Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Continuous Dynamic Obstacle Environment Path Optimization Algorithm for continuous dynamic obstacle environment

Zhiyu Huang
Hunan University of Science and Technology
Xiaoming Wu
Xiangtan Central Hospital
Huaning Song
Hechi University
Xiaoliang Wang
fengwxl@hnust.edu.cn

Hunan University of Science and Technology

Research Article

Keywords: DDPG, Path optimization, reward reshaping method, continuous action space, Embodied Intelligence

Posted Date: February 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8848611/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Continuous Dynamic Obstacle Environment Path Optimization Algorithm for continuous dynamic obstacle environment

Zhiyu Huang[1], Xiaoming Wu[2], Huaning Song[3,*] and Xiaoliang Wang[1,*]

[1]Hunan University of Science and Technology, School of Computer Science and Technology, Xiangtan, 411100, China
[2]Xiangtan Central Hospital, Xiangtan, 411100, China
[3]Hechi University, School of Artificial Intelligence and Manufacturing, Hechi, 547000, China
[*]Corresponding Author: Huaning Song. Email: shn_2008@163.com and Xiaoliang Wang. Email: fengwxl@hnust.edu.cn

Abstract

This paper delves into the challenges of continuous path planning for robots in complex environments, addressing the limitations of traditional algorithms in handling sparse rewards and insufficient exploration.To tackle these issues, we propose an improved Deep Deterministic Policy Gradient (DDPG) algorithm named DDAPG, which integrates a potential energy-based reward reshaping method and an adaptive round Ornstein-Uhlenbeck (OU) noise model.The reward reshaping method enhances the algorithm's ability to provide smooth and continuous rewards, guiding the robot towards the target while avoiding obstacles effectively.Meanwhile, the adaptive OU noise model balances exploration and exploitation by adjusting the noise level according to the training progress, preventing the robot from getting stuck in local optima.Experimental results in a continuous simulation environment demonstrate that the DDAPG algorithm significantly improves convergence speed, reduces collision rates, and enhances adaptability compared to baseline algorithms.Specifically, DDAPG achieves a success rate of over 92% and a low collision rate of 7.625% in reaching target positions while navigating around obstacles.These findings underscore the potential of DDAPG in advancing embodied intelligence for robot path planning, offering a robust framework for deploying robots in real-world scenarios with dynamic and complex environments.

Keywords: DDPG; Path optimization; reward reshaping method; continuous action space; Embodied Intelligence

1 Introduction

The research on robot path planning originates from the demand for automation and intelligence, aiming to solve the core problem of autonomous and safe movement of robots in complex environments. Its development has gone through various stages, from early static environment planning methods such as visual mapping and grid methods, to random sampling algorithms suitable for high-dimensional spaces, and now to intelligent planning methods that integrate deep learning and reinforcement learning. The significance of this research is significant as it marks a crucial leap for robots from passive execution to active perception and decision-making, directly driving practical applications in many fields such as autonomous driving, unmanned aerial vehicle surveying, intelligent warehousing and logistics, and service robots. By optimizing the efficiency, safety, and robustness of paths, path planning technology is not only a technical prerequisite for robots to be deployed in real work, but also one of the core driving forces for improving social production efficiency and automation level. Considering that in continuous path planning problems, compared to discrete action spaces, the continuous action space is larger and the exploration problem faced by

robots is more complex. Robots need to face more choices in the path planning process, which increases randomness and avoids getting stuck in local optimal solutions. Therefore, this paper studies the implementation of further enhancing its exploration ability to better explore and learn continuous actions in the environment, considering the convergence speed and stability of the algorithm, and designing a reasonable reward function to encourage the robot to move towards the target position, punish the robot for colliding with obstacles, and enable the robot to have good path planning ability in different scenarios. At the same time, the research on robot path planning is moving from traditional environment abstract calculation to a new stage of deep integration with embodied intelligence. Embodied intelligence emphasizes the core role of the interaction between the "body" and the environment in intelligent behavior, which means that path planning is no longer an isolated optimal solution calculation, but a embodied decision-making process that relies on the specific form, physical characteristics, and multimodal perception of robots.

2 Related Work

The research on robot path planning originates from the demand for automation and intelligence, aiming to solve the core problem of autonomous and safe movement of robots in complex environments. In this section, we will provide an overview and analysis of these studies. In recent years, traditional path planning algorithms have been widely studied and applied, Khatib et al. proposed the Artificial Potential Field (APF) method, which treats robots as ppapers and introduces potential energy and repulsion functions to describe the motion state of robots and the forces of surrounding obstacles [8]. Yang et al. proposed an improved artificial potential field method, which solves the problem of local minima by modifying the direction and influence range of the gravitational field, adding virtual targets and evaluation functions, and solving the problem of unreachable targets by increasing gravity [24]. Wang et al. proposed an artificial driving vehicle that adapts to different environments based on the characteristics of the driver and the artificial potential field method. Compared with the traditional artificial potential field method, the improved local path planning algorithm obtains more reasonable path curves [19]. Li et al. proposed an OSM robot navigation method that combines road network information and local perception information to address the problem of global path inconsistency with the real environment caused by map errors in OpenStreetMap (OSM). OSM provides road network information and obtains the global path through Dijkstra's algorithm [11]. Lee et al. proposed a method called Finite Distribution Estimation of Dynamic Window (FDEDWA), which avoids dynamic obstacles by estimating the overall distribution of obstacles. The FDEDWA algorithm uses the Frobenius matrix factorization method to estimate the distribution of obstacles and predict their future positions, enabling real-time perception of dynamic environments [10]. Van et al. proposed the Double Deep Q Network (Double DQN) algorithm, which improves the original DQN by decoupling action selection and action evaluation. This method effectively reduces the overestimation bias of Q values, resulting in more stable and reliable learning performance, and has become one of the standard improvements in subsequent research [18]. Wang et al. designed a dueling network architecture (Dueling DQN) that uses a shared encoder followed by two streams that estimate the state value and action advantage respectively. This structure enables the agent to learn the value of the state more effectively without having to worry about the effect of each action, and performs well in scenarios with multiple optional actions [20]. Wang et al. proposed a method for mobile robot path planning in dynamic environments through global guided reinforcement learning, combining traditional global planners (such as A*) with local DQN decision-making. By incorporating global path information into the reward function, the agent is guided to explore efficiently, solving the problem of

convergence difficulties in complex environments [21]. Gök et al. proposed a method for mobile robot path planning in dynamic environments by integrating the Dueling network architecture, Double Q-learning, and priority experience replay. This method combines advanced components of deep reinforcement learning with an efficient experience sampling mechanism. By using priority replay to focus on key experiences and using the Dueling network to more accurately evaluate state value and action advantage, the problem of overestimation of value functions and low learning efficiency in dynamic environments is solved [3]. Gu et al. proposed a method for robot path planning by integrating the Dueling network architecture with the Munchausen reinforcement learning concept, combining a deep Q network with a policy regularization component. By incorporating the logarithmic probability of historical policies into the reward function as a self-motivation term and introducing entropy regularization to encourage exploration, they solved the problem of insufficient exploration and difficulty in improving policy performance in standard deep reinforcement learning algorithms in sparse reward environments [4]. Meng et al. proposed a method for mobile robot path planning using a deep reinforcement learning algorithm, combining the experience replay mechanism with an improved reward function design. By integrating environmental state information and goal-oriented signals into the reward function, the agent is guided to learn effectively, solving the problems of poor adaptability and slow convergence of traditional methods in unknown environments [16]. Li et al. proposed a method to realize path planning of mobile robots by improving the deep Q network, guiding the intelligent agent to learn efficiently [12]. Wu et al. proposed a method for mobile robot path planning in radioactive environments by improving the deep Q-network algorithm, combining deep reinforcement learning with a radiation dose constraint model. By integrating global radiation dose field information and the shortest path goal into the reward function, the agent is guided to explore efficiently while avoiding high-risk areas, solving the difficult problem of balancing path safety and efficiency in complex radioactive environments [22]. Zhang et al. proposed a method for mobile robot path planning based on the Double DQN (DDQN) algorithm improved by experience classification and multi-step learning, combining deep reinforcement learning with the experience priority evaluation mechanism. By classifying and storing key experiences of success and failure and sampling them differentially, and using multi-step TD goals to accelerate value propagation, the intelligent agent is guided to learn more efficiently from high-quality experience, solving the problems of low sample utilization efficiency and unstable algorithm convergence in the traditional experience replay mechanism [28]. Luo et al. proposed a method to achieve path finding for automated guided vehicles in a robotic mobile fulfillment system by guiding a deep Q network through an A algorithm, combining the traditional global planner A with the local DQN decision. By incorporating the global optimal path information generated by A into the reward function, the agent is guided to explore efficiently in a complex warehouse environment, solving the problems of algorithm convergence difficulties and low path planning efficiency in a multi-agent, highly dynamic fulfillment system environment [14]. Yuan et al. proposed a mobile robot path planning method that combines bat algorithm and DWA algorithm to achieve global optimization and dynamic obstacle avoidance goals [27]. Ma et al. proposed a CGAN-RRT algorithm consisting of a conditional generative adversarial network generation model and an improved RRT, which can generate probability distributions of effective feasible paths given map information [15]. Jiang et al. proposed an improved neural network based on DQN architecture. On a 30 × 30 map, DQN ensures that the optimal path between any two points can be successfully found within approximately 9000 rounds [6]. Xin et al. proposed a ladder reward mechanism to replace traditional simple reward functions and introduced it into experience replay based DQN, which significantly improved its adaptability to dynamic environments [23]. Huang et al. proposed a weighted reward function to balance obstacle avoidance and approach to the target,

