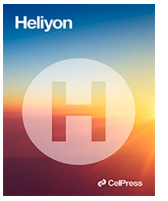
Heliyon 10 (2024) e32167

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

# Path planning of mobile robot based on improved TD3 algorithm in dynamic environment

Peng Li, Donghui Chen, Yuchen Wang, Lanyong Zhang, Shiquan Zhao *

*College of Intelligent Systems Science and Engineering, Harbin Engineering University, No.145 Nantong Street, Harbin, Heilongjiang Province, 15001, China*

ARTICLE INFO

*Keywords:*
Path planning
Dynamic environment
Mobile robot
Improved TD3 algorithm
Deep reinforcement learning

ABSTRACT

This paper proposes an improved TD3 (Twin Delayed Deep Deterministic Policy Gradient) algorithm to address the flaws of low success rate and slow training speed, when using the original TD3 algorithm in mobile robot path planning in dynamic environment. Firstly, prioritized experience replay and transfer learning are introduced to enhance the learning efficiency, where the probability of beneficial experiences being sampled in the experience pool is increased, and the pre-trained model is applied in an obstacle-free environment as the initial model for training in a dynamic environment. Secondly, dynamic delay update strategy is devised and OU noise is added to improve the success rate of path planning, where the probability of missing high-quality value estimate is reduced through changing the delay update interval dynamically, and the correlated exploration of the mobile robot inertial navigation system in the dynamic environment is temporally improved. The algorithm is tested by simulation where the Turtlebot3 robot model as a training object, the ROS melodic operating system and Gazebo simulation software as an experimental environment. Meanwhile, the result shows that the improved TD3 algorithm has a 16.6 % increase in success rate and a 23.5 % reduction in algorithm training time. A generalization experiment was designed finally, and it indicates that superior generation performance has been acquired in mobile robot path planning with continuous action spaces through the improved TD3 algorithm.

## 1. Introduction

Mobile robot is one type of intelligent systems integrating microelectronics, communication, computer science and optics, which are extensively applied in the fields of agriculture, search and rescue, environmental monitoring, etc [1]. For example, unmanned aerial vehicles are used to perform aerial search and rescue, collect terrain data and airborne remote sensing [2], autonomous underwater vehicles are used to explore marine resources and monitor the marine environment [3], and autonomous guided vehicles are used for transportation of logistics warehousing, the detection and grasping of hazardous materials [4]. Path planning is one of the most basic problems that need to be solved before these mobile robots can move and explore autonomously in their task environment. Depending on the task environment, mobile robots search for optimal or suboptimal paths from their respective start states to their respective goal states based on specific constraints and performance criteria, which is known as the path planning problem for robots.

* Corresponding author.
*E-mail addresses:* lipeng@hrbeu.edu.cn (P. Li), chendonghui@hrbeu.edu.cn (D. Chen), wyc19980913@gmail.com (Y. Wang), zhaoshiquan@hrbeu.edu.cn (S. Zhao).

https://doi.org/10.1016/j.heliyon.2024.e32167
Received 3 November 2023; Received in revised form 23 May 2024; Accepted 29 May 2024
Available online 31 May 2024

The key constraint is to ensure that the mobile robots reach the goal position without colliding with each other, and the key performance criteria are the speed and quality of path planning. Furthermore, as the performance requirements for mobile robots increasing, superior working performance in dynamic working environments is needed as well.

Currently, effective path planning algorithms for mobile robots are mainly including search-based algorithms such as Dijkstra algorithm [5], A* algorithm [6], RRT* algorithm [7], and intelligent planning algorithms such as Genetic algorithm [8], Particle Swarm algorithm [9], Ant Colony algorithm [10]. However, the complex changes in the dynamic environment lead to the lack of a priori knowledge and the surge of computation in the planning of traditional algorithms, making it difficult to achieve good path planning. When Reinforcement Learning (RL) algorithms are applied to solve path planning problems, mobile robots realize their goals by interacting with the environment without the necessity to obtain any prior information of environment. The mobile robot acquires its state and environmental information through sensors, and calculates its next action to transition to the next state through the algorithm. Next action will be employed according to the size of the rewards which were the value to evaluate the current action, and continuously learns the action that receives the maximum reward value. Nevertheless, RL algorithms are susceptible to the "dimension explosion" issue in more intricate environment. Deep Learning (DL) shows the capability of effectively processing high-dimensional information, but requires sufficient labeled samples and is difficult to be applied to dynamic environments [11]. Thus, the combination with reinforcement learning and deep learning for end-to-end learning, forming deep reinforcement learning (DRL) with complementary strengths which is capable of performing path planning tasks for mobile robots in dynamic environments while handling high-dimensionl information within the environment efficiently [12].

Following advantages for DRL in dealing with path planning problems in dynamic environments.

(1) Firstly, low requirements needed for the models of robots and sensors themselves, the DRL algorithm only requires to obtain observations of the sensors that reflect the current state information, which will correspond to the following actions. During the learning process, the cycle of state-action to next-state continuously until the optimal strategy is learned, which is an end-to-end decision-making method with the ability to reduce many unnecessary impacts.
(2) Secondly, DRL applications in path planning problems do not require mobile robots to obtain any environmental information in advance, making it effective on solving path planning problems in dynamic environments. On the contrary, traditional path planning algorithms require prior knowledge of environmental information, and usually map the environment by slam algorithms, tending to re-planning with high cost due to the restriction when facing the changing dynamic environments. DRL, an end-to-end method, only needs to find the optimal action corresponding to the observed values, and then robots can still act based on the observed values in dynamic environments.
(3) Finally, DRL interacts with the training environment directly without the need for training through numerous labeled samples, saving time consumption for data collection and effectively avoiding the impact of collected data correlation.

The TD3 algorithm is a widely favored DRL algorithm at present [13]. As an upgraded version of the DDPG algorithm, it enhances the performance and stability of the algorithm by incorporating a twin Q network and a delayed update strategy. Comparing to the DQN algorithm which requires the selection of an action with the maximum Q value from all actions, thus can only handle environments with finite action spaces [14], the TD3 algorithm can handle continuous control tasks. In comparison with the TRPO algorithm [15] and PPO algorithm [16], both are online policy algorithms with relatively low sample efficiency, but the TD3 algorithm is an offline policy algorithm with higher sample efficiency. When juxtaposed with the SAC algorithm [17], which demonstrates commendable performance in striking a balance between exploration and exploitation, the TD3 algorithm triumphs in handling high-dimensional state and action spaces. When a mobile robot executing path planning in a dynamic environment, the robot must undertake continuous control tasks, necessitate a brief training duration, and operate within a complex dynamic environment that requires a high-dimensional environmental state space. Taking all these factors into account, this article opts for the TD3 algorithm to design the path planning method for mobile robots. Based on the characteristics of the algorithm, an improved TD3 algorithm was presented in dynamic environments to deal with the issue of path planning. The main work and improvements made in the paper are as follows.

This paper takes two-wheel differential mobile robot based on TurtleBot3 Burger as the experimental research object for path planning [18], and which can control the robot's vertical movement, circular motion, or in-place rotation from the center of the chassis by adjusting the speed of two independent wheels. Setting specific state space, action space, and reward function parameters based on the simulation environment, which makes the movement of mobile robots more reasonable and facilitate the subsequent simulation experiments. This paper employs an improved TD3 algorithm, optimizing the algorithm through a series of rational strategies. Firstly, priority experience replay and transfer learning are utilized to curtail the training duration. Subsequently, the paper introduces apt OU noise and designs a dynamic delay update strategy, thereby augmenting the convergence speed and training success rate. Ultimately, within a dynamic environment, the path planning simulation experiment of the mobile robot validates the effectiveness of the improved TD3 algorithm in obstacle avoidance and path planning. The generalization capability has been verified empirically, demonstrating that the algorithm possesses robust generalization abilities.

