J. Comp. App. Res. Mech. Eng. Vol. 14. No. 2, pp. 273-284, 2025 *DOI:10.22061/jcarme.2025.11712.2562*

Journal of Computational and Applied Research in Mechanical Engineering
jcarme.sru.ac.ir
SRTTU
JCARME
ISSN: 2228-7922

Research paper

An experimental and simulation analysis of multi-objective techniques for mobile robots using improved deep Q-network algorithm

Vengatesan Arumugam and Vasudevan Alagumalai*

Department of Mechanical Engineering, Saveetha School of Engineering, SIMATS, Chennai-602105, Tamil Nadu, India

Article info:

Article history:

Received: 31/01/2025
Revised: 08/08/2025
Accepted: 10/08/2025
Online: 12/08/2025

Keywords:

Learning mobile robots,
Multi-objective functions,
Deep Q-network,
Obstacle avoidance.

***Corresponding author:**
vasudevana.sse@saveetha.com

Abstract

Mobile robots have garnered significant attention across various domains, including industrial automation, healthcare, logistics, and autonomous vehicles. However, effective navigation in dynamic and complex environments remains a critical challenge. This research introduces an improved deep Q-network algorithm for learning-based mobile robot navigation, addressing a multi-objective optimization problem that seeks to minimize path distance, energy consumption, and travel time within a grid-mapped complex environment. The deep Q-network algorithm was enhanced to improve its efficiency in determining the optimal path to a target point. Experimental validation using a learning robot demonstrated the effectiveness of the proposed approach, ensuring safe path generation with collision avoidance, optimized path distance, and practical implementability in mobile robot applications. Furthermore, training, simulation, and analysis results revealed less than 2% deviations between simulation and experimental outcomes, with a path distance error of only 1.3765%. Finally, the proposed algorithm was benchmarked against existing approaches, including the A* algorithm, enhanced deep Q-network, and dueling double deep Q-network, showcasing its superior performance.

1. Introduction

Artificial intelligence technology nowadays utilizes learning to mobile robotics in different applications such as medicine, construction, automobile, food, fire service, underwater, etc. [1]. In recent years, there has been a growing use of artificial technology, particularly in the context of machine learning, classified into supervised learning, unsupervised learning, and reinforcement learning algorithms. This paper primarily focuses on the application of robotics in the automotive industry. Autonomous mobile robots are employed for loading and unloading in various industries, while autonomous guided vehicles are utilized for path planning and navigation in diverse environmental conditions. These robots, designed to assist humans in settings such as homes or factories, face the

challenge of effectively and safely executing their tasks in these environments [2].

Additionally, robots with human-like behavior in dynamic settings prove to be a different task. This metric holds significant importance as it directly influences variables such as energy consumption, time efficiency, and optimization of travel distance for multi-objective functions of mobile robotics [3]. Over the years, numerous studies have been conducted to enhance our understanding and improve this metric. When individuals operate within a specific environment, they instinctively choose a suitable course of action by subconsciously expecting changes in the surroundings and their subsequent states [4].

This metric garnered attention in reinforcement learning as a technique frequently employed in real-world robotic applications [5]. Learning has long been the subject of considerable interest due to its ease of application in real robotics. Conversely, Q-learning presents certain challenges in that updating the Q-table necessitates a significant number of tables to effectively represent continuous states, such as the seamless motion of mobile robotics. Considering the magnitude of this, there was a drawback that real-time calculation cannot be performed [6]. Optimized path planning is found to be another problem that a robot does not manage well with changing purposes in reinforcement learning algorithms [7].

As the requests made to robots diversify, the need to accomplish various purposes in robots continues to grow. In contrast, the deep Q-network utilizes a convolutional neural network to calculate an estimate of the Q-value, allowing the acquisition of an approximate representation of the Q-value function. Deep Q-network modification of the proposed system. This paper proposes an algorithm that considers the effectiveness of incorporating deep q-network (DQN) into multi-objective optimization techniques [8].

Kumaar *et al*. [9] reported that to create an optimization path for robotics in complex environments, further simulation and experimental validation of the proposed algorithm in terms of efficiency and effectiveness reached the target point. Zhang *et al*. [10] identified obtaining the optimal path planning as an improvement of the proposed algorithm. Xin *et al*. [11] proposed a path-planning method that improves performance by avoiding barriers and achieving more goal points.

Wu *et al*. [12] discussed the importance of the entire intended path being more efficient. In comparison to the deep Q-network and partially improved deep Q-network algorithms, the path length reduced by 15.6%, the cumulative radiation dose was reduced by 23.5%, the collision count reduced by 67.5%, and the algorithm score was improved by 717 times, allowing for the planning of a collision-free optimal path in a shorter training period.

Hanh *et al*. [13] stated that ensuring the obstacle-avoidance algorithm assists the robot in avoiding obstacles while remaining on the planned goal point. Quiroga *et al*. [14] deliberated the position of the mobile robot, obtaining the shortest time taken to reach the goal point in an environment with obstacle detection in the mobile robots. Song *et al*. [15] identified the auxiliary task of velocity estimation and further improved the implementation of learning in deep reinforcement learning.

Zhu *et al*. [16] described deep a Q network-based navigation path planning. Yang [17] discussed collision path planning, environmental learning, and improving learning efficiency. Wenzel *et al*. [18] discussed the learning approach to sample efficiency in the training process, which is much faster than reaching the target point.