using the weighted reward function to balance obstacle avoidance and approach to the target. Two reward thresholds were used to modify abnormal rewards to study path planning for mobile robots in unknown dynamic environments [5]. Yang et al. proposed an N-step priority dual DQN NPDDQN path planning algorithm and experience filtering mechanism, which screens the positive experiences mined and improves their reuse rate, effectively achieving obstacle avoidance in complex environments [25]. Kretzschmar et al. proposed a DDPG algorithm that utilizes deep neural networks based on the DPG algorithm. The DDPG algorithm combines Q-Learning and deterministic policy gradient methods, and approximates the value function and policy function using deep neural networks to achieve planning and control of continuous action spaces [9]. Tai et al. proposed an asynchronous DDPG algorithm to improve the training efficiency of deep reinforcement learning algorithms [17]. Li et al. proposed a method to replace the neural network optimization algorithm in DDPG algorithm with Radam algorithm, and added priority experience replay to improve the success rate and convergence speed of the algorithm [13]. Jiang et al. proposed an interactive deep deterministic policy gradient method based on action guidance to solve the problem of low sample efficiency in traditional deep reinforcement learning [7]. Zhou et al. designed a dual noise mechanism to explore the environment more effectively, embedding LSTM network based on DDPG algorithm to ensure that the model obtains information from previous navigation experience, thereby optimizing the model's decisions [30]. Yang et al. provided a comprehensive overview of the current progress in embodied intelligence, achieving autonomous learning capabilities for robots through deep reinforcement learning, enabling robots to continuously optimize path planning strategies in their interaction with the environment [26]. In summary, when conducting path planning in a continuous obstacle simulation environment, existing traditional DDPG algorithms face the challenge of sparse rewards, which makes it difficult for robots to quickly obtain effective real-time feedback in action continuous path planning, leading to slow convergence speed or inability to converge. At the same time, insufficient exploration efficiency can easily cause the algorithm to fall into local optima, and its adaptability and generalization ability in different environments are limited. Therefore, it is particularly important to design and optimize deep reinforcement learning algorithms for changing environments to improve the performance of robot path planning.

3 Background

Next, a brief introduction will be given to the basic theories and techniques used in this study, including reinforcement learning foundations, Markov decision processes, DDPG algorithms, and embodied intelligence concepts. These basic theories and techniques have important guiding significance for the model establishment and algorithm design of this research.

3.1 Fundamentals of Reinforcement Learning

Reinforcement learning is a technique that enables agents to learn optimal decision-making methods through continuous interaction with the environment. During this process, the intelligent agent continuously executes actions and receives corresponding reward feedback from the environment. Its core goal is to learn a strategy that maximizes long-term cumulative rewards by continuously adjusting behavior, gradually improving decision quality. Due to the dynamic changes in environmental conditions and reward mechanisms, intelligent agents must have the ability to continuously learn and adapt in order to achieve optimal long-term benefits.

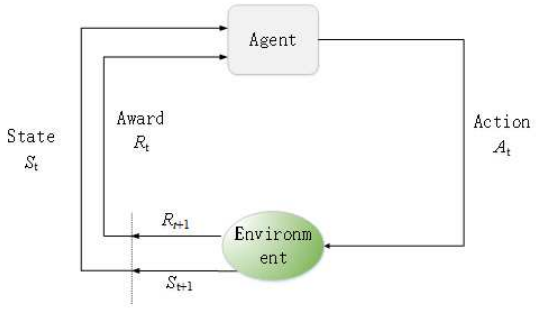


Figure 1: Basic framework of Reinforcement Learning

In the process of reinforcement learning, the agent performs an action in a certain state, and the environment returns a reward and transitions to a new state. By repeatedly iterating this interactive process, the agent gradually masters the optimal action choices to be made in each state, thereby achieving the maximization of long-term cumulative rewards. These five elements together form a Markov Decision Process (MDP). Intelligent agents continuously optimize their behavioral strategies by analyzing historical states, actions, and reward sequences, ultimately achieving optimal decision-making capabilities.

Reinforcement learning is applicable to many practical problems, such as robot control, autonomous driving, gameplay optimization, recommendation systems, etc. Compared to traditional machine learning algorithms, reinforcement learning faces more complex problems because it requires interaction with the environment to obtain feedback information, and the environment itself may also change. Therefore, reinforcement learning needs to adopt specific algorithms and techniques to solve these problems, such as value functions, policy search, Monte Carlo methods, Q-learning, etc.

3.2 Markov Decision Process

Markov decision process is a mathematical model used to handle multi-stage stochastic decision problems. In this framework, decision-makers can periodically or continuously observe the system state and select actions based on the current state at each moment, thereby interacting with the environment. The basic assumption of MDP is that the next state of the system depends only on the current state and the actions taken, and is independent of earlier historical states - a characteristic known as Markov property. The memoryless nature of state transitions allows the decision-making process to progress step by step, with decision-makers continuing to take action after each observation of a new state, gradually approaching a predetermined goal through continuous interaction.

Due to its ability to characterize sequential decision problems in stochastic environments, MDP is widely used in the field of optimization. For example, in robot path planning tasks, the robot needs to continuously select movement actions based on environmental information and interact with the environment to achieve navigation goals. This type of problem is essentially a multi-stage decision-making process, and MDP provides a theoretical basis for formal modeling and solution. In the MDP model, the intelligent agent (i.e. robot) models the path planning problem as an optimization problem of the state action value function by interacting with the external environment (such as maps, obstacles, etc.). Specifically, the robot selects actions based on the current environmental state, obtains feedback based on a reward function after execution, and iteratively learns to ultimately obtain the optimal path strategy.

The so-called "Markov property" refers to the state of a system at any given moment, and its future evolution depends only on the state at that moment, independent of all previous states. In other words, the subsequent development of a random process

is only determined by the current state and subsequent random factors, and historical information does not have a direct impact. Therefore, in Markov decision processes, the next state of the system is determined solely by the current state and the selected action, and is independent of the previous sequence of states.