## 2. Related works

### 2.1. Path planning

In the research field of path planning, two categories are proposed to divide the path planning problem, including global path

planning (basing on prior complete environmental information) and local path planning (when the environment is unknown or partially known, and real-time environment information is obtained through perception). Furthermore, according to the distinction between dynamic and static environments, four types of problems are formed, each of which has representative solutions.

Global static environment path planning: Since the static environment is in a completely known state and will not has much variation, algorithms of this type can find the globally optimal path by utilizing complete environmental information obtained through various methods. Algorithms suitable for this type include graph search algorithms such as A* algorithm and Dijkstra algorithm. In the case of known static environment information, these two algorithms can guarantee finding the shortest path. However, when the environment information is complex, the computational complexity is high, leading to low efficiency. In addition, some have proposed bio-inspired algorithms such as particle swarm optimization algorithm, ant colony algorithm and genetic algorithm to solve problems in complex static environments. Although these bio-inspired algorithms guarantee the success rate of finding the globally optimal path, they sacrifice some local obstacle avoidance capabilities.

Global Dynamic Environment Path Planning: When it comes to path planning in dynamic environment and the specific information about the changing circumstances is accessible, real-time SLAM can be used to implement it. At this time, global dynamic environment path planning algorithms such as RRT* and PRM can be used. RRT* algorithm is based on random sampling and searches for global paths by generating a random tree, which can quickly update path planning when the environment changes. However, the dynamic environment with complexity and the requirement for SLAM map construction, the algorithm has a large computational burden, lower real-time performance, and efficiency.

Local Static Environment Path Planning: When the environment is unknown or partially unknown, global path planning methods cannot be applied without environmental information. At this point, the mobile robot needs to have the ability to obtain real-time environmental information in order to autonomously plan the optimal path. In traditional algorithms, the Dynamic Window Approach [19] ensures the requirements of dynamic model and obstacle avoidance by applying constraints to the velocity space, but its exploration ability is lacking in more complex environments, making it prone to getting stuck in local optima. Additionally, the Artificial Potential Field Method [20] borrows from the mechanical idea of attraction and repulsion to control robot movement, but it also has the risk of getting stuck in local optima. Another approach involves the use of Simulated Annealing [21] to solve the problem of getting stuck in local optima, which is simple and highly adaptable, but is also plagued by slow convergence speed and other issues.

Local dynamic environment path planning: When facing an unknown and constantly changing dynamic environment, the above methods are helpless. At this time, people have gained inspiration from biomimetic algorithms and use neural network algorithms to address the issue of local dynamic environment path planning [22]. However, the neural network algorithm requires a large amount of training to obtain enough samples to perform path planning tasks well. By introducing DRL which interacts with the environment to obtain a large number of training samples, and using neural networks for data sample processing, path planning tasks can be well implemented. Therefore, deep reinforcement learning algorithms have significant advantages in solving the path planning of mobile robots in local dynamic environments.

### 2.2. Deep reinforcement learning (DRL)

Value-based algorithms are mostly applied in discrete environments. However, in scenarios with large and continuous action sets (such as in the field of mobile robot control), these methods struggle to learn optimal results. In policy-based algorithms, the agent formulates a strategy and takes action based on this strategy. DRL algorithms continually train and optimize the policy in order to achieve the maximum reward value. The policy-based algorithms effectively solve the application problems in scenes with large and continuous action sets.

Value-based DRL algorithms, such as the Deep Q Network (DQN), have been adept at handling scenarios with high-dimensional state inputs and low-dimensional action outputs. However, it often overestimates Q-values, leading to unstable training. Subsequently, numerous improved versions have been developed based on the characteristics of the DQN algorithm. The introduction of a double Q network forms the Double DQN network [23], effectively avoiding overestimation of Q-values. The NoisyNet DQN algorithm adds Gaussian noise to the final layer of the network and updates parameters through training of the weight network, significantly reducing computational costs [24]. Policy-based DRL algorithms include the Actor-Critic algorithm, which can learn both policy functions and value functions, but training instability remains an issue. Many advanced algorithms have been developed and improved based on the Actor-Critic framework [25]. The introduction of a trust region forms the TPRO algorithm, solving the problem of excessive policy parameter changes. The introduction of importance sampling techniques and N-step updates forms the PPO algorithm, greatly enhancing training stability. The A3C algorithm introduces multiple threads interacting with the environment and updating asynchronously, reducing data correlation and accelerating convergence speed [26].

Considering the advantages of deep reinforcement learning, some scholars have applied it to the path planning for mobile robots in dynamic environments, achieving corresponding results. Mobile robots explore and utilize complex in dynamic environments, and the algorithm processes high-dimensional state information from the interactions, outputting value functions and action values for the mobile robots. The mobile robots continuously optimize their behavior to plan the optimal path, with the goal of maximizing the reward function. However, existing algorithms still have room for improvement in terms of convergence speed, stability, and generalizability.

For instance, the path planning of indoor robots based on the DQN algorithm discretizes continuous actions, which leads to mediocre control effects and weak exploration abilities. On the other hand, the DQN algorithm tends to overestimate Q-values, which results in poor convergence speed, making it difficult to plan high-quality paths. Zhang et al. introduced Pan/Tilt/Zoom (PTZ) image information into the input of the deep reinforcement learning algorithm, effectively processing indoor environment information and

optimizing path planning convergence efficiency [27]. Path tracking for drones based on the DDPG algorithm faces challenges from environments with unknown disruptions and the generalizability of deep reinforcement learning. Ma et al. improved the DDPG algorithm by using a policy relief to allow the drone to properly explore new environments. They also used a Soft Weight (SW) method to improve the utilization rate of episodes with higher importance and richer information, thus achieving better tracking control accuracy and robustness [28]. For mobile robot path planning based on the SAC algorithm, the sampling efficiency and exploration abilities are high. Adjusting the temperature parameter appropriately can improve the performance of policy learning, but manual adjustment is inefficient. Chen et al. added adaptive behavior to automatically adjust the temperature parameter, allowing the entropy to vary in different states, thus controlling the degree of exploration and reducing the likelihood of learning suboptimal policies to some extent [29]. Fan et al. putted forward an intelligent 3D path planning algorithm according to IFDS, and used DRL to solve the coefficients of IFDS for solving problems about UAV path planning in 3D dynamic environment, and reducing the impact of dynamic obstacles on UAV [30]. Gao et al. implemented the combination of TD3 algorithm with Probability Roadmap (PRM) which is the traditional method subordinated to global path planning and this new local dynamic environment path planning algorithm achieved good results, ensuring a higher success rate for drones to avoid dynamic obstacles and finding the optimal path effectively [31]. Kim et al. putted forward a method of motion planning combined with TD3 and HER, which obtained more seamless and concise paths in the changing dynamic environment, and reduced the sudden changes of routes due to dynamic obstacles [32].

## 3. System model

### 3.1. Markov Decision Process with mobile robot path planning

Markov Decision Process (MDP) [33] is a sequential decision-making mathematical model used to simulate the randomized policies and rewards of intelligent agents in a Markovian environment. A five-elements tuple model $\{S, A, P_{sa}, \gamma, R\}$ is taken to describe the Markov decision process as hypothesis.

The agent-environment interaction in a Markov decision process is shown in Fig. 1.

The decision-making process of mobile robot path planning can be abstracted as a Markov decision process, with continual interaction between the mobile robot and the dynamic environment. The reinforcement learning algorithms can be applied to train the mobile robot about decision-making in relation to the rotational speed of their wheels, and the robot can proactively opt for a trajectory leading to a destination with a significant reward value. Consequently, an efficacious and collision-free planned path can be obtained for the mobile robot.

