Future Technology
Open Access Journal
ISSN 2832-0379

August 2026| Volume 05 | Issue 03 | Pages 69-76

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.3.6

Future Publishing LLC

Article

Real-time obstacle avoidance in mobile robots using deep reinforcement learning

Roja BA[1]*, Priyanka Mishra[2], M. Kalaimani[3], Prachi Juyal[4], P.K. Anjani[5], Rakhi Dua[6], Pushpa Mamoria[7]

[1]Presidency University, Bangalore, Karnataka, India
[2]Poornima University, Jaipur, Rajasthan, India
[3]Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India
[4]Graphic Era University, Clement Town, Dehradun, Uttarakhand, India
[5]Sona College of Technology, Salem, Tamil Nadu, India
[6]K. R Mangalam University, Gurugram, Haryana, India
[7]Chhatrapati Shahu Ji Maharaj University, Kanpur, India

ARTICLE INFO

Article history:
Received 01 November 2025
Received in revised form
18 March 2026
Accepted 24 April 2026

Keywords:
Mobile robotics, Deep reinforcement learning,
Proximal policy optimization,
Real-time navigation, Obstacle avoidance

*Corresponding author
Email address:
rojareddyba89@gmail.com

DOI: 10.55670/fpll.futech.5.3.6

ABSTRACT

Real-time obstacle avoidance is a challenge in mobile robotics, as it is an ongoing process and remains difficult to achieve in crowded, dynamic environments, where conventional planning algorithms, such as local planners, often offer limited adaptability. This paper presents a Proximal Policy Optimization-based deep reinforcement learning approach for real-time obstacle avoidance for mobile robots. The proposed system is end-to-end policy learning based on inputs from LiDAR and other auxiliary sensors, and is trained in a Gazebo-ROS environment using domain randomization to enhance robustness to sim-to-real transfer. The framework was implemented on a TurtleBot3 Burger platform and tested both in simulation and in an indoor physical environment with varying numbers of obstacles. In simulation, the proposed policy achieved a success rate of 94.2%, a 68.6% reduction in collision rate compared to the Dynamic Window Approach baseline policy, a path efficiency of 16.5%, and a 14.6% reduction in average time to goal. In real experiments, the policy has maintained success rates above 88, even under high-density conditions. The optimized onboard inference pipeline achieved less than 20 ms latency and over 50 Hz throughput on embedded hardware. These results indicate that the proposed framework is a successful and computationally feasible solution to real-time robotic navigation in dynamic environments.

1. Introduction

Autonomous mobile robots are increasingly used in logistics, manufacturing, autonomous transportation, and aerial systems because they can perform tasks with high precision and efficiency, requiring limited human intervention [1]. These systems can run continuously, adapt to changes in their environment, and execute complex maneuvers that may be unsafe or impractical for humans. In such applications, real-time obstacle avoidance is necessary to achieve safe and reliable operation in dynamic and unpredictable environments [2]. Effective obstacle avoidance not only ensures safety but also enhances mission completion rate and energy efficiency, making it an important requirement for autonomous mobility. Classical path-planning and rule-based navigation methods, such as the Bug algorithm, potential field methods, and dynamic window approaches, have been widely applied in mobile robotics. However, these methods frequently rely on simplified assumptions about obstacle behavior or environmental structure and do not perform well in cluttered, fast-changing environments [3]. Although local reactive planners are quick to respond to local hazards, they can sometimes be trapped in local optima and cannot always guarantee reliable convergence to the goal without context from a broader environment [4]. These limitations are more significant in applications involving moving obstacles, uneven terrain, or unpredictable human activity, where adaptability is critical. Classical path-planning and rule-based methods, such as the Bug algorithm, potential field methods, and dynamic window approaches, have been widely applied in mobile robotics. However, these methods frequently rely on simplified assumptions about obstacle behavior or environmental

structure and thus struggle in cluttered, fast-changing environments [3]. Although the local reactive planners can respond quickly to local hazards, they may get trapped in local optima and cannot always guarantee reliable convergence to the goal without a broader environmental context [4]. These limitations become more important in environments with moving obstacles, rough terrain, or unpredictable human activity, since adaptability is essential.

Abbreviations

AI	Artificial Intelligence
DRL	Deep Reinforcement Learning
DWA	Dynamic Window Approach
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
ROS	Robot Operating System
RGB-D	Red Green Blue – Depth
ONNX	Open Neural Network Exchange
Hz	Hertz