3.3 DQN algorithm

DQN combines deep neural networks with the Q-learning algorithm. It guides an intelligent agent to make optimal decisions by learning a value function, achieving the first integration of reinforcement learning and deep learning. DQN uses techniques such as experience replay and fixed targets to improve training efficiency and enhance training stability. Fixed targets reduce the correlation of sample data. Specifically, when training a neural network, a separate target network is used to calculate updated targets, rather than directly using the current Q network. This separate target network is copied from the current Q network but remains unchanged for a period of time, known as the "fixed target." Another key feature of DQN is its experience replay mechanism. To better train the neural network, experiences are typically stored as experience tuples in an experience replay buffer. During neural network training, a small batch of experience tuples is randomly selected from the experience replay buffer for learning. This process is called experience replay. Experience replay reduces the correlation of experience data and improves training efficiency. Compared to Q-learning, DQN can handle high-dimensional state spaces, avoiding the curse of dimensionality that plagues traditional Q-learning algorithms. The use of an experience replay pool improves stability and training efficiency and avoids repeated training of related samples. Furthermore, deep reinforcement learning algorithms can perform end-to-end learning, avoiding the uncertainties of manual feature extraction. The DQN algorithm has been widely recognized and successfully applied in the gaming field. The DeepMind team applied the DQN algorithm to Atari games, achieving computer superhuman performance in certain games, becoming one of the most representative applications of DQN. Furthermore, the DQN algorithm is primarily targeted at discrete control reinforcement learning tasks. However, the value function-based DQN algorithm has limitations in its application to problems with continuous action spaces. Unlike value function-based algorithms, the policy gradient algorithm directly optimizes policies by building a policy network that outputs the probability of each action and then updates the policy network parameters by maximizing a cumulative reward function to obtain the optimal policy. Compared to indirect value function optimization methods, the policy gradient algorithm can better handle problems with high-dimensional discrete action spaces and offers improved stability and convergence.

3.4 DDPG Algorithm

The Deep Deterministic Policy Gradient (DDPG) algorithm is a deep reinforcement learning method developed on the basis of the Deterministic Policy Gradient (DPG) algorithm. Compared with DPG, DDPG introduces deep neural networks in the Actor network to approximate the representation strategy and integrates multiple techniques to improve training effectiveness. Specifically, DDPG draws on the experience replay mechanism in DQN to reduce the correlation between samples by storing and randomly sampling historical experience, thereby improving data utilization efficiency and training stability; At the same time, the algorithm also introduces a target network to alleviate the training instability problem caused by target value fluctuations during Q-value target calculation.

In terms of action output, DDPG's Actor network directly outputs a deterministic action value rather than the probability distribution of the action, which makes it particularly suitable for control problems in continuous action spaces. DDPG uses four neural networks, including Actor, Critic, and their corresponding target networks, and

gradually synchronizes the parameters of the target network through soft updates, further enhancing the overall stability of the algorithm.

In deep reinforcement learning tasks, Actor networks tend to converge to local optima, continuously outputting the same action, limiting exploration of other potentially better actions. To address this issue, DDPG introduces a noise mechanism (such as Gaussian noise) during the action selection stage, which makes the actions output by the Actor in the same state have a certain degree of randomness, thereby encouraging the agent to explore a wider action space and avoiding premature falling into local optima. Noise is usually added to the output actions of the Actor network at each time step.

3.5 Embodied Intelligence Concept

In the field of robotics, Embodied Intelligence emphasizes that agents acquire and enhance their intelligence level through interaction with the physical environment. The core of embodied intelligence lies in the ability of robots to learn and adapt through their own perception, motion, and environmental interaction, rather than relying solely on pre-set rules or models. Embodied intelligence emphasizes that robots perceive the environment through sensors and interact with it through actuators, continuously adjusting and optimizing their behavior strategies. Robots can learn autonomously through interaction with the environment and adjust their strategies based on experience to better adapt to complex and changing environments. Embodied intelligence requires robots to quickly adjust their behavior based on real-time feedback information to cope with uncertainty and dynamic changes in the environment.

4 Structure

Considering that in continuous path planning problems, compared to discrete action spaces, the continuous action space is larger and the exploration problem faced by robots is more complex. Robots need to face more choices in the path planning process, which increases randomness and avoids getting stuck in local optimal solutions. Therefore, it is necessary to further enhance its exploration ability to better explore and learn continuous actions in the environment, considering the convergence speed and stability of the algorithm, and designing a reasonable reward function to encourage the robot to move towards the target position, punish the robot for colliding with obstacles, and enable the robot to have good path planning ability in different scenarios.

For this purpose, the DDAPG algorithm is designed in this paper. Firstly, the robot is designed as a Markov decision process in continuous path planning tasks. A simulation robot is used to collect environmental information around it through laser ranging sensors, and a deep neural network is built. Considering the issue of sparse rewards, the reward function should provide smooth and continuous rewards in a continuous action space. A potential energy based reward reshaping method was designed. In order to improve exploration intensity, an adaptive round OU noise model was designed. Finally, a simulation environment was built based on Gazebo to compare the algorithm performance. In this chapter, the DDAPG algorithm was proposed to accelerate convergence speed and reduce collision rate compared to the baseline algorithm, and its adaptive ability was verified in different scenarios.

4.1 Design of Continuous Path Planning Process

The simulation environment built by Gazebo is more in line with the operation of robots in real scenes, and can use higher precision sensors to obtain environmental information, such as using sensors such as LiDAR, thereby improving the accuracy and flexibility of path planning. Turtlebot3 is one of the default robot models in Gazebo,

which is easy to set up and integrate with other models and plugins. The advantage of using Turtlebot3 to simulate robots is that it provides a realistic simulation of physical robots, including dynamics, sensors, and actuators, which can be used to test and validate algorithms in controlled and safe environments. Easy to customize: Turtlebot3 is highly customizable, allowing researchers to modify and add different sensors, actuators, and control mechanisms to test and evaluate various path planning algorithms. ROS Integration: Turtlebot3 is essentially integrated with ROS and can be integrated with other ROS software packages and nodes. This makes it easier to develop and test path planning algorithms using Gazebo simulation software. Turtlebot3 has two main models in the simulation environment: Hamburg style and waffle style.

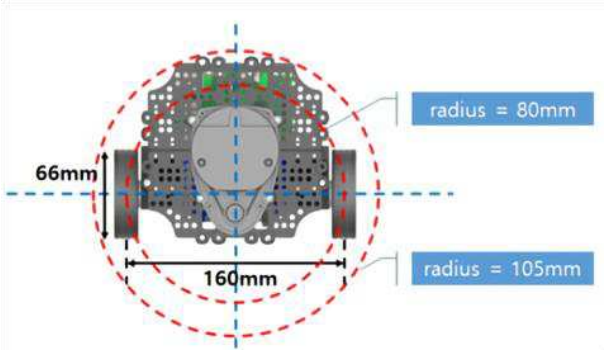


Figure 2: Schematic diagram of bottom plate of Turtlebot3 Burger

In the subsequent simulation experiments, the Turtlebot3 Hamburg style simulation robot was selected as the experimental object, and its motion mode is based on differential drive wheel movement. It uses two wheels to control the movement and rotation of the robot, where the speed of each wheel can be achieved by controlling the speed of the motor. Its kinematic model can be described using basic kinematic equations:

$$v = \left(v_r + v_l\right) / 2 \tag{1}$$

$$w = \left(v_r - v_l\right) / L \tag{2}$$

Among them, v represents the linear velocity of the robot, w represents the angular velocity of the robot, v_r and v_l represents the velocity of the right and left wheels, L representing the distance between the two wheels.These equations describe the motion of robots at different wheel speeds and can be used to control the path planning of robots in continuous simulation environments. It should be noted that the Turtlebot3 burger style robot also has incompleteness, which means it cannot achieve independent control in some directions of motion, such as being unable to move directly in a direction perpendicular to the two wheels. Therefore, when conducting path planning and control, it is necessary to consider the non integrity characteristics of the robot to avoid problems during its movement.The linear motion of the Turtlebot3 hamburger style robot is controlled by two differential drive wheels, which control the forward and backward movement of the robot.The curved motion of the Turtlebot3 hamburger style robot can be achieved by controlling the angular velocity of the robot. When the speed of the left and right driving wheels of the robot is not equal, the robot will rotate. The motion of a robot on a curve can be expressed as:

$$x(t) = x_0 + \frac{v}{\omega}\left(\sin\left(\omega t + \theta_0\right) - \sin\theta_0\right) \tag{3}$$

$$y(t) = y_0 + \frac{v}{\omega}\left(\cos\theta_0 - \cos\left(\omega t + \theta_0\right)\right) \tag{4}$$

Then, Gazebo in ROS was used to build a simulation environment for robot motion, and performance testing was conducted using Turtlebot3 simulation robot. The goal of robot path planning is to avoid obstacles from the starting point and reach the endpoint. In Gazebo, simulation environments can be designed by adding objects, terrain, light sources, etc., and real-world scenes can be simulated by adding obstacles, target points, sensors, etc.

