

Article

Assembly Measurement Path Planning for Mobile Robots Using an Improved Deep Reinforcement Learning

Gang Yuan ^{1,*} , Bo Zhu ², Yi Hu ², Guangdong Tian ³  and Zhiwu Li ⁴¹ School of Mechanical Engineering, Southeast University, Nanjing 211189, China² Aerosun Corporation, Nanjing 211100, China³ School of the Mechanical-Electrical and Vehicle Engineering, Beijing University of Civil Engineering and Architecture, Beijing 100044, China⁴ School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China

* Correspondence: yuangang1224@163.com

Abstract

In addressing the challenges associated with mobile robot path planning during complex product assembly measurements, this study introduces the N-step Priority Double DQN (NDDQN) algorithm, which integrates double Q-learning and an N-step priority strategy to accelerate convergence. This approach aims to improve the obstacle avoidance capabilities of mobile robots while accelerating their learning efficiency. We conducted three grid-based obstacle avoidance simulation experiments of varying scales to compare and analyze the path planning performance of both the proximal policy optimization algorithm and the Deep Q Network algorithm. To accurately simulate real-world robotic measurement scenarios, two Gazebo environments were utilized to validate the effectiveness of our proposed algorithm. Through a comprehensive analysis of simulation results from all three algorithms, we demonstrate that the NDDQN algorithm exhibits significant effectiveness and stability in path planning. Notably, it substantially reduces iteration counts and enhances convergence speeds. This research provides a theoretical foundation for adaptive path planning in mobile robots engaged in complex product assembly measurements.

Keywords: mobile robot; reinforcement learning; assembly measurement; path planning

Received: 26 October 2025

Revised: 18 November 2025

Accepted: 21 November 2025

Published: 22 November 2025

Citation: Yuan, G.; Zhu, B.; Hu, Y.; Tian, G.; Li, Z. Assembly Measurement Path Planning for Mobile Robots Using an Improved Deep Reinforcement Learning. *Appl. Sci.* **2025**, *15*, 12406. <https://doi.org/10.3390/app152312406>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid advancement of Industry 4.0 technologies in the realm of intelligent manufacturing, assembly measurement path planning (AMPP) for complex products has increasingly evolved into a process characterized by digitalization, intelligence, and integrated smart control [1]. The accuracy requirements for AMPP associated with complex products are exceptionally rigorous, presenting a broad global measurement range that complicates precise positioning. Traditionally, the AMPP has predominantly relied on manual processes [2]. However, this conventional approach is characterized by relatively low efficiency. During the AMPP process, numerous measurements, including point application, calibration, and station transfer, must be executed manually [3]. This reliance on manual procedures ultimately results in extended assembly cycles and challenges in ensuring timely product delivery [4,5].

The AMPP is evolving towards a comprehensive measurement approach that integrates multiple technologies and couples various sensors. Mobile robots offer significant

advantages, including substantial payload capacity and high precision in positioning accuracy [6,7]. When combined with the exceptional adaptability and flexibility of human operators, this synergy empowers robots to execute automatic assembly and measurement tasks effectively. This advancement enables flexible, high-precision assembly and measurement processes for complex products. Furthermore, the design of high-precision measurement trajectories is a crucial element in automating the assembly operations for large-scale, complex product sites, as it significantly contributes to ensuring overall assembly accuracy.

Path planning for mobile robots is typically categorized into global path planning and local path planning [8]. The methods employed in global path planning primarily encompass graph search-based [9], random sampling-based [10], heuristic algorithm [11], and artificial intelligence [12]. Notably, the success of global planning methods heavily relies on the availability of prior information regarding the environment [13]. Due to the inherent uncertainty that robots encounter in complex environments, an increasing number of researchers are focusing on local path planning methods. Local path planning involves the exploration of mobile robot pathways by gathering environmental information through sensors, assessing rewards and penalties, and subsequently determining the next action [14]. This process leads to the formulation of a relatively optimal collision-free path. Importantly, local path planning operates independently of prior knowledge about the environment and demonstrates excellent adaptability as well as real-time performance in response to dynamic conditions [15].

Deep reinforcement learning (DRL), recognized as a pivotal technology in intelligent manufacturing, has garnered increasing attention from researchers regarding its application to the challenges of robot path planning [16,17]. Deep learning enables agents to perceive their environment, while reinforcement learning equips them with the capability to devise optimal strategies for addressing complex challenges. Consequently, deep reinforcement learning facilitates dynamic planning in uncertain environments by identifying pathways that optimize reward values to effectively reach target objectives.

However, DRL algorithms still harbor certain shortcomings. The intricacies of reward mechanism design are significantly reliant on expert prior knowledge and necessitate a delicate balance among various objective criteria, including path length, safety considerations, and energy efficiency [18]. Furthermore, the opaque nature of the DRL algorithm engenders a lack of interpretability in its decision-making processes, rendering it challenging to furnish formal guarantees regarding security [19]. Throughout the training process, the convergence performance remains unstable, and the agent is prone to collisions with obstacles. When the next action taken by the agent relies solely on observed environmental data, the DRL path planning algorithm demonstrates insufficient generalization capabilities.

While deep reinforcement learning shows promise, a significant gap exists between general-purpose DRL algorithms and the stringent requirements of industrial-grade assembly measurement. Such applications demand not only path optimality but also provable stability, rapid convergence in training, and reliable operation in semi-structured environments with both static and dynamic obstacles. This paper bridges this gap by proposing the N-step Priority Double DQN (NDDQN) algorithm. The original contribution of this work is threefold.

- (1) It presents a purpose-driven integration of proven DRL components (Double Q-learning, PER, N-step) specifically optimized for the challenges of mobile robot navigation in aerospace assembly environments.
- (2) It provides a comprehensive validation methodology, utilizing both abstract grid worlds and high-fidelity Gazebo simulations that closely mimic real-world measurement scenarios, a step beyond typical maze or simple static environment tests.

- (3) It delivers an efficient and robust path planning solution that demonstrates superior sample efficiency and convergence stability compared to baseline algorithms, making it a practical candidate for real-world industrial deployment.

The rest of this paper is organized as follows: Section 2 presents the theoretical framework and related work. Section 3 is problem description and modeling. Section 4 introduces the improved deep reinforcement learning algorithm. Section 5 presents the experimental verification and analysis. Section 6 summarizes the content of the study and outlines areas for future research.

2. Theoretical Framework

Deep reinforcement learning (DRL) has achieved remarkable advancements in the realm of path planning. As a prominent deep reinforcement learning algorithm, DQN has been successfully applied across various domains, including energy management, workshop scheduling, and robotic obstacle avoidance [20]. It exhibits considerable efficacy in addressing path planning challenges within uncertain environments. Smart et al. [21] were pioneers in applying Q-learning to the problem of robot navigation, developing a straightforward reward function to assist robots in identifying viable solutions. Low et al. [22] introduced an enhanced DQN designed to tackle path planning challenges, thereby improving the convergence speed of Q-learning. However, this approach encounters difficulties in effectively adapting to complex working environments during the path planning process. Cheng et al. [23] proposed a streamlined deep reinforcement learning algorithm for obstacle avoidance, leveraging a robust DQN architecture that effectively addresses both obstacle avoidance and path planning challenges faced by unmanned vessels. Lou et al. [24] advanced a method that integrates DQN with prior knowledge through the design of a fuzzy logic controller, which significantly aids robots in mitigating unproductive exploration and consequently reduces training time. To solve the sequential decision-making problem of ship collision avoidance, Yang et al. [25] improved DQN by sampling priority and adopting non-uniform sampling techniques, subsequently validating their proposed algorithm within a simulation environment. Nakamura et al. [26] utilized DQN to obtain the control strategy for path planning on narrow roads, facilitating the robot's ability to reach its target with minimal speed variations and fewer turns in constrained environments. To achieve collection-free path planning in complex environments, Gu et al. [27] applied M-DQN to address the mobile robots' path planning challenges. By decomposing the network architecture and incorporating an artificial potential field as a reward function, they attained an enhanced learning strategy. Hu et al. [28] proposed a multi-agent deep deterministic policy gradient method to solve the anti-collision problem of AGVs in container terminal transportation, utilizing the Gumbel-softmax technique to discretize scenarios generated by the node network. Li et al. [29] proposed a deep learning algorithm combining Angle search and DQN to achieve precise control over the search direction of mobile robots, thereby enhancing the efficiency of path planning. Wu et al. [30] proposed a path planning method for multi-robot systems based on distributed multi-agent deep reinforcement learning, allowing for local optimal path planning that take into account the behaviors of other robots. To address issues related to low solution efficiency and blind exploration in robot path planning, Li et al. [31] proposed an adaptive Q-learning algorithm based on virtual target guidance. This method incorporates a dual memory mechanism and an adaptive greed factor, effectively improving the convergence speed of the algorithm.