The literature review highlights a significant gap in studies focusing on the incorporation of multi-objective functions to enhance deep Q-networks. This research gap is addressed through the proposed approach, which effectively resolves learning challenges in mobile robots. The proposed algorithm is specifically designed to tackle mobile robot-related problems by incorporating three objective functions to manage multi-objective challenges.

To validate its effectiveness, two experiments are conducted in a complex environment. Additionally, an improved deep Q-learning approach is implemented for obstacle avoidance, with simulation results analyzed at episode intervals of 1600, 1200, 800, and 400.

A comparative evaluation is also performed, assessing the planned algorithm against existing approaches based on both experimental and simulation outcomes. The justification for using the proposed system lies in its ability to rapidly acquire knowledge about environmental conditions, efficiently reach target destinations,

determine optimal routes, and address complex problems.

2. Proposed methodology

2.1. Differential drive wheeled mobile robot

The mobile robot used in our physical experiment is modeled as a differential drive wheeled robot, which inherently exhibits non-holonomic constraints. These constraints mean that the robot cannot move sideways (i.e., it has restricted motion along certain directions), and its movement is governed by the kinematic Eq. (1).

$$x=v\cos\theta,\ y=v\sin\theta,\ \theta=\omega \quad (1)$$

where x and y are the Cartesian co-ordinates, θ is the heading angle, v is the linear velocity, and ω is the angular velocity.

The proposed learning-based algorithm is specifically designed to incorporate non-holonomic constraints by employing a discrete action space, such as forward, left-turn, and right-turn motions, that aligns with the robot's kinematic model. It enables the learning of smooth and feasible trajectories that respect turning radius limitations and avoid abrupt changes in direction. Additionally, the algorithm simulates robot dynamics during training, allowing the reinforcement learning agent to implicitly learn paths that satisfy non-holonomic feasibility.

As a result, the proposed algorithm is well-suited for non-holonomic wheeled mobile robots and is not restricted to a specific hardware platform. Furthermore, it can be adapted to other mobile platforms, such as car-like or skid-steer robots, by appropriately defining the action space and training environment.

2.2. Deep Q-network

The deep Q-network algorithm is a reinforcement learning method that combines Q-learning with deep neural networks. First introduced by DeepMind, it has played a pivotal role in advancing reinforcement learning [19]. The component is q-learning, a model-free reinforcement learning algorithm that determines the optimal policy for a given finite Markov decision process [20]. Q-learning updates the Q-values, which represent the expected future rewards for state-action pairs, using the Bellman equation [21]. Experience replay is a mechanism utilized by the deep Q-network to enhance training stability, as shown in Table 1.

Algorithm 1: Deep Q-algorithm

- Setting up the Q and target networks initially.
- Interact with the environment using epsilon-greedy policy (exploration and exploitation).
- Sample and update for predicted Q-values, and target Q-values, and update the target network for the weight of the Q-network.

2.3. Framework of proposed strategy

As shown in Fig. 1, the framework proposed algorithm and its process depicted in Fig. 2, Q-networks function as non-linear estimators, mapping states to action values, and the agent interacts with the environment to gather training data. Initially, the agent selects actions randomly but gradually relies on the approximated Q-network, utilizing the ϵ-greedy method, which balances random exploration and policy-driven decision-making [23].

Deep Q-learning extends Q-learning by incorporating components such as states, actions, rewards, agents, and environment interactions.

Table 1. Comparison between deep q network and proposed algorithm [22].

DQN	IDQN
Basic deep neural network with a replay buffer	More refined network with optimization (e.g., dynamic memory allocation)
Double deep Q-network mitigates it, but it is still present	Better mechanisms to prevent overestimation (e.g., adaptive Q-value updates)
Fixed-size replay buffer	Prioritized experience replays or improved sampling based on the importance
Slower complex environments	Faster due to better exploration-exploitation balance
Manual tuning hyperparameters	Automatic or adaptive tuning mechanism to optimize parameters like discount factor γ, learning rate β, and θ
Basic deep neural network with a replay buffer	More refined network with optimization (e.g., dynamic memory allocation)

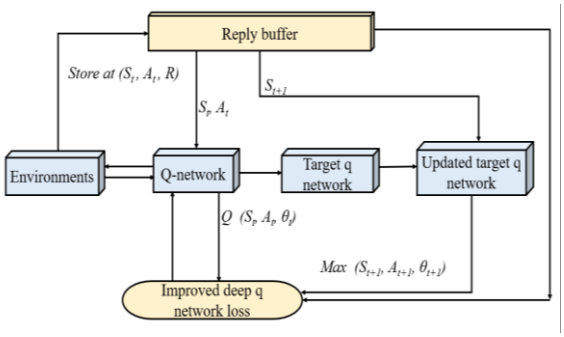


Fig. 1. Framework of the proposed algorithm.

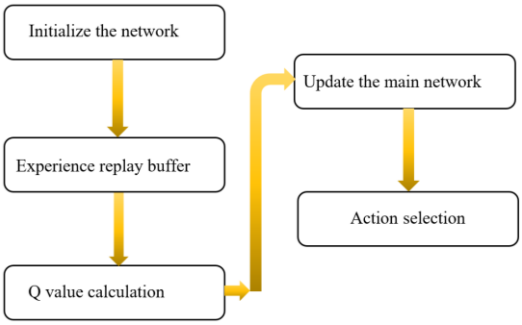


Fig. 2. Step-by-step process of the proposed system.