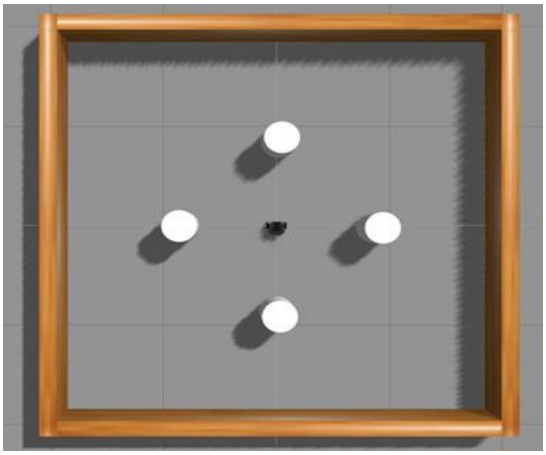


Figure 3: Gazebo simulation scene

Laser ranging sensors were selected to obtain environmental information around the Turtlebot3 simulation robot. Compared to image sensors, laser ranging sensors can accurately measure in various complex environments and obtain information about the surrounding environment through rotational scanning, with high flexibility. Robots can measure the distance and direction between surrounding obstacles and the robot through laser ranging sensors, providing accurate environmental information. Compared to other sensors, laser ranging sensors are not affected by external factors such as light, climate, dust, etc., and can provide stable performance in complex environments.

The robot selects actions to interact with the environment based on its current state and determines the current state. If the maximum number of steps per turn is exceeded, the training environment and the position of the robot are reset. If the robot encounters an obstacle, it will receive a negative reward and continue to learn in the environment. If the robot reaches the target point, it receives a positive reward and then determines whether the deep reinforcement learning algorithm has converged. If it has converged, the program ends. Otherwise, the training environment and the robot's position are reset, and the robot continues to interact with it in the environment until the algorithm converges.

4.2 DDAPG Algorithm Design

This paper studies the problem of continuous path planning, where the robot's actions are continuous. Firstly, the problem is modeled as a Markov decision process, and a reward reshaping method based on potential energy and an adaptive turn OU noise mechanism are designed in combination with the DDPG algorithm, which is named the DDAPG algorithm. The deep neural network is also set up.

4.2.1 Markov Decision Process

State space S: The experiment is based on ROS and uses Gazebo to build a continuous simulation environment. The robot needs to complete the path planning task from the starting point to the target point without colliding with obstacles. Therefore, the state space of the robot is related to the position information of the robot to the target point and obstacles around the robot. In this chapter, the state space of the robot

$$S=\begin{cases}(x_r,y_r)\\ \theta\\ (x_o-x_r),(y_o-y_r)\\ (x_a-x_r),(y_a-y_r)\end{cases} \tag{5}$$

is defined as follows:

Where (x_r,y_r) represents the robot's current position, θ represents the angle and direction of the target point recorded by the robot, $(x_o-x_r),(y_o-y_r)$ represents the distance from the robot's current position to the nearest obstacle, $(x_a-x_r),(y_a-y_r)$ represents the distance from the robot's current position to the target point, d_a represents the distance from the robot's current position to the target point, and d_o represents the distance from the robot's current position to the nearest obstacle. The corresponding position information is calculated using the Euclidean distance formula.

$$d=\sqrt{(x-x_1)^2+(y-y_1)^2} \tag{6}$$

Action Space A: It refers to the actions performed by a robot in a time step, and its output actions are continuous. By outputting continuous action values, the robot can try various combinations of actions to find the optimal path.

Reward Function R: The reward function is used to measure the quality of a robot's execution of a certain action. The robot observes the state of the environment at each time step and decides the next action to take based on the current state and behavior. It also receives a reward value indicating the quality of the current action. At each time step, the robot's reward depends on the actions of state $S(\mathrm{t})$ and execution $A(t)$.

$$R=\begin{cases}R_a=200\\ R_o=-150\\ R_p=1\end{cases} \tag{7}$$

4.2.2 Potential-based Reward Reshaping

Deep reinforcement learning, in the robotic path planning problem, guides the robot to plan a path from its initial position to its target point while avoiding obstacles by maximizing a reward function. Traditional reward functions typically only reward the robot when it reaches the target point. As the complexity of the environment and the state space increase, the robot has a low probability of finding a collision-free path during training. This prevents the robot from gaining sufficient experience from useful samples, leading to a sparse reward problem and slow algorithm convergence.A good reward function should guide the robot to explore. The robot needs to explore different routes and actions in the environment to discover more optimal paths. To encourage exploration, the reward function should provide positive rewards for the agent to choose unexplored actions and states. The reward function should provide a smooth and continuous reward signal in the continuous action space. The reward function should be linked to the specific requirements of the task. This means that the reward function should be designed based on the nature of the task to maximize the agent's progress towards the goal of completing the task.

To address this issue, we devised a potential energy-based reward reshaping method. For

robot path planning, the primary tasks are reaching the goal and avoiding obstacles. To this end, we can design reward functions from these two perspectives. Within the traversable area, we leverage the potential energy concept to reshape the reward function, dividing it into rewards r_θ and r_a for reaching the goal and reward r_o for avoiding obstacles.In a navigable area, the robot is expected to move toward the target as quickly and efficiently as possible, receiving positive rewards as it approaches the target. The robot is motivated by rewards and is more inclined to move toward the target. The rewards for moving toward the target primarily include the direction toward the target and the distance to the target.

$$r_\theta = d / d_a \tag{8}$$

For the distance reward r_θ towards the target, d_a represents the distance from the robot's current position to the target point, and d represents the distance from the robot's starting point to the target point. The design idea is that the closer to the target point, the greater the reward value.

For the orientation reward r_a towards the target, r_h represents the angle between the orientation of the current robot position and the line connecting the target point and the y-axis of the simulation environment. Its value range is $[0, \pi]$. The smaller the angle, the more correct the direction of the robot's movement. α is the reward factor for approaching the target.

$$r_a = \pi - r_h \tag{9}$$

$$r_o = d_o * \beta \tag{10}$$

To ensure the robot avoids collisions and successfully dodges obstacles, a reward function for obstacle avoidance is designed. When the robot approaches an obstacle, it receives a negative reward. This encourages the robot to move away from obstacles, thereby reducing the likelihood of collisions. This also accounts for the interaction between the robot and obstacles in the reward function. The reward for obstacle avoidance is r_o, where d_o represents the distance from the robot's current position to the nearest obstacle, and β is the reward factor for avoiding the obstacle.

The specific reward function is designed as follows:

$$R = \begin{cases} R_a = 200 \\ R_o = -150 \\ R_p = r_\theta * r_a * \alpha + r_o \end{cases} \tag{11}$$

Among them, R_a represents reaching the target point, R_o represents collision with obstacles, and R_p represents movement in the passable area.

4.2.3 Adaptive Round OU Noise

Striking the right balance between exploration and exploitation in reinforcement learning, particularly when faced with robot path planning in continuous action spaces, presents a complex challenge. Unlike discrete action spaces, robots in continuous action spaces have a vast array of actions to choose from, which complicates exploration. If a robot overly relies on known strategies, it may become trapped in local optima and unable to discover a better strategy. On the other hand, if the robot explores the action space too randomly, it may waste significant training time and resources, making it difficult to find a collision-free path. Therefore, a balance between exploration and exploitation is crucial to achieving a global optimal solution.This paper designs an adaptive round-based OU noise model. This model takes into account the different exploration levels required by the robot in different training rounds. In the early stages

of training, the robot needs to randomly explore the environment. As training progresses, the robot can make more use of its existing experience to make decisions.

Gaussian noise is a stationary, independent and identically distributed random process whose probability distribution follows a Gaussian distribution. It has the stationary and independent and identically distributed properties. This means that the Gaussian noise at one point in time is independent of each other, and the mean and variance of the noise remain constant over time.