To enable robots to make adaptive decisions in complex environments, DRL brings new potential to robot path planning. Hua et al. [32] present an advanced hierarchical motion planning strategy that synergistically combines DRL with collaborative design of motion optimization, aiming to achieve efficient and seamless navigation in unfamiliar en-

vironments. Zhao et al. [33] introduced an advanced hierarchical path planning framework that intricately integrates global path planning with local dynamic obstacle avoidance, leveraging the power of deep reinforcement learning alongside the Grey Wolf optimization algorithm. This innovative approach aims to facilitate efficient and secure navigation for robots operating within unpredictable environments. Dawood et al. [34] employed multi-agent reinforcement learning (MARL) to address the intricacies of behavior-based cooperative navigation for mobile robots, successfully achieving a paradigm of safe and efficient collaborative movement. Jonnarth et al. [35] explored the potential of continuous spatial reinforcement learning in the realm of coverage path planning, empowering agents to adeptly adapt to distinct environmental characteristics. Jiao et al. [36] introduced a deep reinforcement learning algorithm featuring automatic entropy adjustment tailored for shearing, aimed at enhancing the efficacy of strategy acquisition in mobile robots by effectively mitigating the evaluation of entropy.

Despite the remarkable advancements of DRL algorithms in the realm of path planning, they still harbor certain shortcomings [37]. To address these issues regarding slow convergence rates and inadequate generalization capabilities in DRL, we introduce N-step priority strategy and double Q-learning techniques. This work aims to propose an N-step priority double DQN learning algorithm (NDDQN), wherein the learning strategy leverages not only the current state of the agent but also its target location. By selecting appropriate actions grounded in this comprehensive understanding, we guide the robot in its pursuit of optimal path. To clearly delineate the positioning and innovation of our method against a broader landscape of DQN variants, a structural comparison is provided in Table 1. Unlike the general-purpose and computationally intensive Rainbow DQN, our proposed NDDQN is a streamlined, task-oriented algorithm that integrates a minimal set of components, Double Q-learning, Prioritized Experience Replay, and N-step returns, specifically chosen to address the core challenges of efficiency, stability, and credit assignment in mobile robot path planning. This design avoids the complexity and computational overhead of other general-purpose frameworks while maintaining high performance and stability, making it particularly suitable for efficient and stable online training in practical robotics applications with limited computational resources.

Table 1. Structural comparison between the proposed algorithm and prior DQN variants.

Algorithm	DQN	Double DQN	Dueling DQN	Rainbow DQN	NDDQN (Ours)
Experience Replay	✓	✓	✓	✓	✓
Target Network	✓	✓	✓	✓	✓
Double	×	✓	✓	✓	✓
Q-learning	×	×	×	✓	✓
Prioritized ER	×	×	×	✓	✓
N-step Learning	×	×	×	✓	✓
Dueling Network	×	×	×	✓	×
Noisy Nets	×	×	×	✓	×
C51	×	×	×	✓	×
Sample Efficiency	Medium	Medium-High	Medium	High	High
Computational Overhead	Medium	Low	Medium	Very High	Low
Primary Design Goal	Stability	Reduce Overestimation	Better Value Estimation	Maximize General Performance	Efficiency & Stability in Robotic Path Planning
Key Characteristic	Foundational	Incremental Improvement	Architectural	Comprehensive & Complex	Streamlined & Task-Oriented

3. Problem Description and Modeling

3.1. Problem Description

The assembly measurement process for complex products is conducted by a mobile measurement robot equipped with a scanner, which automatically scans the workpiece along a predetermined trajectory. Taking the assembly measurement process of an aircraft tube section as an example, the simplified schematic model is illustrated in Figure 1. The measurement robot utilizes the scanner to perform localized scanning operations, ensuring that the scanning path runs parallel to the axis of the large-diameter cylindrical product. Upon completing the scanning process for one measuring unit and transitioning to the next, the robot executes a horizontal movement to facilitate this transfer operation.

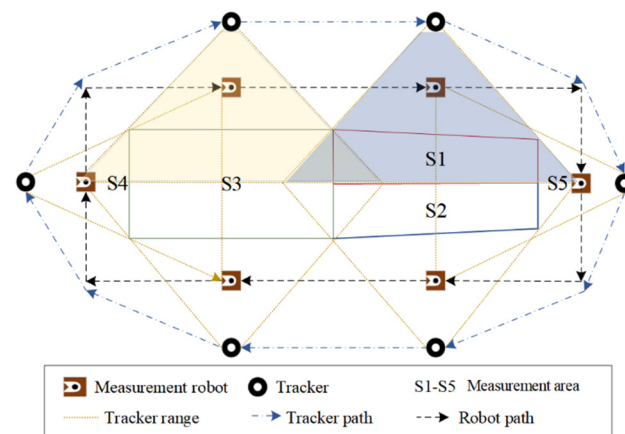


Figure 1. Schematic diagram of the measurement path of the mobile robot.

The measurement task encompasses five distinct measurement units: S1 and S2 are conical surfaces, while S3 is a cylindrical surface. Additionally, S4 and S5 consist of symmetrically arranged circular planes. The measurement area encompasses overlapping regions that enhance image stitching capabilities, thereby improving overall scanning accuracy. The measurement robot conducts shape scanning of each component along a predetermined trajectory path and subsequently acquires point cloud data. The measurement robot and the calibration target are positioned within the visible range of the tracker, which plans its movement trajectory based on an optimized scanning path. The primary function of the tracker is to ascertain the spatial position of the measuring robot, thereby facilitating an indirect determination of the scanner's spatial position. Scanning data is subsequently transferred into a global coordinate system via the laser tracker's measurement coordinate system, effectively achieving unification within the spatial measurement domain. The object under examination undergoes a comprehensive scanning process. It is crucial for the tracker to maintain proximity to this object in order to ensure precise measurements.

3.2. Dynamic Modeling for Mobile Robots

The mobile measurement robot is primarily utilized for the precise scanning of assembly dimensions in AMPP. This system comprises omnidirectional mobile automated guided vehicles (AGVs), industrial robotic arms, distance sensors, visual guidance systems, and end support devices. Equipped with five ultrasonic distance sensors, the mobile measurement robot can effectively determine its directional orientation. The angle between each direction measured by these ultrasonic sensors is set at 45 degrees. The kinematic structure of the robot, along with its integrated sensors, is depicted in Figure 2. In this illustration, XOY represents the global coordinate system, while X'OY' denotes the local coordinate system of the robot. The term W_p refers to the distance between the left and right wheels. When a robot performs a turn, the point around which it rotates is referred to

as the instantaneous curvature center R_c . θ represents the angle between the center of the robot and the global coordinate system. The position of the robot can be expressed using global coordinates (R_x, R_y) and its directional orientation.

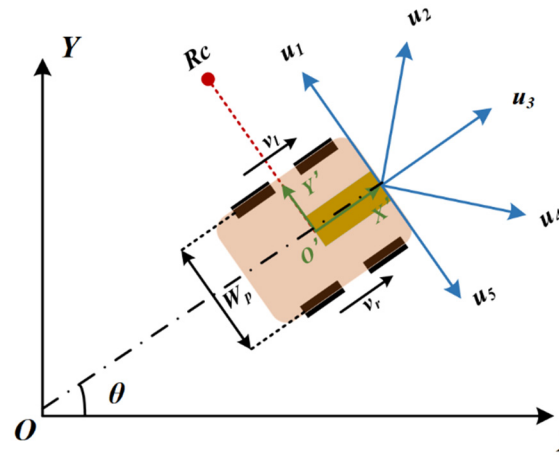


Figure 2. Dynamic structure analysis of the mobile robot.

The mobile robot's movement is facilitated through a differential drive system. Consider that the left front wheel and the left rear wheel of the robot operate at an identical linear velocity, denoted as v_l , while the right front wheel and the right rear wheel maintain a consistent linear velocity represented as v_r . The forward linear velocity v and the turning angular velocity ω of the robot are as shown in Formula (1).

$$\begin{cases} v = \frac{v_r + v_l}{2} \\ \omega = \frac{v_r - v_l}{W_p} \end{cases} \quad (1)$$

When the robot turns, the instantaneous curvature radius R_r between the robot center and the curvature center is calculated as Formula (2):

$$R_r = \frac{W_p(v_r + v_l)}{2(v_r - v_l)} \quad (2)$$

Generally, the motion data directly read by mobile robots is based on the local coordinate system. The motion data in the global coordinate system is obtained through Euler transformation, and the formula is using Formula (3).

$$\begin{bmatrix} R_x \\ R_y \\ \theta \end{bmatrix} = \begin{bmatrix} \cos(\omega dt) & -\sin(\omega dt) & 0 \\ \sin(\omega dt) & \cos(\omega dt) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{cx} \\ R_{cy} \\ \theta \end{bmatrix} \quad (3)$$

In light of the premise of uniform motion, the trajectory generation and assessment for the mobile robot will be directly derived from the sequence of actions undertaken, thereby obviating the need for explicit integration of left and right wheel velocities in computational processes. This comprehensive model is meticulously crafted to furnish a robust theoretical foundation for evaluative indicators while simultaneously establishing interfaces that anticipate future investigative endeavors.

