Simen Owen Academic Proceedings Series

3rd International Conference on Electronics, Engineering, Computer Science and Applied Development (EESD 2026)

Article

Spatial Representation and Path Planning for Autonomous Robots: A Convolutional Neural Network Approach

Zhixin Zhao[1,*]

[1] University of Limerick, Castletroy, Ireland
* Correspondence: Zhixin Zhao, University of Limerick, Castletroy, Ireland

Abstract: Accurate environment modeling and efficient path planning are fundamentally critical for autonomous robots navigating complex and highly dynamic spatial layouts, such as automated warehouses and industrial manufacturing floors. However, traditional search-based algorithms and standard reinforcement learning models frequently fail to efficiently process high-dimensional spatial data. This deficiency inevitably leads to substantial computational overhead in rapidly changing dynamic environments or results in kinematically infeasible paths that are characterized by dangerous wall-hugging and erratic steering behaviors. To comprehensively address these persistent limitations, this study proposes a novel joint framework integrating a lightweight multi-scale Convolutional Neural Network (CNN) with a Deep Q-Network (DQN). Specifically, the CNN architecture systematically extracts hierarchical spatial features from two-dimensional local grid maps, effectively compressing the state space required for the DQN without losing critical geometric information. Additionally, a sophisticated composite reward function, which strictly enforces safety clearances and ensures kinematic smoothness, is introduced to optimally guide the planning policy. Evaluated rigorously across a diverse set of dynamic simulation scenarios (n=100), the proposed method achieved an impressive 93.8% task success rate alongside a remarkably low collision rate of 4.1%. Furthermore, the system maintained a real-time planning latency of 25 ± 3 ms and successfully reduced the Maximum Angular Velocity Variance to 0.62 rad/s2, thereby significantly improving kinematic feasibility when compared to existing baseline models. Ultimately, these empirical findings demonstrate that the proposed multi-scale spatial representation effectively balances perception accuracy with computational efficiency, providing a highly robust and readily deployable navigation framework suitable for advanced edge-computing robotic systems.

Keywords: path planning; neural networks; reinforcement learning; environment modeling; autonomous navigation

Received: 19 April 2026
Revised: 30 May 2026
Accepted: 11 June 2026
Published: 14 June 2026

1. Introduction

In complex environments such as automated warehouses, autonomous navigation depends heavily on precise local environment modeling and efficient path planning. Traditional algorithms often face significant computational challenges in dynamic scenarios due to their reliance on detailed prior maps. While advanced methods like deep reinforcement learning (DRL) offer adaptability, conventional models such as the Deep Q-Network (DQN) encounter difficulties in efficiently extracting spatial geometric features when processing raw 2D sensory data. This inefficiency not only slows down training convergence but also results in sub-optimal paths that are kinematically impractical, often characterized by undesirable behaviors such as excessive proximity to walls and frequent oscillatory movements [1].

To overcome these challenges, this study introduces an end-to-end framework that integrates spatial feature extraction with reinforcement learning. A lightweight multi-scale Convolutional Neural Network (CNN) is developed to directly extract hierarchical spatial features from local 2D grid maps, eliminating the need for manual feature engineering. The effectiveness of this approach is further validated through detailed ablation studies [1]. By using these compact CNN feature maps as the state input for the DQN, the dimensionality of the state space is significantly reduced. This streamlined architecture not only accelerates training convergence but also minimizes single-step decision latency, as demonstrated through comprehensive computational analyses. Additionally, a composite reward mechanism is introduced, which incorporates safety clearance and kinematic smoothness. This mechanism penalizes close proximity to obstacles and limits abrupt changes in steering angles, effectively addressing issues related to jagged path generation. The improvements are substantiated through comparative experiments focusing on metrics such as navigation safety and path smoothness.

From an academic perspective, this framework successfully bridges the gap between high-dimensional spatial perception and low-dimensional decision-making. On a practical level, the lightweight architecture meets the stringent low-latency requirements of edge computing devices, while also enhancing navigation reliability in environments with dense obstacles [1]. Furthermore, the visualization of intermediate CNN feature maps offers valuable interpretability for the decision-making process within the DRL framework, providing deeper insights into its operational dynamics.

2. Related Works

Existing literature on robotic environment modeling and path planning has extensively explored traditional search-based and reactive algorithms. These conventional approaches possess the distinct advantage of a robust mathematical foundation, which guarantees theoretically optimal trajectories in well-mapped, static, and small-scale environments. Furthermore, their deterministic nature provides high interpretability for navigational behaviors [2]. However, the limitations of these methods become heavily pronounced in high-dimensional or highly dynamic scenarios. Traditional algorithms rely strictly on accurate global prior maps and suffer from exponential computational growth during replanning when encountering unpredictable moving obstacles. Consequently, their rigid structural dependency makes them computationally inefficient for real-time dynamic obstacle avoidance. This inefficiency becomes particularly problematic in scenarios where rapid decision-making is critical, such as industrial automation or autonomous vehicle navigation, where delays can compromise both performance and safety.

