International Journal of Robotics and Control Systems
Vol. 5, No. 5, 2025, pp. 2454-2468
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Deep Q-Learning with Custom Reward Shaping for Mobile Robot Navigation in Grid Dynamic Environments

Karam A. Al-Zubaidi [a,1], Ahmed M. Alkamachi [a,2,*], Bahaa Ansaf [a,3]

[a] Department of Mechatronics Engineering, Al-Khawarizmi College of Engineering, University of Baghdad, Baghdad, Iraq
[1] karam.abd2302@kecbu.uobaghdad.edu.iq; [2] ahmed78@kecbu.uobaghdad.edu.iq; [3] bahaa.ansaf@uobaghdad.edu.iq
* Corresponding Author

ARTICLE INFO

Article history
Received September 10, 2025
Revised October 13, 2025
Accepted November 17, 2025

Keywords
Grid-world Navigation;
Dynamic Obstacles;
Reward Shaping in DRL;
Discrete Action Space;
Simulation Limitations

ABSTRACT

One of the difficulties facing robots is navigating in dynamic and unstable environments. Traditional algorithms often fail to plan paths in environments containing dynamic obstacles. Therefore, this study proposes a framework for deep learning in a grid-based environment containing static and dynamic obstacles. The research contribution is summarized in designing an equivalent function that encourages reaching the goal and reduces unnecessary movements. It also assesses the impact of altering obstacle velocities on navigation performance and compares the DQL and DWA approaches. A deep Q-network with two hidden layers is trained using a greedy policy. Simulated LiDAR sensor used for spatial perception. The model is tested in four simulated environments of increasing difficulty, incorporating both static and dynamic obstacles to mimic realistic conditions. The agent that operates by the approach DQL achieved a complete success rate of 100% in environments A, B, and C, and a 90% success rate in environment D, in contrast to the DWA, which recorded success rates of 100%, 90%, 70%, and 60% respectively. Furthermore, DQL demonstrated sustained high performance despite increases in obstacle speed, achieving success rates of 100%, 90%, 70%, and 40% as the speed escalated to six times that of the robot's velocity. The results showed that the DQL approach outperformed in success rates and path efficiency and maintained stability in complex environments. In summary, the DQL framework proposed herein presents a robust and scalable approach for mobile robot navigation, clearly demonstrating significant advantages over conventional methodologies in intricate and unpredictable environments.

1. Introduction

Despite advancements in sensing and computing technology, autonomous driving systems such as mobile robots and autonomous cars have gained significant scholarly interest and are becoming more commercially viable [1]-[3]. Today, the mobile robot is an important component of social progress. They do dangerous and difficult activities for people, including search and rescue, help with epidemic exposed areas and search for distant planets. As a result, establishing the robot's course is a

critical part of its design [4]. In an era of industry-wide intelligence, it is critical to apply an intelligent route-planning algorithm to the field of robot path planning [5]-[7].

One of the most difficult difficulties for mobile robots is self-navigation in dynamic and unstable environments. This differs from scenarios with permanent barriers, in which the robot is familiar with both the impediments and the surroundings. In dynamic environments, the mobile robot's experience is limited, and impediments shift in unforeseen ways [8].

Steering for mobile robots to avoid obstacles has been the subject of many published studies by scholars. In [9]-[11], mobile robots planned their routes using the Dynamic Window Approach (DWA) algorithm. Another research study employed the D* algorithm to determine the course of a mobile robot [12]. There are several prominent pathfinding methods for avoiding static obstacles, such as the Timed-Elastic-Bands (TEB), EBS-A*, A* algorithm, Dijkstra, Probabilistic Roadmap (PRM), Potential Field Method (PFM), RRT, genetic algorithm (GA), Ant Colony Optimization (ACO), and Bi-Population Particle Swarm Optimization (BPPSO) [13]-[23]. Traditional algorithms, such as the Dynamic Window Approach (DWA), face significant obstacles because they assume they can predict obstacle paths within a short time [24], which makes these methods that depend on deterministic or organized motion assumptions are inadequate in situations where barriers move randomly, and this assumption restricts their usefulness in such circumstances [25].

In terms of avoiding moving obstacles, deep reinforcement learning systems outperform classical algorithms [26]. Research on deep reinforcement learning algorithms used for dynamic obstacle avoidance in mobile robots has emerged as a critical area of inquiry due to the increasing demand for autonomous navigation in complex and unpredictable environments [27], [28].

In recent years, there have been significant developments in reinforcement learning, which have contributed to radical changes in path planning and dynamic obstacle avoidance methods, enabling robots to avoid dynamic obstacles efficiently and without prior maps [29]-[31]. The scientific importance of this research is highlighted in the applications of industrial automation, service robots, and self-driving cars, where robots must navigate safely among humans and moving obstacles [32]-[35]. This field has evolved from classical path planning approaches to deep reinforcement learning, including Deep Q-Networks (DQN), Double DQN, and Proximal Policy Optimization (PPO), resulting in more efficient, adaptive, and scalable solutions [36]-[38]. Deep neural networks possess the capability to formulate effective control strategies by utilizing sensor inputs, thereby enabling their adaptation to alterations in the environment without necessitating retraining [39], [40].

With these developments, robots face a problem in avoiding moving obstacles efficiently and reliably due to the inability to predict the movement of obstacles and the limitations of sensor data, and limited generalization to real-world scenarios [41]-[45]. Studies have revealed a knowledge gap in developing algorithms that balance navigation efficiency and safety in real-world environments [46]-[50]. Moreover, there is a conflict of opinion about the best reinforcement learning algorithms and the best reward function design. Some advocate value-based methods such as DQN, while others support policy gradient methods such as TD3, SAC and DDPG for continuous control [51]-[54]. These gaps possess numerous implications, encompassing suboptimal navigational efficacy, heightened probability of collisions, and challenges associated with the transition to practical, real-world implementations [55], [56].

The principal scholarly contribution of the present investigation is the formulation of a navigation framework predicated on Deep Q-Learning (DQL) for a mobile robot agent functioning within a grid-based environment characterized by the presence of both static and dynamic obstacles. The proposed approach introduces a tailored reward function for efficient convergence, discouraging collisions and unproductive actions, and a systematic evaluation under varying obstacle speeds. The work also includes a comparison between DQL with the Dynamic Window Approach (DWA) algorithm and demonstrates the efficiency of each approach in terms of the success rate and number of steps required to reach the goal The simulation integrates both immobile and dynamically relocating impediments, through this, more realistic and increasingly difficult situations are generated. Furthermore, the

emulated LiDAR sensing facilitates spatial cognition, allowing for judicious decision-making predicated on the immediate environment. These augmentations synergistically enhance the agent's capacity to function proficiently in unpredictable and changing conditions.