A quintuple model $\{S, A, P_{sa}, \gamma, R\}$ is considered to depict the decision-making process for the path planning of mobile robot. Within an episode, the mobile robot receives some representation of the environment's state space $S$, including all previous state information such as radar scanning information, collision status details, etc. Then, the mobile robot selects action space $A$, including angular velocity and linear velocity of the robot, where the corresponding Actor Network of the action space updated with algorithm training. The probabilities of the mobile robot to transit to the next state, based on the action $A$ in the state $S$, called the state transition probability $P_{sa}$. As time passing, minimizing the impact of long-term benefits on current evaluations is necessary, which is in line with human pursuit of immediate benefits, due to the increasing uncertainty of long-term benefits. For minimizing the impact of subsequent state returns on the current state evaluation, discount factor, $\gamma$ is introduced to the decision-making process of path planning, which can avoid the algorithm falling into an infinite loop and achieve convergence. As $\gamma$ approaches 1, future rewards are taken into account more strongly, as $\gamma$ approaches 0, immediate rewards are concerned more strongly. The mobile robot accepts a numerical reward $R$, which taken as a consequence of its action.

The decision-making process for the mobile robot path planning is based on the deterministic policy, so the state-value equation is expressed as $V^{\pi}$ , which can be obtained by the following strategy $\pi$ beginning from the state $s$.

$$V^{\pi}(s) = E_{\pi}\left[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \ldots | s_0 = s\right] \tag{1}$$

With the existence of action, equation (1) can be rewritten when the deterministic strategy is applied as action-value equation, and the

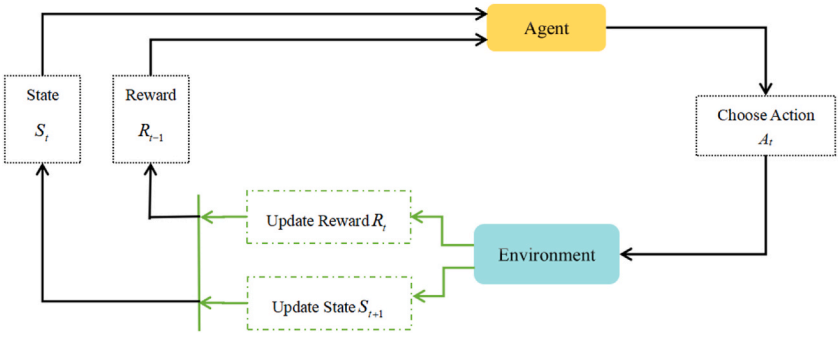



**Fig. 1.** State transition graph of Markov Decision Process.

action is described as $a = \pi(s)$ based on the initial state, as shown in equation (2).

$$Q^{\pi}(s,a) = E_{\pi}\left[R_0 + \gamma R_1 + \gamma^2 R_2 + \ldots \middle| s_0 = s, a_0 = a\right] \tag{2}$$

With the Bellman expectation equation mentioned above, our goal is to maximize these two equations above, so taking the maximum value of the equation yields the optimal value equation $V^*$ and the optimal action value equation $Q^*$, which can be also called Bellman optimality equation, as shown in equation (3).

$$\begin{aligned} V^*(s) &= \max V^{\pi}(s) = \max\left\{R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')\right\} \\ Q^*(s,a) &= \max Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') \max_{a' \in A} Q^*(s', a') \end{aligned} \tag{3}$$

Where, $s'$ is the next state, $a'$ is the action taken in the next state, $P_{sa}(s')$ is the probability of taking action $a$ to transit to state $s'$ from state $s$, and also known as the state transition probability. The strategy adopted in calculating the Bellman optimality equation is the optimal strategy $\pi^*$, as shown in equation (4).

$$\begin{aligned} \pi^*(s) &= \operatorname{argmax}_{a \in A} Q^*(s,a) \\ &= \operatorname{argmax}_{a \in A} R(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') V^*(s') \end{aligned} \tag{4}$$

In the research of mobile robot path planning problem in dynamic environment, the Markov property refers to that the mobile robot selects the optimal action by sensing the state of the surroundings and information of its own position and posture. Based the optimal action, it gets the feedback return value by sensing the changed environment and its own state, and then adjusts the next action, and continues to cycle until a complete optimal strategy is formed.

### 3.2. State space

Mobile robots need to obtain information such as the environment and their own posture to take the optimal action, which is also known as state information, when trained by DRL algorithms. Therefore, we need to design a reasonable state space for the robot and environment to generate input data of neural network. In the path planning task of this paper, state information mainly consists of the following two aspects. On the one hand, environmental information includes the distance between robot and obstacle, and the distance from robot to the target point. On the other hand, it includes its own posture information and motion status. The selected mobile robot laser radar has measurement range in the range of 3.5 m in this paper, and its angle range of detection is 180°taken robot front as reference coordinate system, where angle information is divided into 20 dimensions, with date in range of 9°for each. The robot's state will be determined as a collision state when obstacles within a distance of less than 0.2 m and in range of 180°from the robot front.

The distance between robot and obstacles is measured by laser radar whose principle can be simplified as triangular ranging. The model of triangulation ranging is shown in Fig. 2, where the angle between the emitted light and the radar plane is $\alpha$, the angle between the reflection ray and the radar plane is $\beta$, and the distance between the laser and the receptor is $L = L_1 + L_2$.

Calculating the distance $d$ between laser radar and obstacles, as shown in equation (5).

$$d = L\frac{\sin\alpha \sin\beta}{\sin(\alpha + \beta)} \tag{5}$$

The robot-obstacle distance is implanted as $d_i(t)$ measured by the *i-th* dimensional laser radar of the mobile robot at time *t*. Based on the principles above, we use the Boolean variable *done* as the state variable, which is *True* when collision happened and *False* in normal

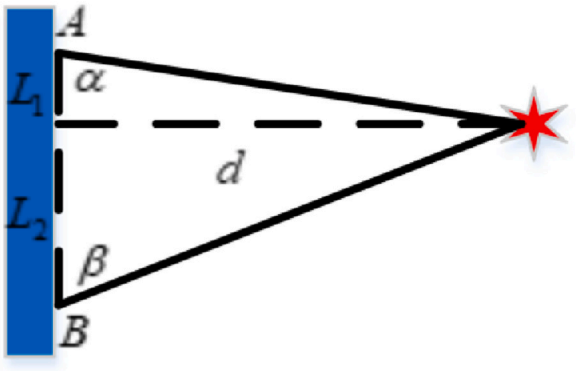



**Fig. 2.** Schematic diagram of laser radar triangulation ranging.

conditions, as shown in equation (6).

$$done = \begin{cases} True, d_i(t) < 0.2m \\ False, d_i(t) > 0.2m \end{cases} \tag{6}$$

In addition to the 20 dimensional information and collision information of the laser radar mentioned above, it is also necessary to know some real-time posture information of the robot, such as the arrival status of the robot, the robot-target distance $d_t$, and the angle $\theta$ between motion direction and the target, as shown in equation (7).

$$\begin{aligned} d_t &= \sqrt{(x_t - x)^2 + (y_t - y)^2} \\ \theta &= angle_t + yaw \end{aligned} \tag{7}$$

Among them, the target point's coordinates within the map are $(x_t, y_t)$, and the coordinates of the robot at time $t$ are $(x, y)$. $angle_t$ indicates the angle which formed by the line connected from the robot to the target point and the x-axis, yaw represents the orientation angle. Combined with 20 dimensional laser information and 1 dimensional collision information, the state space data consists of 24 dimensions information.

### 3.3. Action space

First, the kinematics model of mobile robot is analyzed. Robots in Turtlebot3 series can be simplified as a two wheel differential robot, as the movement of it at time $t$ shown in Fig. 3.