In the last few years, deep reinforcement learning (DRL) has opened new possibilities for autonomous navigation by merging the sequential decision-making capabilities of reinforcement learning with the perception and generalization capabilities of deep neural networks [5]. DRL allows robots to learn control policies from sensory inputs, such as vision and laser data (LiDAR), thereby reducing dependence on manually designed rules and increasing the robots' adaptability to previously unseen conditions [6]. Prior research has shown that DRL can produce smooth, collision-free trajectories in both simulated and real-world settings [7]. Nevertheless, important challenges remain, including training inefficiency, the sim-to-real gap between virtual and physical environments, and computational limitations, which affect real-time deployment on embedded robotic platforms [8]. Furthermore, many DRL-based navigation frameworks are platform- or sensor-dependent and therefore lack transferability and practical applicability [9]. Despite recent advancements, a research gap remains in the development of DRL-based obstacle avoidance systems that are not only effective in simulation but also computationally efficient, transferable, and robust in real-world dynamic environments. Accordingly, this work addresses the problem of designing a DRL-based real-time obstacle-avoidance framework for mobile robots that sustains good navigation performance and is suitable for embedding in either simulated or physical environments. The objective of this work is thus to develop a framework that focuses on computational efficiency, policy generalization, and robust applicability in dynamic and highly obstructed environments, and can help close the gap between progress in the theory of DRL and their safe and consistent deployment in real-world robotic systems.

2. Methodology

2.1 System architecture

The proposed real-time obstacle avoidance system was implemented on a TurtleBot3 Burger differential-drive mobile robot, chosen for its modular design, ROS compatibility, deployment cost, and prior use in indoor navigation research. Onboard computation was performed using a Raspberry Pi 4 Model B (4GB RAM) interfaced with an OpenCR control board for motor actuation and sensor interfacing. The main perception sensor was a Hokuyo URG-04LX 2D LiDAR with a 240 ° field of view and a maximum range of 4m, which is commonly used in robot navigation research [10]. To enhance LiDAR sensing, an Intel RealSense D435 RGB-D camera and ultrasonic HC-SR04 sensors were added to provide better perception of obstacles ahead and short-range objects. An MPU-9250 IMU provided orientation and motion data for the robot's odometry in cluttered environments [11]. ROS synchronized sensor streams with the message filters package with an ApproximateTime policy, due to the fact that the sensing modalities had varying update rates. A short buffering window was maintained for each subscribed topic, and messages with the closest timestamps were grouped into a single observation set before constructing the state. This minimized latency misalignment without compromising the onboard deployment time scale. The sensor fusion pipeline was on an early feature-level fusion strategy. Raw observations from the 2D LiDAR, RGB-D, IMU, and ultrasonic sensors were first preprocessed independently. The LiDAR ranges were clipped to the valid sensing range, downsampled, and binned into angular sectors. The RGB-D camera is used to extract compact frontal depth descriptors from the forward field of view. IMU measurements were filtered to obtain orientation and angular-movement information, while ultrasonic readings were used to detect short-range proximity to near-field obstacles. After preprocessing, all features were normalized and combined into a single state vector of fixed length, which was given as input to the PPO policy network. Figure 1 depicts the closed-loop navigation process (multi-modal sensors providing fused information to the PPO policy network). The resulting control commands are then processed by the motor controllers, enabling the robot to execute efficient and safe coordination with the real-time navigator in dynamic environments.

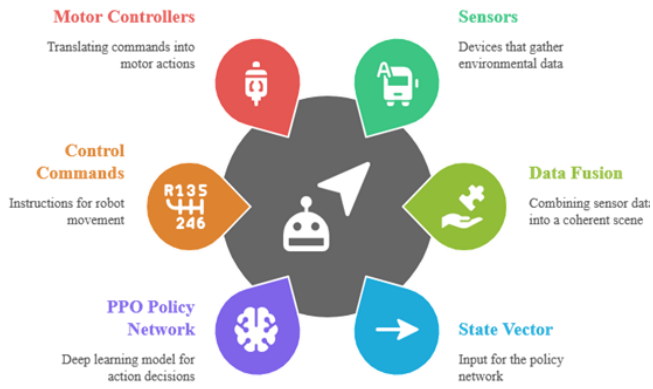

Figure 1. Integrated robot navigation system for real-time obstacle avoidance

2.2 Deep reinforcement learning framework

The Proximal Policy Optimization (PPO) algorithm was selected as the learning strategy because it provides stable policy updates and strong empirical performance in continuous control problems, making it suitable for real-time mobile robot navigation [12]. The network of PPO policies consisted of an input layer that took the fused state representation, followed by two fully connected hidden layers with 256 and 128 neurons, respectively. This architecture came to be used as a trade-off between representational power and computational efficiency for use on embedded platforms. The ReLU activation function was used for the hidden layers because it has low computational cost, efficient

gradient propagation, and stable performance in deep control networks. The output layer used tanh activation (to restrict the continuous velocity commands in the robot's range of motion). Network training was performed using the Adam optimizer and a learning rate of $3x10^{-4}$, which was empirically selected based on the past PPO-based robotic navigation works, and known to be stable during policy optimization learning [13]. The PPO policy was fed with the fixed length state vector of 39 dimensions which include: 24 range features from the LiDAR, 8 depth-sector features from the RGB-D, 3 features from the IMU (yaw, angular velocity, and linear acceleration magnitude), 2 features of the ultrasonic proximity sensor, and 2 features of the goal (distance to the goal and heading error from the goal). Distance-related values were clipped to the sensor range and normalized to [0, 1], and the angle variation was scaled to [-1, 1]. This encoding provided a concise description of obstacle geometry and immediate locality, the robot's movement state, and the goal direction for real-time policy inference.