In the present investigation, Deep Q-Learning (DQL) is chosen as the principal algorithm owing to its comparative straightforwardness and its extensive utilization as a foundational benchmark in the domain of reinforcement learning research. Although more sophisticated algorithms such as Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), or Twin Delayed Deep Deterministic Policy Gradient (TD3) could also be implemented, DQL offers a lucid and interpretable basis for examining the effects of reward architecture, environmental dynamics, and the intricacies of obstacles. This renders it particularly advantageous for establishing a foundational reference that can be further developed in subsequent endeavors toward more advanced methodologies in deep reinforcement learning.

2. Methodology

In this section, the process of training a mobile robot to perform a navigation task in a 15x15 grid simulation environment with static and dynamic obstacles using DQL is outlined. A Custom reward function is employed to direct the mobile robot to its objective by avoiding unnecessary movement to facilitate effective training. Moreover, thanks to the four-directional LiDAR readings, the robot can observe its surroundings without prior knowledge. An ε-greedy policy is employed for the balance of exploration and exploitation, and a two-layer fully connected neural network trained the robot.

2.1. Environment Setup

The mobile robot is trained in a (15×15) grid environment. The robot's initial position in each episode is (0,0), and its goal position is (14,14). The environment contains static obstacles and dynamic obstacles that move randomly in four directions (right, left, up, down). The training episode ends when the mobile robot reaches the goal, collides with a static or dynamic obstacle, or moves 200 steps without reaching the target.

2.2. State Representation

The state (S) is an 8-dimensional vector defined as follows:

$$S = \left[\frac{x_r}{G}, \frac{y_r}{G}, \frac{x_g}{G}, \frac{y_g}{G}, L_1, L_2, L_3, L_4\right] \quad (1)$$

Where (x_r, y_r) is current robot position, (x_g, y_g) is goal position, G is grid size, $(\frac{x_r}{G}, \frac{y_r}{G}, \frac{x_g}{G}, \frac{y_g}{G})$ is normalized robot and goal position, $L_i \in [0, 1]$ is normalized LiDAR reading in direction i (front, back, right, left) where $L_i = \frac{d_i}{R}, d_i$ is the number of cells to the nearest obstacle, and R is the LIDAR range (3 cells).

2.3. Action Space

The robot has four discrete actions $a \in \{0,1,2,3\}$ that map to the movement directions:

$$a = \begin{cases} 0 \rightarrow (1,0) & \text{Move right} \\ 1 \rightarrow (-1,0) & \text{Move left} \\ 2 \rightarrow (0,1) & \text{Move up} \\ 3 \rightarrow (0,-1) & \text{Move down} \end{cases} \quad (2)$$

2.4. Reward Function and Euclidean Distance

2.4.1. Reward Function

In contrast to traditional reward functions that reward a robot only when it reaches a goal, this function is designed to motivate the robot to accomplish the goal and limit useless movements by imposing a penalty on movements that cause the robot to move away from the goal. In return, it provides a small reward when it approaches the goal and severe penalties for collisions, making the mobile robot more efficient at avoiding obstacles. The reward function is delineated as follows:

$$r = \begin{cases} +10.0 & \text{If the robot reaches the goal} \\ -5.0 & \text{If the robot hits a dynamic obstacle or moves out of bounds} \\ +0.2 & \text{If the robot moves closer to the goal} \\ -0.2 & \text{If the robot moves farther from the goal} \\ -0.1 & \text{If the distance to the goal remains the same} \end{cases} \quad (3)$$

2.4.2.Euclidean Distance

To assess if the robot has moved closer or farther from the target, the Euclidean distance between the current robot location and the goal is determined before and after the movement. The Euclidean distance is defined as follows:

$$\partial = \sqrt{(x_r - x_g)^2 - (y_r - y_g)^2} \quad (4)$$

Where (∂) is the robot's Euclidean distance from the target.

2.4.3.Distance Difference

The change in distance $(\Delta\partial)$ is used to evaluate the reward value:

$$\Delta\partial = \partial_{old} - \partial_{new} \quad (5)$$

Where ∂_{old} is the distance to the goal before taking action and ∂_{new} is the distance to the goal after taking action.

Interpretation:

- If $\Delta\partial>0$: The robot is getting closer $\rightarrow +0.2$
- If $\Delta\partial<0$: The robot is moving away $\rightarrow -0.2$
- If $\Delta\partial=0$: No improvement $\rightarrow -0.1$

2.5. Q-Network Architecture

Contrary to (DQL) models, which have simple networks, we used a deep Q-Network (DQN) with two hidden layers (128-64) fully connected, which gives the model a higher ability to learn complex patterns. The Deep Q-Network (DQN) used has the following structure in Fig. 1.

- Input layer: 8 neurons representing the robot's state, which includes normalized positions of the robot and goal, along with Lidar sensor readings.
- Hidden layers: It contains two layers, 128 and 64 neurons, connected in series, and both use the RELU activation function.
- Output layer: 4 neurons corresponding to the Q-values of the four possible actions (up, down, left, right), using a linear activation function.
- Optimizer and loss function: The Adam optimizer is used to train the network with a learning rate of 0.001, and the loss function is mean squared error (MSE).

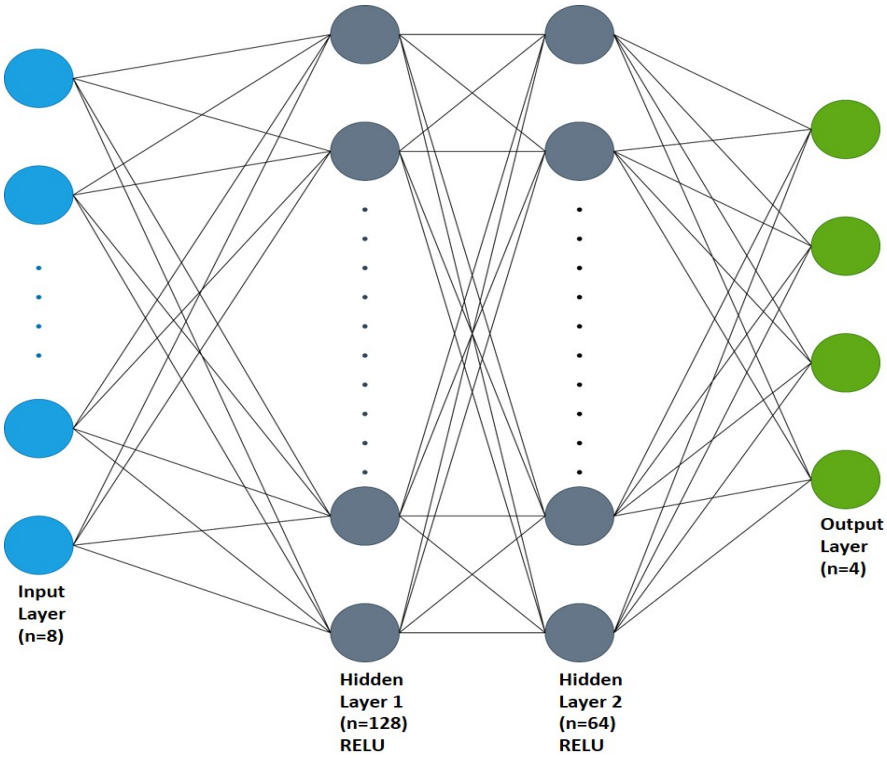

Fig. 1. Structure of the neural network model

2.6. Training Procedure

Training is performed over N=3000 episodes, with each episode limited to a maximum of T=200 steps. At each step, the agent selects an action using the ε-greedy policy:

Start with ε= 0.99, decay by 0.995 per episode, down to 0.01 to become like this:

$$\varepsilon = \max\left(\varepsilon_{\min}, \varepsilon_{*}\varepsilon_{\text{decay}}\right) \tag{6}$$

At each step, the agent selects an action using:

$$a = \begin{cases} \text{If rand } () < \varepsilon & \text{random action} \\ \text{otherwise} & \arg\ \max_a Q(s,a;\Theta) \end{cases} \tag{7}$$

Where $Q(s,a;\Theta)$ is the estimated Q-value for taking action a in state s, representing the expected cumulative future reward and Θ is denotes the trainable parameters (weights and biases) of the Q-network.

Transitions are stored in a replay memory.

A mini-batch is sampled to update the Q-values using the Bellman equation:

$$\text{target} = \begin{cases} \text{It terminal state} & r \\ \text{otherwise} & r + \gamma \cdot \max_{a'} Q(S',a';\Theta) \end{cases} \tag{8}$$

Where r is the instantaneous reward obtained while doing action a in state s, γ is the discount factor (0.95), which controls the significance of future rewards, S' is The next state reached after taking action a, a' is all possible actions that can be taken in the next state S', $max_{a'}Q(S',a')$: The maximum predicted Q-value for the next state S' across all a' possible actions and "terminal state": Refers to a state in which the episode ends, such as reaching the goal or colliding with an obstacle.

The Q-network is updated by minimizing the mean squared error between the target value and the current prediction, using the following loss function:

$$L(\theta) = (\text{target} - Q(s, a; \theta))^2 \quad (9)$$

In Fig. 2, the sequence of operations performed by the DQL algorithm is shown.

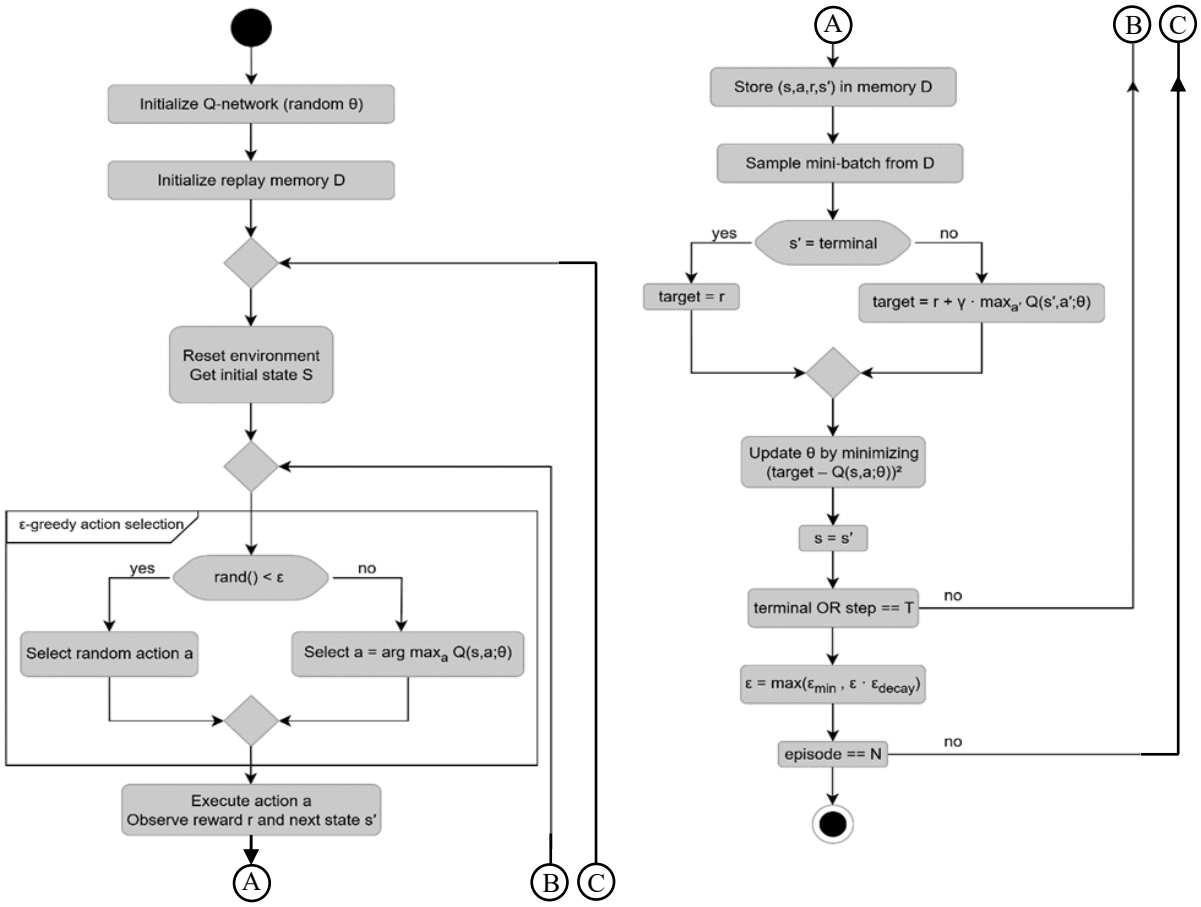

Fig. 2. Flowchart of the deep Q-learning algorithm

2.6.1.Computational Settings

The training parameters are meticulously chosen to guarantee stable convergence during the learning process. The replay memory is configured to accommodate 2000 experiences, with a minibatch size of 32 drawn at each iteration of the update. The Q-network underwent training utilizing the Adam optimizer, which is implemented with a learning rate of 0.001 and mean squared error (MSE) as the loss function. The discount factor γ is maintained at a constant value of 0.95 in order to effectively balance the trade-off between immediate and future rewards.

The system is implemented in Python 3.12.7. The neural network is trained and built using the TensorFlow library, and the simulation environment is designed using the Matplotlib library. All operations are performed on a system provided with an Intel i7 CPU and 8GB of RAM.

3. Results and Discussion

This section reviews the analysis of model training data using the DQL algorithm, evaluates the trained model in four different difficulty environments, and compares the results with a model running with the DWA algorithm. The comparison focuses on two main criteria: the number of successes and the average steps taken to reach the goal in each of the four environments. The results are analyzed,

along with images of the environment and images of the performance of each model in each environment. The performance of the model is evaluated by increasing the obstacle speed.