However, in continuous robot path planning, it's necessary to consider noise simulation for long time series. However, Gaussian noise is independent and identically distributed, meaning the noise added to each action selection is independent and varies significantly. This results in frequent oscillations and regressions during exploration, hindering the robot's ability to reach its goal.OU noise, unlike Gaussian noise, is an autoregressive noise with historical information. It exhibits stable autocorrelation and a tendency to regress toward the mean. This allows it to maintain a certain degree of continuity and smoothness during state space exploration. As the action approaches the target value, the noise gradually shifts in that direction, resulting in a more focused exploration. Compared to Gaussian noise, it is more suitable for robot simulation path planning problems. The probability density function and noise model are expressed as follows:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{12}$$

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t \tag{13}$$

Where x_t represents the current state, μ represents the mean to be returned, θ represents the speed of mean return, σ represents the intensity of the noise, and dW_t represents the standard Brownian motion, which can be obtained by integral calculation. The integral solution is as follows:

$$x_t = \mu + (x_0 - \mu)e^{-\theta t} + \sigma \int_0^t e^{-\theta(t-s)} W_s \tag{14}$$

Since the OU process is a mean reversion process, its expectation gradually returns to the mean μ over time, and the variance will tend to zero at an infinite time scale. Its integral and variance are as follows:

$$\mathrm{E}[x_t] = x_0 e^{-\theta t} + \mu(1 - e^{-\theta t}) \tag{15}$$

$$Var[x_t] = \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t}) \tag{16}$$

Adding OU noise to the output of the policy network yields the following:

$$A(t) = \mu(S(t)) + \dot{o}_o(t) \tag{17}$$

Taking into account the need for different degrees of exploration of the robot in different training rounds, the robot needs to randomly explore the environment in the early stage of training. As the training progresses, the robot needs to make more use of existing experience to make decisions. For this reason, this chapter designs an adaptive round OU noise model and designs the noise impact factor λ, so that the robot's exploration degree in the environment continues to decrease with the increase in the number of training rounds. By adjusting the noise impact factor λ, the exploration degree of the OU noise is changed. The robot can have different exploration degrees for different numbers of training rounds, so as to balance the relationship between

exploration and utilization throughout the training process. In the initial training stage, the noise impact factor is large, which means that the amplitude of the OU noise is large, and the robot's actions are more random, which helps the robot explore the environment. As the training progresses, the noise impact factor gradually decreases, resulting in a decrease in the amplitude of the OU noise, and the robot's actions tend to be more smooth and stable.

$$\lambda = \frac{W - W_e}{W} \tag{18}$$

Where W_e is the number of rounds of the current robot path planning experiment, and W is the total number of rounds of the set robot path planning experiment. The following results are further obtained:

$$A(t) = \mu(S(t)) + \lambda \partial_o(t) = \mu(S(t)) + \mathrm{N}_{eps} \tag{19}$$

4.2.4 Deep Neural Network Design

For the Actor network in the DDAPG algorithm, the neural network is designed to consist of an input layer, three fully connected hidden layers, and an output layer. In this neural network, the input layer receives the current state of the robot as input, and the hidden layer uses the ReLU activation function. The middle layer of the neural network consists of three fully connected layers, containing 400, 300, and 300 neurons, respectively. The output layer is set up with different activation functions as needed to predict policies or value functions, etc. In the last layer of the policy neural network, the Tanh function is used as the activation function to output the action A selected by the neural network, making the output action continuous and the robot's movement smoother. The main difference between the policy target network and the policy estimation network is their different weights. The weights of the policy target network are fixed and periodically copied from the policy estimation network. The purpose of this is to prevent instability in the policy estimation network during training. The loss function of the Actor network is defined as follows:

$$J(\omega) = -\frac{1}{D_b}(y_i - Q_\theta(S_i, A_i))^2 \tag{20}$$

The Critic network in the DDPG algorithm is used to evaluate the value of the current state. This network consists of a Q target network and a Q estimation network, both of which have the same structure. In this neural network, the input layer receives the robot's state S and action A as input. The middle layer of this neural network consists of three fully connected layers, containing 400, 300, and 300 neurons respectively. The activation functions of these layers are all ReLU functions, and the output layer outputs the Q value obtained from the current state S and action A. The loss function of the Critic network is defined as follows:

$$L(\theta) = \frac{1}{D_b}(y_i - Q_\theta(S_i, A_i))^2 \tag{21}$$

To train the two networks in DDAPG, the algorithm uses target networks: a target policy network and a target Q-value network. The parameters of these target networks are updated using soft updates, rather than the conventional hard updates, where the target network parameters are periodically copied from the estimation network. Soft updates use a smoothing approach to update the target network parameters. With each update, only a small subset of weights are updated, rather than all of them.

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{22}$$

$$\omega' \leftarrow \tau\omega + (1-\tau)\omega' \tag{23}$$

Where τ is the soft update rate, which causes the target network's parameters to slowly approach those of the estimated network over time. Soft updates improve the algorithm's stability. Because the target network's parameters change relatively little, training avoids drastic fluctuations. Smooth updates of the target network also allow for better estimation of the cumulative reward.

4.2.5 DDAPG Algorithm Flow

At the beginning of the DDAPG algorithm, the relevant initialization work is carried out by clearing the experience replay area and randomly initializing Q network θ and policy network ω. The target Q network θ' is set to be the same as θ, and the parameter ω' of the target policy network is set to be the same as ω.Using D_b to represent the size of the sampling small batch, the robot's learning process is divided into W rounds, each round consisting of T steps. After algorithm initialization, the loop begins. At the beginning of each round, for all rounds, the initialization state is $S(1)$ and the OU noise N of the action strategy is calculated to obtain λ, and all obstacles, target points, and other information are initialized. Then the algorithm will perform a loop iteration process, using an adaptive round OU noise model at each step t of the robot to solve the exploration and utilization problems of the robot, and avoid getting stuck in local optimal solutions.

When it chooses to execute $A(t)$, the robot receives a reward value and enters $S(t+1)$, placing the experience value in the experience replay pool D. After W_s rounds of training, the learning process is carried out. For the experience replay pool D, a random sampling method is used to train the Q network and policy network using small batches of D_b. The Q network is updated using stochastic gradient descent, the strategy network is updated using stochastic gradient ascent (SGA), and the Adam optimizer is used. Compared to the DQN algorithm, the DDAPG algorithm adopts the dual network mechanism of Q network and policy network, and therefore uses a soft update method to update the target network smoothly to update the weights of the neural network. Finally, the trained network is used for decision-making. By taking the current state as input and outputting the action in that state, the path in the current 3D environment map can be obtained.

