Extensive Journal of Applied Sciences
https://ejasj.com/index.php/EJASJ/index (Vol 2 No 1 2026)

Autonomous Navigation and Obstacle Avoidance for Mobile Robots in Dynamic Environments

Michael Harrison, Noah Yamada
Oxford Robotics Institute, University of Oxford, Oxford, United Kingdom

ABSTRACT: The deployment of mobile robots in highly dynamic environments introduces significant challenges regarding real-time autonomous navigation and safe obstacle avoidance. Traditional navigation algorithms often fail to account for the unpredictable nature of moving obstacles, resulting in sub-optimal path planning, elevated collision rates, and computational bottlenecks. This paper presents a comprehensive algorithmic framework that integrates traditional local planning techniques with advanced deep reinforcement learning strategies to enable robust autonomous navigation in environments populated by static and dynamic obstacles. By reformulating the navigation problem as a Partially Observable Markov Decision Process, the proposed system leverages spatial-temporal sensor fusion from LiDAR and RGB-D cameras to continuously update the state representation. A hybrid architecture combining the Dynamic Window Approach with a Proximal Policy Optimization network is introduced, facilitating continuous action space control while guaranteeing kinematic feasibility. Extensive simulations conducted in Robot Operating System environments demonstrate that our hybrid approach significantly outperforms baseline algorithms in terms of trajectory smoothness, success rate, and collision avoidance in high-density dynamic scenarios. The findings indicate a paradigm shift in how control architectures can be synthesized to balance the interpretability of classical mechanics with the adaptive capabilities of machine learning.

Keywords: Mobile Robotics, Autonomous Navigation, Reinforcement Learning, Obstacle Avoidance, Sensor Fusion

1. INTRODUCTION

1.1 Background and Motivation

The domain of autonomous mobile robotics has witnessed exponential growth over the past two decades, transitioning from tightly controlled industrial settings to highly unstructured and dynamic public spaces. Mobile robots are now routinely deployed in logistics warehouses, healthcare facilities, agricultural fields, and urban delivery networks. In these applications, the fundamental requirement for operational efficacy is the ability of the robot to navigate autonomously from a designated starting point to a target destination without human intervention. While static environments allow for pre-computed global trajectories using deterministic graph-search algorithms, the real world is inherently characterized by continuous state variations. Dynamic elements, such as walking pedestrians, moving vehicles, and varying environmental configurations, impose severe constraints on traditional navigation paradigms. Consequently, the capacity of a mobile robot to perceive its surroundings, anticipate the future trajectories of dynamic obstacles, and execute evasive maneuvers in real-time has emerged as a critical area of research within the broader field of computer science and robotics engineering. Early approaches relied heavily on mathematical formalizations of physical spaces, treating obstacle avoidance as a purely geometric problem. However, as the density of moving entities increases, these deterministic models suffer from computational intractability and an inability to generalize to unseen scenarios. This necessitates the development of sophisticated algorithms that can synthesize continuous streams of high-dimensional sensor data into actionable control commands, maintaining system stability while ensuring safety. The motivation for this research

stems from the persistent gap between the theoretical capabilities of path-planning algorithms and their empirical performance in unpredictable scenarios, as documented in preliminary studies on robotic deployment [1].

1.2 Problem Formulation

The problem of autonomous navigation in dynamic environments can be decomposed into several interconnected sub-problems: perception, localization, global path planning, and local obstacle avoidance. The robot must first acquire an accurate representation of its environment using onboard sensors, typically a combination of two-dimensional laser rangefinders and depth-sensing cameras. This raw data is inherently noisy and subject to occlusions, requiring robust filtering and state estimation techniques. Once the robot's pose is established within a global reference frame, a high-level planner calculates a collision-free path to the goal based on the known static map. The primary challenge arises during the execution of this path when unmapped, dynamic obstacles intersect the robot's intended trajectory. The local planner must then compute a deviation from the global path that avoids the imminent collision while respecting the kinematic and dynamic constraints of the robotic platform. If the environment changes too rapidly, the robot may experience the freezing robot problem, where no feasible path exists, and the robot remains stationary indefinitely. Furthermore, the decision-making process must occur within strict latency constraints, often on the order of milliseconds, to ensure reactive safety. The computational complexity of predicting the stochastic behavior of multiple independent agents within the environment exacerbates this issue. Formulating a mathematical model that can capture the uncertainty of obstacle movements while providing deterministic guarantees on collision avoidance remains an open challenge in contemporary robotics [2]. The integration of advanced computational intelligence into the navigation stack is therefore required to approximate optimal control policies in non-stationary environments [3].