3.1. Training Analysis

Training data from over 3,000 episodes is analyzed. Fig. 3 illustrates an analysis of the reward values throughout the entire training process. The moving average of the reward values is observed in Fig. 3a. It can be seen that the rewards started from negative values and then began to improve until they stabilized between 12.5 and 13.4, indicating the agent's improvement during the progress of the training episodes. Statistics of reward values across all training episodes are depicted in Fig. 3b.

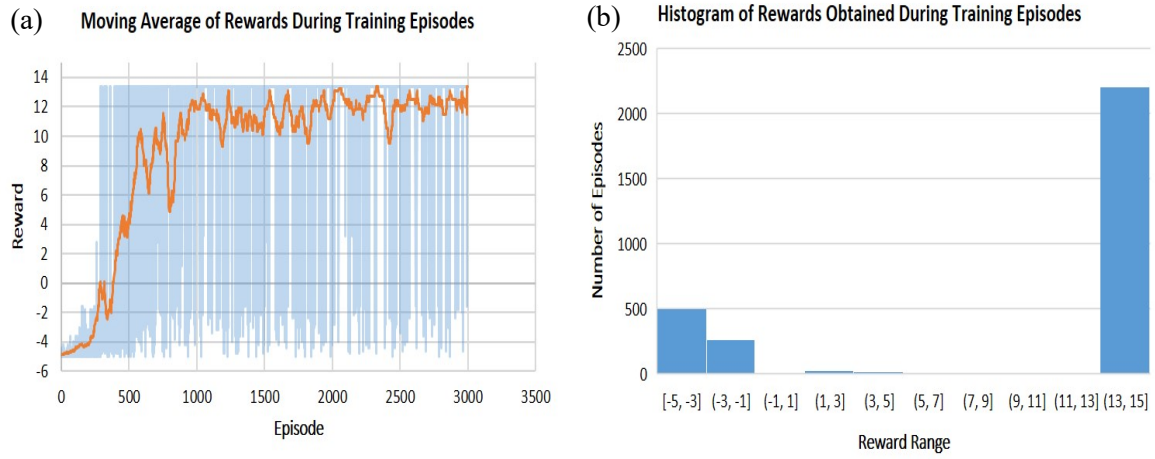

Fig. 3. Reward performance analysis during training: (a) moving average of rewards and (b) histogram of rewards obtained

As shown in Fig. 4, the agent's success rate in reaching the goal increased. From Fig. 4a and Fig. 4b, it is observed that the success rate from episodes 200-300 is 2%, while the success rate in the last 100 episodes of training (2900-3000) reached a success rate of 94%. The success curve as it increases is illustrated in Fig. 4c. Statistics of the number of successes and failures over 3,000 training sessions are shown in Fig. 4d. The green figure represents the number of successes, and the red figure represents the number of failures to reach the goal.

3.2. Experimental Environments

Four (15×15) grid environments of varying difficulty are designed to assess the efficiency of the trained model. Each environment differed in the number of static and dynamic obstacles it contained. Fig. 5 illustrates the environments used in the test and the locations of the obstacles.

3.3. Quantitative Comparison

A comparison is made between the DQL algorithm and the DWA algorithm in the environments shown in Fig. 5, where each algorithm is tested in each environment 10 times, and the comparison is made on the basis of two main criteria: the number of successes and the average steps taken to reach the goal. The average steps is calculated by the sum of the number of steps in successful tests divided by the number of successful tests. Table 1 shows the test results.

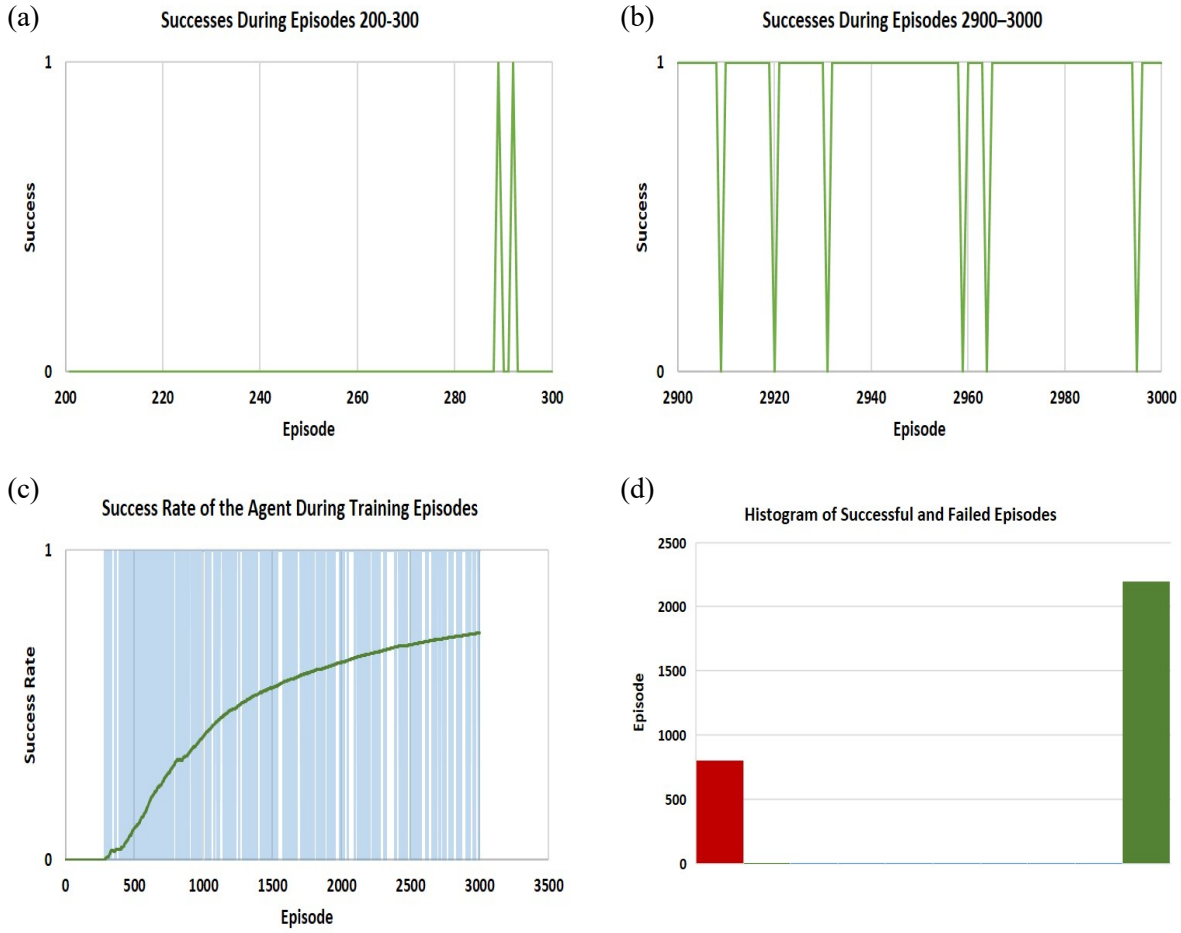

Fig. 4. (a) Successes during episodes 200-300, (b) successes during episodes 2900-3000, (c) success rate of the agent during training episodes, and (d) histogram of successful and failed episodes