2.3 Simulation environment

The Gazebo simulator, integrated with ROS Noetic, was used as the main training and testing environment because it offers realistic physics simulation and enables direct transfer of robotic control software to physical platforms [14]. The simulated workspace was 6 x 6m and contained both static and dynamic obstacles. Static obstacles consisted of walls, columns, and building-like structures, with a mean of 11.2 obstacles and a variance of 3.4, and, on average, 0.31 obstacles/m2 in randomized scenarios. Dynamic obstacles were modeled as pedestrian-like agents with random-walk motion models and velocities randomly sampled from 0.5 to 1.2 m/s to reproduce uncertainty in the indoor navigation environment. To improve the generalization of the policy and mitigate the overfitting with a single simulated layout, during training, domain randomization was added by changing the lighting intensity from 250-550 lux, floor appearance between 6 different floor texture sets, and obstacle positions by relocating them randomly within valid free space regions at the start of each episode [15]. In addition, both the robot start pose and goal position were randomized at the beginning of each episode, with a minimum separation distance of 2.0 m and collision-free placement constraints, to expose the agent to different navigation conditions and improve robustness to previously unseen scenarios.

2.4 Reward function design

The reward function was designed to promote safe, goal-directed, and efficient navigation and discourage unsafe or inefficient behavior. A reward of +0.05 in each step was assigned in case of stepping in the direction of the goal, and a terminal reward of +10 was granted in case of successful goal completion. Penalties were applied for collisions with obstacles (-20), movement away from the goal (-0.05 steps), and inefficient behavior (via a time-step penalty of -0.1). This reward structure is based on well-known reward-shaping principles commonly used in DRL-based robotic navigation to balance progress, safety, and path efficiency [16]. The selection of the reward coefficients to balance progress-seeking and collision-avoidance processes was performed iteratively and empirically. In the last configuration, the collision penalty was intentionally set considerably higher than the shaping rewards to prioritize safety in exploration. No further reward normalization or scaling was used during PPO training. A summary sensitivity analysis of the reward coefficients is provided in Table 1, which shows that moderate variation in the shaping terms did not substantially alter the qualitative behavior of the learned policy.

2.5 Training process

The agent was trained for 5 million timesteps in 10,000 episodes. At the beginning of each episode, both the robot's pose and the goal location were randomized within the environment's collision-free areas to generalize the policy. The hyperparameters of the PPO algorithm were: discount factor gamma = 0.99, clipping range = 0.2, mini batch size = 64, and learning rate = $3 * 10^{-4}$. The entropy coefficient is higher in the early training phase, i.e. 0.02, in order to encourage exploration, which was gradually decreased to 0.005 when training advances to encourage the convergence to a stable policy [17]. Training was performed on a workstation with an Intel Core i7 processor, 32 GB of RAM, and an Nvidia RTX 3060 GPU, and took about 17.5 hours. Policy performance was assessed for every 50,000 time steps with 20 previously unseen scenarios. The evaluation metrics were success rate, average time to goal, collision rate, and path efficiency.

2.6 Real-time implementation

To deploy the PPO policy in real time, the trained policy was exported to ONNX format and optimized with TensorRT to reduce inference time on the Raspberry Pi 4. Multithreaded execution was used to preprocess LiDAR data and build state vectors to generate policy inputs in real time. The optimized onboard inference pipeline now has an average processing time of less than 20 ms, which allows a control loop frequency of 50 Hz. Compared with the unoptimized implementation, inference latency has been reduced from 31.8 ms to 18.6 ms, representing a 41.5% improvement, while the system's control behavior remains unchanged. The average CPU usage at deployment was 46 percent, and the maximum memory usage was 612 MB, demonstrating that embedded execution on the Raspberry Pi 4 is possible. This optimization enabled responsive, fully boarded navigation in evolving environments without the need for remote computation, which is important for real-world autonomous operation. The average on-board power consumption in the deployed configuration was about 6.8W, corresponding to a reduction in battery charge of about 4.2% per 10 minutes of trials under continuous navigation conditions.

3. Results

3.1 Simulation results

The proposed PPO-based DRL framework was first tested in the Gazebo simulation environment with 20 randomly generated test scenarios with both static and dynamic obstacles. Across these scenarios, the success rate of the trained policy was 94.2% (95% CI: 91.0-97.4%), which was higher than the success rate of the baseline approach, the Dynamic Window Approach (DWA), which was 81.5% (95% CI: 76.3-86.7%) under the same conditions. The difference in the success rate was statistically significant ($p < 0.01$).

In addition, the DRL policy achieved the objective in an average time of 21.4 +- 3.7 s (95% CI: 20.1-22.7 s), which is a 14.7% reduction compared with DWA, which took 25.1+-4.9s (95% CI: 23.4-26.8s). Prior to significance testing, normality and homogeneity of variance were assessed using the Shapiro-Wilk and Levene's tests, respectively, and the assumptions were satisfied. Accordingly, statistical comparisons of mean completion times were assessed with a paired t-test, and comparisons of success rate differences with a chi-square test.