Obviously, the robot will move in a straight line when the wheels at the same speed, and conversely move in a circular motion. Where, $O$ is seen as the center of the circular motion, $\theta_1$ is the angle that mobile robot bypasses in $\Delta t$ time, $\theta_3$ is the offset angle perpendicular to the direction of it. $l$ is the distance between the center points of the left and right wheels, and $d$ is the distance of approximate straight-line which is the further travelling length of right wheel than the left. $r$ is the radius of the circular motion trajectory of the center point of the mobile robot. $\omega$ is the angular velocity, $v_l$ is the left wheel motion velocity, $v_r$ is the motion velocity of right, and $v$ is the movement velocity of center of mass (the center point of the two wheels).

According to the equation $v = \omega \cdot r$, the relationship between left wheel speed, right wheel speed, and center of mass speed can be obtained, as shown in equation (8).

$$\begin{aligned} \omega &= \frac{v}{r} = \frac{v_r}{r + l/2} = \frac{v_l}{r - l/2} \\ v &= \frac{v_l + v_r}{2} \end{aligned} \tag{8}$$

Based on the geometric principle of similar triangle, the angular velocity can be obtained, as shown in equation (9).

$$\omega = \tan\theta_1 = \frac{v_r - v_l}{l} \tag{9}$$

The radius $r$ of the robot's circular motion can be obtained from equations (8) and (9).

$$r = \frac{v}{\omega} = \frac{l \cdot (v_l + v_r)}{2(v_r - v_l)} \tag{10}$$

According to equation (10), the motion mode of two-wheel differential mobile robot is determined by the speed difference between the two wheels, and if circular motion is carried out, the center $O$ of the motion can also be determined.

At the same time, the $v_l, v_r$ can be resolved by analyzing the velocity $v$ of the robot's center of mass, as shown in equation (11).

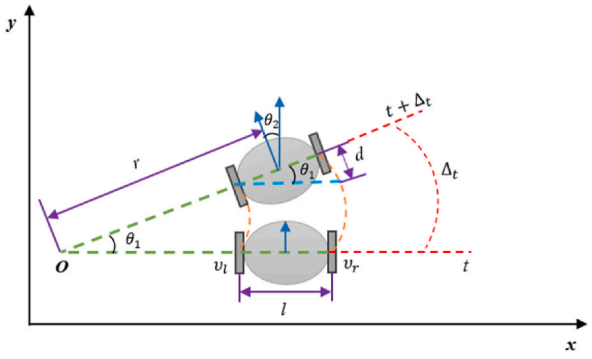



**Fig. 3.** Mobile robot motion model.

$$\begin{bmatrix} v_r \\ v_l \end{bmatrix} = \begin{bmatrix} v + \frac{l}{2}\omega \\ v - \frac{l}{2}\omega \end{bmatrix} = \begin{bmatrix} 1 & \frac{l}{2} \\ 1 & -\frac{l}{2} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{11}$$

The equation above is called the inverse kinematics equation. Based on the kinematics equation, the speed of the two driving wheels can be calculated by the use of real-time speed information $[v, \omega]^T$ of the robot and the inverse kinematics equation.

The motion information which calculated based on the kinematics model includes the angular velocity value and linear velocity value. To ensure the reasonableness of motion, they are continuous within a certain range and cannot exceed certain thresholds. So this paper sets the maximum linear velocity to 0.5 m/s, the maximum angular velocity of the robot to 0.3 rad/s.

### 3.4. Reward function

In the situation of dynamic environment about mobile robot path planning problems, a reasonable reward function can enable the mobile robot to quickly reach the destination point, and must have continuity due to the continuous motion process is continuous. Mobile robots will continuously pursue higher reward values during training to discover the optimal path to achieve the destination point. Therefore, the reward function designed consists of three parts, corresponding to three states: collision with obstacles, reaching the target area, and path planning process. The specific values are as shown in equation (12).

$$reward = \begin{cases} -200, d_o \leq 0.2 \\ 100, d_t \leq 0.2 \\ 300 \times (d_t(t-1) - d_t(t)), others \end{cases} \tag{12}$$

$d_0$ is the robot-obstacle straight-line distance, and when $d_0 \leq 0.2m$, in that case, collision will be imminent between the obstacle and robot, and gains $-200$ reward value according to the method. $d_t$ as the robot-target distance, and when $d_t \leq 0.2m$, that is, the robot accomplishes the capture of goal point, then algorithm gives a 100 reward value. In the normal navigation state of the robot, due to the continuous motion, the difference between distance $d_t(t-1)$ at previous action moment and distance $d_t(t)$ at current action moment is used as the measure of evaluating the action quality. The positive or negative situation of the difference also reflects whether the robot is moving towards or away from the target point, and this value is multiplied by 300 times as the reward value.

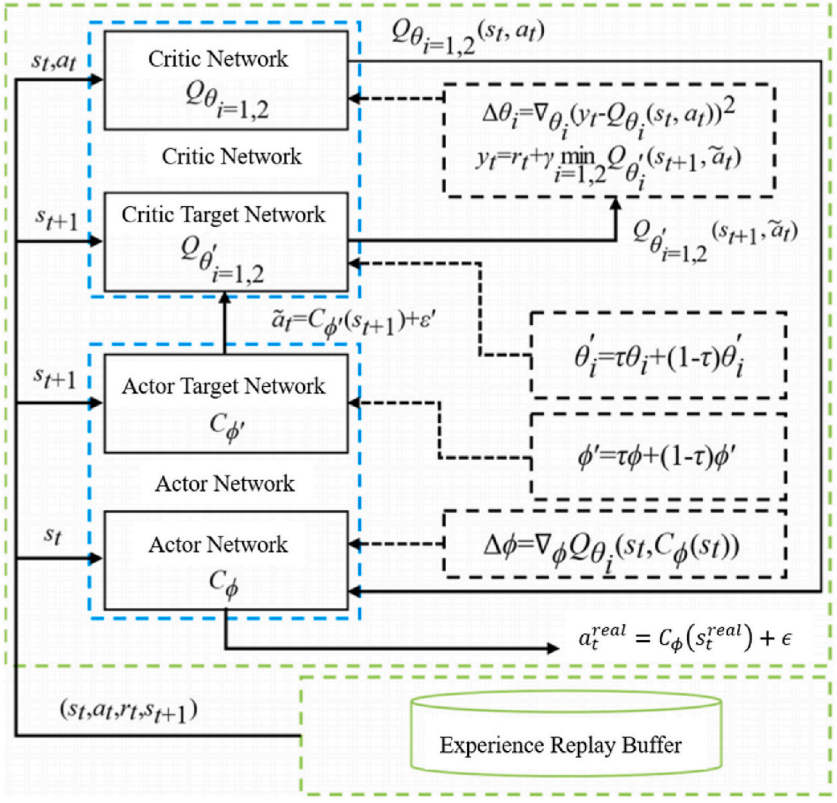



**Fig. 4.** TD3 algorithm framework.

## 4. Planning methods

### 4.1. TD3 algorithm

The TD3 (Twin Delayed Deep Deterministic Policy Gradient Algorithm) algorithm is applied in this paper. Three improvement methods compared to the DDPG algorithm are as follows: (1) Delayed policy updates; (2) Clipped double-Q method; (3) Target policy smoothing regularization.

The overall framework of TD3 method is illustrated in Fig. 4.

Firstly, the clipped double-Q method can be applied to calculate the corresponding Q value through two independent critical networks, then the subsequent Critic Network is updated according to the smaller Q value. The second point involves a delayed strategy update, whereby the Critic Network undergoes multiple updates until it approaches relative stability, followed by the Actor Network performing an update. This delay in updating enables the Actor Network to more accurately identify the maximum Q estimate rather than updating at a sub-optimal point. Thirdly, the smooth regularization, where through adding noise to the target actions, allows TD3 to own the capability of appropriately smoothing the target strategy and preventing the over-fitting of the policy network.