3.4. Quantitative Analysis

The experimental results in Table 1 provide a clear view of how the DQL algorithm performs compared to the traditional DWA approach across environments of increasing complexity. In the simpler environments A and B, both algorithms performed well in terms of success rate. The average number of steps for both methods in these settings is close, indicating that neither approach needed to make significant detours. However, the difference becomes more apparent in environments C and D. These maps introduced tighter passages and higher obstacle density, both static and dynamic. DWA performance degraded significantly: its success rate dropped to 70% and 60% in environments C and D, respectively. More notably, in Environment C, DWA required over 114 steps to complete the task in successful trials, indicating inefficient path planning and repeated detours.

Table 1. Comparative performance metrics for DQL and DWA in each test environment

Environment	DQL		DWA	
	Success rate	Average Steps	Success rate	Average Steps
Environment A	10/10	29.6	10/10	29.2
Environment B	10/10	31.6	9/10	30.4
Environment C	10/10	32.6	7/10	114.666
Environment D	9/10	31.75	6/10	36.1

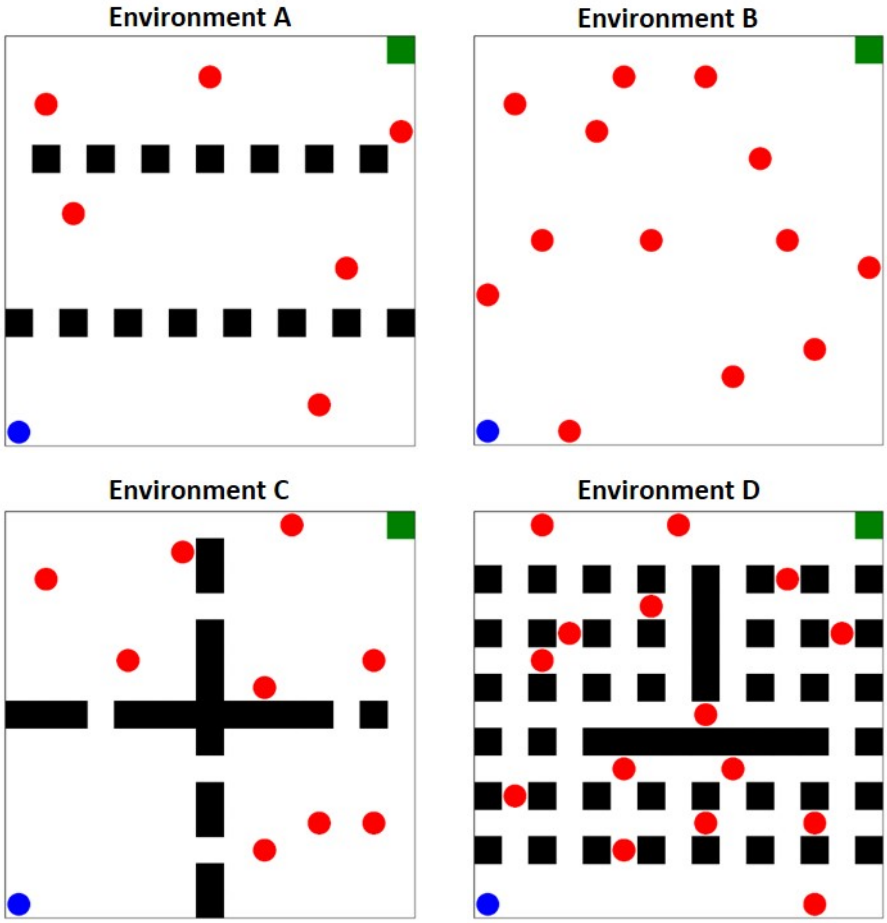

Fig. 5. Testing environments with varying levels of complexity

In contrast, DQL maintained a near-perfect success rate and showed stable performance across all environments. The number of steps increased slightly in more complex maps, but remained within a narrow, consistent range (≈30–33 steps). This consistency suggests that DQL is able to generalize its learned navigation policy and make strategic decisions, even under dynamic and unpredictable conditions. There are a few main reasons why DQL performance has gotten better: it is trained on dynamic obstacles, it uses a state representation that includes LIDAR-simulated directional sensing, and its reward function encourages movement toward goals while punishing collisions or unnecessary wandering. These factors let the agent guess how its actions might affect things beyond its immediate surroundings, which is different from DWA, which only looks at short-term predictions.

In conclusion, the findings show that the suggested DQL method works, especially in situations when standard approaches do not work well. Deep reinforcement learning is strong and flexible enough to handle real-world navigation challenges because it can keep high success rates, stable behaviour, and efficient paths even when things get tough. Fig. 6 illustrates the behaviour of the algorithms in each environment.

The findings substantiate the practicality of the proposed methodology within authentic contexts necessitating instantaneous decision-making amidst fluctuating conditions. For instance, this framework could be utilized in autonomous robotic systems operating in intelligent warehouse environments or internal logistics networks within manufacturing facilities, where the positioning of impediments is in a state of constant flux due to the mobility of personnel or other robotic entities. Furthermore, the system could be applied in service-oriented robots functioning in medical institutions or retail establishments, which must navigate securely amidst individuals and moving apparatus.

The DQL model's ability to maintain stable performance in complex environments demonstrates its suitability for applications that require safe and rapid navigation without the need for a pre-map. However, its practical application requires additional developments, including handling sensor noise generated by real sensors and improving the time response to achieve real-time navigation.