This dynamic model provides the foundational kinematics for the subsequent reinforcement learning agent. It is important to note that the model and the ensuing path planning validation operate under a key assumption of a perfectly flat and level ground surface, which is characteristic of the targeted aerospace assembly workshop environment. Given this ideal terrain, combined with the robot's low operating speed and a design that ensures a low center of gravity, the lateral and dynamic stability of the robot during the

generated maneuvers is considered inherently maintained. Therefore, explicit stability analysis concerning the risk of tipping was outside the scope of this study, which focuses on navigation and obstacle avoidance in a structured, flat environment.

4. Improved Deep Reinforcement Learning Algorithm

4.1. Markov Process

The path planning problem for mobile robot installation and testing can be regarded as a Markov process. In each time step, the robot agent will choose an action to change the environmental state, which is represented as a five-tuple (S, A, p, r, γ) . The environment returns the next state s_{t+1} and reward r of the robot agent, and the robot agent selects the next action a_{t+1} based on the reward. Under the time step t , the state of the environment is s_t , the agent selects action a_t , and the environment provides the next state s_{t+1} , the next reward r_{t+1} and the discount factor γ , where the discount factor is used for the discount reward, and $\gamma \in [0, 1]$. The set of all states is the state space $S = \{s_1, s_2, \dots, s_n\}$, and the set of all actions is the action space $A = \{a_1, a_2, \dots, a_n\}$. The probability distribution of each state behavior is strategy π . The robot agent selects action a through strategy π , and $\pi(a|s)$ represents the probability of choosing action a in state s . The transfer function $p(s'|s, a)$ represents the probability of obtaining the next state s' under state s and behavior a , and $r(s, a)$ represents the reward for choosing behavior a under state s , as Formula (4).

$$r(s, a) = \mathbb{E}[r_{t+1}|s_t, a_t] = \sum_r r \sum_{s'} p(s'|s, a) \quad (4)$$

The cumulative rewards of the robot agent are as follows.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (5)$$

where the discount factor of step k is $\gamma_t^k = \prod_{i=1}^k \gamma_{t+i}$, and the goal of reinforcement learning is to maximize the cumulative reward G_t .

The value estimation of reinforcement learning is defined as a value function, which is divided into state value functions and action value functions. The state value function represents the expected reward of the strategy in the state, and the action value function represents the expected reward of the strategy taking actions in the state, as shown in Formulas (6) and (7). Formula (6) is a state value function, representing the expected cumulative return of state s under strategy π . Formula (7) is the action value function, representing the expected cumulative return of performing action a in state s and following strategy π .

$$\begin{aligned} v_{\pi}(s) &\approx \mathbb{E}[G_t|S_t = s] = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t \right] \\ &= \sum \pi(a|s) \sum p(s'|s, a) [r_t + \gamma v_{\pi}(s')] \end{aligned} \quad (6)$$

$$q_{\pi}(s, a) \approx \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t, a_t \right] \quad (7)$$

The relationship between v_{π} and q_{π} can be expressed as $v_{\pi} = \max_{a \in A} q_{\pi}(s, a)$. The optimal strategy $\pi^*(s)$ selects the action a with the maximum action value for each state s , as shown in Formula (8).

$$\pi^*(s) = \operatorname{argmax}_{a \in A} v_{\pi}(s'|s, a) = \operatorname{argmax}_{a \in A} q_{\pi}(s, a) \quad (8)$$

The optimal state value and the optimal action value can be calculated as

$$v^*(s) = \max_a \sum_{s'} p(s'|s, a) [r + \gamma v^*(s')] \quad (9)$$

$$q^*(s, a) = \max \mathbb{E}[r_{t+1} + \gamma v^*(s') | s_t, a_t] \quad (10)$$

Therefore, the goal of reinforcement learning can be transformed into solving the optimal state value $v^*(s)$ and action value $q^*(s, a)$.

4.2. Double-Q Learning

When the state space or action space is excessively large, accurately estimating the value of each action becomes a challenging task. Consequently, the incorporation of deep neural networks to approximate the action value function has given rise to what is known as deep reinforcement learning. In the context of deep Q-networks, the optimization objective is to progressively minimize the discrepancy between the estimated return value and the predicted action value. This involves updating network weight w through loss function optimization, as illustrated in Formula (11).

$$L(w) = \mathbb{E}_{\pi} \left[\left(r + \gamma \max_{a'} (s', a'; w) - q_{\pi}(s, a; w) \right)^2 \right] \quad (11)$$

In the process of enhancing DQN, the introduction of a target network serves to improve stability. In DQN's update mechanism, the neural network utilized for estimating rewards is identical to that used for action value estimation. Consequently, it leads to fluctuations in the estimated values as parameters vary. The updating of DQN's action values can easily result in an overestimation bias. To address this issue of overestimation, Double Q-learning employs a strategy where actions are selected using the evaluation network while utilizing the target network for reward estimation. Accordingly, the loss function is represented as shown in Formula (12).

$$L(w) = \mathbb{E}_{\pi} \left[\left(r + \gamma \max_{a'} q_{w_t}(s', \arg \max q_w(s', a')) - q_w(s, a) \right)^2 \right] \quad (12)$$

where q_{w_t} is the target network.

4.3. Priority Experience Reply

Experience replay serves as a storage mechanism that retains actions, behaviors, and rewards within a discrete time step. During the training process, samples are drawn from the stored experiences based on specific criteria, after which the network is updated utilizing these sampled experiences. This paper employs prioritized experience replay (PER) to enhance the weight priority of the stored experiences. In DQN, the temporal difference (TD) error represents the discrepancy between the target value and the current value. A larger TD error significantly influences backpropagation. Conversely, a smaller TD error results in reduced influence of that sample on gradient descent updates. Refining this concept further, prioritized experience replay can expedite the convergence speed of the network. This work introduces SumTree to store samples in priority experience replay. All empirical samples are stored at the bottom leaf nodes, where the leaf nodes preserve the sample priorities. The top of the SumTree is the sum of the experiences of all samples. Then, the loss function of the sample priorities becomes Formula (13).

$$L(w) = \mathbb{E}_{\pi} \left[w_j \left(r + \gamma \max_{a'} q_{w_t}(s', \arg \max q_w(s', a')) - q_w(s, a) \right)^2 \right] \quad (13)$$

where w_j is the priority weight of the j th sample. After performing gradient update on the Q network, the TD error needs to be recalculated and the SumTree updated.

4.4. Priority N-Step Update Strategy

The above-mentioned method is a one-step update method, which only uses the information of the current time step and the next time step, that is

$$G_t = r_{t+1} + \gamma v_t(s_{t+1}) \quad (14)$$

To accelerate the learning process, an N-step update strategy is adopted and multi-step information is used for updates. The return value of step N is expressed as

$$G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n v_{t+n-1}(s_{t+n}) \quad (15)$$

Then, the loss function is as Formula (16).

$$L(w) = \mathbb{E}_{\pi} \left[w_j \left(r_t^n + \gamma^n \max_{a'} q_{w_t}(s_{t+n}, \arg \max q_w(s_{t+n}, a')) - q_w(s, a) \right)^2 \right] \quad (16)$$

The integration of the N-step return strategy is mathematically and conceptually critical for addressing a fundamental challenge in robotic path planning: temporal credit assignment. In our task, the most critical rewards, collision penalty and success reward, are highly sparse and often occur many steps after the causative actions. The conventional one-step TD target G_t only propagates reward information one step per update, leading to slow learning and difficulty in associating early actions with long-term consequences.

The N-step return $G_t^{(n)}$ provides a more informed bootstrap target by directly incorporating the cumulative reward over a multi-step trajectory. This effectively shortens the temporal distance between actions and their significant outcomes. For instance, an action that places the robot on a trajectory towards an obstacle may only result in a collision N steps later. With one-step updates, this negative outcome is poorly attributed. With N-step updates, the connection is made explicit, allowing the agent to rapidly learn to avoid such sequences. This leads to faster policy convergence and more stable learning, which is empirically confirmed by our ablation study, where the removal of N-step returns caused the most significant degradation in training speed.