The Critical Network and Critical Target Network each have two networks, and equation (13) can be obtained in accordance with the holistic framework of TD3 algorithm.

$$\begin{cases} \Delta\theta_i = \bigtriangledown\theta_i(y_t - Q_{\theta i}(s_t, a_t))^2 \\ y_t = r_t + \gamma \min\limits_{i=1,2} Q_{\theta_i'}(s_{t+1}, \widetilde{a}_t) \\ \widetilde{a}_t = C_{\varphi'}(s_{t+1}) + \varepsilon' \end{cases} \tag{13}$$

Among the algorithm formula above, $\varepsilon'$ is the additional random noise of target policy smoothing regularization which obeys the normal distribution clip $(N(0,\sigma), -c, c)$ after truncation, where $N$ represents a normal distribution and $\sigma$ is a variance parameter of this equation, the truncated value is $c$ with boundary greater than zero, and $r_t$ is the reward value of current time. Furthermore, $\gamma$ is a discount factor which represents the value impact proportion in the present moment. Among formula (13), the upcoming action $a_t$ will be calculated and outputted at present state $s_t$ through Actor Network $C_\varphi$, the objective action $\widetilde{a}_t$ will be outputted through Actor target Network $C_{\varphi'}$ according to the next state $s_{t+1}$. Analogously, the Q value of $Q_{\theta_i}(s_{t+1}, a_t)$ and the target Q value of $Q_{\theta_i'}(s_{t+1}, \widetilde{a}_t)$ determine the quality of the action in the current state, and then help the Actor Network to update the policy. The actual value $y_t$ is obtained by selecting the minimum Q values calculated by the two Critic target networks and applying it to the equation of $\Delta\theta_i = \nabla\theta_i\left(y_t - Q_{\theta_i}(s_t, a_t)\right)^2$. This formula updates $\theta_i$ based on the change $\Delta\theta_i$ of Critic Network parameters which obtained through the gradient increase of the Loss equation $\left(y_t - Q_{\theta_i}(s_t, a_t)\right)^2$. The experience information obtained in such step, quasi the current state, action, reward value, and next step state, is preserved in tuple format to an experience replay pool. When updating the parameters of the Actor Network and Critic Network, the experience replay pool is sampled randomly to update the data.

Performing soft updates on the Actor Network and four target networks as the following formula (14).

$$\begin{cases} \Delta\varphi = \bigtriangledown_\varphi Q_{\theta_i}(s_t, C_\varphi(s_t)) \\ \theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \\ \varphi' \leftarrow \tau\varphi + (1-\tau)\varphi' \end{cases} \tag{14}$$

Where, $\tau(\tau \leq 1)$ is the soft update attenuation coefficient. Soft update insures the iterative updating of the target network, utilizing the linear combination of the current network parameters $\varphi$, $\theta_i$ and target network parameters $\varphi'$ , $\theta_i'$. Using soft update, the parameters of the target network change gradually and the target values change smoothly. Then, provided that in every iteration the target network is refreshed, the algorithm possesses capabilities of sustaining the certain degree of stability. The smaller coefficient of soft update attenuation, more stability will the algorithm possesses, the slower variation will the target network owns, and the more sluggish convergence rate will the algorithm gets.

### 4.2. Improving TD3 algorithm

There are some shortcomings in the traditional algorithm theory mentioned above.

(1) The experience replay pool is designed to solve the problem of strong correlation between environmental samples, which makes it difficult to converge. It mixes the current environmental feedback data samples with the previously obtained environmental samples, updates the network parameters through random extraction from the mixed samples during training, thereby achieving the effect of accelerating algorithm convergence. However, the experience replay pool contains a large number of failed experiences of collision obstacles generated by robot early training, which putted into the experience replay pool with rare successful experiences in the early stage will result in a low success rate of subsequent training and exceedingly extent the training time.

(2) The robot starts training from completely random actions at each time, and the sample data in training is irregular, which will lead to poor training efficiency, especially the poor ability to converge needs, and over-fitting is easy to occur when the number

of training rounds is small. Subsequently, the generalization performance of traditional algorithms in mobile robot path planning problems is not ideal, and the trained results and parameters in a single environment are difficult to directly apply to solve path planning problems in different environments.

(3) In the standard TD3 algorithm, Gaussian noise is inherently applied. Due to the independence of the Gaussian noise in the exploration process, the agent's action at each step is a random action completely following the Gaussian distribution. The benchmark of this algorithm is typically depended on simulation tasks, and there is often no need to consider time discretization issues. When the TD3 algorithm is confronted with minimal time discretization granularity, simply the discretization granularity amplified, which lets the independent noises still play their parts effectively. Therefore, in terms of performance evaluation for the TD3 algorithm, relatively favorable results can be obtained with Gaussian noise, without the necessity of integrating time-series-related noise like OU noise. However, in the process of the mobile robot path planning, the motor speed control is an inertial process. The inertial process is a typical low-pass filter, which filters out Gaussian noise with great differences between the two steps, which resulting problems such as low robot's exploration efficiency, poor training efficiency and effectiveness, and easily trapping the robot into local optima with the actions and selection strategies set in this way.

(4) In the traditional TD3 algorithm, the estimation of Critic Network value affects the update direction and step size of the Actor Network strategy, determining the quality of policy updates. Due to the dynamic changes in the performance of robots during training, delaying updates with a fixed step size is clearly not flexible and accurate enough. Especially when the Q value estimation is suboptimal, vicious circle will be the perishing result caused by the chain reaction, that of the suboptimal selection from the corresponding strategy and the next Q value estimation will be affected by the action in suboptimal. And to a certain extent, TD-error reacts the amplitude of update in the Critic Network. At a certain time, acquiring the oversized TD-error, signifies that the Critical Network has undergone a significant update, and implies that the current Q estimate is suboptimal. In this case, the Actor Network demands more delay intervals for updating in order to wait for a better Q estimate. And it has a high possibility to skip the optimal Q estimation, resulting in failure to update in a timely manner when there is too much delay in updating the Actor Network.

In this paper, the methods of prioritized experience replay and transfer learning are introduced, aiming to problems (1) and (2) above. A noise that simulates OU noise is designed in the algorithm, aiming to problem (3). Meanwhile, a dynamic delayed update strategy is designed and applied in response to problem (4). Specific steps are as follows.

1) **Prioritized experience replay**: In path planning, training will prioritize extracting experiences that have better training effects on mobile robot path planning. However, to ensure sample diversity, it is not enough to only extract the best experiences, but rather the probability of experience selection is directly proportional to the quality of the results whose effect is evaluated by TD-error.

So, the method of random prioritized experience replay (PER) is appended to the study to solve the sampling issue [34]. Where, through measuring the indicator time difference error TD-error $r_t + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \widetilde{a}_t) - Q_{\theta_i}(s_t, a_t)$, we select the experience with a large reward value greedily, but it is easy to cause over-fitting to produce superior TD-error and lead to a loss of diversity issue. Therefore, to address this issue, the method employs random priority sampling, bias and importance sampling techniques Which guarantees the probability of sampling with empirical priority increases monotonically as the reward value, while also guaranteeing a non-zero probability for even the lowest-priority experience to be sampled. The probability of sampling is shown in equation (15).

$$P(i) = \frac{p_i^a}{\sum p_k^a} \tag{15}$$

Where, $p_i = |\delta_t + \varepsilon|$ is the $i$-th experience priority namely the probability of being drawn, $\delta_t$ is TD-error, $\varepsilon$ as minuscule value which used to guard the probability of experiences with TD-error 0 being extracted against 0, $\alpha$ as the hyperparameter of control sampling is employed to modify the prioritization (when $\alpha = 0$, the probability are the same value as uniform sampling), and the value range of $k$ is (0, $i$+1).

To improve the traversal speed of the experience pool, the SumTree data structure is chosen. While conducting sampling, the gross priority value of the root nodes is divided by *batch_size* to create *batch_size* intervals. Then pick out numbers from each interval in stochastic that can perform sampling efficiently.