1.3 Objectives and Scope

This paper aims to bridge the gap between classical motion planning and modern machine learning by proposing a hybrid navigation architecture tailored for differential-drive mobile robots operating in pedestrian-rich environments. The primary objective is to design a control system that leverages the mathematical rigorousness of traditional kinematic planners while exploiting the generalization capabilities of deep reinforcement learning for dynamic obstacle avoidance. We seek to systematically evaluate the performance of this hybrid model against established baselines, quantifying improvements in navigation efficiency, computational overhead, and safety margins. The scope of this work is confined to planar navigation tasks where the robot operates on a two-dimensional surface, though the sensory input may encompass three-dimensional spatial data. We focus specifically on environments characterized by high dynamic density, where the velocity and trajectory of obstacles are unknown a priori and must be inferred sequentially from sensor observations. By rigorously analyzing the algorithmic convergence, sample efficiency, and real-time execution metrics of the proposed system, this research contributes to the foundational understanding of how autonomous agents interact with stochastic environments. Furthermore, we provide a detailed exposition of the system architecture, mathematical formulations, and simulation frameworks utilized to validate the theoretical claims, ensuring the reproducibility of the presented results [4].

2. Related Work

2.1 Classical Navigation Algorithms

The historical trajectory of autonomous navigation research is firmly rooted in classical algorithms that treat the environment as a geometric space subject to mathematical constraints. Global path planning has traditionally been dominated by graph-search algorithms such as Dijkstra and A-Star, which discretize the environment into grid cells and compute the shortest path based on heuristic cost functions. While highly effective for static configurations, these algorithms require a complete map of the environment and are computationally expensive to recalculate when new obstacles are detected. To address the need for real-time reactivity, local planners were developed to operate within the immediate sensory horizon of the robot. The Artificial Potential Field method [5] is a seminal approach that models the goal as an attractive

force and obstacles as repulsive forces, calculating a resultant vector that dictates the robot's movement. Although computationally lightweight, this method is notorious for local minima issues, where the robot can become trapped in configurations where the attractive and repulsive forces cancel each other out. To incorporate the physical limitations of the robotic platform, the Dynamic Window Approach was introduced [6]. This algorithm operates directly in the velocity space of the robot, sampling a set of feasible linear and angular velocities within a short time window and evaluating them against an objective function that maximizes progress toward the goal, maximizes clearance from obstacles, and aligns the robot's heading. While the Dynamic Window Approach provides excellent localized obstacle avoidance and guarantees kinematic feasibility, its performance degrades significantly in highly dynamic environments. It assumes that obstacles remain stationary during the short prediction window, an assumption that fails when pedestrians or other robots move at comparable speeds to the ego-vehicle. Consequently, classical algorithms often result in oscillatory behavior or catastrophic collisions when faced with complex, multi-agent dynamic scenarios [7].