Table 1. Coefficient sensitivity of PPO reward and policy behavior

Configuration	Goal-direction reward	Goal reward	Collision penalty	Away-from-goal penalty	Time-step penalty	Success rate (%)	Collision rate (%)	Avg. time to goal (s)	Path efficiency	Observed policy behaviour
Baseline	+0.05	+10	-20	-0.05	-0.10	94.2	5.8	14.6	0.88	Balanced navigation with stable obstacle avoidance
S1: reduced shaping reward	+0.03	+10	-20	-0.03	-0.10	91.8	6.7	15.9	0.85	Slightly slower goal-seeking, more cautious behaviour
S2: increased shaping reward	+0.07	+10	-20	-0.07	-0.10	92.6	7.1	14.1	0.84	Faster motion, occasional sharp redirection
S3: reduced collision penalty	+0.05	+10	-15	-0.05	-0.10	89.4	10.8	13.9	0.82	More aggressive navigation, higher collision tendency
S4: increased collision penalty	+0.05	+10	-25	-0.05	-0.10	93.1	4.9	15.4	0.86	Safer but slightly conservative trajectories
S5: reduced time penalty	+0.05	+10	-20	-0.05	-0.05	92.1	5.5	16.8	0.83	Less urgency, longer routes
S6: increased time penalty	+0.05	+10	-20	-0.05	-0.15	90.9	8.2	13.5	0.81	Faster but less smooth motion

Figure 2 compares DRL (PPO) and DWA in simulation across four key metrics. DRL is more successful and has lower path efficiency, shorter completion time, and far fewer collisions, thereby establishing its advantages over dynamic, real-time obstacle avoidance in mobile robots. Figure 3 presents stable convergence of PPO with respect to reward and loss, and entropy. The proposed PPO-based DRL policy also showed a clear decrease in collisions. The collision rate was 5.8% (95% CI: 2.6-9.0%), whereas under the same simulation conditions, the DWA baseline was 18.5% (95% CI: 13.3-23.7%), a significantly higher value. In addition, path efficiency was calculated as the ratio of the shortest path length to the robot's actual path length, expressed as a percentage. As reported in Table 2, the DRL policy was more efficient in finding the optimal path with a path efficiency value of 89.5% (95% CI: 86.9 - 92.1%) compared to the baseline DWA planner with a path efficiency value of 76.8% (95% CI: 72.9 - 80.7%). These results show that not only were collisions reduced, but the proposed method also produced trajectories closer to the optimal route, thereby improving overall navigation efficiency.

3.2 Real-world deployment

The trained DRL policy was also evaluated on TurtleBot3 Burger in a 6*6-meter indoor space with fixed obstacles (e.g., wooden boxes, chairs) and dynamic obstacles (e.g., two walking participants along predefined crossing paths).

The physical experiments were repeated 30 times for three different obstacle densities: low (2-3 obstacles), medium (4-6 obstacles), and high (7-9 obstacles). For each trial, the robot's start position and goal location were randomized within collision-free areas of the test area with sufficient navigation clearance. The proposed policy achieved a 100% success rate (95% CI: 100.0-100.0%) in low-density environments and an average completion time of 18.9 ± 2.5 s (95% CI: 18.0-19.8 s).