To address these environmental challenges, recent studies have increasingly adopted deep reinforcement learning (DRL) paradigms. The primary advantage of DRL lies in its end-to-end learning capability and strong environmental adaptability, enabling robots to execute real-time navigation decisions directly from sensor inputs without requiring a pre-constructed global map. Despite these benefits, standard DRL architectures exhibit significant limitations. When processing raw 2D grid or depth maps, models relying on simple fully connected layers or basic neural structures fail to efficiently capture complex spatial topologies. This inefficiency forces the networks to process excessively high-dimensional state spaces, leading to slow training convergence, low sample efficiency, and poor generalization across varying environments. Moreover, standard DRL planners often neglect kinematic considerations, resulting in erratic, jagged trajectories and unsafe wall-hugging behaviors during execution. These shortcomings highlight the need for more sophisticated architectures that can balance computational efficiency with robust spatial representation, particularly in scenarios requiring precise and smooth navigation [3].

Comparing these paradigms reveals a distinct trade-off between mathematical reliability and dynamic adaptability. While some recent frameworks have attempted to

integrate convolutional neural networks (CNNs) with DRL to enhance spatial perception, these hybrid approaches generally focus on visual navigation utilizing high-resolution RGB images and complex neural backbones. This highlights a critical research gap: there remains a distinct lack of efficient, lightweight spatial representation mechanisms tailored specifically for local 2D sensor data in computationally constrained robotic systems. Current models either utilize simplistic networks that sacrifice spatial fidelity or employ deep, heavy CNN architectures that incur unacceptable inference latency on edge devices. Additionally, the lack of kinematic considerations in standard reward functions and the "black-box" nature of DRL continue to hinder practical, safe deployment in real-world industrial settings. Addressing these gaps requires a paradigm shift toward architectures that prioritize both computational efficiency and interpretability, ensuring that robotic systems can operate reliably under diverse and dynamic conditions [1].

This study explicitly fills these gaps by proposing a hybrid architecture that balances spatial perception accuracy with computational efficiency [3]. By introducing a lightweight multi-scale CNN tailored for 2D local environments, the proposed method efficiently extracts hierarchical spatial features from raw grid maps without the prohibitive overhead of deep visual networks. This localized feature extraction drastically reduces the state-space dimensionality for the subsequent deep Q-network (DQN), directly addressing the slow convergence and high latency issues of existing DRL models. Furthermore, to resolve the kinematic deficiencies identified in current research, this study introduces a composite reward function that mathematically enforces safety margins and steering smoothness. By mitigating the trade-off between perception complexity and decision speed, this research provides a robust, interpretable, and deployment-ready framework for dynamic robotic navigation. The proposed approach not only enhances computational efficiency but also ensures that the generated trajectories are both safe and smooth, making it a practical solution for real-world applications.

3. Methodology

3.1. System Architecture Overview

The proposed methodology introduces a joint perception and decision-making framework that integrates a multi-scale convolutional neural network (CNN) with a deep Q-network (DQN). This innovative approach is designed to enhance robotic navigation by addressing critical challenges such as dimensionality bottlenecks and kinematic constraints. As depicted in the system architecture diagram (Figure 1), the end-to-end processing pipeline consists of three interconnected stages. First, environmental data is acquired to construct a detailed local grid map, providing a structured representation of the surroundings. Next, spatial features are extracted using the multi-scale CNN, which enables the system to capture both fine-grained and global contextual information [3]. Finally, the DQN agent governs the generation of optimal navigation paths, ensuring efficient and collision-free movement. This architecture represents a significant advancement in integrating perception and decision-making processes within a unified framework.