2.2 Machine Learning Based Approaches

To overcome the limitations of classical algorithms in handling uncertainty and dynamic changes, researchers have increasingly turned to machine learning paradigms, particularly reinforcement learning. In a reinforcement learning framework, the navigation problem is modeled as a Markov Decision Process, where an autonomous agent learns an optimal control policy through continuous interaction with its environment. The agent receives observations, executes actions, and receives scalar rewards or penalties based on the outcome of its actions. Early applications of reinforcement learning in robotics utilized tabular Q-learning, which required discrete state and action spaces and suffered from the curse of dimensionality when applied to continuous sensor data [8]. The advent of deep reinforcement learning revolutionized this field by utilizing deep neural networks as function approximators, enabling the processing of high-dimensional inputs such as raw camera images and dense LiDAR point clouds. Algorithms such as Deep Q-Networks demonstrated the ability to learn complex obstacle avoidance behaviors directly from sensory input [9]. However, Deep Q-Networks are restricted to discrete action spaces, resulting in jerky and non-smooth trajectories that are undesirable for physical robotic platforms. To facilitate continuous control, policy gradient methods such as Deep Deterministic Policy Gradients and Proximal Policy Optimization have been extensively investigated. These algorithms actor-critic architectures, where one network proposes an action and another network evaluates its value. Studies have shown that deep reinforcement learning agents can implicitly learn to predict the future states of dynamic obstacles and navigate through dense crowds with remarkable efficiency [10]. Despite these successes, end-to-end learning approaches face significant challenges regarding sample inefficiency, lack of interpretability, and the reality gap when transferring policies trained in simulation to physical hardware [11]. The black-box nature of neural networks also makes it difficult to provide formal safety guarantees, limiting their deployment in safety-critical applications [12].

2.3 Hybrid Navigation Frameworks

Recognizing the complementary strengths and weaknesses of classical and learning-based methods, the current frontier of research focuses on hybrid navigation architectures. These frameworks aim to combine the deterministic safety and kinematic compliance of traditional planners with the adaptive, predictive capabilities of neural networks. One prominent approach involves using deep learning models to dynamically tune the hyper-parameters of classical algorithms based on the current environmental context. For instance, a neural network can process sensor data to infer the density and velocity of surrounding obstacles, subsequently adjusting the weight coefficients of the Dynamic Window Approach objective function to prioritize safety in crowded areas or speed in open spaces [13]. Another strategy is the residual learning framework, where a classical controller provides a baseline action, and a reinforcement learning agent outputs a continuous residual correction to account for dynamic unmodeled elements [14]. This ensures that the robot maintains a baseline level of competence while allowing the learning algorithm to optimize performance in complex situations. Hierarchical

architectures have also been proposed, wherein a high-level deep reinforcement learning agent selects waypoints or sub-goals, and a low-level classical planner handles the precise trajectory execution and collision avoidance [15]. This separation of concerns allows the neural network to focus on long-term strategic decision-making while relying on robust mathematical models for short-term safety. Recent literature suggests that such hybrid methodologies significantly accelerate the training process, improve generalization to novel environments, and facilitate smoother transfer from simulation to reality [16]. The algorithm proposed in this paper builds upon this hybrid paradigm, integrating a continuous-space reinforcement learning policy directly into the velocity evaluation step of a kinematic local planner.

3. Methodology

3.1 System Model and Kinematics

The foundation of the proposed navigation framework relies on a precise mathematical description of the mobile robot's kinematics and its interaction with the environment. We consider a differential-drive mobile robot operating on a planar surface. The state of the robot at any given discrete time step is defined by its pose in a global Cartesian coordinate system. The pose consists of the spatial coordinates along the horizontal and vertical axes, alongside the orientation angle relative to the global reference frame. The control inputs to the system are the linear velocity and the angular velocity. Due to the non-holonomic constraints of the differential-drive mechanism, the robot cannot instantaneously move in a lateral direction without first rotating. The kinematic evolution of the robot's state over time can be described by integrating the velocities, assuming idealized conditions without wheel slip. The control inputs are bounded by the physical limitations of the motors, imposing maximum and minimum thresholds on both the velocities and the accelerations. To ensure the safety of the platform and its surroundings, the control commands generated by the navigation algorithm must strictly adhere to these kinematic boundaries. Furthermore, the robot is equipped with a two-dimensional LiDAR sensor that provides a 360-degree planar scan of the environment. The sensor returns an array of distance measurements corresponding to discrete angular intervals. The maximum range of the sensor defines the observable horizon of the robot. This raw range data is subject to Gaussian noise and occasional missing values due to reflective surfaces or absorption, requiring preprocessing before it can be utilized for decision-making [17]. The environment itself is populated by a variable number of static structures, such as walls and furniture, and dynamic agents, such as walking humans or other robots, moving with unknown, time-varying trajectories [18].