4.5. The Framework of NDDQN Algorithm

The neural network architecture used for function approximation in our NDDQN agent is a fully connected Deep Q-Network (DQN). We deliberately adopted a standard Multi-Layer Perceptron (MLP) design rather than a more complex contemporary architecture. This decision was motivated by a focus on isolation of variables, to ensure that the performance gains observed in our experiments are attributable to our core algorithmic contributions (the synergistic integration of Double Q-learning, PER, and N-step returns) rather than to the representational capacity of a specialized network. This architecture strikes a balance between model expressiveness and training stability, reducing the risk of overfitting and contributing to the reliable convergence observed in our results, a paramount consideration for developing robust solutions for industrial applications.

Figure 3 illustrates the framework predicated on the NDDQN algorithm. We have devised a path planning model for mobile robots utilizing NDDQN, framing the path planning conundrum as a Markov decision process. Subsequently, the mobile robot acquires environmental data through an array of sensors and meticulously calculates both the direction and distance to obstacles as well as target objects. The mobile robot selects action values from the replay buffer by employing prioritized experience replay while simultaneously referencing the reward function. These action values and states are then processed through layers of neural networks, culminating in the output of Q values aimed at minimizing loss functions, which are subsequently fed back into the neural network for parameter updates. Upon reaching its designated endpoint, it assesses whether convergence has been achieved. If convergence is confirmed, the program concludes; otherwise, it persists in generating new target endpoints and engaging with its environment until

termination occurs. Ultimately, this interaction enables the mobile robot to gather training data and refine its learning based on sampled information.

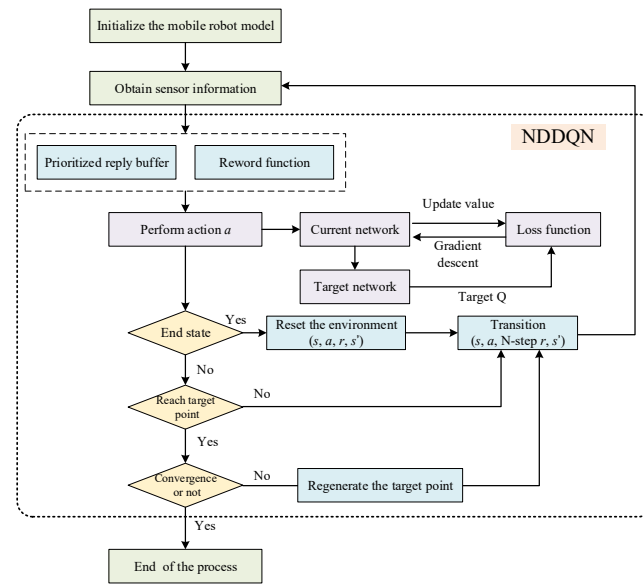


Figure 3. The framework of proposed algorithm.

5. Experimental Verification and Analysis

5.1. Experimental Environment

To verify the effectiveness of the proposed NDDQN algorithm, comparative experiments were conducted in two training environments, namely the virtual Grid Map environment and the Gazebo simulation environment. The Grid Map experiment builds a GUI environment model on the built-in standard library of PyCharm 2024. The experiment adopted a $1\text{ m} \times 1\text{ m}$ grid map, where black squares represent obstacles, blank areas are passable zones, the lower left corner is the starting point, the upper right corner is the target point, and the black lines throughout the map represent the grid. The developed grid environments are 10×10 , 20×20 and 30×30 , respectively. The robot agent is located on the map by the coordinates of the two diagonal points.

The Gazebo simulation environment similar to the real-world environment is: an environment with static obstacles (E1); An environment containing both static and dynamic obstacles (E2). The Gazebo robot serves as a mobile measurement platform for testing in a simulated environment. The environment E1 has 13 fixed rectangular obstacles, with white objects serving as static obstacles. The layout of environment E2 is more complex. Two blue cylinders are randomly placed in the environment as moving obstacles. Six target points (P1 to P6) are placed in the unobstructed area within the environment.

The total number of training rounds of the simulation experiment is set to 1000 episodes. When the agent reaches the target or collides, the current round terminates. The training is completed when the total reward of the agent reaches a stable and satisfactory level. The computer environment is 2.1 GHz Intel Core i7-12700, 128 GB of memory, and the operating system is windows11. Various path planning algorithms are implemented using Python 3.12 and invoke the built-in interfaces. TensorFlow is used to define neural network architectures, enabling models to gain GPU-accelerated advantages during the training process. In all the simulation experiments, the same robot model parameters were set. The mobile robot is trained by different algorithms, and then experiments are conducted using the empirical dataset generated from the training to verify the performance of the proposed algorithms. To ensure the statistical significance and reproducibility of the results, all simulation experiments for each algorithm and scenario were independently

repeated three times with different random seeds. The performance metrics reported in the following sections, unless otherwise specified, are presented as the mean value along with the standard deviation derived from these three runs. Table 2 presents the network hyperparameters of the NDDQN algorithm, detailing both network and training parameters. The fully connected layers of the network are managed substantial amounts of data, with the first layer containing 256 units, the second with 256 units, and the third with 128 units. This study employs a maximum epoch limit of 1000 across all training algorithms and establishes a maximum exploration step threshold of 800 for continuous learning as well as algorithm feasibility evaluation. The mobile robot learning rate is fixed at 0.005. To facilitate efficient training and enhance the learning rate of the NDDQN algorithm, we set the minimum batch learning size to 64 and the discount factor to 0.95. The proposed algorithm network updates occur at a 0.001 update rate between the current and target networks. The optimal distance that a mobile robot should maintain while in motion is set at 0.5 m. Comparative simulation experiments were conducted, respectively, for the NDDQN algorithm, the proximal policy optimization (PPO) algorithm and the DQN algorithm [24] in different scenarios, and the convergence was analyzed.

Table 2. This NDDQN Hyperparameters.

Hyperparameter	Value
The first layer network nodes	256
The second layer network nodes	256
The third layer network nodes	128
Max episodes	1000
Max steps	800
Learning rate (α)	0.0005
Mini-batch	64
Discount factor (γ)	0.95
Network update rate	0.001
The safe distance	0.5 m
Playback cache storage capacity	50,000

5.2. Deep Learning Parameter Settings

The settings of deep reinforcement learning include states, goals, behaviors and reward functions. The status is determined by the information obtained by the ultrasonic sensor. The target includes the distance d from the current position of the robot to the target point, as well as the relative direction θ' of the target point relative to the robot. The calculation formula is as follows.

$$d = \sqrt{(R_x - T_x)^2 + (R_y - T_y)^2} \quad (17)$$

$$\theta' = \begin{cases} \arctan \frac{T_x - R_x}{T_y - R_y} - \theta(T_x - R_x) \geq 0 \\ \arctan \frac{T_x - R_x}{T_y - R_y} + \pi - \theta(T_x - R_x) < 0 \end{cases} \quad (18)$$

where (R_x, R_y) and (T_x, T_y) are the coordinates of the robot and the target point relative to the global coordinate system.

For the behavior of mobile robots, it is manifested as five actions: forward movement, left turn, right turn, quick left turn and quick right turn. The forward linear velocity v and the turning angular velocity ω corresponding to the action are shown in Table 3. Each action lasts for 1 s. After the agent completes the current action, it will re-read the status and target information and proceed to the next action.

Table 3. Robot Actions and Corresponding Speed.

Action	Linear Velocity (v)	Angular Velocity (ω)
Move forward	0.8 m/s	0 rad/s
Turn left	0.8 m/s	−0.5 rad/s
Turn right	0.8 m/s	0.5 rad/s
Turn left quickly	0.8 m/s	−1 rad/s
Turn right quickly	0.8 m/s	1 rad/s

When setting the reward function, the reward depends on the distance of the obstacle, the collision with the obstacle, and the comprehensive score of reaching the target point. The reward function includes a negative reward r_1 for collision, a reward r_2 for reaching the target point, and a positive reward r_3 for maintaining a safe distance. To prevent the agent from colliding with the surrounding environment during movement, when the robot agent collides, it will receive a negative reward of $r_1 = -100$. To encourage agents to move towards the target point, the closer the agent is to the target, the higher the reward will be. When the agent successfully reaches the target position, a positive reward of $r_2 = 100$ will be directly given. When the agent approaches the collision obstacle, the states of the robot agent are divided into the safe distance state S and the non-safe distance state NS . To reduce the possibility of colliding with obstacles, the definition of the safety distance reward is as follows.

$$r_3 = \begin{cases} 5, & S \rightarrow S \\ -5 & S \rightarrow NS \\ 10, & NS \rightarrow S \end{cases} \quad (19)$$

Therefore, when the agent reaches the target point, the comprehensive reward function is as Formula (20).