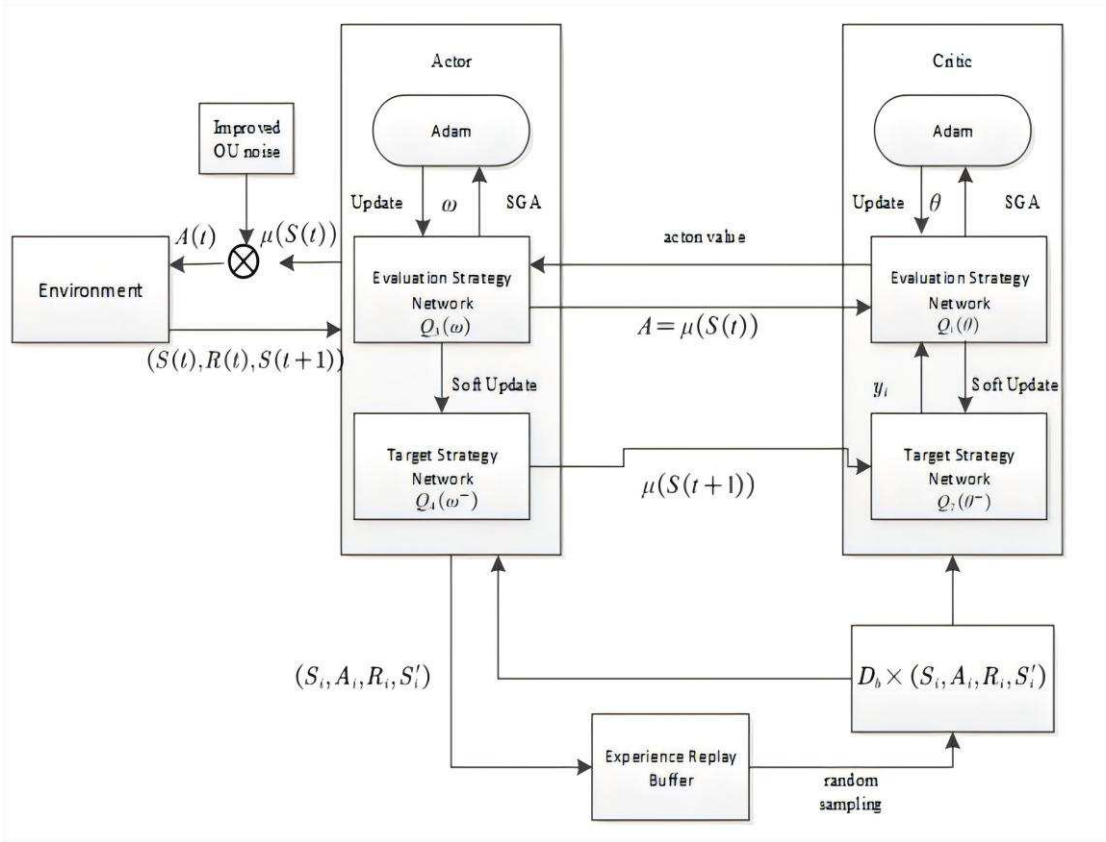


Figure 4: Schematic diagram of DDAPG algorithm

5 Experiment

5.1 Simulation Scenarios And Evaluation Indicators

5.1.1 Simulation Scenarios

Table 1: Main tools and version models of experiment

Experimental Tools	Version Model
Operating System	Ubuntu16.04
CPU	12th Gen lntel(R) Core(TM) i5-12600KF
GPU	NVIDIA GeForce RTX 3060 Ti
Gazebo	9.18.0
Python	2.7
Torch	0.4.0
Robot	Turtlebot 3 Hamburger

This experiment aims to verify the effectiveness of the DDAPG algorithm in a continuous simulation environment. Compared with a two-dimensional grid simulation environment, the simulation environment built by Gazebo is more consistent with the robot's movement in a real environment. However, deep reinforcement learning faces the problem of sparse rewards in the problem of robot path planning, which leads to low exploration efficiency of the deep reinforcement learning algorithm and very slow convergence speed, or even no convergence. To overcome this problem, this chapter designs a potential energy-based reward reshaping method. In order to prevent the robot from falling into a local optimal solution, an adaptive round-based OU noise model is introduced in the action selection stage to enhance the robot's exploration degree.

Table 1 lists the main tools and versions of the continuous simulation environment experiment.

Table 2: The experiment mainly involves parameters

Parameter Representation	Setting Values
Maximum angular velocity	1.5 rad/s
Linear speed	0.15 m/s
Dimension of state space	28
Dimensions of the action space	1
Experience pool size	1000000
Actor learning rate	0.0001
Critic learning rate	0.001
Mini-batch size	64
Number of rounds	800
Soft update speed	0.005
Maximum number of steps per round	500

5.1.2 Comparison Algorithms

In order to evaluate the effectiveness of the proposed DDAPG algorithm, other DRL algorithms were compared. For the designed DDAPG algorithm, all neural networks have one input layer, three hidden layers, and one output layer. All hidden layers are fully connected layers, and the reward function is designed based on the potential energy reward reshaping method. In the exploration phase, the algorithm adopts the designed adaptive round OU noise strategy. The baseline algorithms for comparison mainly include: 1) DDPG algorithm combined with Gaussian noise. 2) The dDDPG algorithm designed an adaptive exploration method based on incorporating OU noise [2]. 3) The D-DDPG algorithm integrates the dueling network into the critic network and divides the Q-value into two parts for calculation, adding random noise [29]. 4) The LSTM-DDPG algorithm [1] integrates LSTM into the DDPG algorithm framework. For all DRL algorithms, the input is 24 dimensional perception information of laser ranging, the current position of the robot, the angle and direction of the target point recorded by the robot, the information from the current position to the nearest obstacle position, and a total of 28 dimensions of information from the current position to the target point position. The output is angular velocity, with a range value of $w \in [-1.5rad/s, 1.5rad/s]$.

5.1.3 Evaluation Indicators

In order to evaluate the performance and convergence of the robot path planning algorithm in this study, the following indicators were defined for comparative analysis of the algorithms.

Total reward: The total reward value of the robot in each round.

Collision rate: The proportion of robot encounters with obstacles in the total number of turns.

Success rate: The proportion of turns in which the robot does not collide during the process from the starting point to the target point in the total number of turns.

Maximum Reward: The highest total reward value for the number of turns in which the robot does not collide during the process from the starting point to the target point.

5.2 Comparison And Analysis of the Impact of Discount Factors on Performance

Taking into account the influence of discount factor γ on the performance of DRL

algorithm, the DDAPG algorithm experiment under different discount factors γ is designed to consider its performance. First, by setting the discount factor γ in the algorithm to 0.80, 0.90 and 0.99 respectively, the experiment is compared in the simulation environment of Figure 5. When the Turtlebot3 robot hamburger is performing the path planning task, it moves according to the DDAPG algorithm. The wooden boards and walls in the environment are obstacles, the red square is the target position of the robot, and the red curve is a collision-free path planned by the robot from the starting point to the target point.

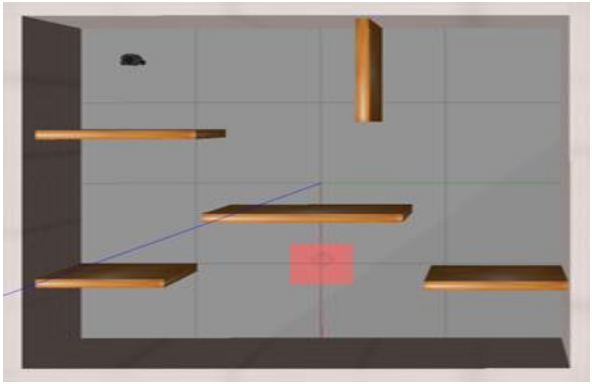


Figure 5: Continuous path planning simulation environment

Table 3: Performance comparison of DDAPG algorithm

	Collision rate	Success rate	Maximum Reward
0.80	36.5%	63.5%	259
0.90	21.5%	78.5%	260
0.99	7.25%	92.75%	261

The performance of the DDAPG algorithm under different discount factors was tested. When $\gamma = 0.99$ was used, the maximum reward value, collision rate and success rate were better than others. Therefore, in the subsequent experiments, discount factor $\gamma = 0.99$ was selected for the experiment.

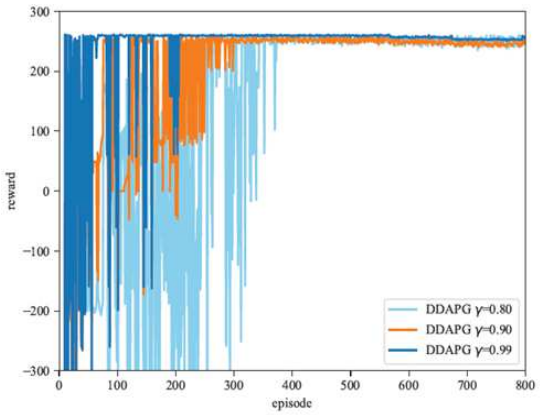
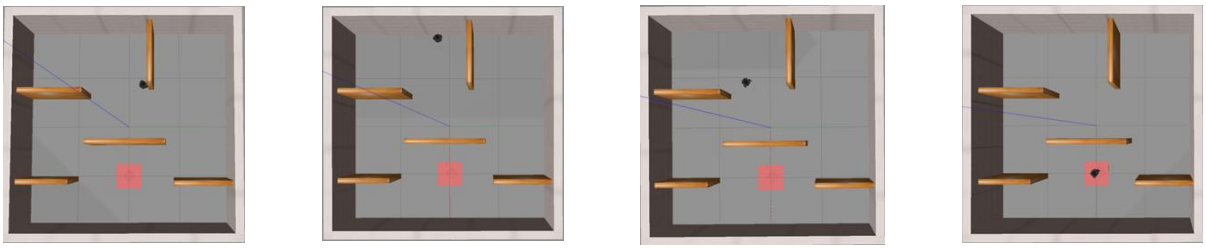