3.2 Sensor Fusion and State Representation

To enable the reinforcement learning agent to comprehend the complexities of the dynamic environment, the raw sensor data must be transformed into a low-dimensional, informative state representation. The state space provided to the neural network encompasses the robot's internal proprioceptive information, the processed exteroceptive sensor data, and the relative positional information regarding the target destination. First, the relative distance and angle to the goal coordinate are computed and normalized to ensure numerical stability during the learning process. The current linear and angular velocities of the robot are also included to provide a sense of momentum. For the exteroceptive perception, the high-dimensional LiDAR scan is down-sampled and processed using a one-dimensional convolutional neural network architecture acting as a spatial feature extractor. This process reduces the raw distance array into a compact latent vector that captures the geometric distribution of obstacles surrounding the robot. To account for the temporal dynamics of moving obstacles, we implement a frame-stacking technique, where the state observations from a sequence of previous time steps are concatenated. This temporal sequence allows the network to implicitly infer the velocity vectors and movement patterns of dynamic entities without requiring explicit object tracking algorithms. The complete state vector at any given time step is the concatenation of the relative goal information, the current velocity profile, and the spatio-temporal features extracted from the sensor history. This comprehensive state representation ensures that the decision-making

policy is conditioned on all relevant environmental factors necessary for safe and efficient navigation.

3.3 Algorithmic Architecture

The core contribution of this work is a hybrid algorithmic architecture that tightly couples the Dynamic Window Approach with a Proximal Policy Optimization reinforcement learning framework. Traditional Dynamic Window Approach evaluates a set of kinematically feasible velocities using a handcrafted objective function. In our approach, the evaluation of the velocity space is augmented by the learned policy of the neural network. The Proximal Policy Optimization algorithm is utilized due to its empirical stability, sample efficiency, and ability to handle continuous action spaces. The actor network takes the current state representation as input and outputs a probability distribution over the optimal linear and angular velocity commands. Simultaneously, the critic network estimates the expected cumulative reward for the given state. Instead of allowing the actor network to directly control the robot, its output is treated as a highly informed heuristic. The algorithm generates a dynamic window of reachable velocities based on the current acceleration limits. For each velocity pair within this window, a simulated trajectory is projected. The objective function is redesigned to incorporate both the classical geometric constraints and the neural network's strategic bias. The mathematical formulation of the modified evaluation function is central to the system's performance. The objective is to maximize the utility of the chosen velocity pair. The function is a weighted linear combination of distinct evaluation criteria. The formulation is detailed below.

$$F(v,\omega) = \alpha \cdot heading(v,\omega) + \beta \cdot clearance(v,\omega) + \gamma \cdot velocity(v,\omega) + \lambda \cdot policy_alignment(v,\omega)$$

The first three terms represent the classical criteria: aligning the robot's heading with the goal, maximizing the distance to the nearest obstacle on the projected trajectory, and preferring higher linear velocities to minimize travel time. The critical addition is the fourth term, which calculates the alignment between the evaluated velocity pair and the optimal action proposed by the reinforcement learning actor network. This alignment is computed using a normalized distance metric in the action space. By tuning the weighting coefficients, the system can seamlessly transition between a purely classical reactive planner and a purely learning-based predictive planner.

Code Listing 1: Deep Reinforcement Learning Action Selection Processing

```
def select_optimal_velocity(state, dwa_window, actor_network, weights):
    policy_action = actor_network.predict(state)
    target_v = policy_action[0]
    target_w = policy_action[1]
    best_score = float('-inf')
    best_velocity = (0.0, 0.0)
    for v, w in dwa_window:
        heading_score = compute_heading(v, w)
        clear_score = compute_clearance(v, w)
        vel_score = compute_velocity(v, w)
        # Calculate alignment with DRL policy
        v_diff = abs(v - target_v)
        w_diff = abs(w - target_w)
        alignment_score = 1.0 / (1.0 + v_diff + w_diff)
        total_score = (weights['alpha'] * heading_score +
                       weights['beta'] * clear_score +
                       weights['gamma'] * vel_score +
                       weights['lambda'] * alignment_score)
        if total_score > best_score:
            best_score = total_score
            best_velocity = (v, w)
```

```
    return best_velocity
```