$$r = r_1 + r_2 + r_3 \quad (20)$$

5.3. Simulation Results Analysis in Grid Map

To verify the effectiveness of the proposed NDDQN algorithm in solving the path planning problem, simulation experiments were conducted using grid obstacles of different scales to verify the effectiveness of the proposed algorithm in solving the path planning problem. To achieve the simulation scenarios of small-scale, medium-scale and large-scale examples, grid simulation examples of three scenarios, namely 10×10 , 20×20 and 30×30 , are constructed. In this paper, four parameters, namely the total substitution value of the path, the maximum completion time, the total length of the path and the total number of turns of the path, are adopted as the evaluation indicators for solving the quality of the path. After 800 generations of training, the convergence of the models trained by NDDQN, DQN and PPO was compared. The comparison of the average rewards of the three algorithms is shown in Figures 4–6. In small-scale simulation examples, the convergence speeds of the three algorithms do not differ much and basically achieve convergence after 100 generations of training. In the medium-scale simulation results, NDDQN has a relatively fast convergence speed and a high reward value. The three algorithms reach convergence around the 130th generation. In the large-scale simulation results, the iteration speed of NDDQN is significantly accelerated. It can avoid obstacles and reach the target point more quickly, and the number of training generations required to reach the target is smaller than that of the other two algorithms. With the increase in environmental complexity, the number of convergence generations required by the agent has a continuous increasing trend, and the final reward value is also increasing.

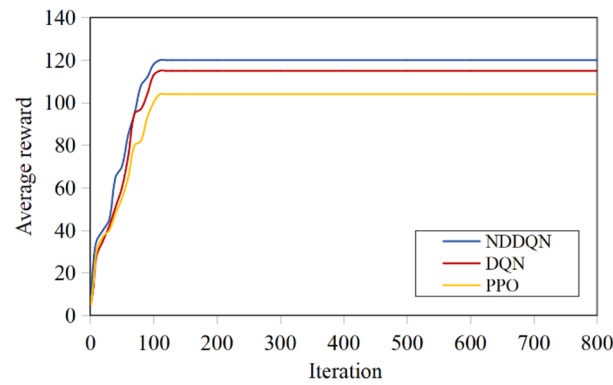


Figure 4. The average reward value of training in the 10×10 scenario.

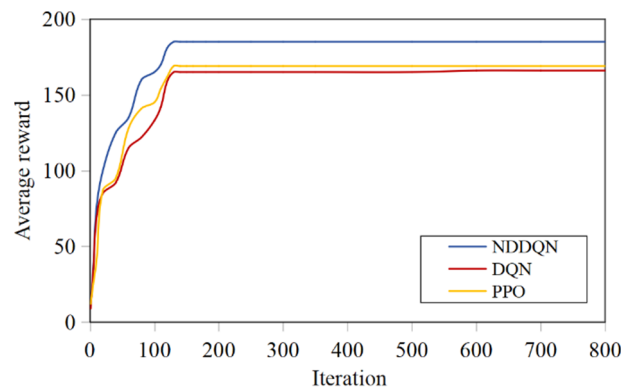


Figure 5. The average reward value of training in the 20×20 scenario.

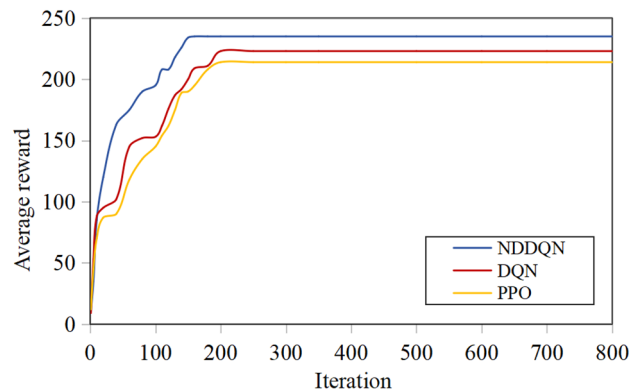


Figure 6. The average reward value of training in the 30×30 scenario.

To compare the stability of the models trained by the three methods, 50 tests were conducted on the trained models under the same conditions. The average results are shown in Table 4. It can be seen from the training results of the three scenarios that the arrival time, total path length, turning times and average curvature of NDDQN are significantly lower than those of the other two algorithms, and the total reward values are all higher than those of DQN and PPO algorithms. The powerful learning ability of NDDQN is more prominent in complex scenarios, reaching the destination with the fewest turns and the highest reward value. The paths selected by the three algorithms when successfully reaching the target point are shown in Figures 7–9. Among them, the blue line represents the NDDQN planning path, the yellow line represents the DQN planning path, and the red line represents the PPO planning path. It can be seen from the simulation paths that the paths for NDDQN and DQN to reach the target point are similar, and the number of turns required

to reach the target is relatively small. Although the path planned by the DQN algorithm is similar to that of the NDDQN algorithm, the path length is longer, so the time is longer. The path planning obstacle avoidance ability of the PPO algorithm is more prominent. However, during the exploration process, there are more turns, resulting in a larger total path length and an increase in the time to reach the destination accordingly. To sum up, the experimental simulation results demonstrate the effectiveness of the NDDQN algorithm, which can help mobile robots obtain more effective obstacle avoidance experience in the task of measurement path planning.

Table 4. Path-solving Performance Indicators of Different Example.

Grid	Algorithm	Total Reward	Time	Total Length	Number of Turns	Average Curvature
10×10	NDDQN	121.3 ± 5.2	19.41 ± 0.85	14.33 ± 0.41	3.0 ± 0.0	4.2 ± 0.3
	DQN	115.7 ± 8.1	19.69 ± 1.12	14.55 ± 0.62	3.3 ± 0.6	4.4 ± 0.5
	PPO	104.5 ± 10.3	20.33 ± 1.54	14.66 ± 0.78	4.3 ± 0.6	5.8 ± 0.9
20×20	NDDQN	183.5 ± 6.8	38.46 ± 1.92	28.78 ± 0.87	5.0 ± 0.0	3.8 ± 0.2
	DQN	166.2 ± 9.5	38.78 ± 2.15	29.02 ± 1.03	5.7 ± 0.6	4.1 ± 0.4
	PPO	169.8 ± 12.1	41.31 ± 2.88	30.25 ± 1.45	7.3 ± 0.6	5.2 ± 0.7
30×30	NDDQN	233.8 ± 8.1	53.06 ± 2.65	40.05 ± 1.20	7.3 ± 0.6	3.9 ± 0.3
	DQN	221.4 ± 11.7	57.40 ± 3.44	42.32 ± 1.69	10.0 ± 1.0	5.5 ± 0.8
	PPO	214.6 ± 15.2	57.96 ± 4.06	43.17 ± 2.15	9.0 ± 1.0	4.8 ± 0.6

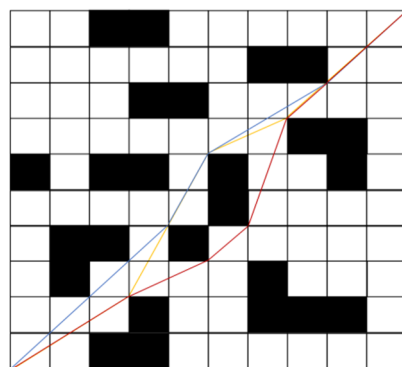


Figure 7. The 10×10 raster simulation results of three algorithms.

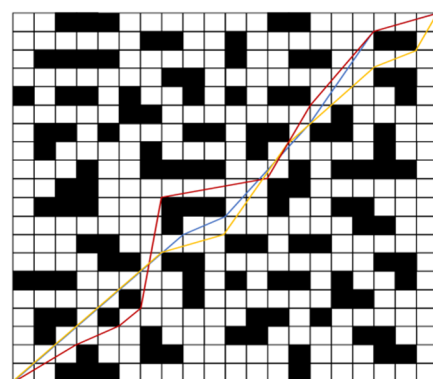


Figure 8. The 20×20 raster simulation results of three algorithms.

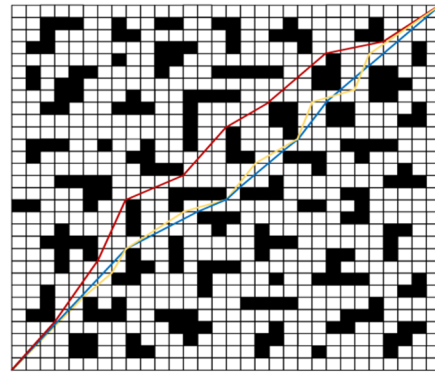


Figure 9. The 30×30 raster simulation results of three algorithms.