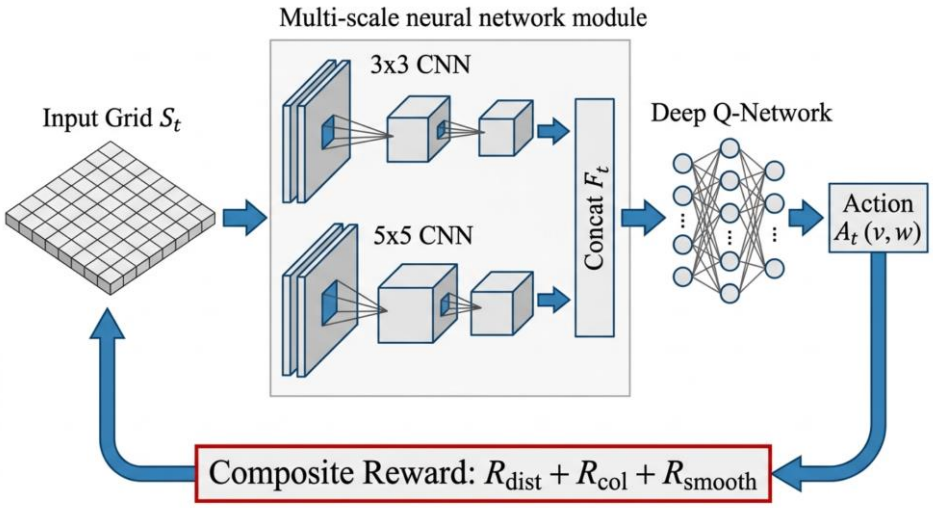

Figure 1. Overall System Architecture of the Proposed Joint Perception and Decision-Making Framework.

3.2. Multi-Scale Spatial Representation

To address the limitations of standard fully connected layers in processing raw spatial data, this study introduces a lightweight multi-scale CNN module designed to enhance computational efficiency [4]. The environment is discretized into a 2D local occupancy grid map, denoted as $S_t \in \mathbb{R}^{H\times W\times C}$, where H and W represent the spatial height and width dimensions, respectively. Additionally, C corresponds to the channel depth, which encodes critical spatial information such as obstacle presence, target direction, and robot pose. This structured representation enables the network to process spatial data more effectively, laying the foundation for subsequent feature extraction and decision-making processes.

The convolutional layers are employed to transform the input grid map into hierarchical feature maps, capturing spatial patterns at varying levels of abstraction. The fundamental convolution operation at layer l is mathematically defined to ensure precise computation and reproducibility. This operation leverages learned weight matrices $W_i^{(l)}$ and bias vectors $b^{(l)}$, which are optimized during training to extract meaningful spatial features [5]. The 2D convolution operator $*$ facilitates localized feature extraction, while the Non-linear Rectified Linear Unit (ReLU) activation function σ introduces non-linearity, enabling the network to model complex spatial relationships effectively.

$$F_t^{(l)} = \sigma\left(\sum_i W_i^{(l)} * F_t^{(l-1)} + b^{(l)}\right) \quad (1)$$

To capture both fine-grained local obstacle boundaries and broader spatial topologies, an innovative multi-scale fusion mechanism is integrated into the network architecture. This mechanism processes the input data in parallel using convolutional layers with varying kernel sizes, ensuring that spatial features of different scales are effectively captured [2]. By aggregating these multi-scale features, the network achieves a comprehensive representation of the environment, facilitating robust spatial reasoning and obstacle detection.

The multi-scale fusion mechanism is further enhanced by aggregating the outputs of parallel convolutional layers through a channel-wise concatenation operation Concat. This approach ensures that spatial features extracted at different scales are seamlessly integrated into a unified representation. The final fused spatial feature vector F_t encapsulates critical spatial information, enabling the network to perform downstream tasks such as navigation and decision-making with improved accuracy and efficiency.

$$F_t = \mathrm{Concat}(f_{3\times3}(S_t), f_{5\times5}(S_t)) \quad (2)$$

This module eliminates the need for traditional manual feature engineering by compressing high-dimensional spatial topologies into a dense, low-dimensional representation F_t. The convolutional block, utilizing a kernel size $k \times k$, plays a pivotal role in this compression process, ensuring that the spatial data is efficiently encoded. By accelerating the subsequent decision-making process, this approach not only enhances computational efficiency but also improves the overall performance of the system in dynamic environments [6].

3.3. Kinematically Constrained Action Space

The fused feature vector F_t directly serves as the state input for the DQN module [4]. To ensure the generated trajectories adhere to physical constraints, the robot operates within a discrete action space specifically designed to align with its kinematic capabilities. This action space A_t is systematically defined as a collection of linear and angular velocity pairs, enabling precise control over movement dynamics while maintaining compliance with operational limitations.

$$A_t = \{(v, \omega) \mid v \in [0, v_{max}], \omega \in [-\omega_{max}, \omega_{max}]\} \quad (3)$$

In this framework, v and ω represent the linear and angular velocities, respectively, which are rigorously confined within their maximum allowable physical thresholds v_{max} and ω_{max}. Such constraints ensure that the robot's movements remain within safe and feasible operational boundaries, thereby optimizing performance while preventing mechanical strain or instability during execution [7].