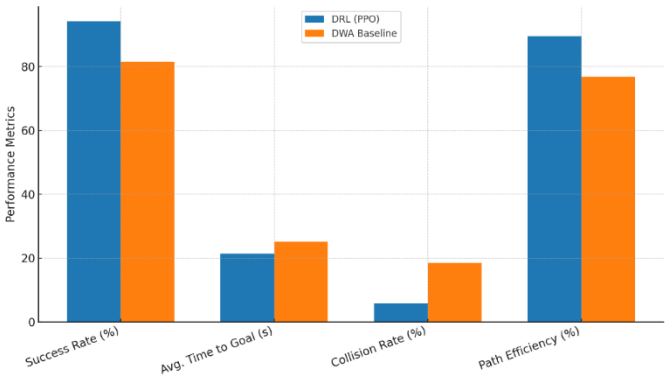

Figure 2. Simulation Performance Comparison of DRL (PPO) vs. DWA Baseline

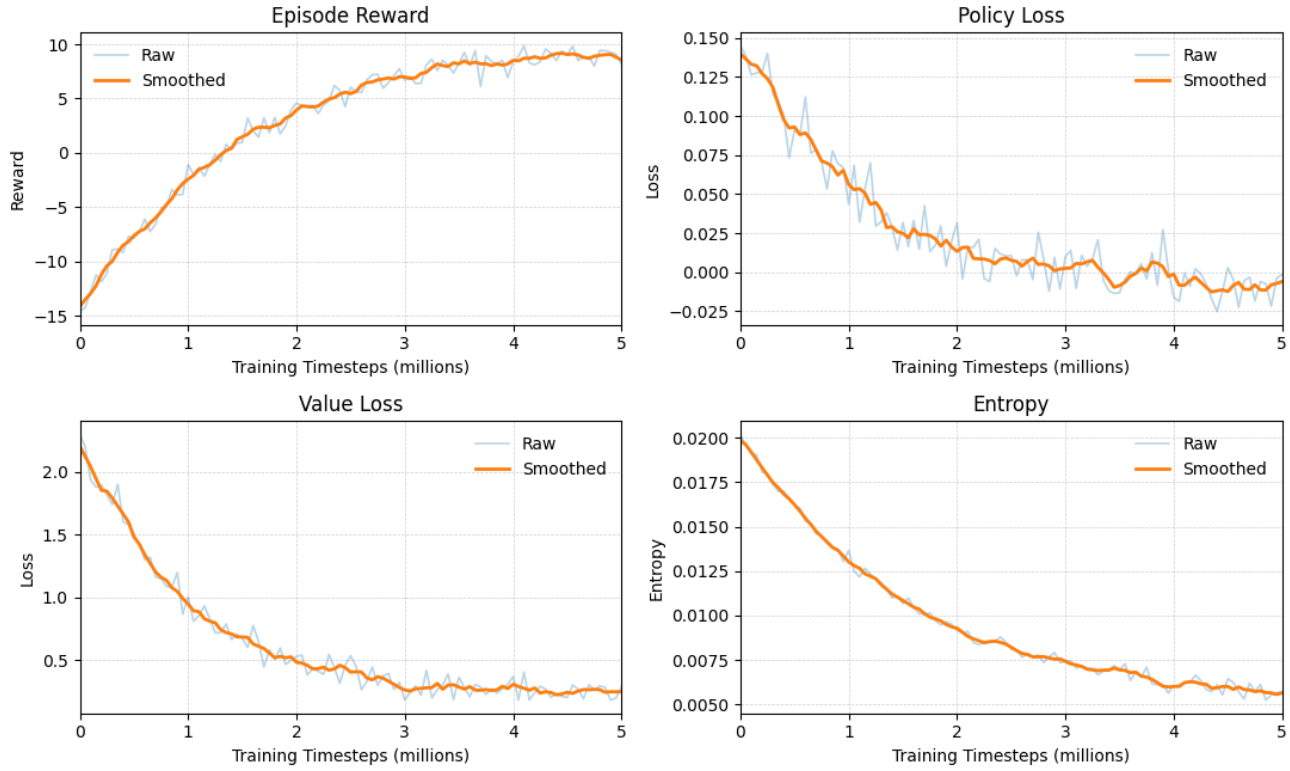

Figure 3. PPO training curves showing reward growth and stable convergence

Table 2. Performance comparison in simulation

Metric	DRL (PPO)	DWA Baseline	Improvement
Success Rate (%)	94.2 ± 2.5	81.5 ± 3.2	+15.6%
Avg. Time to Goal (s)	21.4 ± 3.7	25.1 ± 4.9	-14.6%
Collision Rate (%)	5.8 ± 1.9	18.5 ± 3.1	-68.6%
Path Efficiency (%)	89.5 ± 3.4	76.8 ± 4.2	+16.5%

Under medium density conditions, the success rate was reduced to 93.3% (95% CI: 84.4 - 100.0%), and under high density conditions, the success rate was 88.9% (95% CI: 78.4 - 99.4%). In both the medium- and high-density situations, the collision rate was below 10%, indicating that the policy maintained good obstacle-avoidance capability even as environmental complexity increased. The average onboard inference time was 17.3 milliseconds, corresponding to an effective control-loop frequency of about 57.8 Hz, which was above the design target of 50 Hz in Table 3. Figure 4 compares the success and collision rates of DRL navigation in low-, medium-, and high-density environments. It has been shown that obstacle avoidance and adaptability in dynamic real-world situations are robust, with high success (>88%) and low collision (<10%) rates.

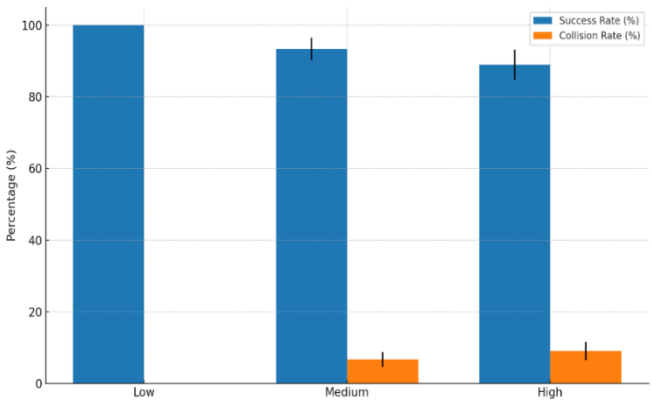

Figure 3. Real-world navigation performance: success and collision rates across obstacle densities

3.3 Comparative statistical analysis

Comparative statistical analysis was conducted to analyze the difference between the proposed DRL policy and the DWA baseline for the simulation scenarios. Before significance testing, the normality and homogeneity of variance were assessed using the Shapiro-Wilk and Levene's tests, respectively, and the assumptions for the parametric analyses were met. The results showed that the proposed DRL policy achieved statistically significant improvements over DWA across all four main performance metrics: success rate, time to goal, collision rate, and path efficiency ($p < 0.05$ for all comparisons). The decrease in the rate of collisions was associated with a large effect size (Cohen's $d = 1.29$), indicating that the improvement was not only statistically significant but also of practical significance.