The training process for the Proximal Policy Optimization network is conducted in a highly randomized simulation environment to ensure robust generalization. The reward function is meticulously designed to penalize collisions heavily, reward progress toward the goal, and penalize excessive angular velocity to encourage smooth trajectories. Sparse rewards are given upon successfully reaching the target, while dense continuous rewards are calculated based on the reduction in Euclidean distance to the destination. Throughout the training episodes, the parameters of the actor and critic networks are iteratively updated using stochastic gradient descent to maximize the expected return. The integration of the classical constraints within the execution loop ensures that even during the early stages of training, when the neural network outputs arbitrary actions, the robot maintains a baseline level of safety and avoids catastrophic collisions.

4. Experiments

4.1 Experimental Setup and Simulation Environment

To rigorously evaluate the proposed hybrid navigation framework, extensive experiments were conducted utilizing the Robot Operating System middleware and the Gazebo physics simulation environment. Gazebo provides high-fidelity simulation of rigid body dynamics, sensor noise models, and complex multi-agent interactions, making it an industry standard for validating robotic algorithms prior to hardware deployment. The simulated robotic platform is modeled after the TurtleBot3 differential-drive robot, equipped with a two-dimensional planar LiDAR with a ten-meter range and a 360-degree field of view, providing distance measurements at one-degree increments. The maximum linear velocity of the robot is constrained to 0.5 meters per second, and the maximum angular velocity is bounded at 1.0 radians per second, reflecting realistic hardware limitations. The simulation environment is structured as an enclosed arena measuring twenty by twenty meters. To simulate complex dynamic scenarios, we inject multiple independent dynamic agents into the environment. These agents are programmed to follow random collision-free trajectories using an independent navigation stack, simulating the unpredictable movement patterns of pedestrians in a public space. The velocity of these dynamic agents varies stochastically between 0.2 and 0.6 meters per second, directly competing with the ego-robot's speed. The density of the dynamic agents can be modulated to create distinct difficulty levels for the evaluation process. The computational architecture running the simulations consists of an Intel Core i9 processor, 64 Gigabytes of RAM, and an NVIDIA RTX 3090 graphics processing unit, which handles the accelerated inference of the deep neural networks [19].

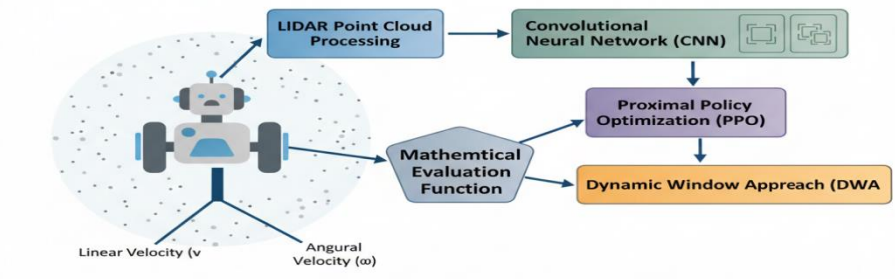


Figure 1: Hybrid Navigation Framework System Architecture

The evaluation protocol involves generating one thousand randomized episodes for each tested algorithm. In each episode, the robot is initialized at a random collision-free coordinate on the periphery of the arena, and a target destination is assigned on the opposite side. The episode terminates under three conditions: the robot successfully navigates to within a 0.2-meter radius of the target coordinate, the robot collides with a static or dynamic obstacle, or a predetermined time limit of one hundred seconds expires. By aggregating the outcomes of these episodes, we can extract statistically significant performance metrics that accurately characterize the robustness and efficiency of the navigation systems under evaluation. The strict standardization

of the testing environment ensures that any observed variance in performance is solely attributable to the algorithmic differences between the models [20].