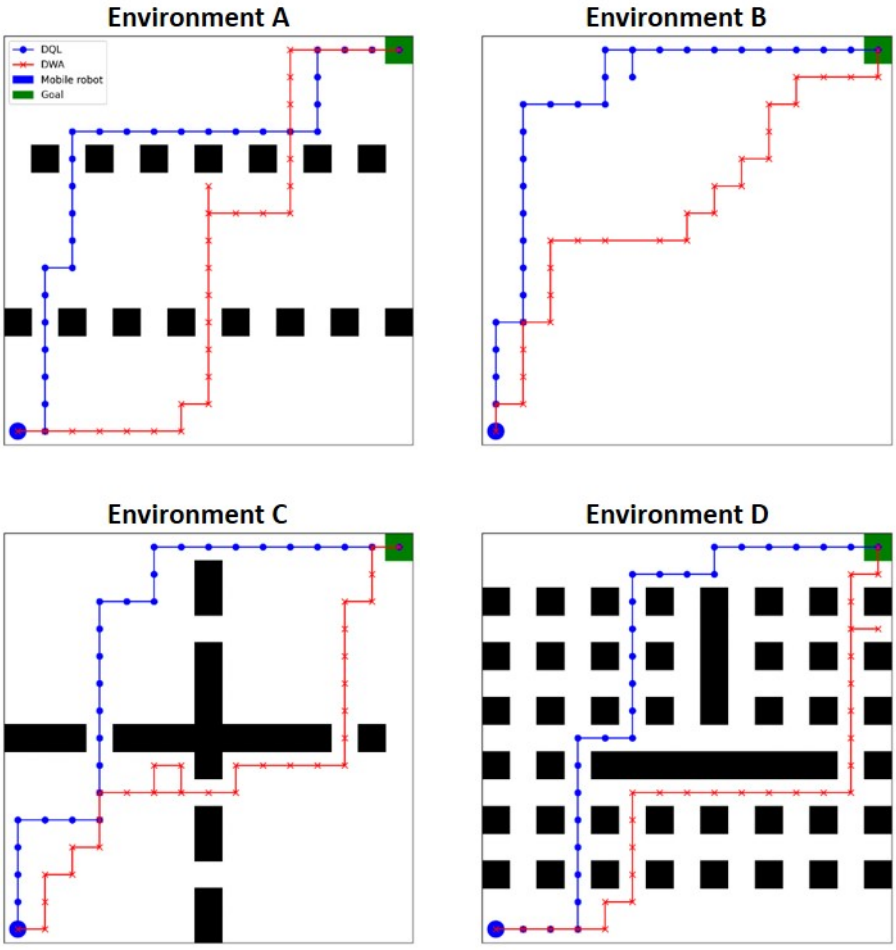

Fig. 6. Trajectories of DQL vs. DWA in multiple environments

3.5. Model Limitations in High-Speed Obstacle Scenarios

To discover the limits of the trained model, it is tested in a 15×15 grid environment with varying speeds of dynamic obstacles. The speed of the obstacles is doubled to 6 times faster than the mobile robot. The experimental results are presented in Table 2. The performance of the trained model when the obstacle speeds changed. When the obstacle speed is the same as the mobile robot speed, the success rate is 100%. The rate decreased to 90% when the obstacle speed is increased to twice the mobile robot speed. The performance of the mobile robot decreased significantly when the speed is increased to 3 times, until the success rate reached 70%. However, when the obstacle speed is increased to 4 times, a sharp drop in the success rate appears, becoming 40%. That is, most attempts failed to reach the target, and the success rate continued to decrease the more we doubled the speed, until it reached complete failure. When the obstacle speed is doubled to 6 times the mobile robot speed, the success rate becomes 0%.

By analyzing the results above, the model maintained ideal performance even when the obstacle speed is doubled, and good performance when the obstacle speed is tripled. Performance degradation at higher obstacle speeds is attributed to the agent's limited sensing horizon and the absence of predictive avoidance in the reward structure. Failures mainly occurred due to sudden collisions with fast-approaching obstacles.

Table 2. Impact of obstacle speed on model success rate.

Obstacle Speed	Success
1x mobile robot speed	10/10
2x mobile robot speed	9/10
3x mobile robot speed	7/10
4x mobile robot speed	4/10
5x mobile robot speed	2/10
6x mobile robot speed	0/10

The experiment opens the door for future enhancements, such as training under variable-speed conditions or using prediction-based modules to improve anticipation. It is important that the gradual change in performance emphasizes the realism of the assessment setup and exposes the agent's possible scalability to the distribution of the more complex real world.

4. Conclusion and Future Work

This study presented a navigation model for a mobile robot operating with theDeep Q-Learning (DQL) algorithm that navigates in a network environment containing fixed and moving obstacles. The research contributions are represented in three points: (i) designing an environment that combines fixed and moving obstacles, (ii) designing a dedicated reward function to encourage the robot to move towards the goal and reduce useless and unsafe movements, (iii) evaluating the model under variable obstacle speeds to assess its robustness.

Test results show that the DQL approach outperforms the DWA approach. Specifically, DQL attained success rates of 100% across environments A–C and 90% in environment D, with stable performance averaging between 30 and 33 steps per trial. In juxtaposition, the success rate of DWA diminished to 70% in environment C and 60% in environment D, accompanied by a considerable increase in trajectory lengths (up to 114 steps). Moreover, during the obstacle speed stress assessment, the proposed system sustained elevated success rates at up to twice the obstacle speed, although a pronounced decline in performance was observed at greater speeds.

Notwithstanding these encouraging findings, various constraints necessitate recognition. The existing configuration is predicated on a grid framework characterized by discrete actions, simplified LiDAR sensing, and the absence of sensor noise or kinematic limitations. Future work will address these limitations through four directions: develop the model using advanced algorithms such as PPO or SAC, integrate prediction modules to estimate the obstacle speed and trajectory, implement the approach in 3D platforms (e.g., Gazebo, ROS) to increase realism and conduct tests on real robots to evaluate robustness under realistic conditions.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] K. Almazrouei, I. Kamel, and T. Rabie, "Dynamic obstacle avoidance and path planning through reinforcement learning," *Applied Sciences*, vol. 13, no. 14, art. 8174, 2023, https://doi.org/10.3390/app13148174.

[2] K. Alnabulseih, A.-E.-R. Abd-El-Rehim, M. B. Badawi, and R. Elgohary, "Enhancing disaster response efforts with YOLOv8-based human detection in mobile robotics," *International Integrated Intelligent Systems*, vol. 2, no. 1, Jan. 2025, https://doi.org/10.21608/iiis.2025.292458.1036.

[3] G. Fragapane, H.-H. Hvolby, F. Sgarbossa, and J. O. Strandhagen, "Autonomous mobile robots in hospital logistics," in *Advances in Production Management Systems: The Path to Digital Transformation and Innovation of Production Management Systems*, B. Lalic, V. Majstorovic, U. Marjanovic, G. von Cieminski, and D. Romero, Eds. Cham: Springer International Publishing, 2020, pp. 672–679, https://doi.org/10.1007/978-3-030-57993-7_76.

[4] V. T. Ha and V. Q. Vinh, "Experimental research on avoidance obstacle control for mobile robots using Q-learning (QL) and deep Q-learning (DQL) algorithms in dynamic environments," *Actuators*, vol. 13, no. 1, art. 26, 2024, https://doi.org/10.3390/act13010026.

[5] C. Wang, X. Yang, and H. Li, "Improved Q-learning applied to dynamic obstacle avoidance and path planning," *IEEE Access*, vol. 10, pp. 92879–92888, 2022, https://doi.org/10.1109/ACCESS.2022.3203072.

[6] R. Raj and A. Kos, "A comprehensive study of mobile robot: History, developments, applications, and future research perspectives," *Appl. Sci.*, vol. 12, no. 14, p. 6951, Jul. 2022, https://doi.org/10.3390/app12146951.