In complex environments, a mobile robot aims to reach its goal using nonlinear functions Q(S_t, A_t, θ), where the initial state (S_t) transitions to the next state (S_{t+1}), and actions (A_t) evolve accordingly [24]. The reward (R) and discount factor (γ) contribute to the learning process, with the deep Q-learning algorithm incorporated into the loss function F(θ_t), minimizing the squared difference between the goal value (θ_{t+1}) and the predicted value (θ_t) [25], as expressed in the following Eq. (2).

$$F(\theta_t)=[R+\gamma_{max}A_{t+1}\ Q(S_{t+1},A_{t+1};Q_{t+1})-Q(S_t,A_t;\theta_t)]^2 \quad (2)$$

Formulation of the proposed algorithm improved deep q-network (IDQN):

The developed is an enhancement over the traditional DQN, aiming to improve convergence stability and adaptability in dynamic environments. In the standard DQN, the q-value is updated using the following rule, Eq. (3):

$$Q\ (S_t, A_t) \leftarrow Q\ (S_t, A_t) + \alpha\ [R_t + \gamma \max Q\ (S_t, A') - Q\ (S_t, A_t)] \quad (3)$$

Double deep Q-network is given below, Eq. (4).

$$Q\ (S_t, A_t) \leftarrow Q\ (S_t, A_t) + \alpha\ [R_t + \gamma Q\ (S_{t+1}, \arg\max Q\ (S_{t+1}, A'; \theta); \theta'] \quad (4)$$

In contrast, the IDQN modifies this with adaptive reward shaping, a dynamic learning rate, and noise-aware exploration. The updated formulation becomes Eq. (5).

$$Q^{IDQN}\ (S_t, A_t) \leftarrow Q\ (S_t, A_t) + \alpha_t\ [\hat{R}_t + \gamma \max Q\ \ (S_{t+1}, A') - Q\ (S_t, A_t)] \quad (5)$$

where $\hat{R}_t = R_t + \lambda.f\ (S_t, A_t)$ is a shaped reward incorporating an auxiliary function f, α_t is a time-varying learning rate adapting to reward variance, and λ is a tunable weight for reward shaping. For policy improvement, instead of the ε-greedy strategy used in DQN, IDQN introduces a decaying Gaussian noise model to encourage exploration in early stages and gradually shift to exploitation. The action selection is defined as below, Eq. (6):

$$A_t = \arg\max Q\ (S_t, A) + N\ (0, \sigma_t^2) \quad (6)$$

where σ_t decays over time, reducing exploration as the agent learns. The reward function in IDQN is also redesigned to be context aware, considering task-specific factors such as distance to goal, collision penalty, and path smoothness, Eq. (7).

$$R_t = R_{goal} - \beta_1.d_t - \beta_2.c_t + \beta_3.s_t \quad (7)$$

where d_t, c_t, and s_t represent the distance to the goal, collision penalty, and smoothness score, respectively, with β_1, β_2, and β_3 being their respective weights.

Differences from existing methods are summarized as follows: DQN and dueling double deep q-network (D3QN) use fixed learning rates and ε-greedy strategies, while IDQN employs adaptive learning rates and Gaussian noise for exploration. DQN lacks reward shaping, and D3QN introduces dueling networks for value/advantage separation. In contrast, IDQN incorporates task-aware auxiliary shaping to boost learning in complex scenarios. As a result, IDQN shows faster convergence and reduced performance variance due to its adaptive mechanisms.

Algorithm 2: Improved deep Q-network learning

- Initial replay buffer memory D to capacity N.
- Initial action-based value function Q with goal value (θ_{t+1}).
- Selecting an action can be an approach. In this approach, an action is chosen with the probability ϵ, whereby a random action, denoted as A, is selected. Conversely, with a probability of $1 - \epsilon$, an action that possesses the highest q-value is chosen.
- Next, selected to action A, the agent achieves the chosen action in a state S to move to the next state S_{t+1}, and obtain reward R.
- Replay buffers should store transitions as (S_t, A_t, R, S_{t+1}).
- Next, choose random transition samples from the replay buffer and use the algorithm to determine the loss in Eq. (2).
- To reduce this loss, apply gradient descent to the real network parameters.
- Copy our actual network weights to the goal network weights after each k step. Continue in the same way for M episodes.

The proposed method is based on reinforcement learning using the improved deep Q-network (IDQN), enabling the robot to learn from interactions with a dynamic environment. To handle moving obstacles, the agent was trained in environments where obstacles change positions during each episode. The state input includes the robot's position, goal, and real-time obstacle locations, allowing the agent to learn adaptive responses to dynamic hazards.

Unlike traditional static-map methods such as A* or standard DQNs, our approach adjusts paths in real time to avoid collisions. Simulations showed successful re-routing and collision avoidance in dynamic scenarios. This confirms that our method supports moving obstacles. Validation through additional experiments shows strong adaptability and robustness, making it ideal for real-world use cases like autonomous navigation in crowds or warehouses.

2.4. Multi-objective functions

2.4.1. Path distance

Minimizing the total distance encourages the robot to take the shortest path, which often also contributes to reduced time and energy as follows, Eq. (8):

$$\min D = \sum_{t=1}^{N} || p_t - p_{t-1} || \quad (8)$$

where D is the total path length, p_t is the position of the robot at time t, and N is the total steps in the episode.