5.4. Simulation Results Analysis in Gazebo

In the E1 and E2 test environments, there are six selected navigation points (P) and two moving obstacle points (D). The robot's task is to independently explore the environment, traverse from position $P1$ to $P6$, and then return to the initial position S without colliding with obstacles. To verify the effectiveness of the proposed NDDQN algorithm, we conducted path planning simulations in two Gazebo simulation environments. In scenario E1, the robot achieved path exploration from position $P1$ to $P6$ and returned to the target position S , as shown in Figure 10. In scenario E2, the robot completed the path exploration from position $P1$ to $P6$, avoided two dynamic obstacles, and successfully returned to the target position S , as shown in Figure 11.

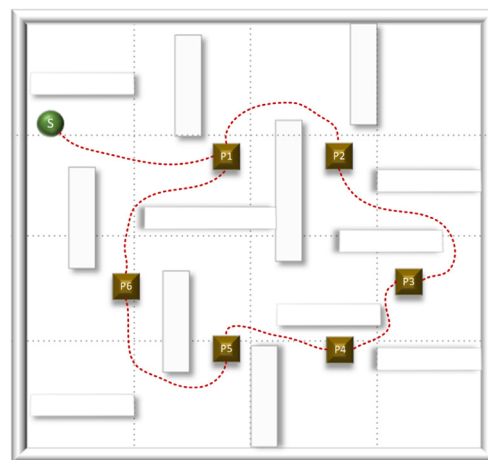


Figure 10. Generated path for NDDQN algorithm in E1.

In the static environment, the robot meticulously devises a smooth and nearly optimal trajectory from S to $P6$ and back, exemplifying the algorithm's remarkable efficiency in navigating obstacle-free pathways. As illustrated in Figure 10, the robot's trajectory is direct and streamlined, epitomizing a goal-oriented path characterized by minimal redundancy. In stark contrast, the dynamic environment presents two mobile obstacles, as depicted in Figure 11. The generated path clearly reveals noticeable detours and recalibration points where the robot adeptly anticipated and circumvented collisions prior to advancing towards its target and returning to S . These detours are far from arbitrary. They represent strategic avoidance maneuvers executed with precision to uphold a safe distance from the projected trajectories of dynamic obstacles. This vividly underscores the algorithm's robustness and real-time responsiveness when confronted with dynamic impediments. The comparative analysis conclusively demonstrates that while the algorithm excels at

identifying optimal paths within static contexts, its paramount strength lies in its predictive avoidance capabilities amidst dynamic scenarios.

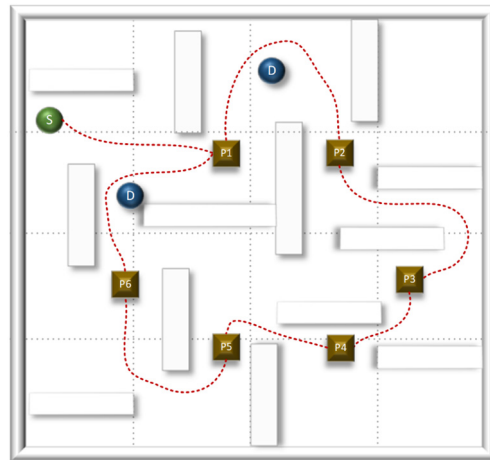


Figure 11. Generated path for NDDQN algorithm in E2.

The ability of the cumulative reward response algorithm path planning for robot agents. To more intuitively reflect the characteristics of each agent reward, Figures 12–14 present the total rewards of the three algorithms in the E1 environment. As shown in Figure 14, the NDDQN algorithm is significantly superior to other algorithms. The reward converging to 2600 is significantly higher than 1000 for PPO and 1700 for DQN. Similarly, in the more complex E2 environment, the performance gap is even greater. The superior performance of NDDQN in both environments proves that it can learn the optimal strategy more effectively. The simplicity of PPO limits its performance. Although DQN performs well, it is still surpassed by the efficient learning ability of NDDQN.

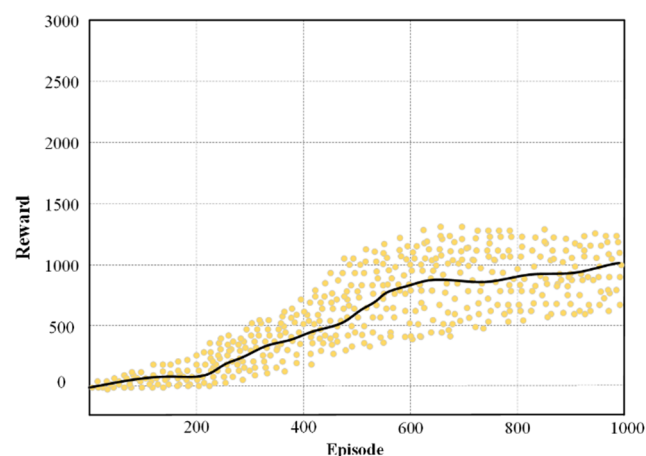


Figure 12. The robot's reward for each episode based on PPO algorithm.

Table 5 compares the success rates and convergence times of the PPO, DQN, and NDDQN algorithms used for path planning in two scenarios. For the E1 scenario, the success rate of the NDDQN algorithm is 34.4% higher than that of the DQN algorithm and 58.5% higher than that of PPO. Table 4 simultaneously compares the convergence speeds of the three algorithms. The shortest time required for NDDQN to obtain the final target point is 114 min, while PPO and DQN take 248 and 153 min, respectively. For the E2 scenario, the introduction of dynamic obstacles leads to a decline in the performance of all algorithms. The success rate of NDDQN was 60.2% and 39.4% higher than that of PPO and DQN, respectively. NDDQN converges the fastest and completes the path task

in 142 min. This performance outperforms the DQN algorithm, which takes 193 min to converge, and is faster than PPO, which takes 286 min. These simulation results prove that NDDQN performs superior in both dynamic and static environments, demonstrating a faster convergence speed and learning efficiency.

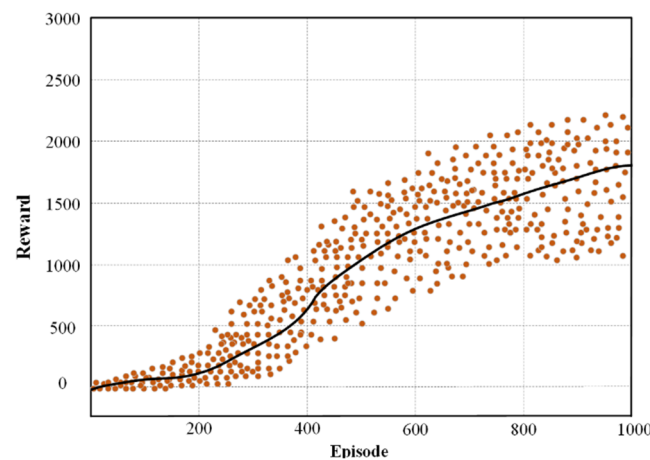


Figure 13. The robot's reward for each episode based on DQN algorithm.

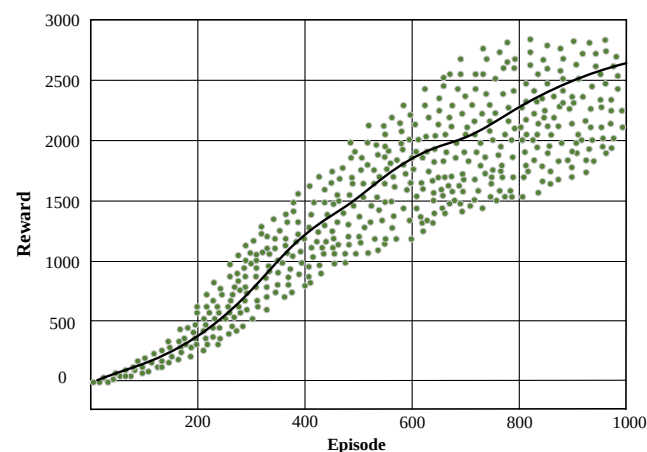


Figure 14. The robot's reward for each episode based on NDDQN algorithm.

Table 5. Comparison of Three Algorithms on Different Environments.