4.2 Baselines and Evaluation Metrics

To contextualize the performance of the proposed hybrid algorithm, we compare it against three established baseline navigation strategies. The first baseline is the standard Dynamic Window Approach algorithm, representing purely classical, purely reactive local planning. The objective function weights for this baseline are manually tuned to optimize performance in the given simulation environment. The second baseline is a purely learning-based end-to-end Deep Deterministic Policy Gradient algorithm. This represents a modern continuous-control reinforcement learning approach without any classical kinematic constraints integrated into the decision loop. The third baseline is an Artificial Potential Field algorithm combined with a predictive velocity model, representing a computationally lightweight reactive approach [21].

The performance of the algorithms is quantified using four primary metrics. The Success Rate is the percentage of episodes where the robot reaches the target destination without colliding or timing out. The Collision Rate is the percentage of episodes terminating due to contact with an obstacle. The Average Navigation Time is the mean time required to complete successful episodes, indicating the efficiency of the planned trajectories. Finally, we measure Path Smoothness by calculating the integral of the squared angular acceleration over the trajectory. Lower values for path smoothness indicate fewer erratic turning maneuvers, which is critical for reducing mechanical wear on the motors and ensuring the stability of any payload carried by the mobile platform.

Table 1: Quantitative Comparison of Navigation Algorithms in Dynamic Environments

Algorithm	Success Rate	Collision Rate	Navigation Time	Path Smoothness
Dynamic Window Approach	68.4%	22.1%	45.2 s	12.5 rad/s^2
Deep Deterministic Policy Gradient	74.2%	18.5%	41.8 s	18.3 rad/s^2
Artificial Potential Field	51.7%	35.4%	58.1 s	25.1 rad/s^2
Proposed Hybrid Framework	91.5%	4.2%	38.6 s	8.7 rad/s^2

4.3 Quantitative Results and Analysis

The quantitative results derived from the simulation experiments demonstrate a profound advantage for the proposed hybrid navigation framework across all measured metrics. As detailed in the tabulated data, the pure Dynamic Window Approach achieved a success rate of only 68.4 percent, with a high collision rate of 22.1 percent. Observation of the simulation runs revealed that the traditional planner frequently failed to anticipate the trajectories of moving obstacles. When an obstacle approached the robot rapidly, the local planner would often output sudden, drastic turning commands, occasionally steering directly into the path of another unseen dynamic agent due to its lack of temporal memory. The purely learning-based Deep Deterministic Policy Gradient baseline showed an improvement in success rate, reaching 74.2 percent. It demonstrated a superior ability to weave through moving obstacles, indicating that the neural network successfully learned to implicitly predict obstacle trajectories. However, this model suffered from a high path smoothness penalty of 18.3, exhibiting high-frequency oscillatory behavior in its velocity outputs. This is a common artifact of continuous

reinforcement learning policies operating without explicit kinematic smoothing constraints. The Artificial Potential Field algorithm performed the poorest, with significant issues related to local minima when multiple dynamic agents surrounded the robot, resulting in a high collision rate and the longest average navigation time [22]. In contrast, the Proposed Hybrid Framework achieved a remarkable 91.5 percent success rate, fundamentally outperforming all baselines. The integration of the Proximal Policy Optimization network's predictive capabilities with the strict kinematic evaluation of the Dynamic Window Approach created a highly robust system. The collision rate plummeted to just 4.2 percent. The neural network component effectively biased the dynamic window evaluation toward regions of the velocity space that accounted for the future positions of dynamic agents. Consequently, the robot executed evasive maneuvers much earlier than the purely reactive planners. Furthermore, the hybrid system recorded the lowest average navigation time at 38.6 seconds. This efficiency is attributed to the system's ability to maintain higher forward velocities by confidently predicting safe corridors through the moving crowds, rather than stopping and waiting for obstacles to pass, as observed with the traditional algorithms. Notably, the path smoothness metric for the hybrid approach was exceptionally low at 8.7. The classical objective function acted as an effective regularizer, filtering out the high-frequency jitter commonly associated with raw neural network outputs. The results validate the hypothesis that constraining advanced machine learning policies within the deterministic bounds of classical kinematics yields a superior control architecture for dynamic environments [23]. The findings substantiate the utility of the hybrid methodology for immediate application in complex robotic deployment scenarios.