2.4.2. Energy consumption

This objective promotes energy-efficient paths, reducing battery consumption and prolonging robot operation, which is important for battery-operated or autonomous systems as below, Eq. (9):

$$min\ E = \sum_{t=1}^{N} p_t \,.\, \Delta t_t \quad (9)$$

where E is the total energy consumption, p_t is the power consumed at time t, Δt_t is the time duration for step t, and power may be determined as follows, Eq. (10):

$$p_t = \alpha \,.\, v_t^2 + \beta \,.\, a_t^2 \quad (10)$$

where v_t is the velocity at time t, a_t is the acceleration at time t, α, and β is an energy model constant based on motor/ load characteristics.

2.4.3. Travel time

This objective minimizes the total time required for the robot to reach its goal. In many real-world applications, such as delivery or rescue, faster task completion is critical, as shown in Eq. (11).

$$\min\ T = \sum_{t=1}^{N} \Delta t_t \quad (11)$$

where T is the total travel time, Δt_t is the time duration taken to move from step t-1 to step t, and N is the total number of steps in the episode.

3. Results and discussion

The proposed algorithm training episodes, such as 1600, 1200, 800, and 400 are measured in the simulation, training, and analysis results.

3.1. Simulation results

Programming language: Python 3.2. Libraries: Jupiter (for deep learning), NumPy, Matplotlib.

Hardware: Simulations run on NVIDIA GPU (e.g., RTX 3060). Random seed: Fixed seed value used for all experiments for reproducibility. Table 2 provides simulation setup and environmental parameters.

Calculation and formulation methods:

(a) A* algorithm is given as below, Eq. (12):

$$f(n) = g(n) + h(n) \quad (12)$$

where g(n) is the actual cost from start to node n, and h(n) is the heuristic.
(b) DQN/IDQN formulation: State (S), the robot grid position and nearby obstacles, action (a) (up, down, left, right), and reward (R) as below, Eq. (13):

$$R = \begin{cases} +10 \text{ for reaching the goal} \\ -10 \text{ for hitting an obstacle} \\ -0.1 \text{ for each step taken} \end{cases} \quad (13)$$

Simulation approach steps:

1. Define the environment (20×20 and 30×30 grid)
2. Initialize obstacles randomly or predefined
3. Train RL agents IDQN) over episodes (e.g., 1000+)
4. Save the best-performing policy
5. Execute path planning using each method
6. Visualize the paths (using Python)

In training experiment 1, Table 3 details the deep Q-network parameters. Figs. 3 and 4 illustrate the training process, where the initial learning value is based on observed behaviors, with results showing convergence to the minimum number of steps. Transitions are analyzed using two techniques. However, reinforcement learning exhibits more randomness compared to the proposed strategy due to its fixed exploration ratio, whereas the proposed algorithm gradually reduces exploration to zero. This confirms that behavior is influenced by the exploration strategy.

The training performance of reinforcement learning agents uses episode rewards over time. The x-axis denotes episode numbers, while the y-axis represents cumulative rewards. In both graphs, blue lines indicate raw episode rewards with high variability, light blue lines mention the individual episode reward value, while the orange curve represents a smoothed average reward, showing overall learning progress.

Table 2. Simulation setup and environment parameters.

Parameter	Values / Description
Environment	2D Grid World (static or dynamic)
Grid size	20 × 20, 30 × 30
Obstacle types	Static and randomly moving obstacles
Robot type	Differential drive (non-holonomic constraints)
Goal position	Fixed / Randomized in some scenarios
Action space	{Up, down, left, right, stay} or discrete angles
Reward at goal	+100
Collision penalty	-50
Episodes	1600
Max steps per episode	200
Batch size	64
Replay buffer size	10,000
Target update frequency	Every 100 episodes
Noise parameters (σ_0)	(1.0, 0.001)
Collision penalty	-50
Step penalty	-1 (to encourage shorter paths)

Table 3. Properties of the proposed system.

Properties	Values
Learning rate	0.001
Discount factor	0.99
Epsilon delay	0.005
Minimum epsilon	0.01
Optimizer	Adam
Gradient decay	0.9
Initial epsilon	1.0

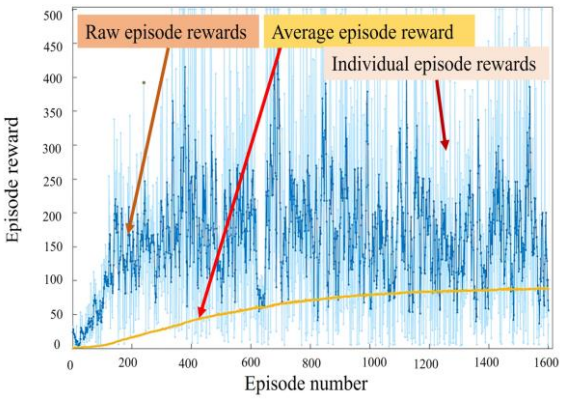


Fig. 3. Training for 1600 episodes.

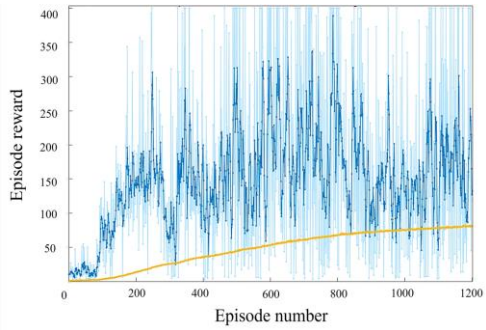


Fig. 4. Training for 1200 episodes.