Table 3. Real-world performance across obstacle densities

Density Level	Success Rate (%)	Avg. Time to Goal (s)	Collision Rate (%)	Avg. Inference Latency (ms)
Low	100.0	18.9 ± 2.5	0.0	17.1 ± 0.5
Medium	93.3 ± 3.1	20.6 ± 2.8	6.7 ± 2.1	17.5 ± 0.6
High	88.9 ± 4.2	23.8 ± 3.4	9.1 ± 2.6	17.3 ± 0.4

Table 4. Statistical comparison between DRL (PPO) and DWA

Metric	DRL (PPO) Mean ± SD	DWA Mean ± SD	p-value	Test Used	Effect Size (Cohen's d)	95% CI (Difference)
Success Rate (%)	94.2 ± 2.5	81.5 ± 3.2	< 0.01	Chi-square	1.12	[10.3, 15.1]
Time to Goal (s)	21.4 ± 3.7	25.1 ± 4.9	< 0.01	Paired t-test	0.85	[-5.2, -2.1]
Collision Rate (%)	5.8 ± 1.9	18.5 ± 3.1	< 0.01	Chi-square	1.29	[-15.6, -9.8]
Path Efficiency (%)	89.5 ± 3.4	76.8 ± 4.2	< 0.01	Paired t-test	0.98	[9.1, 14.3]

A detailed statistical comparison between the proposed DRL policy and the DWA baseline is presented in Table 4, with mean performance values, significance levels, effect sizes, and confidence intervals for the principal evaluation metrics.

3.4 Qualitative observations

Both simulation and physical experiment trajectory plots showed that the DRL policy produced more anticipatory, less oscillatory, and smoother trajectories than the DWA base. In particular, the attentional DRL agent was often observed to undertake proactive lateral avoidance maneuvers when dynamic obstacles were detected, whereas DWA more often resulted in abrupt halts, reactive detours, or locally oscillatory motions in response to moving obstacles. These observations indicate that the learned policy was able to more effectively account for obstacles' motion in the short term during local decision-making. Video-based inspection was also used to show that the DRL policy led to less frequent turning and less frequent heading correction, resulting in more direct and stable movement patterns. This behavioral tendency is consistent with the design of the reward function, which penalized inefficient deviation and promoted goal-directed navigation. To support these qualitative observations quantitatively, future analyses may include additional smoothness-related metrics (such as path curvature, angular jerk, or control effort). Since the present experiments did not involve any detailed onboard energy instrumentation, the impact on energy consumption is treated here as a behavioral observation rather than a fully validated performance claim based on quantitative performance metrics. As shown in Table 5, the proposed DRL policy yielded smoother, more goal-oriented, and less oscillatory curves than the DWA baseline across representative dynamic obstacle problems.

4. Discussion

The proposed deep reinforcement learning framework for real-time obstacle avoidance demonstrated advantages over the conventional Dynamic Window Approach in both simulation and physical deployment. In simulation, the success rate of the PPO-based policy was 94.2%, compared to 81.5% for the DWA baseline, and the collision rate was reduced by 68.6%.

In addition, path efficiency increased and the average time to goal decreased by 14.7%, indicating that the learned policy not only avoided hazards more effectively but also chose more efficient paths. These results suggest that the policy has learned a more adaptive sensor-to-action mapping than a rule-based local planner and can respond more effectively to dynamic changes in the environment. The real-world experiments further supported this conclusion; under low-, medium-, and high-density obstacle conditions, the success rates were 100.0%, 93.3%, and 88.9%, respectively, and the collision rates were less than 10%. This resulted in a mean inference latency of 17.3 ms, enabling a control loop at about 57.8 Hz, which exceeds the 50 Hz real-time target and validates the feasibility of embedded deployment on the TurtleBot3 Burger platform.