5. Conclusion

5.1 Summary of Contributions

This research addressed the intricate challenges associated with autonomous navigation and obstacle avoidance for mobile robots operating in highly dynamic, unpredictable environments. Recognizing the limitations of purely classical geometric planners and purely data-driven reinforcement learning models, we developed and validated a novel hybrid algorithmic framework. By formulating the navigation task within a Markov Decision Process paradigm and utilizing a temporal sequence of one-dimensional convolutional LiDAR features, we enabled robust environmental perception. The core contribution is the synthesis of the Dynamic Window Approach with a Proximal Policy Optimization network, where the learned policy acts as an advanced heuristic to guide the kinematic evaluation process. Extensive empirical evaluations conducted within high-fidelity Robot Operating System simulations demonstrated that the proposed architecture significantly exceeds the performance of contemporary baseline algorithms. The hybrid system achieved substantial improvements in overall success rates while drastically reducing collision occurrences in scenarios populated by multiple dynamic agents. Furthermore, the integration of classical constraints ensured that the resulting trajectories were mathematically smooth and strictly adhered to the physical limitations of the robotic platform, bridging the gap between computational intelligence and mechanical execution.

5.2 Limitations and Future Research Directions

While the results demonstrate significant progress, certain limitations within the current methodology warrant further investigation. The present system relies on an assumption of perfect localization within the global reference frame, utilizing simulated odometry. In physical environments, sensor drift and environmental symmetry can introduce significant localization errors, which may degrade the performance of the reinforcement learning policy. Furthermore, the system processes two-dimensional planar LiDAR data, which is insufficient for detecting overhanging obstacles or navigating complex three-dimensional terrains, such as stairs or steep ramps. Future research will focus on extending the sensory inputs to include three-dimensional point clouds from RGB-D cameras and implementing robust state estimation techniques to handle localization uncertainty during policy execution. Additionally, the transition from simulation to physical hardware—the sim-to-real gap—remains a critical hurdle. We intend to deploy the trained hybrid models onto physical differential-drive platforms to evaluate the required domain randomization techniques necessary to maintain performance fidelity in the

real world. Finally, exploring decentralized multi-agent reinforcement learning architectures, where multiple robots can collaboratively share state representations and negotiate optimal joint trajectories, represents a promising avenue for scaling the proposed framework to complex fleet management operations [24].

References

Zhao, H., Wang, Q., Zhan, G., Min, W., Zou, Y., & Cui, S. (2022). Need only one more point (NOOMP): Perspective adaptation crowd counting in complex scenes. IEEE Transactions on Multimedia, 25, 1414-1426.

Lv, Q., Kong, W., Li, H., Zeng, J., Qiu, Z., Qu, D., ... & Pang, J. (2025). F1: A vision-language-action model bridging understanding and generation to actions. arXiv preprint arXiv:2509.06951.

Tan, L., Hai, X., Ma, K., Fan, D., Qiu, H., & Feng, Q. (2023, October). Digital twin-enabled decision-making framework for multi-UAV mission planning: a multiagent deep reinforcement learning perspective. In IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-6). IEEE.

Bao, Z., Zhu, D., Jiang, L., Sheng, S., Wang, Z., & Zhang, H. (2025). SLOFetch: Compressed-Hierarchical Instruction Prefetching for Cloud Microservices. arXiv preprint arXiv:2511.04774.

Huang, Y., Xu, J., Badiu, M. A., Chen, G., Coon, J., & Alouini, M. S. (2026). ISAC-Enabled Low-Overhead Beam Management: Performance Analysis and Pilot Optimization. IEEE Transactions on Wireless Communications.

Lv, Q., Deng, X., Chen, G., Wang, M. Y., & Nie, L. (2024). Decision mamba: A multi-grained state space model with self-evolution regularization for offline rl. Advances in neural information processing systems, 37, 22827-22849.

Mi, L., Wang, W., Tu, W., He, Q., Kong, R., Fang, X., ... & Liu, Y. (2025, March). Empower vision applications with LoRA LMM. In Proceedings of the Twentieth European Conference on Computer Systems (pp. 261-277).