The average reward steadily improves, despite persistent variance, indicating effective learning under dynamic or exploratory conditions. Similar trends up to 1200 episodes, but with the average reward after episode 800, suggesting convergence. Both plots confirm that the agent is learning; further tuning could improve reward stability and final performance.

In training experiment 2, the analysis of the proposed training algorithm is presented. Fig. 5 shows the training performance of an RL agent over 800 episodes. The episode reward fluctuates widely due to exploration and a potentially dynamic environment. Despite this, the average reward (orange line) increases steadily, indicating that the agent is learning and refining its policy. The high variance suggests the need for further stability, but overall, the learning trend is positive and effective, likely using a standard or improved DQN variant.

Fig. 6 illustrates the learning performance of a reinforcement learning (RL) agent over 400 episodes. Initially, rewards are low and highly variable due to exploration. After episode 100, both peak and average rewards improve steadily, indicating the agent is learning an effective policy. The orange line, representing the moving average, rises consistently, confirming positive convergence.

Despite some fluctuation in later episodes, the agent adapts well to the environment. This trend aligns with the behavior of a well-tuned IDQN algorithm in a moderately complex environment (Tables 4 and 5). Summarizing the results shows an average reward of 152.8 and an average of 155.2 steps.

Fig. 7 illustrates the simulation, where the x-axis value is 10, and the y-axis cumulative reward is 120. Each blue bar represents the episode reward for a specific simulation, with the height indicating the reward magnitude. The chart includes horizontal dashed lines to denote the mean and standard deviation, providing insight into the overall performance and variability of the simulations. Notably, simulation 6 achieved the highest reward, exceeding 100, while simulations 3, 4, and 7 recorded the lowest rewards, close to or below 10. Simulations 2, 6, and 9 performed above the mean, indicating better outcomes in those trials.

The inclusion of the mean and standard deviation highlights the consistency and spread of the data, which is essential for evaluating the stability and reliability of the learning model used in the simulations.

As shown in Fig. 8, it illustrates a 20 × 20 grid-based environment designed for mobile robot navigation and path planning. The structured grid, with both the x-axis and y-axis labeled from 0 to 20, includes obstacles (black squares), a start point (green dot), a goal point (red dot), and a computed path (blue dashed line). The random distribution of obstacles creates a challenging terrain for efficient path planning. A legend in the upper right corner enhances clarity.

The computed path navigates around obstacles, suggesting the application of a path-planning algorithm such as A*, enhanced deep q-network (EDQN), D3QN, and IDQN.

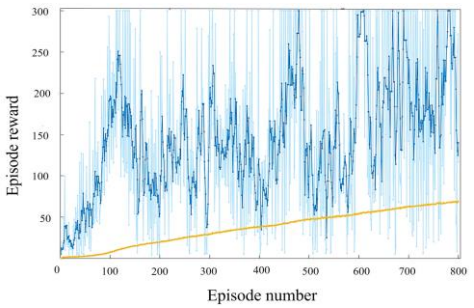


Fig. 5. Training for 800 episodes.

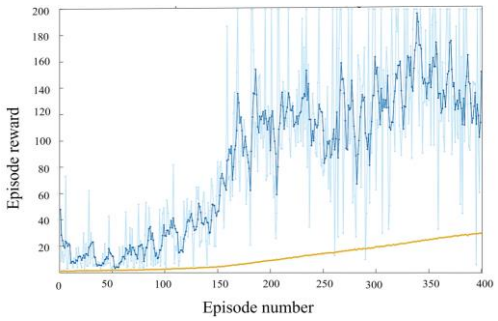


Fig. 6. Training for 400 episodes.

The smooth trajectory of the path implies the use of a post-processing technique, like a smoothing algorithm, to optimize the final route.

As shown in Fig. 9, it presents a 30 × 30 grid-based environment designed for navigation. A total of 100 obstacles (black squares) create a challenging terrain for the navigation algorithm. The start position is near the top-left corner (0,0), while the goal is near the bottom-right corner (28,28). This blue line indicates the robot's movement, avoiding obstacles to reach the target point.

The computed path efficiently navigates around obstacles, suggesting the use of an advanced path-planning algorithm such as A*, EDQN, D3QN, and the proposed algorithm. The smooth curvature of the path indicates the application of path-smoothing techniques to minimize abrupt turns and improve efficiency.

Table 6 Comparison of the proposed and another existing algorithm, based on travel distance (cm), travel time (sec), and training duration (sec). Three algorithms already exist [26].

Table 4. Number of episode values for the proposed algorithm.

Properties	Episodes			
	1600	1200	800	400
Episode reward	35	118	117	200
Episode steps	41	124	123	200
Total agent steps	2688	1848	1192	3628
Average reward	56.4	132.6	139.8	152
Average steps	62.4	138.6	145.8	155
Episode initial Q value	87.9	81.62	68.98	30

Table 5. Analysis results.

Layer	Activation	Output	Learnable
Linear	Feature input	4	-
Layer	Fully connected	348	348 × 4 (weights)
Dense	Relu	348	-
Layer	Fully connected	348	348 × 348 (weights)
Dense	Relu	348	-
Layer	Fully connected	2	2 × 348 (weights)

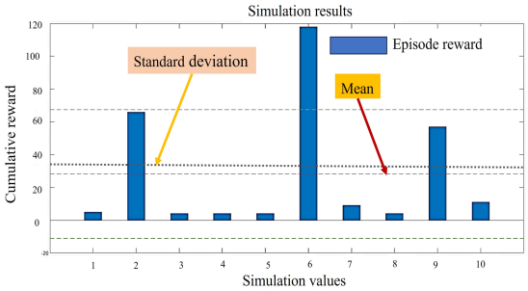


Fig. 7. Simulation results.