Method	Success Rate (%)		Convergence Time (min)	
	E1	E2	E1	E2
PPO	34.2 ± 4.5	28.4 ± 5.1	248 ± 22	286 ± 35
DQA	58.3 ± 3.8	49.2 ± 4.7	153 ± 15	193 ± 24
NDDQA	92.7 ± 2.1	88.6 ± 2.8	114 ± 12	142 ± 16

5.5. Ablation Study

To validate the efficacy of each integrated strategy employed in this study and its contribution to overall performance, we conducted a series of ablation experiments focusing on double Q -learning (DQL), priority experience replay (PER), and N -step returns. The objective of these experiments is to quantitatively assess the impact of omitting any given strategy on the algorithm's performance, thereby elucidating its essentiality. The experimentation was carried out within the Gazebo environment, allowing us to compare five distinct configurations: (1) Our model: NDDQN; (2) - N -step: excluding n -step returns while relying solely on single-step TD errors; (3) -PER: eliminating PER in favor of uniform

random sampling; (4) -DQL: removing DQL to revert to the target value computation method utilized by traditional DQN; and (5) Benchmark DQN: employing the standard DQN algorithm as a baseline for performance comparison. All models were configured with an identical set of hyperparameters, ensuring consistency across tests. Each configuration underwent independent execution for ten iterations to garner statistically significant results. In assessing asymptotic performance, we elected to utilize average final returns. Sample efficiency and learning speed were measured by the number of training steps required to achieve predetermined performance thresholds. Finally, stability and robustness were quantified through evaluating the standard deviation across multiple return runs.

The experimental results are shown in Table 6. All strategies are key contributors to the overall performance of the proposed algorithm, but their main aspects of influence are different. The *N*-step model has the most significant damage to the convergence speed of the algorithm. The number of training steps required to reach the threshold is approximately 45% higher than that of the NDDQN model, indicating that the multi-step reward mechanism plays a core role in accelerating temporal credit allocation and policy optimization. The -PER shows the most severe degradation in terms of final performance and stability, with the lowest average return and the largest standard deviation. This highlights the irreplaceability of PER in enhancing sample efficiency and avoiding training stagnation by prioritizing the replay of high-learning-value experiences. Although -DQL can achieve a final performance close to that of NDDQN model, its stability significantly decreases, verifying the crucial role of -DQL in alleviating *Q* value overestimation and ensuring smooth convergence in the learning process. Ultimately, the NDDQN model significantly outperformed all ablation variants and benchmark models in all evaluation dimensions. This not only demonstrated the effectiveness of each component but also indicated a positive synergy among them, collectively forming an efficient, stable, and robust reinforcement learning agent.

Table 6. Quantitative comparison of ablation experiment results.

Model	Average Returns	Training Steps	Standard Deviation
-N-step	450.2 ± 25.1	55.80 ± 6.50	25.1
-PER	420.5 ± 45.7	72.10 ± 8.90	45.7
-DQL	485.3 ± 22.5	41.00 ± 5.00	22.5
Baseline DQN	400.8 ± 52.0	>100.00	52.0
NDDQN	00.0 ± 15.3	38.500 ± 4.20	15.3

5.6. Sensitivity Analysis

To address the robustness of the proposed NDDQN algorithm against hyperparameter variations, a sensitivity analysis was conducted. Following the ablation study, we investigated the impact of three key hyperparameters: the learning rate (α), the discount factor (γ), and the target network update rate. The experiments were performed in the 20×20 grid environment, and the performance was evaluated based on the average cumulative reward over 50 episodes after training convergence. For each parameter, we maintained the other parameters at their optimal values and varied the target parameter across a defined range. The results are summarized in Table 7. The algorithm demonstrated moderate sensitivity to the learning rate. A value that is too low significantly slowed down the learning process, while a value that is too high caused instability and prevented the model from converging to an optimal policy. The range between 0.0001 and 0.001 provided stable and high-performing results, with 0.0005 being a robust choice. The performance showed a high degree of robustness and a predictable trend with respect to the discount factor. As γ decreased, the agent became more short-sighted, leading to a gradual decline in the average

reward. This smooth degradation confirms the theoretical role of γ and suggests that the algorithm's behavior is controllable and understandable. The algorithm exhibited very low sensitivity to the target network update rate. The final performance remained largely unaffected across the tested range, indicating that DQN framework effectively stabilizes the training process. A slower update rate slightly increased the time to convergence but did not impact the final policy quality. In conclusion, the sensitivity analysis confirms that the NDDQN algorithm is not fragile to hyperparameter tuning. It maintains robust performance across a reasonably wide range of settings for its most critical parameters, which enhances its practicality for real-world applications where optimal parameter selection might not always be feasible.

Table 7. Sensitivity analysis of key hyperparameters.

Hyperparameter	Value Range	Optimal Value	Robustness
Learning Rate (α)	0.0001–0.01	0.0005	Moderate
Discount Factor (γ)	0.9–0.99	0.95	High
Network Update Rate	0.0005–0.01	0.001	Very High

6. Conclusions

This study presents a deep reinforcement learning-based path planning algorithm for mobile robots in complex aerospace assembly and measurement tasks. This study demonstrates that the path planning challenges in mobile robot assembly measurement are effectively addressed by a deep reinforcement learning agent designed around three core principles: estimation stability, sample efficiency, and temporal credit assignment. The NDDQN algorithm embodies these principles through the synergistic integration of Double Q-learning, Prioritized Experience Replay, and N-step returns. This specific combination was shown to be crucial, as our ablation study confirmed, for generating reliable, collision-free paths while significantly reducing the training time in high-fidelity simulations, a critical advantage for industrial application. While the comparative experiments were conducted against foundational baselines (DQN, PPO), the superior performance of NDDQN should be interpreted through the lens of domain-specific requirements in robotic assembly measurement. The meaningful performance gains achieved by NDDQN are not merely a result of algorithmic complexity but stem from a deliberate design that aligns with the core challenges of the manufacturing domain. Future research will focus on tackling complex path planning problems involving multiple robots and numerous target points while analyzing the generalization capability of robotic collision avoidance against diverse types of obstacles to further enhance adaptive path planning abilities. To ultimately realize the goal of deployment, future endeavors must encompass the meticulous performance profiling of representative robotic computing platforms.

Author Contributions: Conceptualization, G.Y. and B.Z.; methodology, G.Y. and B.Z.; software, G.Y.; validation, B.Z., Y.H. and G.T.; formal analysis, G.Y.; investigation, B.Z.; resources, Y.H.; data curation, B.Z.; writing—original draft preparation, G.Y.; writing—review and editing, Z.L.; visualization, G.T.; supervision, Z.L.; project administration, G.Y.; funding acquisition, G.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding within the National Natural Science Foundation of China under Grant No.52405537, Humanities and Social Science Fund of Ministry of Education of China under Grant No.24YJCZH446, the National key Research and Development Program of China under Grant No. 2023YFE0198600, and the Jiangsu Excellent Postdoctoral Program under Grant No. 2023ZB634.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data can be obtained by contacting the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

(R_x, R_y)	The robot's position of the global coordinate.
(T_x, T_y)	The target point's position of the global coordinate.
v_l	The linear velocity of the left front wheel and the left rear wheel of the robot.
v_r	The linear velocity of the right front wheel and the right rear wheel.
v	The forward linear velocity of the robot.
ω	The turning angular velocity of the robot.
R_c	The instantaneous curvature center.
R_r	The instantaneous radius of curvature.
θ	The Angle between the center of the robot and the global coordinate system.
r	The reward of the robot agent.
s_t	The state of robot agent.
a_t	The action of the robot agent.
s_{t+1}	The next state of robot agent.
a_{t+1}	The next action of the robot agent.
r_{n+1}	The next reward of the robot agent.
γ	The discount factor, and $\gamma \in [0, 1]$.
S	The state space of robot.
A	The action space of robot.
π	The strategy of each state behavior.
$\pi(a s)$	The probability of choosing action a in state s .
$p(s' s, a)$	The probability of obtaining the next state s' under state s and behavior a .
$r(s, a)$	The reward for choosing behavior a under state s .
G_t	The cumulative reward of robot agent.
$\pi^*(s)$	The optimal strategy of robot agent.
$v^*(s)$	The optimal state value of robot agent.
$q^*(s, a)$	The optimal action value of robot agent.
w	The updating network weight.
$L(w)$	The loss function.
q_{w_i}	The target network.
w_j	The priority weight of the j th sample.
α	Learning rate.