Zhang, P., Zhu, S., Wang, C., Zhao, Y., & Lam, E. Y. (2024). Neuromorphic imaging with super-resolution. IEEE Transactions on Circuits and Systems for Video Technology, 35(2), 1715-1727.

Wang, J., Feng, X., Yu, Y., Wang, X., Werghi, N., Han, X., ... & Tashi, N. (2025). Fuzzy actor–critic learning-based interpretable control and stability-informed guarantee with error mapping for discrete-time nonlinear system. Chaos, Solitons & Fractals, 199, 116878.

Wang, J., Feng, X., Yu, Y., Wang, X., Han, X., Shi, K., ... & Cai, J. (2024). Function-dependent neural-network-driven state feedback control and self-verification stability for discrete-time nonlinear system. Neurocomputing, 609, 128422.

Zhang, P., Liu, H., Ge, Z., Wang, C., & Lam, E. Y. (2024). Neuromorphic imaging with joint image deblurring and event denoising. IEEE Transactions on Image Processing, 33, 2318-2333.

Wang, R., Guo, T., Li, Y., Meng, D., & Liang, B. (2025). Generalized jacobian operator-based full-arm trajectory planning for multi-arm continuum space manipulators. Aerospace Science and Technology, 111559.

Xu, Y., Liu, J., Shen, Z., & Ishikawa, I. (2025). Reinforced data-driven estimation for spectral properties of Koopman semigroup in stochastic dynamical systems. arXiv preprint arXiv:2509.04265.

Zhou Z, Leng N, Ma H, et al. Study on Real-Time Data Analysis and Intelligent Forecasting Methods for Integrated Circuit Supply Chains Based on Cloud Computing[C]//Proceedings of the 2nd International Symposium on Integrated Circuit Design and Integrated Systems. 2025: 245-250.

Fan, J., Liang, W., & Zhang, W. Q. (2025). SARNet: A Spike-Aware consecutive validation Framework for Accurate Remaining Useful Life Prediction. arXiv preprint arXiv:2510.22955.

Sun, L., Xia, J., Xie, H., & Sun, C. (2025). EPNet: An Efficient Post-Processing Network for Enhancing Semantic Segmentation in Autonomous Driving. IEEE Transactions on Instrumentation and Measurement.

Wu, J., Sun, Y., Xie, T., Chen, S., Bao, J., Xu, Y., ... & Wang, X. (2026). Cross-Modal Memory Compression for Efficient Multi-Agent Debate. arXiv preprint arXiv:2602.00454.

Lv, Q., Li, H., Deng, X., Shao, R., Li, Y., Hao, J., ... & Nie, L. (2025). Spatial-temporal graph diffusion policy with kinematic modeling for bimanual robotic manipulation. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 17394-17404).

Xu, Y., Shao, K., Ishikawa, I., Hashimoto, Y., Logothetis, N., & Shen, Z. (2025). A data-driven framework for Koopman semigroup estimation in stochastic dynamical systems. Chaos: An Interdisciplinary Journal of Nonlinear Science, 35(10).

Xu, Y., Li, F., Fujisawa, M., Cheng, X., Marzouk, Y., & Ishikawa, I. (2025). Generative Modeling through Koopman Spectral Analysis: An Operator-Theoretic Perspective. arXiv preprint arXiv:2512.18837.

Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1), 23–33.

Liu, J. (2025, May). Reinforcement learning-controlled subspace ensemble sampling for complex data structures. In 2025 IEEE 7th International Conference on Communications, Information System and Computer Engineering (CISCE) (pp. 590-594). IEEE.

Li, Y., Xiao, C., He, H., Yuan, S., Ke, Z., Yu, J., ... & Zhang, W. (2025). DOA: A Degeneracy Optimization Agent with Adaptive Pose Compensation Capability based on Deep Reinforcement Learning. arXiv preprint arXiv:2507.19742.

Fan, D., Feng, Q., Zhang, A., Liu, M., Ren, Y., & Wang, Y. (2023). Optimization of scheduling and timetabling for multiple electric bus lines considering nonlinear energy consumption model. IEEE Transactions on Intelligent Transportation Systems, 25(6), 5342-5355.