[7] N. Demir, P. Demircioğlu, and İ. Böğrekci, "Advancing industry 4.0 with ROS: A case study on autonomous mobile robot technological advancements," *Int. J. 3D Print. Technol. Digit. Ind.*, vol. 8, no. 1, pp. 130–142, Apr. 2024, https://doi.org/10.46519/ij3dptdi.1366132.

[8] J. Zhong, M. Zhang, Z. Chen, and J. Wang, "Dynamic obstacle avoidance for mobile robots based on 2D differential Euclidean signed distance field maps in park environment," *World Electr. Veh. J.*, vol. 15, no. 7, art. 320, 2024, https://doi.org/10.3390/wevj15070320.

[9] Z. Zhang, Y. Xue, N. Figueroa, and K. Åkesson, "Gradient field-based dynamic window approach for collision avoidance in complex environments," *arXiv preprint* arXiv:2504.03260, 2025, https://arxiv.org/abs/2504.03260.

[10] H. Qin, S. Shao, T. Wang, Y. Li, N. Wang, and C. Yao, "An improved dynamic window approach for mobile robot dynamic path planning," in *Proc. 12th Int. Conf. CYBER Technol. Autom. Control Intell. Syst. (CYBER)*, Jul. 2022, pp. 348–353, https://doi.org/10.1109/CYBER55403.2022.9907305.

[11] J. Kong and J. Cheng, "Path planning of mobile robots based on the fusion of an improved A* algorithm and a dynamic window approach," in *Proc. 6th IEEE Inf. Technol. Netw. Electron. Autom. Control Conf. (ITNEC)*, Feb. 2023, pp. 968–973, https://doi.org/10.1109/ITNEC56291.2023.10082080.

[12] J. Guo, L. Liu, Q. Liu, and Y. Qu, "An improvement of D* algorithm for mobile robot path planning in partial unknown environment," in *Proc. 2nd Int. Conf. Intell. Comput. Technol. Autom.*, 2009, pp. 394–397, https://doi.org/10.1109/ICICTA.2009.561.

[13] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *Proceedings of the 2015 European Control Conference (ECC)*, 2015, pp. 3352–3357, https://doi.org/10.1109/ECC.2015.7331052.

[14] Z. Liu, Z. Chang, C. Qiao, H. Chen, and G. Zong, "Improved bidirectional A* and time-elastic band algorithm based mobile robot path planning," in *Proc. 37th Chin. Control Decis. Conf. (CCDC)*, May 2025, pp. 1–6, https://doi.org/10.1109/CCDC65474.2025.11090416.

[15] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, "The EBS-A* algorithm: An improved A* algorithm for path planning," *PLoS One*, vol. 17, no. 2, pp. 1–27, 2022, https://doi.org/10.1371/journal.pone.0263841.

[16] F. Wang, W. Sun, P. Yan, H. Wei, and H. Lu, "Research on path planning for robots with improved A* algorithm under bidirectional JPS strategy," *Appl. Sci.*, vol. 14, no. 13, art. 5622, 2024, https://doi.org/10.3390/app14135622.

[17] S. Alshammrei, S. Boubaker, and L. Kolsi, "Improved Dijkstra algorithm for mobile robot path planning and obstacle avoidance," *Comput. Mater. Contin.*, vol. 72, no. 3, pp. 5939–5954, 2022, https://doi.org/10.32604/cmc.2022.028165.

[18] L. Qiao, X. Luo, and Q. Luo, "An optimized probabilistic roadmap algorithm for path planning of mobile robots in complex environments with narrow channels," *Sensors*, vol. 22, no. 22, p. 8983, Nov. 2022, https://doi.org/10.3390/s22228983.

[19] C. Ren, F. Fu, C. Yin, Z. Yan, R. Zhang, and Z. Wang, "Improved artificial potential field method based on robot local path information," *Int. J. Adv. Robot. Syst.*, vol. 21, no. 5, Sep. 2024, https://doi.org/10.1177/17298806241278172.

[20] S. Lei, T. Li, X. Gao, P. Xue, and G. Song, "Research on improved RRT path planning algorithm based on multi-strategy fusion," *Sci. Rep.*, vol. 15, no. 1, pp. 1–19, 2025, https://doi.org/10.1038/s41598-025-92675-5.

[21] M. N. Ab Wahab, A. Nazir, A. Khalil, W. J. Ho, M. F. Akbar, M. H. M. Noor, and A. S. A. Mohamed, "Improved genetic algorithm for mobile robot path planning in static environments," *Expert Systems with Applications*, vol. 249, art. 123762, 2024, https://doi.org/10.1016/j.eswa.2024.123762.

[22] R. Rashid, N. Perumal, I. Elamvazuthi, M. K. Tageldeen, M. K. A. A. Khan, and S. Parasuraman, "Mobile robot path planning using Ant Colony Optimization," in *Proc. 2nd IEEE Int. Symp. Robot. Manuf. Autom. (ROMA)*, Sep. 2016, pp. 1–6, https://doi.org/10.1109/ROMA.2016.7847836.

[23] B. Tao and J.-H. Kim, "Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy," *J. King Saud Univ.–Comput. Inf. Sci.*, vol. 36, no. 2, p. 101974, Feb. 2024, https://doi.org/10.1016/j.jksuci.2024.101974.

[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997, https://doi.org/10.1109/100.580977.

[25] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Rob. Res.*, vol. 28, no. 8, pp. 933–945, Aug. 2009, https://doi.org/10.1177/0278364909340445.

[26] J. Choi, G. Lee, and C. Lee, "Reinforcement learning-based dynamic obstacle avoidance and integration of path planning," *Intell. Serv. Robot.*, vol. 14, no. 5, pp. 663–677, 2021, https://doi.org/10.1007/s11370-021-00387-2.

[27] Y. Xi, "Research on autonomous mobile robot navigation technology based on deep reinforcement learning," *Highlights Sci. Eng. Technol.*, vol. 114, pp. 108–113, Oct. 2024, https://doi.org/10.54097/krgznc69.

[28] M.-F. R. Lee and S. H. Yusuf, "Mobile robot navigation using deep reinforcement learning," *Processes*, vol. 10, no. 12, p. 2748, Dec. 2022, https://doi.org/10.3390/pr10122748.

[29] 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. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36, https://doi.org/10.1109/IROS.2017.8202134.

[30] T. Yu, "Progress in the application of deep reinforcement learning in path planning and control of mobile robots," *Sci. Technol. Eng. Chem. Environ. Prot.*, vol. 1, no. 2, Apr. 2025, https://doi.org/10.61173/3sjk6h37.

[31] 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, Oct. 2023, https://doi.org/10.3390/vehicles5040078.

[32] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dube, "Robot navigation in crowded environments using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5671–5677, https://doi.org/10.1109/IROS45743.2020.9341540.

