vs. real images, where the folders correspond to `real_A` (raw simulation), `fake_B` (CycleGAN output), and `real_B` (real images).

A.3.2 Structural Similarity Index (SSIM)

Structural Similarity Index (SSIM) measures perceptual similarity between two images based on luminance, contrast, and structural components. Given two corresponding image patches u and v, SSIM is defined as:

$$\mathrm{SSIM}(u, v) = \frac{(2\mu_u\mu_v + C_1)(2\sigma_{uv} + C_2)}{(\mu_u^2 + \mu_v^2 + C_1)(\sigma_u^2 + \sigma_v^2 + C_2)}, \tag{A.4}$$

where μ_u, μ_v are local means, σ_u^2, σ_v^2 are local variances, and σ_{uv} is the local covariance. The stabilizing constants are typically defined as $C_1 = (K_1 L)^2$ and $C_2 = (K_2 L)^2$, where L is the pixel dynamic range.

Implementation details. SSIM is computed using `skimage.metrics.structural_similarity`. Each image is first converted to grayscale (8-bit luminance) and SSIM is computed on the resulting 2D arrays using the library default settings. The overall SSIM score between two folders is the arithmetic mean over all paired images:

$$\overline{\mathrm{SSIM}} = \frac{1}{N} \sum_{t=1}^{N} \mathrm{SSIM}(I_t^{(1)}, I_t^{(2)}), \tag{A.5}$$

where N is the number of image pairs and $I_t^{(1)}$, $I_t^{(2)}$ denote the t-th paired grayscale images from the two folders.

Image pairing. Images are paired by sorting filenames in each folder and matching them by index. Therefore, the two folders must contain the same number of images and a consistent filename ordering to ensure correct correspondence. As with KID, SSIM is reported for (i) raw simulation vs. real and (ii) CycleGAN output vs. real comparisons.

Appendix B

Experimental System Configuration

B.1 Robot Platforms

Two rail-guided patrol robot platforms were used for the real-world evaluation. Both robots share equivalent locomotion and motion control mechanisms for rail-based patrol, while differing primarily in their onboard sensing and camera configurations. This design allows the proposed framework to be examined under different visual sensing conditions without changing the underlying drive system.

B.1.1 Robot 1

Robot 1 is a compact rail-guided patrol robot designed for indoor monitoring tasks. It is equipped with multiple fixed cameras oriented toward different viewing directions, enabling simultaneous observation of lateral and downward regions beneath the rail. In addition to vision sensors, the robot integrates multiple environmental sensors for facility monitoring.

Table B.1: Specifications of Robot 1

Item	**Specification**
Size / Weight	400 mm × 470 mm × 430 mm, 18 kg
Driving speed	0.6, 1.2, 1.5 m/s (max. operation time: 1 hour)
Environmental sensors	Temperature/Humidity, CO_2, VOC, PM (1.0, 2.5, 10)
RGB cameras	3 cameras (left, right, bottom)
	1/2.9" CMOS, 1465 × 1088, FOV 60°
Thermal camera	320 × 240, −40°C to 330°C, FOV 60°
Battery	17 Ah, 0.75C

(a) (b)

Figure B.1: Rail-guided patrol robot platforms used in the experiments. The figure shows Robot 1 and Robot 2, which share equivalent rail-guided drive mechanisms but differ in their camera configurations. Robot 1 employs multiple fixed RGB cameras oriented to the left, right, and downward directions, whereas Robot 2 is equipped with a bottom-mounted pan–tilt camera module enabling flexible viewpoint control.

B.1.2 Robot 2

Robot 2 is a larger patrol robot platform designed for extended operation and enhanced sensing capability. While its rail-guided locomotion function is equivalent to that of Robot 1, Robot 2 employs a pan–tilt camera module and higher-resolution visual sensors to provide a wider and more flexible field of view.

B.2 Vision Sensor Configuration Comparison

Although both robots share equivalent rail-guided locomotion mechanisms, their camera configurations differ significantly. Robot 1 employs three fixed RGB cameras oriented to the left, right, and downward directions, whereas Robot 2 utilizes a pan–tilt camera module mounted beneath the robot to achieve flexible viewpoint control. Table B.3 summarizes the main differences.