References

- Chen, Z.; Li, T.; Jiang, Y. Image-based visual servoing with collision-free path planning for monocular vision-guided assembly. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 7508717. [\[CrossRef\]](#)
- Monaco, E.; Rautela, M.; Gopalakrishnan, S.; Ricci, F. Machine learning algorithms for delaminations detection on composites panels by wave propagation signals analysis: Review, experiences and results. *Prog. Aerosp. Sci.* **2024**, *146*, 100994. [\[CrossRef\]](#)
- Ali, M.; Das, S.; Townley, S. Robot differential drive navigation through probabilistic roadmap and pure pursuit. *IEEE Access* **2025**, *13*, 22118–22132. [\[CrossRef\]](#)
- Zhao, M.; Zhang, Q.B.; Zhao, T.; Tian, H. Research on reducing assembly errors in inductive encoders with observer phase detector phase-locked loop. *IEEE Sens. J.* **2023**, *23*, 28744–28751. [\[CrossRef\]](#)
- Sui, S.C.; Zhu, X.S. Digital measurement technique for evaluating aircraft final assembly quality. *Sci. Sin. Technol.* **2020**, *50*, 1449–1460. [\[CrossRef\]](#)
- Unhelkar, V.V.; Dorr, S.; Bubeck, A.; Lasota, P.A.; Perez, J.; Siu, H.C.; Boerkoel, J.C.; Tyroller, Q.; Bix, J.; Bartscher, S.; et al. Mobile robots for moving-floor assembly lines design, evaluation, and deployment. *IEEE Robot. Autom. Mag.* **2018**, *25*, 72–81. [\[CrossRef\]](#)
- Yuan, G.; Lv, F.; Shi, J.; Tian, G.; Feng, Y.; Li, Z.; Pham, D.T. Integrated optimisation of human-robot collaborative disassembly planning and adaptive evaluation driven by a digital twin. *Int. J. Prod. Res.* **2024**, 1–19. [\[CrossRef\]](#)

8. Xiang, Y.; Jin, X.; Lei, K.; Zhang, Q. Research on energy-saving and efficiency-improving optimization of a four-way shuttle-based dense three-dimensional warehouse system based on two-stage deep reinforcement learning. *Appl. Sci.* **2025**, *15*, 11367. [\[CrossRef\]](#)
9. Liu, Q.; Wang, C.G. A graph-based pipe routing algorithm in aero-engine rotational space. *J. Intell. Manuf.* **2015**, *26*, 1077–1083. [\[CrossRef\]](#)
10. Ni, J.; Ge, Y.; Zhao, Y.; Gu, Y. An improved multi-UAV area coverage path planning approach based on deep Q-Networks. *Appl. Sci.* **2025**, *15*, 11211. [\[CrossRef\]](#)
11. Chi, W.Z.; Ding, Z.Y.; Wang, J.K.; Chen, G.; Sun, L. A generalized voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots. *IEEE Trans. Ind. Electron.* **2022**, *69*, 4926–4937. [\[CrossRef\]](#)
12. Tong, X.L.; Yu, S.E.; Liu, G.Y.; Niu, X.; Xia, C.; Chen, J.; Yang, Z.; Sun, Y. A hybrid formation path planning based on A* and multi-target improved artificial potential field algorithm in the 2D random environments. *Adv. Eng. Inform.* **2022**, *54*, 101755. [\[CrossRef\]](#)
13. Qin, J.; Qin, J.; Qiu, J.; Liu, Q.; Li, M.; Ma, Q. SRL-ORCA: A socially aware multi-agent mapless navigation algorithm in complex dynamic scenes. *IEEE Robot. Autom. Lett.* **2024**, *9*, 143–150. [\[CrossRef\]](#)
14. Soualhi, T.; Crombez, N.; Ruichek, Y.; Lombard, A.; Galland, S. Learning decentralized multi-robot pointGoal navigation. *IEEE Robot. Autom. Lett.* **2025**, *10*, 4117–4124. [\[CrossRef\]](#)
15. Guralnik, D.P.; Stiller, P.F.; Zegers, F.M.; Dixon, W.E. Plug-and-play cooperative navigation: From single-agent navigation fields to graph maintaining distributed MAS controllers. *IEEE Trans. Autom. Control* **2024**, *69*, 5262–5277. [\[CrossRef\]](#)
16. Bai, Z.; Pang, H.; He, Z.; Zhao, B.; Wang, T. Path planning of autonomous mobile robot in comprehensive unknown environment using deep reinforcement learning. *IEEE Internet Things J.* **2024**, *11*, 22153–22166. [\[CrossRef\]](#)
17. Ding, X.; Chen, H.; Wang, Y.; Wei, D.; Fu, K.; Liu, L.; Gao, B.; Liu, Q.; Huang, J. Efficient target assignment via binarized SHP path planning and plasticity-aware RL in urban adversarial scenarios. *Appl. Sci.* **2025**, *15*, 9630. [\[CrossRef\]](#)
18. Ejaz, M.M.; Tang, T.B.; Lu, C.-K. Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning. *IEEE Sens. J.* **2021**, *21*, 2230–2240. [\[CrossRef\]](#)
19. Zhou, C.M.; Huang, B.D.; Fränti, P. A review of motion planning algorithms for intelligent robots. *J. Intell. Manuf.* **2022**, *33*, 387–424. [\[CrossRef\]](#)
20. Zhang, L.; Zhang, Y.J.; Li, Y.F. Mobile robot path planning based on improved localized particle swarm optimization. *IEEE Sens. J.* **2021**, *21*, 6962–6972. [\[CrossRef\]](#)
21. Smart, W.D.; Kaelbling, L.P. Effective reinforcement learning for mobile robots. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; Volume 4, pp. 3404–3410.
22. Low, E.S.; Ong, P.; Cheah, K.C. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robot. Auton. Syst.* **2019**, *115*, 143–161. [\[CrossRef\]](#)
23. Cheng, Y.; Zhang, W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* **2018**, *272*, 63–73. [\[CrossRef\]](#)
24. Lou, P.; Xu, K.; Jiang, X.; Xiao, Z.; Yan, J. Path planning in an unknown environment based on deep reinforcement learning with prior knowledge. *J. Intell. Fuzzy Syst.* **2021**, *41*, 5773–5789. [\[CrossRef\]](#)
25. Yang, X.; Han, Q. Improved DQN for dynamic obstacle avoidance and ship path planning. *Algorithms* **2023**, *16*, 220. [\[CrossRef\]](#)
26. Nakamura, T.; Kobayashi, M.; Motoi, N. Path planning for mobile robot considering turnabouts on narrow road by deep Q-network. *IEEE Access* **2023**, *11*, 19111–19121. [\[CrossRef\]](#)
27. Gu, Y.; Zhu, Z.; Lv, J.; Shi, L.; Hou, Z.; Xu, S. DM-DQN: Dueling munchausen deep Q network for robot path planning. *Complex Intell. Syst.* **2023**, *9*, 4287–4300. [\[CrossRef\]](#)
28. Hu, H.T.; Yang, X.R.; Xiao, S.C.; Wang, F.Y. Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning. *Int. J. Prod. Res.* **2023**, *61*, 65–80. [\[CrossRef\]](#)
29. Li, Z.G.; Han, S.; Chen, Y.J.; Ning, X.G. Mobile robots path planning algorithm based on angle searching and deep Q-network. *Acta Armamentarii* **2025**, *46*, 1–14.
30. Wu, Q.C.; Lin, R.; Ren, Z.W. Distributed multirobot path planning based on MRDWA-MADDPG. *IEEE Sens. J.* **2023**, *23*, 25420–25432. [\[CrossRef\]](#)
31. Li, Z.Y.; Hu, X.T.; Zhang, Y.L.; Xu, J. Adaptive Q-learning path planning algorithm based on virtual target guidance. *Comput. Integr. Manuf. Syst.* **2024**, *30*, 553–568.
32. Hua, H.; Wang, Y.; Zhong, H.; Zhang, H.; Fang, Y. Deep reinforcement learning-based hierarchical motion planning strategy for multirotors. *IEEE Trans. Ind. Inform.* **2025**, *21*, 4324–4333. [\[CrossRef\]](#)
33. Zhao, H.; Guo, Y.; Li, X.; Liu, Y.; Jin, J. Hierarchical control framework for path planning of mobile robots in dynamic environments through global guidance and reinforcement learning. *IEEE Internet Things J.* **2025**, *12*, 309–333. [\[CrossRef\]](#)
34. Dawood, M.; Pan, S.; Dengler, N.; Zhou, S.; Schoellig, A.P.; Bennewitz, M. Safe multi-agent reinforcement learning for behavior-based cooperative navigation. *IEEE Robot. Autom. Lett.* **2025**, *10*, 6256–6263. [\[CrossRef\]](#)

35. Jonnarth, A.; Johansson, O.; Zhao, J.; Felsberg, M. Sim-to-real transfer of deep reinforcement learning agents for online coverage path planning. *IEEE Access* **2025**, *13*, 106883–106905. [[CrossRef](#)]
36. Jiao, T.; Hu, C.; Kong, L.; Zhao, X.; Wang, Z. An improved HM-SAC-CA algorithm for mobile robot path planning in unknown complex environments. *IEEE Access* **2025**, *13*, 21152–21163. [[CrossRef](#)]
37. Lin, X.; Yan, J.; Du, H.; Zhou, F. Path planning for full coverage of farmland operations in Hilly and Mountainous areas based on the dung beetle optimization algorithm. *Appl. Sci.* **2025**, *15*, 9157. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.