[33] K. Li, J. Chen, D. Yu, T. Dajun, X. Qiu, J. Lian, R. Ji, S. Zhang, Z. Wan, B. Sun, F. Ni, and J. Han, "Deep Reinforcement Learning-based Obstacle Avoidance for Robot Movement in Warehouse Environments," in *Proc. 2024 IEEE 6th Int. Conf. Civil Aviation Safety and Information Technology (ICCASIT)*, 2024, pp. 342–348, https://doi.org/10.1109/ICCASIT62299.2024.10828100.

[34] Z. Zhou, J. Ren, Z. Zeng, J. Xiao, X. Zhang, X. Guo, Z. Zhou, and H. Lu, "A safe reinforcement learning approach for autonomous navigation of mobile robots in dynamic environments," *CAAI Transactions on Intelligence Technology*, pp. 1-16, 2024, https://doi.org/10.1049/cit2.12269.

[35] C. Moorthy, S. A, G. V, M. Manoj, V. Naveen, and P. H. Krishnan, "Comprehensive exploration of enhanced navigation efficiency via deep reinforcement learning techniques," in *Proc. 2nd Int. Conf. Intell. Cyber Phys. Syst. Internet Things (ICoICI)*, Aug. 2024, pp. 1087–1094, https://doi.org/10.1109/ICoICI62503.2024.10696134.

[36] Z. Zhang, "Path planning algorithms for mobile robots based on deep reinforcement learning," *Highlights Sci. Eng. Technol.*, vol. 114, pp. 49–55, Oct. 2024, https://doi.org/10.54097/fa2r8310.

[37] A. Rana and B. Kaveendran, "Deep Reinforcement Learning with PPO for Autonomous Mobile Robot Navigation Using ROS 2 Framework," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 13, no. 7, Jul. 2025, pp. 2119-2125, https://doi.org/10.22214/ijraset.2025.73330.

[38] S. M. Bankar, "Review on optimizing robotic navigation with deep reinforcement learning algorithms," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 23s, pp. 226–237, Mar. 2025, https://doi.org/10.52783/jisem.v10i23s.3698.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, https://doi.org/10.1038/nature14236.

[40] A. Gharbi, "A dynamic reward-enhanced Q-learning approach for efficient path planning and obstacle avoidance in mobile robotics," *Appl. Comput. Informatics*, 2024, https://doi.org/10.1108/ACI-10-2023-0089.

[41] R. Raj and A. Kos, "Dynamic obstacle avoidance technique for mobile robot navigation using deep reinforcement learning," *Int. J. Emerg. Trends Eng. Res.*, vol. 11, no. 9, pp. 307–314, Sep. 2023, https://doi.org/10.30534/ijeter/2023/031192023.

[42] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *Proc. 2021 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021, pp. 6057–6063, https://doi.org/10.1109/ICRA48506.2021.9561462.

[43] T. A. Fahmy, O. M. Shehata, and S. A. Maged, "Trajectory aware deep reinforcement learning navigation using multichannel cost maps," *Robotics*, vol. 13, no. 11, p. 166, Nov. 2024, https://doi.org/10.3390/robotics13110166.

[44] F. Hart, M. Waltz, and O. Okhrin, "Missing velocity in dynamic obstacle avoidance based on deep reinforcement learning," *arXiv preprint* arXiv:2112.12465, Dec. 2021, https://doi.org/10.48550/arXiv.2112.12465

[45] Z. Wang and M. Wang, "Mobile robot navigation in an unknown environment based on deep reinforcement learning," in *Proc. 5th Int. Conf. Intell. Comput. Human-Comput. Interact. (ICHCI)*, IEEE, Sep. 2024, pp. 630–634, https://doi.org/10.1109/ICHCI63580.2024.10808044.

[46] Z. Liu, "Research on robot path planning and obstacle avoidance algorithm in dynamic environment based on deep reinforcement learning," *Appl. Comput. Eng.*, vol. 103, no. 1, pp. 86–93, Nov. 2024, https://doi.org/10.54254/2755-2721/103/20241048.

[47] X. Gao, L. Yan, Z. Li, G. Wang, and I.-M. Chen, "Improved deep deterministic policy gradient for dynamic obstacle avoidance of mobile robot," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 53, no. 6, pp. 3675–3682, Jun. 2023, https://doi.org/10.1109/TSMC.2022.3230666.

[48] W. Zhu, X. Gao, H. Wu, J. Chen, X. Zhou, and Z. Zhou, "Design of multimodal obstacle avoidance algorithm based on deep reinforcement learning," *Electronics*, vol. 14, no. 1, p. 78, Dec. 2024, https://doi.org/10.3390/electronics14010078.

[49] P. Jiang, J. Ma, Z. Zhang, and J. Zhang, "Multi-sensor fusion framework for obstacle avoidance via deep reinforcement learning," in *Proc. 2nd Int. Conf. Electr. Eng. Control Sci. (IC2ECS)*, IEEE, Dec. 2022, pp. 153–156, https://doi.org/10.1109/IC2ECS57645.2022.10088073.

[50] Y. Hu, S. Wang, Y. Xie, Y. Wang, and T. Xiong, "Motion-prediction-based obstacle avoidance method for mobile robots via deep reinforcement learning," in *Proc. 48th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, IEEE, Oct. 2022, pp. 1–5, https://doi.org/10.1109/IECON49645.2022.9968446.

[51] W. Hu, Y. Zhou, and H. W. Ho, "Mobile robot navigation based on noisy N-step dueling double deep Q-network and prioritized experience replay," *Electronics*, vol. 13, no. 12, p. 2423, Jun. 2024, https://doi.org/10.3390/electronics13122423.

[52] P. Li, D. Chen, Y. Wang, L. Zhang, and S. Zhao, "Path planning of mobile robot based on improved TD3 algorithm in dynamic environment," *Heliyon*, vol. 10, no. 11, p. e32167, Jun. 2024, https://doi.org/10.1016/j.heliyon.2024.e32167.

[53] Y. Liu, C. Wang, C. Zhao, H. Wu, and Y. Wei, "A soft actor-critic deep reinforcement-learning-based robot navigation method using LiDAR," *Remote Sens.*, vol. 16, no. 12, p. 2072, Jun. 2024, https://doi.org/10.3390/rs16122072.

[54] S. Zhang, W. Tang, P. Li, and F. Zha, "Mapless path planning for mobile robot based on improved deep deterministic policy gradient algorithm," *Sensors*, vol. 24, no. 17, p. 5667, Aug. 2024, https://doi.org/10.3390/s24175667.

[55] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-Rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Prague, Czech Republic, 2021, pp. 6456–6463, https://doi.org/10.1109/IROS51168.2021.9636226.

[56] D. Martinez, L. Riazuelo, and L. Montano, "Deep reinforcement learning oriented for real world dynamic scenarios," *arXiv preprint* arXiv:2210.11392, 2022, https://doi.org/10.48550/arXiv.2210.11392.