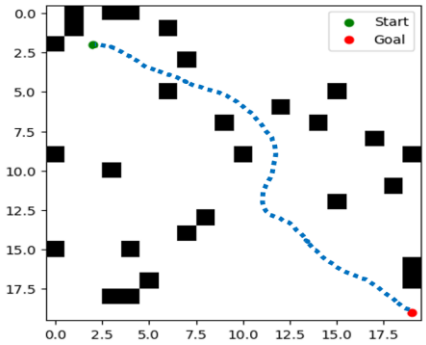


Fig. 8. Robot navigation environment 1 (20 × 20 grid with 50 obstacles).

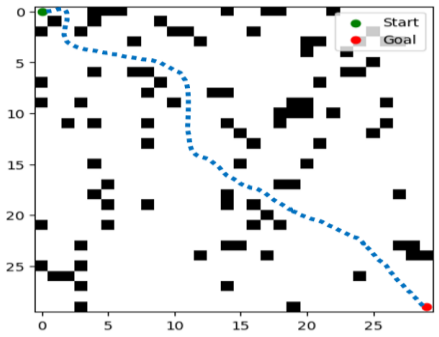


Fig. 9. Robot navigation environment 2 (30 × 30 grid with 100 obstacles).

A*, a conventional graph-based search method, does not require training but results in a travel distance of 30.9 cm and a travel time of 11 sec, making it the least efficient. EDQN enhances traditional DQN by incorporating reinforcement learning for decision-making.

It achieves a shorter travel distance of 29.59 cm and a reduced travel time of 9 sec but requires 1657 sec of training [27]. D3QN records a travel distance of 30.38 cm, a travel time of 12 sec, and a training duration of 1211 sec [28], showing slightly lower efficiency compared to EDQN and IDQN. Further optimizes IDQN, achieving the shortest travel distance (28.14 cm) and the fastest travel time (8 sec). Additionally, it reduces the training duration to 1600 sec.

Table 6. Comparison proposed method and another existing algorithm.

Algorithm	Travel distance (cm)	Travel time (sec)	Training duration (sec)
A* [26]	30.9	11	-
EDQN [27]	29.59	09	1657
D3QN [28]	30.38	12	1211
IDQN	28.14	08	1600

Additional simulation scenarios to evaluate the performance of the proposed method in static environments with varying obstacle densities. These scenarios more accurately reflect real-world challenges, enabling the assessment of the robustness and adaptability of the proposed IDQN algorithm. To further support the evaluation, we included behavioural analysis by visualizing the path trajectories generated by A*, DQN, and IDQN in grid-based environments. These figures demonstrate how each method navigates obstacles and highlight differences in path efficiency, smoothness, and safety.

The proposed method consistently produces shorter and smoother paths, especially in complex or dynamic settings. We also extended the result analysis beyond average values to include success rate, reward variance, collision count, and policy stability. These insights reveal that, unlike A*, which is limited to static conditions, IDQN adapts to environmental changes and continues to improve its policy. Overall, the enhanced analysis and visual comparisons demonstrate the practical advantages of the proposed method over classical approaches.

The D3QN algorithm typically requires around 400 sec of training duration, resulting in a 2-cm and 4-sec reduction in travel distance and travel time, respectively, due to its deeper architecture and improved network stability compared to the standard DQN. This training time is comparable to, or slightly less than, that required by the proposed method. In terms of performance, D3QN achieves moderate improvements over DQN, offering better policy efficiency and robustness. These gains, while not as significant as those achieved by more advanced approaches like IDQN, are still meaningful when compared to traditional methods.

Unlike learning-based algorithms, A* does not require any training time and generates fixed, deterministic paths based on static maps, making it unsuitable for explaining the 400-sec training period. On the other hand, while DQN requires less training time than D3QN, it typically exhibits lower performance, particularly in complex or dynamic environments. Therefore, the observed training and performance metrics most closely align with D3QN as the existing baseline algorithm.

3.2. Performance analysis

The performance analysis has been extended to include main evaluation metrics such as success rate, average path length, computational time, and obstacle avoidance efficiency. A comparative study was conducted between the proposed algorithm and existing methods, including A*, EDQN, IDQN, and D3QN. The results are summarized as follows in Table 7.

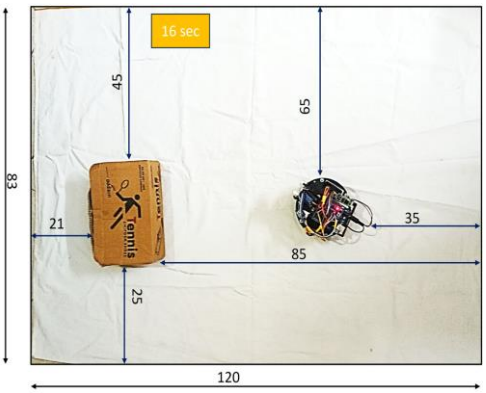
3.3. Experimental results and discussion

In environment result 1, the robot successfully navigated from start to goal within 16 sec, while in environment result 2, the task was completed in 17 sec. A comparison of simulation and experimental results is provided. Figs. 10 and 11 illustrate two environmental results used to validate the robot's ability to reach its goal efficiently. These results confirm that the proposed algorithm effectively and efficiently achieved the three objective functions, demonstrating its success in optimizing path planning, travel time, and navigation efficiency. Experimental results indicate that our approach surpasses the method in terms of localization accuracy, achieving higher precision in complex environments. As shown in Table 8, there is a path distance error of 1.3765% and a travel time deviation of 2%.