Due to the TD-error introduced to the PER, the probability and expected value of data extraction changed by this. Therefore, it is necessary to introduce importance sampling and annealing factor $\beta$, and modify weight is corrected by importance sampling as shown in equation (16).

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta} \tag{16}$$

Where, $N$ is the overall amount samples of the experience pool, and $P(i)$ is the sampling probability, $\beta$ is the annealing factor which is gradually trends towards 1 with updates and in range of (0,1), the non-uniform probability is completely compensated when $\beta = 1$. For stability, standardizing the weights to $\delta_i = \frac{1}{\max_i w_i}$. By multiplying the actual sampling weight with the standardized weight, quasi $w_i \cdot \delta_i$, and which can correct and restore the sampling probability distribution disrupted by PER. The feature of importance sampling under nonlinear fitting is that large step size will lead to too steep gradient, so small step size is generally employed.

The PER algorithm is mainly used to increase the probability of sampling better experiences in an experience pool with a large amount of invalid experiences, thereby shortening the training time. The path planning of mobile robots based on the SAC algorithm achieves exploration of the environment by maximizing the entropy of the policy. Compared with the TD3 algorithm, it has better balance advantages of exploration and utilization, so the quality of early experiences in the experience pool is relatively high. The addition of PER has a limited improvement on the quality of its sampling experience, so whether it is added has little impact. However, the TD3 algorithm has weaker exploration and utilization capabilities in the early stage, and the introduction of PER has obvious effects on the improvement of sampling quality.

2) **Transfer learning:** In generally speaking, transfer learning is taking the pre-trained mode as the initial model of the new model training [35]. Witch the development of deep learning, the design of neural networks has become increasingly complex, and the time to achieve convergence has also increased. Therefore, the introduction of transfer learning into deep learning can save substantial computing time and space, thus significantly reducing the training time of complex models. According to learning methods, there are four types of transfer learning: instance-based transfer, feature-based transfer, model-based transfer and relationship-based transfer.

In the context of the path planning task, we select model-based transfer learning method seeing as the involving of training calculation of neural networks. First, the TD3 algorithm is used for training in an obstacle free environment. After training, the mobile robot learning gains the capability to arrive at the target point, then saves the ability to approach the target point in the form of parameters to neural networks. Finally, the saved network parameters are used as initial weights of the TD3 algorithm for training in dynamic obstacles using transfer learning.

3) **OU noise:** Superimposed noise does not affect the output action due to the characteristic of TD3 algorithm, where exploration noise can be customized to increase the exploration desire of mobile robots in the exploration section. OU noise is suitable for the inertial systems, especially when dealing with a minute degree of time discretization. Additionally, it plays a significant role in preserving the integrity of the actual systems. Thereby, OU noise is introduced into TD3 algorithm, and which can generate time series related exploration, then enhance the exploration efficiency of reinforcement learning processes in inertial system environments. The form of its differential equation is as shown in equation (17).

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t \tag{17}$$

The state value will be maintained near the mean and corrected for major deviations. Introducing OU noise can enable mobile robots to explore in a small swing left and right manner when taking action, improving the convergence efficiency of the algorithm.

4) **Dynamic delayed updates:** The difference between the moving average value of the current Loss value and the current Loss value in the Critic Network is used as the standard to evaluate the update amplitude of the Critic Network. It suggests that the network value estimation at this time is suboptimal when the Critical Network is significantly updated next time, and the above difference will be larger. Therefore, setting a strategy of more steps in the Actor Network to delay updates and wait for better value estimation. But if there is too much delay in updating the Actor Network, there is a higher probability of missing high-quality value estimates and not being able to update network parameters in a timely manner. By calculating its exponentially weighted moving average (EWMA), the delay update interval can be changed accordingly. EWMA does not require saving all past values compared to traditional averages, resulting in a significant reduction in computational complexity, as shown in equation (18).

$$\nu_t = \beta\nu_{t-1} + (1-\beta)\rho_t \tag{18}$$

Where, $\rho_t$ is the practical value at time $t$, coefficient $\beta$ is weighted decline rate which getting faster descent with smaller value, $\nu_t$ is the EWMA value at time $t$. According to the Loss equation in equation (13) and equation (18), the moving average value $E_t$ of the Critic Network Loss at time $t$ can be calculated as shown in equation (19).

$$E_t = \beta\frac{1}{t-1}\sum_{n}^{t-1}(y_n - Q_{\theta i}(s_n,a_n))^2 + (1-\beta)\frac{1}{t}\sum_{n}^{t}(y_n - Q_{\theta i}(s_n,a_n))^2 \tag{19}$$

The difference between the moving average value $E_t$ of Loss and the current Loss value can reflect the update amplitude. However, in actual simulation experiments, this difference is usually very small, making it difficult to directly reflect the update amplitude of the Critic Network. Therefore, we need to enlarge it to an interval that can effectively reflect the update amplitude. Setting $\alpha$ as the magnification, and the measurement standard range for the update amplitude as shown in equation (20).

$$range = \alpha\left(e^{\frac{1}{t}\sum_{n}^{t}(y_n - Q_{\theta i}(s_n,a_n))^2} - E_t\right) \tag{20}$$

According to experimental testing, the range can be expanded to between (1,10) when the $\alpha$ value is 50, making the data more

convenient to process and resulting in better algorithm performance. Finally, evaluate the update amplitude based on the size of the range value, and set the next update step size according to the amplitude as shown in equation (21).

$$step = \begin{cases} 1, range \in (1,4) \\ 2, range \in (4,7) \\ 3, range \in (7,10) \end{cases} \tag{21}$$

### 4.3. Improving TD3 algorithm framework

By integrating the four improved algorithms described above for mobile robot path planning based on the TD3 algorithm, the training time of the algorithm is effectively shortened, the exploration and utilization are balanced, and the success rate of path planning is improved. The overall framework of the algorithm is shown in Fig. 5.

5. Experiments and Results.

This section introduces the construction and configuration of the simulation environment, and the training results of improved TD3 algorithm. For verifying the validity of the improved TD3 algorithm in mobile robot path planning problem in dynamic environments, we conducted comparative experiments using the improved TD3 algorithm and other methods. All environmental configurations for this simulation experiment are listed in Table 1.

### 4.4. Establishment of experimental platform and simulation environment

Using Gazebo software under ROS system as a simulation experimental platform, this simulation platform can build a simulation environment that greatly restores the real environment, reducing the disparity between simulation and reality. Drawing the site plan through the editing interface shown in Gazebo simulation platform, and generate the sdf files. Then insert the square model as the obstacle and add the motion control plug-in to realize the dynamic change of the obstacle, and generate the world files.

Furthermore, the robot used in the simulation experiment is the TurtleBot3 Burger series robot. According to the kinematics model and laser ranging model established in chapter 3, the urdf and xacro files are compiled to build the TurtleBot3 Burger robot model, which can be visualized in the software of Rviz and Gazebo with the characteristics and functions for simulation, and achieve the realization of interactive functions with the simulation environment. Fig. 6 show the visualization models of the TurtleBot3 Burger robot in Rviz and Gazebo, respectively.

Then, a dynamic simulation environment was built based on Gazebo, as shown in Fig. 7.