Table B.2: Specifications of Robot 2

Item	**Specification**
Maximum driving speed	2.0 m/s
Optimized speed modes	Patrol (0.5 m/s), Manual (0.5–1.0 m/s), Emergency (2.0 m/s)
Position repeatability	≤ 5 mm
Localization	Absolute localization system for zone recognition
Turning radius	500 mm
Maximum slope	16°
Battery	Li-ion rechargeable battery with wireless charging
Charging system	150 W, 8.0–58.4 V DC
Communication	5G / LTE / Wi-Fi
Operating temperature	−30°C to 70°C
Ingress protection	IP56
Gas sensors	CH_4, CO, CO_2, H_2
Sound sensors	4-channel digital MEMS microphone
HD camera	1920 × 1080, 32× optical / 32× digital zoom
Thermal camera	640 × 480, −20°C to 130°C
Pan–tilt module	Pan: 0–360°, Tilt: 0–180°

Table B.3: Comparison of vision sensor configurations

Specification	**Robot 1**	**Robot 2**
Camera configuration	Fixed multi-camera	Pan–tilt camera module
Number of RGB cameras	3 (left, right, bottom)	1
Camera mobility	Fixed	Pan 360°, Tilt 180°
RGB resolution	1465 × 1088	1920 × 1080
Thermal camera	320 × 240	640 × 480
Field of view	60° (fixed)	Variable (pan–tilt)

B.3 Image Processing and Robot Control Server

All visual processing, policy inference, and patrol speed command generation were performed on an external processing server. The server received image streams from the robot-mounted cameras, executed the trained deep neural networks, and transmitted the resulting speed commands to the robots. No online learning or parameter updates were performed during deployment.

Table B.4: Specifications of the image processing and robot control server

Item	**Configuration for Robot 1**	**Configuration for Robot 2**
Operating system	Linux	Linux
CPU	Intel Xeon ES-1630	Dual Intel Xeon processors (8 cores total)
Memory	16 GB	64 GB TruDDR4 2666 MHz
GPU	NVIDIA RTX 2060	NVIDIA RTX 3080 (12 GB)
Storage	SSD 2 TB	SSD 512 GB + HDD 3 TB
Network	Ethernet 1 Gb (2 ports)	Ethernet 1 Gb (4 ports)

Bibliography

[1] S. Halder and K. Afsari. Robots in inspection and monitoring of buildings and infrastructure: A systematic review. *Applied Sciences*, 13(4):2304, 2023.

[2] M. O. Macaulay and M. Shafiee. Machine learning techniques for robotic and autonomous inspection of mechanical systems and civil infrastructure. *Autonomous Intelligent Systems*, 2(1):1–8, 2022.

[3] United Nations Office for Disaster Risk Reduction (UNDRR). Disaster resilience scorecard for cities: Detailed level assessment. Technical report, UNDRR, 2021.

[4] United Nations Development Programme (UNDP) and United Nations Office for Disaster Risk Reduction (UNDRR). Data and digital maturity for disaster risk reduction. Technical report, UNDP-UNDRR, 2022.

[5] United Nations Office for Disaster Risk Reduction (UNDRR). Building risk knowledge: Disaster data portal, 2023.

[6] T. Rakha and A. Gorodetsky. Review of unmanned aerial system (uas) applications in the built environment: Towards automated building inspection procedures using drones. *Automation in Construction*, 93:252–264, 2018.

[7] M. Tavakoli, C. Viegas, L. Sgrigna, and A. T. Almeida. Scala: Scalable modular rail based multi-agent robotic system for fine manipulation over large workspaces. *Journal of Intelligent and Robotic Systems*, 89:421–438, 2018.

[8] G. P. S. Carvalho, M. F. S. Xaud, I. Marcovistz, A. F. Neves, and R. R. Costa. The doris offshore robot: Recent developments and real-world demonstration results. *IFAC-PapersOnLine*, 50(1):11215–11220, 2017.

[9] H. Lee, J. Kwon, M. Shin, S. Lee, N. Y. Chong, and W. Yang. Development of rail-guided smart patrol system for surveillance and monitoring of facilities safety. In *Proceedings of the 2023 IEEE/SICE International Symposium on System Integration (SII)*, pages 1–6, 2023.

[10] Abdelfetah Hentout, Abderraouf Maoudj, and Ahmed Kouider. Shortest path planning and efficient fuzzy logic control of mobile robots in indoor static and dynamic environments. *Romanian Journal of Information Science and Technology*, 27(1):21–36, 2024.