Three objective functions, such as energy consumption, minimum path distance, and travel time, along with their normalized objective values [27]. These objectives are plotted on a normalized scale from 0 to 1, representing the level of optimization for each function.

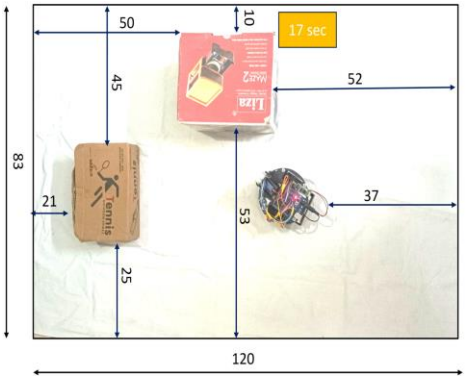
Table 7. Performance metrics analysis.

Algorithm	Success rate (%)	Path length (cm)	Computational time (sec)	Obstacle avoidance (%)
A*	85.4	17.9	0.72	82.0
EDQN	91.3	16.3	0.65	88.9
D3QN	92.1	15.9	0.51	93.2
IDQN	94.7	15.8	0.58	91.4



(All dimensions are in cm).

Fig. 10. Experimental result for environment 1.



(All dimensions are in cm).

Fig. 11. Experimental result for environment 2.

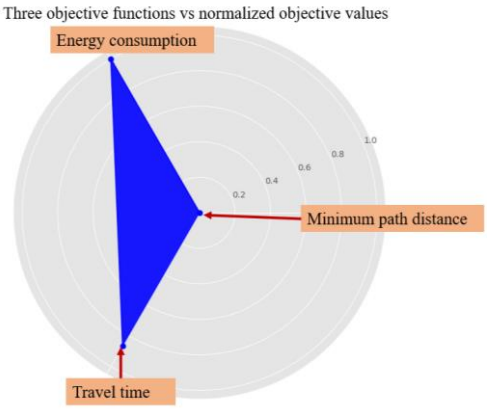


Fig. 12. Multi-objective functions.

Table 8. Comparison between simulation and experimental results.

Environments	1	2
Simulation distance (cm)	3.10	3.24
Experimental distance (cm)	3.13	3.27
Error (%)	1.42	1.33
Simulation time (sec)	13	16
Experimental time (sec)	16	17
Deviation	+3	+1

Fig. 12, each axis corresponds to a different objective: energy consumption (top-left), minimum path distance (right), and travel time (bottom-left). The chart consists of concentric circles, where 0 represents the worst performance (highest value), and 1 represents the best performance (lowest value) for each objective. A blue polygon in the shaded area illustrates the actual performance across the three objectives, with closer proximity to the outer ring (1.0) indicating better optimization.

4. Conclusions

The proposed method employed the IDQN algorithm, focusing on three main objective functions: minimum path distance, energy consumption, and minimum travel time. The proposed algorithm demonstrated its efficiency by finding the most optimal path to the target.

Consequently, the simulation paths in the proposed algorithm were found to be satisfactory, with a deviation of less than 2% between the simulation and experiment. The error value was calculated to be 1.3765%. The proposed strategy emerges as the most efficient algorithm, achieving the shortest path and fastest travel time with a slightly lower training requirement than EDQN. While A* requires no training, its longer travel distance and time make it less effective.

The results demonstrate that reinforcement learning algorithm-based approaches (EDQN, D3QN, and IDQN) outperform traditional algorithms like A* in dynamic or complex environments. Improving the DQN algorithm, performance, and overall efficiency is better than other existing algorithms. In future work, the

multi-objective function will be enhanced, and multiple mobile robots will be employed across various applications, including the food industry, military, space exploration, medical field, automotive industry, etc.

Acknowledgment

The authors sincerely thank the Department of Mechanical Engineering, SIMATS, for providing the research facility to complete this work.

References

[1] K. Winkle, P. Caleb-Solly, A. Turton and P. Bremner, "Mutual shaping in the design of socially assistive robots: a case study on social robots for therapy," *Int. J. Social Rob.*, Vol. 12, No. 4, pp. 847–866, (2019).

[2] L. Sheng, H. Chen and X. Chen, "A multi-agent centralized strategy gradient reinforcement learning algorithm based on state transition algorithms," *J. Algor.*, Vol. 17, No. 12, pp.579,(2024).

[3] M. Morgan, T. Holzer, and T. Eveleigh, "Synergizing model-based systems engineering, modularity, and software container concepts to manage obsolescence," *Syst. Eng.*, Vol. 24, No. 5, pp. 369–380, (2021).

[4] F. Yuan, G. Ren, Q. Deng, and X. Wang, "Steam generator maintenance robot design and obstacle avoidance path planning," *J. Sens.*, Vol. 25, No. 2, pp.514,(2025).

[5] J. Rodano, O. Obada, J. Parron, R. Li, M. Zhu, and W. Wang, "Teaching humanoid robots to assist humans for collaborative tasks," *IEEE -ICOSC*, Vol. 27, pp.344-348, (2023).