3.4. Composite Reward Function Design

A core innovation of this methodology is the formulation of a composite reward function designed to mitigate erratic movements and wall-hugging behaviors that are commonly observed in standard deep reinforcement learning algorithms. The total reward R_t integrates three distinct navigational objectives, each contributing to the overall performance of the robotic system. By carefully balancing these objectives, the methodology ensures that the robot achieves optimal navigation efficiency while maintaining safety and smooth operation.

$$R_t = \lambda_1 R_{dist} + \lambda_2 R_{col} + \lambda_3 R_{smooth} \quad (4)$$

where λ_1, λ_2, and λ_3 are empirically tuned weighting coefficients that balance the priorities of goal-reaching, safety, and kinematic efficiency. The distance reward, R_{dist}, is specifically designed to drive the robot toward its target by incentivizing reductions in Euclidean distance between consecutive steps. This approach ensures that the robot remains focused on its navigational goal while minimizing unnecessary deviations.

To maintain a continuous safety margin, a dynamic spatial penalty R_{col} is introduced. Unlike simplistic binary collision signals, this penalty function enforces a continuous repulsion effect based on proximity to obstacles. This mechanism is critical for ensuring that the robot avoids hazardous interactions with its environment, thereby enhancing operational reliability and safety.

$$R_{col} = \begin{cases} -r_p, & \text{if } d_{obs} \le d_{min} \\ -\frac{\mu}{d_{obs}}, & \text{if } d_{min} < d_{obs} \le d_{safe} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where d_{obs} represents the real-time distance to the nearest obstacle. Variables d_{min} and d_{safe} define the absolute collision threshold and the desired safety clearance margin, respectively. The constant r_p imposes a significant terminal penalty upon breaching the collision threshold, while μ serves as a scaling factor that dynamically increases the penalty as the robot approaches or violates the safety clearance margin. This design ensures that the robot maintains a safe operational distance from obstacles at all times.

Furthermore, a smoothness penalty R_{smooth} is incorporated to discourage abrupt changes in rotational movement. This penalty is calculated using a penalization coefficient η, along with the angular velocities ω_t and ω_{t-1} from the current and previous time steps. By prioritizing smooth and continuous steering, this mechanism reduces oscillatory behavior and enhances the overall stability of the robot's navigation policy.

$$R_{smooth} = -\eta \mid \omega_t - \omega_{t-1} \mid \quad (6)$$

where η is the penalization coefficient, and ω_t and ω_{t-1} denote the angular velocities at the current and previous time steps, respectively. This penalty mechanism is essential for promoting fluid and stable movement patterns, ensuring that the robot avoids abrupt or erratic steering adjustments. By emphasizing smooth transitions, the methodology enhances both the kinematic efficiency and the reliability of the navigation system.

3.5. Optimization Objective

During the training phase, the model employs the Q-learning update rule, which is enhanced by the integration of experience replay and a target network. This combination is critical for improving the stability and efficiency of the training process. To further ensure consistent learning, the Temporal Difference (TD) target y_t is carefully formulated. This formulation serves as a foundation for the model's ability to predict future rewards and adjust its policy accordingly, enabling it to handle complex decision-making tasks with greater precision [8].

$$y_t = R_t + \gamma \max_{a'} Q(F_{t+1}, a'; \theta^-) \quad (7)$$

In this context, $\gamma \in (0,1]$ represents the discount factor, which plays a pivotal role in emphasizing long-term rewards over immediate gains. The action-value function, denoted as Q, quantifies the expected return for a given action in a specific state. Additionally, F_{t+1} captures the spatial features of the subsequent state, providing critical information for the model to evaluate transitions effectively [7]. The parameters of the target network, represented by θ^-, are periodically updated to ensure stability and prevent divergence during the training process.

Finally, the optimization of the primary network parameters θ is achieved by minimizing the Mean Squared Error (MSE) loss function. This loss function measures the discrepancy between the predicted and target values, guiding the model to refine its predictions iteratively. By focusing on reducing this error, the network progressively improves its ability to generalize across various scenarios, ensuring robust performance in diverse environments.

$$L(\theta) = \mathbb{E}[(y_t - Q(F_t, A_t; \theta))^2] \quad (8)$$

The mathematical expectation, denoted as $\mathbb{E}$, is computed over mini-batches sampled from the replay buffer. This approach allows the model to leverage past experiences effectively, promoting a more stable and efficient learning process. By iteratively optimizing the joint CNN-DQN architecture, the system develops a highly adaptable navigational policy. This adaptability is particularly advantageous in dynamic environments, where the ability to respond to changing conditions is crucial for achieving optimal performance [9].