Figure 6: Total reward value of DDAPG with different discount factors

5.3 Convergence Comparison And Analysis

For continuous path planning problems, the designed DDAPG algorithm was tested for convergence, collision rate, and other performance characteristics, and compared with baseline algorithms: DDPG, dDDPG, D-DDPG, and LSTM-DDPG.Training a deep reinforcement learning algorithm is a long and time-consuming process. To provide a more intuitive understanding of the DDAPG training process, we visualized a portion of

it. In the early stages of DDAPG training, the simulated robot has very limited knowledge of the environment. Due to its lack of experience, the robot is often unfamiliar with obstacles and boundaries, leading to frequent collisions. These collisions result in the DDAPG algorithm imposing large "penalties" on the robot to help it quickly learn to avoid obstacles.



(**a**) (**b**) (**c**) (**d**)

Figure 7: Scenes during the robot's operation: (**a**) collision with obstacles; (**b**) collision with boundaries; (**c**) avoiding obstacles; (**d**) reaching the target point

During each training round, when the robot's movement exceeds the preset maximum step count threshold of 500 steps, or when the robot successfully reaches the target point, the simulation environment is reset and a new round of training begins. As training progresses, the robot's performance gradually improves, becoming more familiar with the environment and becoming better able to avoid obstacles and approach the target. With increasing training, the robot acquires the ability to avoid obstacles and reach the target point.

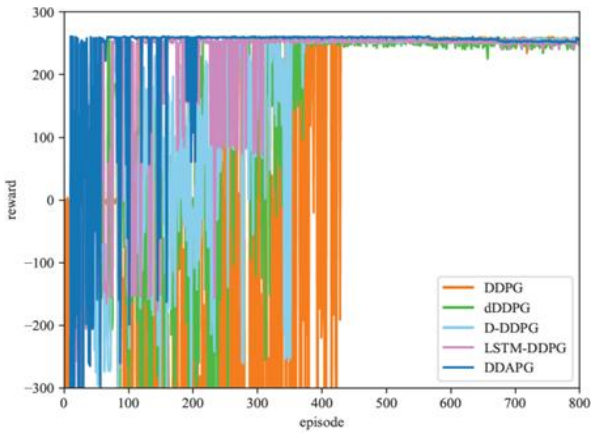


Figure 8: The total reward of the algorithms in continuous environment

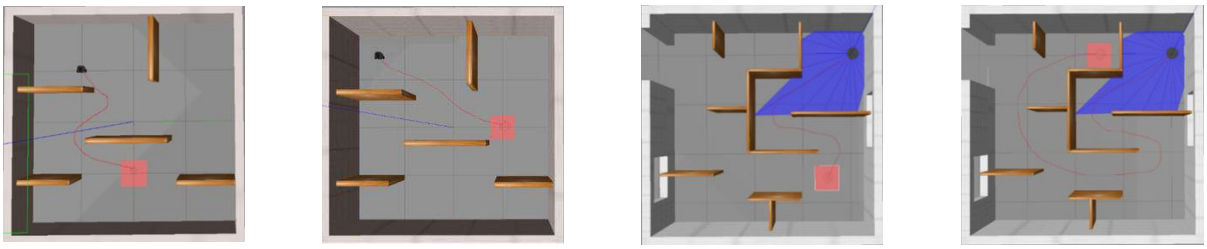
The DDAPG algorithm uses adaptive OU noise and a designed noise influence factor with autocorrelation and a tendency to regress toward the mean. This increases the exploratory nature of the robot's actions, maintaining a certain degree of continuity and smoothness. This results in varying degrees of exploration in different training rounds. In the early stages of training, the noise influence factor is large, causing the robot to explore the environment more randomly. However, as training progresses, the noise influence factor gradually decreases, allowing the robot to rely more on prior experience. This helps improve the algorithm's convergence speed and path planning success rate. The DDAPG algorithm can better adapt to the training needs of different stages and maintain a balanced relationship between exploration and exploitation throughout the training process. Furthermore, a reward reshaping method based on task characteristics addresses the problem of sparse rewards, making it easier for the robot to understand the nature of the task, guiding it toward the goal and avoiding obstacles,

helping to improve the algorithm's convergence speed and reduce the collision rate.

Table 4: The performance algorithms in continuous environment

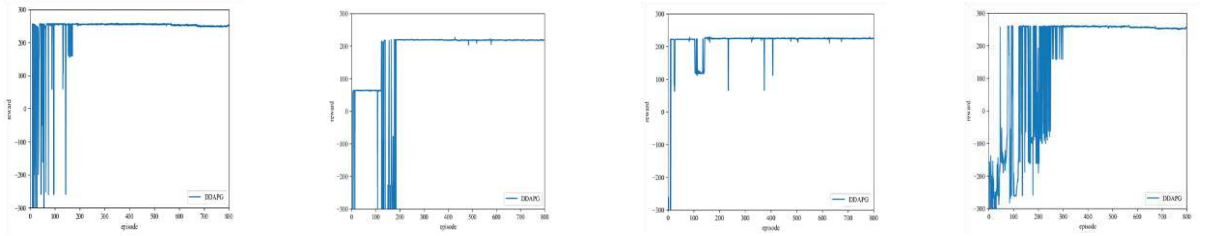
	Collision rate	Success rate	Maximum Reward
DDPG	44.75%	55.25%	261
D-DDPG	38.25%	61.75%	259
dDDPG	43.5%	56.5%	259
LSTM-DDPG	18.25%	81.75%	259
DDAPG	7.625%	92.375%	261

In a continuous simulation environment, environmental factors may change, such as the number, size, shape, and distribution of obstacles. Testing the adaptive capabilities of the DDAPG algorithm helps verify whether the algorithm can work effectively in different scenarios. Four scenarios were set up for experiments to change the control variables of different environmental information: (1) Changing the position of the starting point. (2) Changing the position of the target point. (3) In an environment where the number of obstacles, the target point, and the starting point all change. (4) Changing the position of the target point based on (3). The detailed design and experiment are as follows.



(**a**) (**b**) (**c**) (**d**)

Figure 9: Adaptive ability test scenarios, (**a**) Scenario 1; (**b**) Scenario 2; (**c**) Scenario 3; (**d**) Scenario 4



(**a**) (**b**) (**c**) (**d**)

Figure 10: Adaptive ability test of DDAPG total reward, (**a**) Scenario 1; (**b**) Scenario 2; (**c**) Scenario 3; (**d**) Scenario 4

The performance of the DDAPG algorithm in different scenarios. In scenarios 1, 2, and 3, the DDAPG algorithm achieved success rates of 78.75%, 92.125%, and 78.375%, respectively. The maximum rewards achieved by the DDAPG algorithm in these scenarios were 227.10, 229.69, and 264.09, respectively. This indicates that the DDAPG algorithm has a high success rate in these different simulation scenarios and can plan a collision-free path, demonstrating its robustness under diverse environmental conditions.

Table 5: The performance algorithms in continuous environment

	Collision rate	Success rate	Maximum Reward

Scenario 1	0.6625%	93.375%	259
Scenario 2	21.25%	78.75%	227
Scenario 3	0.7875%	92.125%	229
Scenario 4	21.625%	78.375%	264

In summary, the DDAPG algorithm demonstrates excellent adaptability and robustness for path planning in continuous simulation environments. It can adapt to different environmental scenarios, plan collision-free paths, and demonstrates good performance in continuous action spaces.