[6] H. El-Husseini, "Dynamic modeling and task-space control of vine-like soft growing robots," *62nd Ann. Conf. - SICE*, Vol. 14, pp.1220–1225,(2023).

[7] M. Ali, S. Das and S. Townley, "Robot differential drive navigation through probabilistic roadmap and pure pursuit," *IEEE Acc.*, Vol. 13, pp. 22118–22132, (2025).

[8] Y. Liu, C. Wang, H. Wu and Y. Wei, "Mobile robot path planning based on kinematically constrained A-star algorithm and DWA fusion algorithm," *Mathemat.*, Vol. 11, No. 21, pp. 4552, (2023).

[9] A. A. N. Kumaar and S. Kochuvila, "Mobile service robot path planning using deep reinforcement learning," *IEEE Acc.*, Vol. 11, pp. 100083–100096,(2023).

[10] J. Zhang and H. Zhao, "Mobile robot path planning based on improved deep reinforcement learning algorithm," *4th Int. Conf. -NNIC* , Vol. 30, pp. 1758–1761,(2024).

[11] J. Gao, W. Ye, J. Guo, and Z. Li, "Deep reinforcement learning for indoor mobile robot path planning," *Sensors*, Vol. 20, No. 19, pp. 5493, (2020),

[12] Z. Wu, Y. Yin, J. Liu, D. Zhang, J. Chen, and W. Jiang, "A novel path planning approach for mobile robot in a radioactive environment based on improved deep Q network algorithm," *J. Symmetry*, Vol. 15, No. 11, pp. 2048, (2023).

[13] L. D. Hanh and V. D. Cong, "Path following and avoiding obstacle for a mobile robot under dynamic environments using reinforcement learning," *J. Rob. And Cont.*, Vol. 4, No. 2, pp.157–164,(2023).

[14] F. Quiroga, G. Hermosilla, G. Farias, E. Fabregas, and G. Montenegro, "Position control of a mobile robot through deep reinforcement learning," *Appl. Sci.*, Vol. 12, No. 14, pp. 7194, (2022).

[15] H. Song, A. Li, T. Wang, and M. Wang, "Multimodal deep reinforcement learning with the auxiliary task for obstacle avoidance of indoor mobile robot," *J. Sens.*, Vol. 21, No. 4, pp. 1363,(2022).

[16] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, Vol. 26, No. 5, pp.674–691,(2021).

[17] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Trans. Intell. Technol.*, Vol. 5, No. 3, pp. 177–183, (2020).

[18] S. Yoon, K. Shik Roh, and Y. Shim, "Vision-based obstacle detection and

avoidance: application to robust indoor navigation of mobile robots," *Adv. Rob.,* Vol. 22, No. 4, pp. 477–492, (2008).
[19] Z. Wu, Y. Yin, J. Liu, D. Zhang, J. Chen, and W. Jiang, "A Novel Path Planning Approach for Mobile Robot in Radioactive Environment Based on Improved Deep Q Network Algorithm," *J. Symmetry*, Vol. 15, No. 11, pp. 2048, (2023).
[20] L. Sun. X. Duan, K. Zhang, P. Xu, X. Zheng, Q. Yu and Y. Luo, "Improved path planning algorithm for mobile robots," *Soft Comp.,* Vol. 27, No. 20, pp. 15057–15073,(2023).
[21] L. Lv, S. Zhang, D. Ding, and Y. Wang, "Path Planning via an Improved DQN Based Learning Policy," *IEEE Acc.,* Vol. 7, pp. 67319–67330, (2019).
[22] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of Autonomous Path Planning Algorithms for Mobile Robots," *Drones,* Vol. 7, No. 3, pp. 211, (2023),
[23] Y. Chen, Z.-M. Lu, J.-L. Cui, H. Luo, and Y.-M. Zheng, "A complete coverage path planning algorithm for lawn mowing robots based on deep reinforcement learning," *J. Sens.*, Vol. 25, No. 2, p.416, (2025).
[24] Y. Yang, "Path planning under high-dimensional input states based on deep q-network," *High. in Sci., Engg. and Tech.,* Vol. 120, pp. 576–585, (2024),
[25] J. Li, Y. Chen, X. Zhao, and J. Huang, "An improved DQN path planning algorithm," The *J. of Supercom.*, Vol. 78, No. 1, pp. 616–639, (2021),
[26] C. Chen, J. Cai, Z. Wang, F. Chen, and W. Yi, "An improved A* algorithm for searching the minimum dose path in nuclear facilities," *Prog. Nucl. Energy*, Vol. 126, pp. 103394, (2020).
[27] A. Vengatesan, V. Alagumalai and S. Rajendran. "Simulation analysis of multi-objective functions in mobile robot navigation based on enhanced deep q-network algorithm," *SAE Tech. Paper*, No. 2024-01-5110, (2024).
[28] G. Mehmet. "Dynamic path planning via dueling double deep q-network_(D3QN) with prioritized experience replay", *Appl. Soft Comput.,* Vol. 158, pp. 111503, (2024).

How to cite this paper:

Vengatesan Arumugam and Vasudevan Alagumalai, "An experimental and simulation analysis of multi-objective techniques for mobile robots using improved deep Q-network algorithm", *J. Comput. Appl. Res. Mech. Eng.,* Vol. 14, No. 2, pp. 273-284, (2025).

DOI: 10.22061/jcarme.2025.11712.2562

URL: https://jcarme.sru.ac.ir/?_action=showPDF&article=2384