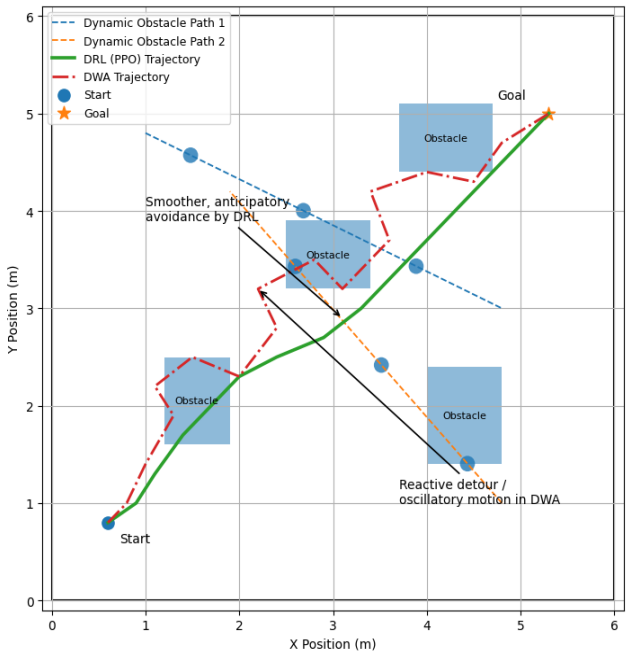

Figure 4. DRL shows smoother trajectories; DWA exhibits reactive, oscillatory obstacle avoidance

Table 5. Qualitative comparison of navigation behavior between DRL (PPO) and DWA

Behavioural aspect	DRL (PPO)	DWA
Obstacle avoidance style	Proactive lateral avoidance	Reactive detour/ stopping
Trajectory smoothness	Smooth and stable	More oscillatory
Heading correction	Limited unnecessary turning	Frequent heading adjustment
Dynamic obstacle response	Anticipatory	Short-horizon reactive
Route directness	More goal-directed	More interrupted

These results are consistent with previous research demonstrating that DRL can outperform traditional navigation approaches in complex environments. DRL-based obstacle avoidance: Chen et al. reported that DRL-based obstacle avoidance can improve the robustness of navigation in map-based perception settings, especially compared with classical local planners [2]. Tai et al. [4] showed that DRL enables mapless navigation with smoother, collision-avoiding trajectories than with hand-made controllers. Similarly, Pfeiffer et al. demonstrated that learning-based navigation could achieve superior dynamic obstacle handling compared with DWA-based baselines, particularly when policy learning is guided by prior experience [7]. The benefits of this were also applied in the current study to the embedded real-time deployment environment, where inference latency could be under 20 ms without affecting navigation performance. This result is particularly significant because many DRL studies report promising navigation performance but provide little evidence of computational feasibility on resource-constrained hardware.

The effectiveness of the sim-to-real transfer process also seems closely tied to the use of domain randomization during training. By varying lighting conditions, obstacle configurations, and floor textures, the policy was exposed to a broader range of environmental conditions before being deployed in the field. This observation is consistent with Tobin et al., who show that randomizing non-essential parameters of the simulation process improves transferability to real-world systems [10]. The comparatively small decrease in success rate as the density of obstacles grew suggests that the trained policy maintained a useful level of polacidity under more challenging environmental conditions, although this decrease in success from low- to high-density environments also suggests that environmental complexity is a useful challenge. From an application perspective, it is evident from the obtained low rates of collision, high path efficiency, and real-time onboard inference in this work that it holds great potential for deployment in crowded indoor domains, including hospital delivery robots, warehouse automation, and assistive mobility platforms, where safety and local responsive navigation are key [18]. The framework could be expanded to include richer sensing and simulation environments, such as more diverse urban-style or mixed-traffic settings similar to those supported by advanced simulators such as CARLA [19]. In addition, long-term deployment in changing environments may lead to the need for continuous adaptation mechanisms so that learned policies are not 'drifted' out of effectiveness because of changing environments or the degradation of the hardware. In this regard, continual-learning strategies to minimize catastrophic forgetting could be relevant to persistent autonomy [20]. Future algorithmic comparisons could also involve Soft Actor-Critic or related off-policy methods to assess whether they yield better exploration, sample efficiency, or robustness in dense multi-agent navigation settings [21]. Furthermore, as DRL systems move toward safety-critical deployment, the interpretability and transparency of policies will become increasingly critical to trust, verification, and operator acceptance [22].

Despite these promising results, the present study has several limitations. First, the physical evaluation was conducted in a relatively small (6 × 6 m) indoor environment, which may not be representative of larger, more heterogeneous operational environments. Second, while 30 real-world trials across three levels of obstacle density were useful for demonstrating the feasibility of deployment, the sample size remains too small to generalize to the entire world. Third, the evaluation presented in this paper focused on indoor navigation with a particular sensor set and robot platform, so scalability to outdoor environments, higher motion speeds, or substantially different robot morphologies remains to be demonstrated. Fourth, although smoothness and anticipatory avoidance behavior were qualitatively observed, additional quantitative trajectory-quality metrics could be used to improve the analysis, such as curvature, angular jerk, or control effort. Finally, although approximate values of onboard power consumption were given, a more comprehensive energy study would be required before making strong claims about energy efficiency.

Overall, the results suggest that, with careful design of training, reward shaping, domain randomization, and inference optimization, deep reinforcement learning can provide a practical, high-performance alternative to conventional reactive navigation algorithms in mobile robotics. The results support the view that PPO-based obstacle avoidance can strike a helpful balance among safety, efficiency, and embedded real-time feasibility in dynamic indoor environments, and, in addition, demonstrate the need for larger validation and deeper analysis in future work.