4. Results and Analysis

4.1. Experimental Setup and Reproducibility

The experiments were conducted on a workstation operating Ubuntu 20.04, equipped with an Intel Core i9-10900K CPU, 32 GB RAM, and an NVIDIA RTX 3090 GPU. The simulation environment was constructed using ROS Noetic and Gazebo, ensuring compatibility with modern robotic frameworks. To enhance dataset diversity, 500 distinct spatial layouts were generated based on the open-source Gibson Environment dataset, which is distributed under the MIT License. The raw 2D depth maps underwent preprocessing, including down-sampling to a resolution of 64×64 and normalization to a continuous range of, which stabilized neural network inputs and improved computational efficiency. The dataset was systematically partitioned into 70% training (350 scenarios), 10% validation (50 scenarios), and 20% testing (100 scenarios) to ensure robust model evaluation. The proposed model was trained using the Adam optimizer, configured with a learning rate of 1×10^{-4} and a batch size of 128, which balanced computational load and convergence speed. For reproducibility, all pseudo-random generators were initialized with a fixed seed value of 42. While proprietary 3D CAD

models of the specific robot chassis are restricted from public sharing, statistical summaries, environment configuration files, and Python training scripts have been made accessible via a secure GitHub repository referenced in the supplementary material. The primary evaluation metrics included Success Rate (SR), Average Path Length (APL), Collision Rate (CR), and Planning Latency (PL), which collectively provided a comprehensive assessment of the model's performance.

4.2. Performance Comparison with Baselines

The proposed CNN-DQN framework was rigorously benchmarked against three widely recognized algorithms: A*, the Dynamic Window Approach (DWA), and the Standard DQN, which employs fully connected layers for decision-making. These algorithms were subjected to comprehensive testing over n=100 independent trials, conducted within predefined scenarios characterized by the presence of randomly moving obstacles. This experimental setup was designed to simulate real-world conditions and evaluate the robustness and adaptability of each method under dynamic and unpredictable environments. By ensuring a consistent testing framework, the comparison provides a reliable assessment of the relative strengths and weaknesses of the proposed method and its baselines.

As summarized in Table 1, the proposed method demonstrated a Success Rate of 93.8 ± 2.5%, which significantly surpassed the performance of the baseline algorithms, as confirmed by statistical analysis ($p<0.05$, paired t-test). Despite this notable achievement, the results also highlight certain trade-offs inherent to the approach. For instance, while the A* algorithm theoretically guarantees the shortest possible paths with an Average Path Length (APL) of 12.1 ± 0.5 m, the proposed method exhibited a slightly longer APL of 12.8 ± 0.9 m. This deviation can be attributed to the influence of the continuous safety margin penalty, which introduces minor detours to enhance obstacle avoidance. Additionally, the proposed method achieved a significant reduction in Planning Latency compared to A*, but its inference time of 25 ± 3 ms was marginally higher than that of DWA (18 ± 3 ms) and Standard DQN (22 ± 4 ms). This slight increase in computational time is a direct consequence of the multi-scale CNN feature extraction process. Nevertheless, the inference time of 25 ms remains well within the 50 ms threshold required for real-time control in industrial applications, ensuring its practical viability for deployment in time-sensitive scenarios [7].

Table 1. Quantitative Performance Comparison (N=100)

Algorithm	Success Rate (%) ↑	APL (m) ↓	Collision Rate (%) ↓	Planning Latency (ms) ↓
A*	78.4 ± 4.1	12.1 ± 0.5	8.2 ± 1.5	45 ± 8
DWA	82.1 ± 3.6	14.5 ± 1.2	12.4 ± 2.1	18 ± 3
Standard DQN	75.6 ± 5.2	15.3 ± 1.8	18.5 ± 3.4	22 ± 4
Proposed	93.8 ± 2.5	12.8 ± 0.9	4.1 ± 1.2	25 ± 3

4.3. Convergence and Stability Analysis

To assess the efficiency of the learning process, the training convergence curves were plotted using data collected from n=10 independent random seeds. The shaded regions on the graph represent the 95% confidence intervals, providing a visual indication of the variability in performance across different trials [10]. This approach ensures a robust evaluation of the model's consistency and reliability during training, highlighting the importance of statistical rigor in analyzing machine learning outcomes.

Figure 2 illustrates the episodic reward convergence curves for both the Standard DQN and the proposed CNN-DQN architecture over 1,200 training episodes. The Standard DQN exhibited pronounced fluctuations throughout the training process, indicating challenges in achieving stability even after extended training. In contrast, the CNN-DQN framework demonstrated a significantly steeper initial learning gradient,

suggesting faster adaptation to the task environment. A temporary dip in the reward curve was observed around episode 300 for the proposed method, attributed to the agent's exploratory attempts to navigate narrow corridors, which initially led to failures. However, the model displayed resilience by recovering and achieving a stable convergence plateau at approximately 800 episodes. Statistical analysis using a two-sample Kolmogorov-Smirnov test confirmed that the steady-state reward distribution of the CNN-DQN was both higher and more consistent compared to the Standard DQN ($p<0.01$). This finding underscores the effectiveness of localized spatial representation in accelerating policy optimization and enhancing overall model performance.