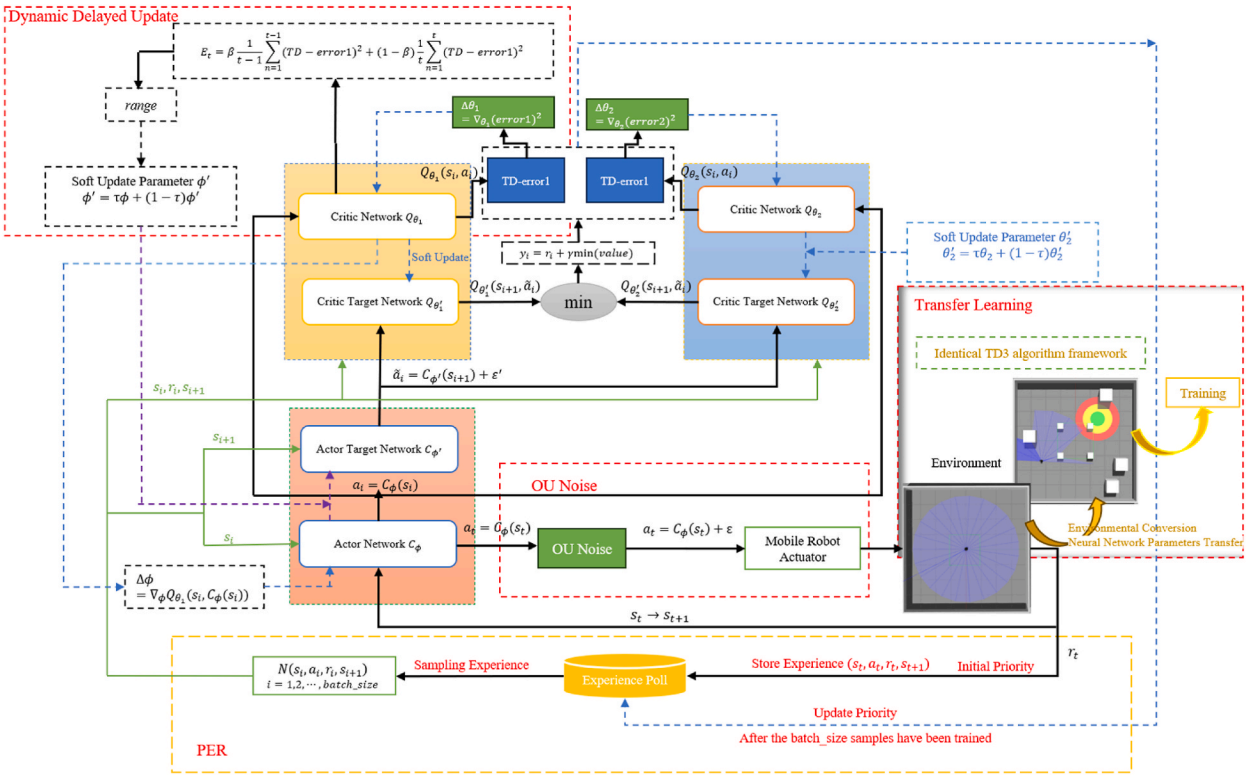



**Fig. 5.** Improved TD3 algorithm framework.

**Table 1**
Environmental Configuration for simulation experiments.

| Configuration | Values |
|---|---|
| Operating System | Ubuntu18.04 |
| CPU | R7-5800H |
| GPU | GeForce RTX 3070 |
| RAM | 16G |
| Robot Platform | ROS-melodic |
| Machine Learning Open Source Library | TensorFlow 1.14.0 |
| Open Source Artificial Neural Network Library | Keras 2.2.4 |
| Reinforcement Learning Experimental Environment Library | Gym |
| Project Programming Language | Python 2.7 |

Among them, the overall task space is 8 × 8 square area, including 4 cubic obstacles with a side length of 0.3 m and 4 cubic obstacles with a side length of 1 m. Small obstacles are distributed in the inner circle of the task space, while large obstacles are distributed in the outer circle of the task space. And they perform uniform motion of 0.5 m/s on their specified paths. In the simulation, setting the horizontal direction to the left as the positive direction of the y-axis, and the vertical direction to the up as the positive direction of the x-axis. The initial position of the mobile robot is (−2,2), and the target point is a colored circular area consisted of red, yellow, and blue, where the center point is the center point of the green circle, and its coordinates are set to (2,-2). In actual simulation, the navigation task of a mobile robot is to start from its initial position, select action strategies based on the designed algorithm, and bypass obstacles in a dynamic environment for finding reasonable path to reach the target. When the whole robot steps into the green circular scope, the navigation task is completed. The robot continuously learns and explores in the simulation environment, and if it reaches the target point successfully, encounters an obstacle, or exceeds 300 steps per round of training, then the scenario is reset and the next round of training begins. The basic parameters of the robot simulation experiment are set as shown in Table 2.

### *4.5. Results and analysis on simulation training*

DDPG, TD3 and improved TD3 algorithms are employed to train and compare mobile robots in dynamic environment, and the trained strategy set is compared and tested under the same environment according to the comparative experimental methods in Ref. [28]. The environment set can better test whether the robot can perceive the movement of obstacles based on changes in environmental information, reflecting the effectiveness and universality of the algorithm.

Fig. 8 (1), Fig. 8 (2), Fig. 8 (3), Fig. 8 (4) show the variation of reward values with variation of rounds when training robots in dynamic environments of each algorithm.

As shown in Fig. 8 (1), in dynamic environment, the convergence of DDPG training is non-ideal, the algorithm has not yet reached convergence even after 1200 rounds of training. Furthermore, the robot will still have a relatively high collision rate with obstacles in the subsequent stage of training through observing the training of the robot in the Gazebo visual interface. As shown in Fig. 8 (2), the TD3 algorithm has a convergence trend after 800 rounds, but it is not obvious, and the random exploration process of the TD3 algorithm is long, demonstrating the ability of robot to approach the target point under the algorithm grows slowly, which ultimately affects the convergence speed. From the visual simulation interface, the mobile robot has experienced a long period of random exploration, which has learned a little capability to achieve the target point, and a few rounds have successfully reached the target point after 800 rounds. However, as shown in Fig. 8 (3), after adding priority experience replay and transfer learning, a certain degree of convergence is achieved after 600 rounds. From the visual simulation interface, after 600 rounds, the robot approaches the target point in nearly half of the rounds, meanwhile be capable of reaching the target point in most of the time after 800 rounds. After that, as shown in Fig. 8 (4), the success rate and convergence degree of training have been further improved through the improved TD3 algorithm. After 500 rounds, there has been a good convergence effect, and the convergence speed in the early stage is also faster, the reward value converges to 2700 in the final. From the visual simulation interface, the robot reaches the target point most of the time

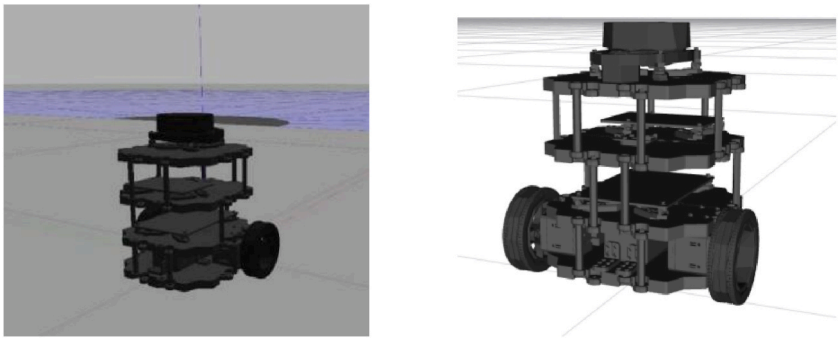


**Fig. 6.** Turtlebot3 visualization model.

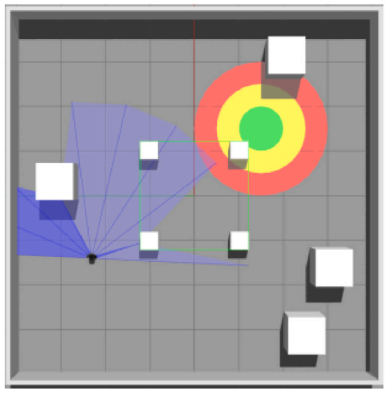


**Fig. 7.** Dynamic simulation environment.

**Table 2**
Parameters setting for simulation experiments.

| Parameters | Values |
|---|---|
| Total Rounds of Training | 1200 |
| Maximum Training Steps Per Round | 300 |
| Replay Buffer Size | 100000 |
| Starting Steps of Experience Replay | 8000 |
| Batch Size | 128 |
| Learning Rate of Actor Network | 0.0001 |
| Learning Rate of Critic Rate | 0.001 |
| Discount Factor | 0.99 |
| Initial Interval of Dynamic Delayed Updates | 2 |
| Adjustment Ratio of Noise level | 0.5 |