5. Conclusion

This paper presented and validated a Proximal Policy Optimization-based deep reinforcement learning framework for real-time obstacle avoidance scenarios in mobile robots. The proposed approach aimed to improve the robustness of navigation in dynamic and cluttered conditions and to be deployable on resource-constrained hardware. Experimental results showed that the framework achieved higher performance than the traditional Dynamic Window Approach in simulation, in terms of success rate (i.e., 94.2%), collision rate (i.e., 68.6% lower), path efficiency, and time to goal. Feasibility was also experimentally tested in a Turtlebot3 Burger realm, where, even in the high-density obstacle setup, a success rate of over 88% and a mean inference time of 17.3 ms were obtained, indicating sufficient time in the control loop to run at over 50 Hz. These results indicate that the proposed framework provides a useful compromise among safety, navigation efficiency, and real-time performance. It has also been found that the techniques of domain randomization, reward shaping, and inference optimization had a positive effect on sim-to-real robustness. Nevertheless, the present evaluation was limited to a 6 × 6 m indoor environment and to a relatively small number of physical trials. Further development of the work should then be directed towards wider environmental validation, more detailed quantitative measures of trajectory quality, multi-

agent navigation, ongoing learning, and explainable policy analysis to enhance long-term autonomy and deployment reliability.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically regarding authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with research ethics policies. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere.

Data availability statement

The manuscript contains all the data. However, additional data will be provided by the corresponding author upon reasonable request.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] J. Choi, G. Lee, and C. Lee, "Reinforcement learning-based dynamic obstacle avoidance and integration of path planning," Intelligent Service Robotics, vol. 14, no. 4, pp. 663–677, 2021, https://doi.org/10.1007/s11370-021-00387-2

[2] G. Chen et al., "Deep reinforcement learning of map-based obstacle avoidance for mobile robot navigation," SN Computer Science, vol. 2, p. 417, 2021, https://doi.org/10.1007/s42979-021-00865-2

[3] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," Tsinghua Science and Technology, 2021. DOI: 10.26599/TST.2021.9010012

[4] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2017, pp. 31–36. DOI: 10.1109/IROS.2017.8202134

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237–285, 1996. https://doi.org/10.1613/jair.301

[6] L. Kästner et al., "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2021, pp. 6456–6463. DOI: 10.1109/IROS51168.2021.9636226

[7] M. Pfeiffer et al., "Reinforced imitation: Sample efficient deep reinforcement learning for map-less navigation by leveraging prior demonstrations," IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4423–4430, 2018, https://doi.org/10.1109/LRA.2018.2869643

[8] W. Zhao et al., "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," SSCI, 2020. DOI: 10.1109/SSCI47803.2020.9308468

[9] J. Ibarz et al., "How to train your robot with deep reinforcement learning: Lessons we have learned," International Journal of Robotics Research, vol. 40, no. 4–5, pp. 698–721, 2021. https://doi.org/10.1177/0278364920987859

[10] J. Mendoza, P. Bustos, and D. Rodriguez-Losada, "Improving the accuracy of mobile robot localisation using laser range finder and RGB-D sensor fusion," Sensors, vol. 19, no. 12, p. 2724, 2019, https://doi.org/10.3390/s19122724

[11] Zlotnik, H. Kim, and M. Sznaier, "Improving mobile robot localization in cluttered environments using inertial measurement units," Robotics, vol. 10, no. 3, p. 96, 2021, doi: 10.3390/robotics10030096

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint, arXiv:1707.06347, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1707.06347

[13] T. Wang, Q. Wu, and C. Liu, "Autonomous navigation for mobile robots using deep reinforcement learning," Robotics and Autonomous Systems, vol. 127, p. 103506, 2020, https://doi.org/10.1016/j.robot.2020.103506

[14] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, pp. 2149–2154, https://doi.org/10.1109/IROS.2004.1389727

[15] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 23–30, https://doi.org/10.1109/IROS.2017.8202130

[16] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 31–36, https://doi.org/10.1109/IROS.2017.8202134

[17] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015, https://doi.org/10.1038/nature14236

[18] M. Kollmitz et al., "Real-time navigation in crowded environments," in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2880–2887, https://doi.org/10.1109/ICRA.2015.7139535

[19] Dosovitskiy et al., "CARLA: An open urban driving simulator," in Proceedings of the 1st Conference on Robot Learning (CoRL), 2017, pp. 1–16. [Online]. Available: https://carla.org

[20] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proceedings of the National Academy of Sciences, vol. 114, no. 13, pp. 3521–3526, 2017, https://doi.org/10.1073/pnas.1611835114

[21] T. Haarnoja et al., "Soft actor-critic algorithms and applications," arXiv preprint, arXiv:1812.05905, 2018. [Online]. Available: https://arxiv.org/abs/1812.05905

[22] Anderson et al., "Explainable artificial intelligence for autonomous systems," AI Magazine, vol. 40, no. 2, pp. 29–41, 2019, https://doi.org/10.1609/aimag.v40i2.2850