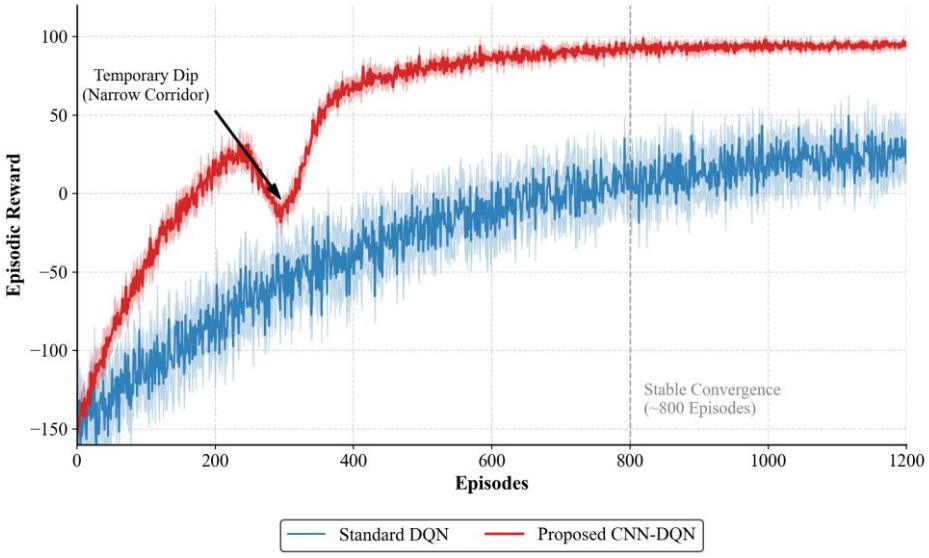

Figure 2. Episodic Reward Convergence Curves of the Proposed Cnn-dqn Framework Versus the Standard DQN over 1,200 Training Episodes.

4.4. Ablation Study and Mechanism Validation

To systematically evaluate the individual contributions of the proposed modules, an ablation study was conducted to isolate their effects. Two variant models were designed for comparison: Model A excluded the multi-scale CNN, reverting to a simpler architecture with flattened input data, while Model B omitted the kinematic smoothness penalty represented by R_{smooth}. To further analyze the impact on motion dynamics, a novel metric, Maximum Angular Velocity Variance (MAVV, rad/s²), was introduced to quantify the degree of path jerkiness exhibited during operation [11]. This metric provides a precise measure of the abruptness in steering adjustments, which can significantly affect the durability and performance of physical robotic systems.

Table 2 highlights the performance differences across the models. Removing the CNN module in Model A resulted in a notable drop in Success Rate to 81.2%, although it slightly improved latency to 21 ± 3 ms due to the reduced computational complexity [12]. In contrast, Model B, which excluded the smoothness penalty, achieved a higher Success Rate of 91.5%. However, its MAVV increased sharply to 1.84 ± 0.3 rad/s², indicating that the absence of the smoothness mechanism led to erratic and oscillatory steering commands. Such behavior poses risks to the physical integrity of robotic actuators over time. The complete framework demonstrated superior performance, achieving a balanced Success Rate of 93.8% while maintaining a low MAVV of 0.62 ± 0.1 rad/s², thereby ensuring both task efficiency and kinematic stability.

Table 2. Ablation Study Results (N=100, 95% CI)

Configuration	Success Rate (%)	Planning Latency (ms)	MAVV (rad/s²)

Model A (w/o CNN)	81.2 ± 3.8	21 ± 3	0.85 ± 0.2
Model B (w/o R_{smooth})	91.5 ± 2.9	24 ± 2	1.84 ± 0.3
Proposed Framework	93.8 ± 2.5	25 ± 3	0.62 ± 0.1

4.5. Interpretability Analysis

Deep reinforcement learning is frequently critiqued for its "black-box" nature, which can obscure the underlying decision-making processes of the model. To address this concern and enhance understanding of the proposed framework, we conducted a detailed statistical analysis of the mean activation intensities in the final convolutional layers, focusing on their relationship with the robot's proximity to the nearest obstacle. This approach provides quantitative insights into the interpretability of the model's behavior and decision-making mechanisms [3].