6 Conclusion

The continuous path planning algorithm based on improved DDPG (DDAPG) proposed in this paper is highly compatible with the concept of embodied intelligence. Through the design of a reward reshaping method based on potential energy and an adaptive noise model, the DDAPG algorithm has achieved efficient path planning for robots in continuous motion spaces, demonstrating strong self-learning, adaptability, and robustness. The experimental results have verified the advantages of DDAPG algorithm in convergence speed, collision rate, and adaptability, providing strong support for the application of embodied intelligence in the field of robot path planning. Future research can further explore the application of DDAPG algorithm in more complex environments and combine it with other embodied intelligence technologies (such as transfer learning, multimodal perception, etc.) to enhance the overall intelligence level of robots.

Author Contributions: The contribution of each author is as follows: Study conception and design: Zhiyu Huang, Xiaoliang Wang; Algorithm implementation and framework design: Zhiyu Huang, Huaning Song; Technical assistance and results analysis: Zhiyu Huang, Xiaoliang Wang; Draft manuscript preparation: Zhiyu Huang, Xiaoming Wu; Review and editing: Xiaoming Wu. All authors reviewed the results and approved the final version of the manuscript.

Funding: This paper is supported by the Scientific Research Fund of National Natural Science Foun 35 dation of China (Grant 62372168), the Hunan Provincial Natural Science Foundation 36 of China (Grants 2023JJ30266), Key Laboratory of AI and Information Processing, Education Department of Guangxi Zhuang Autonomous Region (Hechi University)(Grants 2024GXZDSY010).

Data Availability: The data and materials used in this article are derived from publicly accessible databases and previously published studies cited throughout the text.

Declarations

Competing interests: The authors declare no competing interests.

References

1. Chao W, Han D, Jie X. Multi-rotor UAV autonomous trac-king and obstacle avoidance based on improved DDPG[C]. 2021 2nd International Conference on Artificial Intellige-nce and Computer Engineering (ICAICE). IEEE, 2021: 261-267.
2. Dong Y, Zou X. Mobile robot path planning based on imp-roved DDPG reinforcement learning algorithm[C]. 2020 I-EEE 11th International Conference on software engineerin-g and service science (ICSESS). IEEE, 2020: 52-56.
3. 3.Gök M. Dynamic path planning via Dueling Double Deep Q-Network (D3QN) with prioritized experience replay[J]. Applied Soft Computing, 2024, 158: 111503.
4. 4.Gu Y, Zhu Z, Lv J, et al. DM-DQN: Dueling Munchausen deep Q network for robot path planning[J]. Complex & Int-elligent Systems, 2023, 9(4): 4287-4300.
5. 5.Huang R, Qin C, Li J L, et al. Path planning of mobile rob-ot in unknown dynamic continuous environment using re-ward-modified deep Q-network[J]. Optimal Control

Appli-cations and Methods, 2023, 44(3): 1570-1587.
6. 6.Jiang J, Zeng X, Guzzetti D, et al. Path planning for astero-id hopping rovers with pre-trained deep reinforcement lear-ning architectures[J]. Acta Astronautica, 2020, 171: 265-279.
7. 7.Jiang D, Fang Z, Cheng C, et al. Action guidance-based de-ep interactive reinforcement learning for AUV path planni-ng[C]. 2022 International Conference on Machine Learning, Control, and Robotics (MLCR). IEEE, 2022: 158-165.
8. 8.Khatib O. Real-time obstacle avoidance for manipulators a-nd mobile robots[C]. Proceedings. 1985 IEEE international conference on robotics and automation. IEEE, 1985, 2: 500-505.
9. 9.Kretzschmar H, Spies M, Sprunk C, et al. Socially compli-ant mobile robot navigation via inverse reinforcement lear-ning[J]. The International Journal of Robotics Research, 2016, 35(11): 1289-1307.
10. 10.Lee D H, Lee S S, Ahn C K, et al. Finite distribution estim-ation-based dynamic window approach to reliable obstacle avoidance of mobile robot[J]. IEEE Transactions on Indus-trial Electronics, 2020, 68(10): 9998-10006.
11. 11.Li J, Qin H, Wang J, et al. Openstreetmap-based autonom-ous navigation for the four wheel-legged robot via 3d-lidar and ccd camera[J]. IEEE Transactions on Industrial Electr-onics, 2021, 69(3): 2708-2717.
12. 12.Li J, Chen Y, Zhao X N, et al. An improved DQN path pla-nning algorithm[J]. The Journal of Supercomputing, 2022, 78(1): 616-639.
13. 13.Li P, Ding X, Ren W. Research on path planning of cloud robot in dynamic environment based on improved ddpg al-gorithm[C]. 2021 China Automation Congress (CAC). IE-EE, 2021: 3561-3566.
14. 14.Luo L, Zhao N, Zhu Y, et al. A* guiding DQN algorithm f-or automated guided vehicle pathfinding problem of roboti-c mobile fulfillment systems[J]. Computers & Industrial E-ngineering, 2023, 178: 109112.
15. 15.Ma N, Wang J, Liu J, et al. Conditional generative adversa-rial networks for optimal path planning[J]. IEEE Transacti-ons on Cognitive and Developmental Systems, 2021, 14(2): 662-671.
16. 16.Meng H, Zhang H. Mobile robot path planning method ba-sed on deep reinforcement learning algorithm[J]. Journal of Circuits, Systems and Computers, 2022, 31(15): 2250258.
17. 17.Tai L, Paolo G, Liu M. Virtual-to-real deep reinforc-ement learning: Continuous control of mobile robots for mapless navigation[C]. 2017 IEEE/RSJ internati-onal conference on intelligent robots and systems (I-ROS). IEEE, 2017: 31-36.
18. 18.Van Hasselt H, Guez A, Silver D. Deep reinforceme-nt learning with double q-learning[C]. Proceedings of the AAAI conference on artificial intelligence. 2016, 30(1).
19. 19.Wang S, Lin F, Wang T, et al. Autonomous vehicle path planning based on driver characteristics identifi-cation and improved artificial potential field[C]. Act-uators. MDPI, 2022, 11(2): 52.
20. 20.Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]. Int-ernational conference on machine learning. PMLR, 2016: 1995-2003.
21. 21.Wang B, Liu Z, Li Q, et al. Mobile robot path plann-ing in dynamic environments through globally guid-ed reinforcement learning[J]. IEEE Robotics and Au-tomation Letters, 2020, 5(4): 6932-6939.
22. 22.Wu Z, Yin Y, Liu J, et al. A novel path planning approach for mobile robot in radioactive environment b-ased on improved deep Q network algorithm[J]. Sy-mmetry, 2023, 15(11): 2048.
23. 23.Xin B, He C. DRL-based improvement for autonom-ous UAV motion path planning in unknown environ-ments[C]. 2022 7th International Conference on Co-ntrol and Robotics Engineering (ICCRE). IEEE, 2022: 102-105.
24. 24.Yang W, Wu P, Zhou X, et al. Improved artificial p-otential field and dynamic window method for amp-hibious robot fish path planning[J]. Applied Sciences, 2021, 11(5): 2114.
25. 25.Yang J, Ni J, Xi M, et al. Intelligent path planning of underwater robot based on reinforcement learning[J]. IEEE Transactions on Automation Science and En-gineering, 2022, 20(3): 1983-1996.
26. 26.Liu Y, Chen W, Bai Y, et al. Aligning cyber space with physical world: A comprehensive survey on e-mbodied ai[J]. IEEE/ASME Transactions on Mecha-tronics, 2025.
27. 27.Yuan X, Yuan X, Wang X. Path planning for mobile robot based on improved bat

algorithm[J]. Sensors, 2021, 21(13): 4389.
28. 28.Zhang X, Shi X, Zhang Z, et al. A DDQN path plan-ning algorithm based on experience classification an-d multi steps for mobile robots[J]. Electronics, 2022, 11(14): 2120.
29. 29.Zhao P, Zheng J, Zhou Q, et al. A dueling-DDPG ar-chitecture for mobile robots path planning based on laser range findings[C]. Pacific Rim International C-onference on Artificial Intelligence. Cham: Springer International Publishing, 2021: 154-168.
30. Zhou Q, Lyu L, Liu H. Deep reinforcement learning with long-time memory capability for robot mapless navigation[C]. 2022 IEEE 25th International Confer-ence on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2022: 1215-1220.