[11] Andrei Mateescu, Dragos Constantin Popescu, Ioana-Livia Stefan, Ioana Miruna Vlasceanu, Alina Claudia Petrescu-Nita, Ioan Stefan Sacala, and I. Dumitrache. Machine learning control for assistive humanoid robots using blackbox optimization of pid loops through digital twins. *Romanian Journal of Information Science and Technology*, 28(1):63–76, 2025.

[12] Geeta Yadav and Kolin Paul. Architecture and security of scada systems: A review. *arXiv preprint arXiv:2001.02925*, 2020.

[13] W. F. Cheung, T. H. Lin, and Y. C. Lin. A real-time construction safety monitoring system for enhancing workers' safety. *Sensors*, 18(2):436, 2018.

[14] Zhaoyi Xu and Joseph Homer Saleh. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *arXiv preprint arXiv:2008.08221*, 2020.

[15] A. Haleem. Encouraging safety 4.0 to enhance industrial culture. *Discover Sustainability*, 5(1):85, 2024.

[16] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Andrés Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): A systematic review. *arXiv preprint arXiv:2006.00093*, 2020.

[17] Leilani H. Gilpin, David Bau, Ben Yuan, Arun Bajwa, Michael Specter, and Lalana Kagal. Classification of explainable artificial intelligence methods through their output formats. *Informatics*, 8(3):30, 2021.

[18] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[19] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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 (KDD)*, pages 1135–1144. ACM, 2016.

[20] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 4765–4774, 2017.

[21] Stephen James and Andrew J Davison. Sim-to-real via simulated+unsupervised (s+u) learning with domain randomization. *arXiv preprint arXiv:1903.01458*, 2019.

[22] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 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)*. IEEE, 2017.

[23] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[24] M. Bengel, K. Pfeiffer, B. Graf, et al. Mobile robots for offshore inspection and manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3317–3322, 2009.

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

[26] G. Carvalho, G. Freitas, et al. Doris—a mobile robot for inspection and monitoring of offshore facilities. In *Proceedings of the Congresso Brasileiro de Automática*, 2014.

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

[28] K. Cho and D.-H. Lee. Design and position control of overhang-type rail mover using dual blac motor. *Energies*, 14(1000), 2021.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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 (KDD)*, pages 1135–1144. ACM, 2016.

[31] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[32] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2020.

[33] Unity-Technologies. Robotics-warehouse. `https://github.com/Unity-Technologies/Robotics-Warehouse`, 2022. Accessed: 2023-03-02.

[34] Unity Technologies. Uma2 - unity multipurpose avatar. `https://assetstore.unity.com/packages/3d/characters/uma-2-unity-multipurpose-avatar-35611`, 2022. Accessed: 2023-03-02.

Publications

International Journals

[1] **Hosun Lee**, Jeasung Kwon, Nak Young Chong and Woosung Yang, "Explainable and Transferable Deep Reinforcement Learning for Adaptive Patrol of Rail-Guided Robot System", *PeerJ Computer Science (***PeerJ CS***)*, 2025 (Accepted)

International Conferences

[2] **Hosun Lee**, Jeasung Kwon, Minjun Shin, Sungon Lee, Nak Young Chong and Woosung Yang, "Development of Rail-guided Smart Patrol System for Surveillance and Monitoring of Facilities Safety", *2023 IEEE/SICE International Symposium on System Integration (***SII***)*, Atlanta, GA, USA, pp. 1-6, 2023

[3] **Hosun Lee**, Jeasung Kwon, Sungon Lee, Nak Young Chong and Woosung Yang, "Information-Based Patrol Speed Control Method for Rail-Guided Robot System Using Deep Deterministic Policy Gradient Algorithm", *Intelligent Autonomous Systems 18 (***IAS 2023***)*, Lecture Notes in Networks and Systems, vol 794, 2024

[4] **Hosun Lee**, Jeasung Kwon, Nak Young Chong and Woosung Yang, "Explainable Deep Reinforcement Learning for Patrol Speed Control of Rail-Guided Robot System", *The 12th International Conference on Robot Intelligence Technology and Applications (***RiTA 2024***)*, Ulsan, Korea, pp. 1-8, 2024