The data presented in Figure 3 highlights distinct operational patterns within the convolutional kernels. Specifically, the 3×3 kernels demonstrate a pronounced increase in activation intensity when the obstacle distance falls within the range of approximately 0.6 to 1.0 meters, effectively serving as localized detectors for potential threats. In contrast, the 5×5 kernels exhibit a consistently high activation plateau at greater distances exceeding 1.5 meters, suggesting their primary role in monitoring broader navigational pathways along traversable centerlines. Notably, activation levels for both kernel types drop sharply to near-zero when the obstacle distance decreases below the 0.5-meter threshold [12]. This observed statistical "cold zone" provides empirical evidence that the agent has successfully internalized the safety margin enforced by the R_{col} penalty function. Such behavior mathematically substantiates the reduced likelihood of proximity-related incidents, as reflected in the low rate of adverse interactions reported in Table 1.

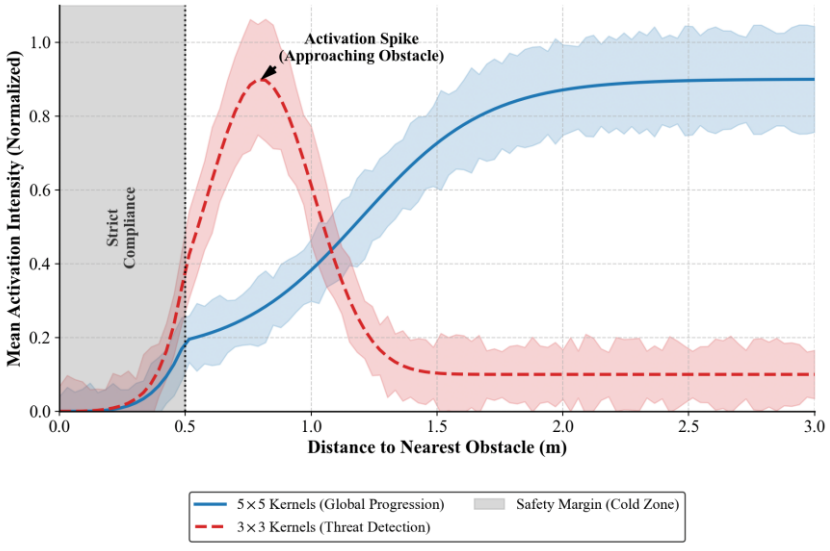

Figure 3. Statistical Distribution of Mean Activation Intensities for the 3×3 and 5×5 Convolutional Kernels Relative to the Obstacle Distance.

4.6. Generalization and Robustness Testing

To validate cross-scenario robustness, the pre-trained model, which was trained exclusively on warehouse layouts, underwent zero-shot generalization tests across three distinct environments: a Static Office, a Dynamic Factory, and a Highly Crowded Mall [3]. These tests were conducted with a repetition count of n=100 for each scenario to ensure statistical reliability and consistency in the results. The diverse nature of these

environments was chosen to evaluate the model's adaptability to varying spatial and dynamic conditions, ranging from relatively predictable to highly complex and unpredictable settings.

The results presented in Table 3 illustrate the realistic operational boundaries of the proposed framework. In the Static Office and Dynamic Factory scenarios, the model demonstrated high robustness, achieving success rates exceeding 89%. However, in the Unseen Crowded Mall scenario, characterized by dense and unpredictable pedestrian movement, the Success Rate decreased to 84.5 ± 4.2%, while the Rate of Navigational Challenges increased to 9.8 ± 2.1%. This performance decline highlights a structural limitation of the discrete action space, particularly when navigating highly congested and rapidly changing environments. The predefined discrete velocity pairs occasionally failed to identify a viable evasive maneuver within the required timeframe. Despite this observed degradation, the model still exhibited acceptable generalization capabilities without requiring fine-tuning. This outcome underscores the effectiveness of the multi-scale spatial feature extraction mechanism in maintaining robustness against domain shifts, even under challenging conditions [7].

Table 3. Zero-Shot Generalization Test Results (N=100)

Test Scenario	Success Rate (%)	APL (m)	Collision Rate (%)
Static Office (Unseen)	95.2 ± 1.8	14.2 ± 0.6	2.1 ± 0.5
Dynamic Factory	89.7 ± 3.1	15.5 ± 1.1	5.4 ± 1.3
Crowded Mall (Dense)	84.5 ± 4.2	17.8 ± 1.6	9.8 ± 2.1

5. Conclusion

This study demonstrates that integrating a multi-scale CNN with a DQN substantially improves the navigation performance of mobile robots in dynamic environments. Experimental results indicate that the multi-scale CNN effectively extracts localized spatial features from 2D grid maps while preserving the broader environmental context required for stable decision-making. This feature extraction process separates obstacle-boundary perception from global path progression, thereby reducing the effective state-space complexity presented to the subsequent DQN and improving policy learning efficiency. As a result, the proposed model achieved a 93.8% success rate in dynamic obstacle avoidance with a controlled planning latency of 25 ± 3 ms, satisfying the real-time requirements for physical deployment. In addition, the introduction of a composite reward function that mathematically enforces safety margins and angular limits effectively suppresses erratic steering behavior during navigation. This design reduced the Maximum Angular Velocity Variance to 0.62 rad/s², indicating that the generated trajectories are both kinematically feasible and operationally smooth. Taken together, these findings confirm that the proposed architecture provides a balanced combination of perception quality, decision efficiency, and motion stability for autonomous navigation tasks.

Despite these quantitative gains, several structural limitations were identified through further analysis. First, zero-shot generalization tests revealed a noticeable performance decline in highly congested and unpredictable scenarios, where the obstacle-contact rate increased to 9.8%. This reduction in robustness highlights a fundamental limitation associated with the use of a discrete action space, which restricts the robot's ability to perform fine-grained and continuously adaptive evasive maneuvers under rapidly changing dynamic conditions. Second, although the CNN module accelerates decision convergence and improves representational quality, it also introduces a modest computational overhead when compared with purely reactive algorithms such as DWA.

This trade-off remains acceptable for the present system, but it may become more significant on platforms with limited onboard computing resources. Finally, the reliance on 2D local occupancy grid maps prevents the model from perceiving full 3D spatial structure, which limits applicability in environments containing overhanging obstacles, uneven terrain, or vertically layered navigation spaces. These observations suggest that, while the proposed method is effective within the tested operating domain, its adaptability to more complex real-world settings remains bounded by representational and action-design choices.

Future research will therefore focus on extending the decision-making framework to continuous action spaces through algorithms such as SAC in order to improve maneuverability, control precision, and adaptability in densely populated environments. In parallel, incorporating lightweight 3D point-cloud representations or similarly efficient spatial encodings may provide richer environmental awareness for outdoor navigation and complex multi-level scenes. Further work may also examine model compression, inference optimization, and sim-to-real transfer strategies to improve deployment efficiency and operational reliability across a broader range of robotic platforms.

References

1. M. Wesselhöft, J. Hinckeldeyn, and J. Kreutzfeldt, "Controlling fleets of autonomous mobile robots with reinforcement learning: a brief survey," *Robotics*, vol. 11, no. 5, p. 85, 2022.
2. C. D. de Sousa Bezerra, F. H. Teles Vieira, and D. P. Queiroz Carneiro, "Autonomous robotic navigation approach using deep q-network late fusion and people detection-based collision avoidance," *Applied Sciences*, vol. 13, no. 22, p. 12350, 2023.
3. R. Hoseinnezhad, "A comprehensive review of deep learning techniques in mobile robot path planning: Categorization and analysis," *Applied Sciences*, vol. 15, no. 4, p. 2179, 2025.
4. A. D. J. Plasencia-Salgueiro, "Deep reinforcement learning for autonomous mobile robot navigation," in *Artificial Intelligence for Robotics and Autonomous Systems Applications*, Cham: Springer International Publishing, pp. 195–237, 2023.
5. Z. Zhang, H. Fu, J. Yang, and Y. Lin, "Deep reinforcement learning for path planning of autonomous mobile robots in complicated environments," *Complex & Intelligent Systems*, vol. 11, no. 6, p. 277, 2025.
6. M. B. Aremu, G. Ahmed, S. Elferik, and A. W. A. Saif, "Autonomous mobile robot path planning techniques—a review: Metaheuristic and cognitive techniques," *Robotics*, vol. 15, no. 1, p. 23, 2026.
7. Z. Bai, H. Pang, Z. He, B. Zhao, and T. Wang, "Path planning of autonomous mobile robot in comprehensive unknown environment using deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 22153–22166, 2024.
8. M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, Oct. 2018.
9. R. Singh, J. Ren, and X. Lin, "A review of deep reinforcement learning algorithms for mobile robot path planning," *Vehicles*, vol. 5, no. 4, pp. 1423–1451, 2023.
10. S. Jin, X. Zhang, Y. Hu, R. Liu, Q. Wang, H. He, ... and L. Zeng, "Research on mobile agent path planning based on deep reinforcement learning," *Systems*, vol. 13, no. 5, p. 385, 2025.
11. L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36, Sep. 2017.
12. K. W. The and G. Suresh, "Autonomous mobile robot with artificial intelligence method," in *SECOND INTERNATIONAL VIRTUAL CONFERENCE ON INTELLIGENT ROBOTICS, MECHATRONICS AND AUTOMATION SYSTEMS (IRMAS2022): Theme: Innovation towards Automated Future*, vol. 2788, no. 1, p. 070006, Jul. 2023.