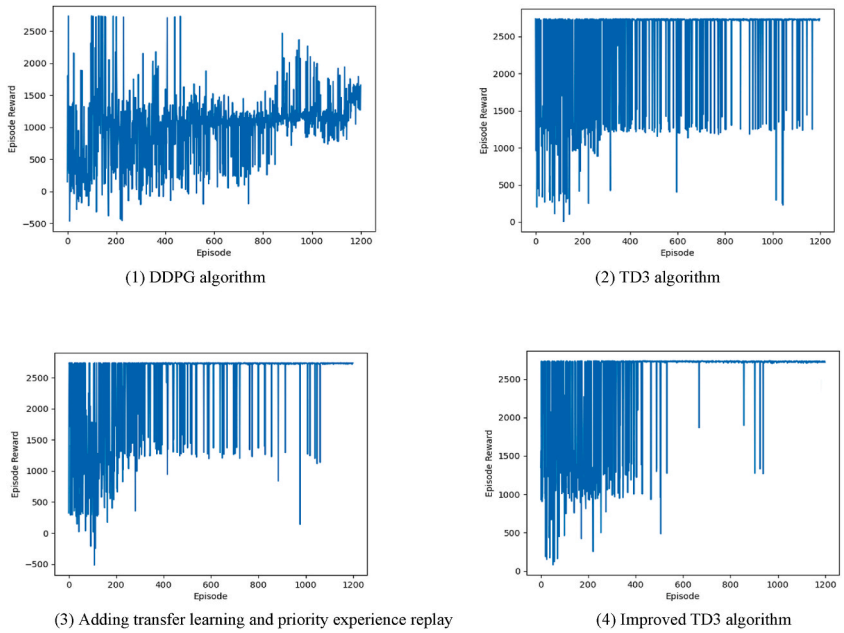



**Fig. 8.** Reward values of different algorithms.

after 500 rounds.

There are two modes set in the algorithm, one is the training mode mentioned above, and the other is to conduct 1200 rounds of independent navigation tests on the trained models in a dynamic environment, and which is called the testing mode. The following figure shows the path planning simulation process of the robot from the starting position to the target point after bypassing all obstacles

in the test mode, as well as the robot's movement trajectory. The mobile robot starts from the start point Fig. 9 (1), vias the intermediate point Fig. 9 (2), arrives the target point Fig. 9 (3), forms a complete motion trajectory Fig. 9 (4). According to Fig. 9, the mobile robot follows positive direction of x-axis firstly. Subsequently, through bypassing the small dynamic obstacle at the initial position (1,1), the robot turns to negative direction of y-axis, and trends to target points.

Fig. 10 is a line chart of the collision rate obtained from running one times in test mode, with forty rounds as a statistical cycle, determining the collision ratio of the mobile robot in each cycle. Evidently, the DDPG algorithm demonstrates a relatively high collision rate in dynamic environment, fluctuating between 25 % and 35 %. Observation from Gazebo simulation reveals that in dynamic environment, mobile robot based on the DDPG algorithm tends to move towards the inner circle of small obstacles, hence they are very prone to collision with small obstacles. The original TD3 algorithm also exhibits a high collision rate in this environment, ranging between 20 % and 28 %. This is primarily attributed to the algorithm's inefficient utilization of experience, the path exploration range is large, and it is easy to collide with large external obstacles and small obstacles in the upper left corner. The collision avoidance rate of the two improved TD3 algorithms has been greatly improved, especially the completely improved TD3 algorithm, which has an impressive balance between exploration and utilization, and the collision rate with obstacles is basically maintained at about 7 %. Therefore, it can be concluded that mobile robots based on the improved TD3 algorithm in dynamic environment have significantly improved performance in avoiding dynamic obstacles.

Fig. 11 represents the statistical values of the number of successful arrivals at the destination for each algorithm, based on five times running in test mode. Table 3 shows the average performance parameters obtained by running 5 times in test mode. Absolutely, the statistical values of each algorithm's success in reaching the destination can be used to calculate the success ratio of each algorithm. Similarly, the statistical values of collisions with obstacles for each algorithm can be used to derive the collision ratio between each algorithm and the obstacles. In summary, the improved TD3 algorithm has a 22.4 % increase in success rate compared to the DDPG algorithm in the same dynamic simulation environment designed in the paper, and the total navigation duration has been reduced by 2 h. Compared to the TD3 algorithm, it has improved the success rate by 16.6 % and shortened the time of training navigation by 3 h. The experimental outcome validates the improved performance of the modified TD3 methods, which effectively solves the issues of weak efficiency and low success rate caused by overestimation, sample error, and other issues. Indeed, based on the collision rate, it can be deduced that the majority of unsuccessful rounds are due to collisions with obstacles. Only a small number of rounds are deemed unsuccessful because the exploration step length exceeded the maximum value.

Fig. 12 appears to be a comparative chart showing the trajectories of each selected algorithm's successful journey to the destination. Table 4 shows the time and step size needed to mobile robot navigation to approach the target point within each algorithm

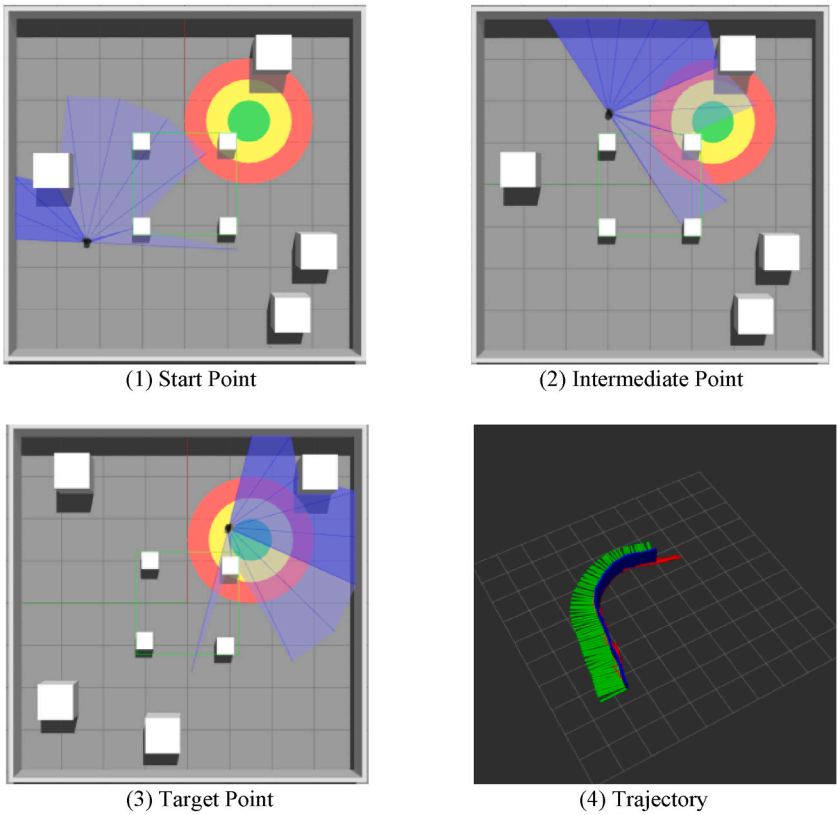



**Fig. 9.** Dynamic environment simulation process and movement trajectory.

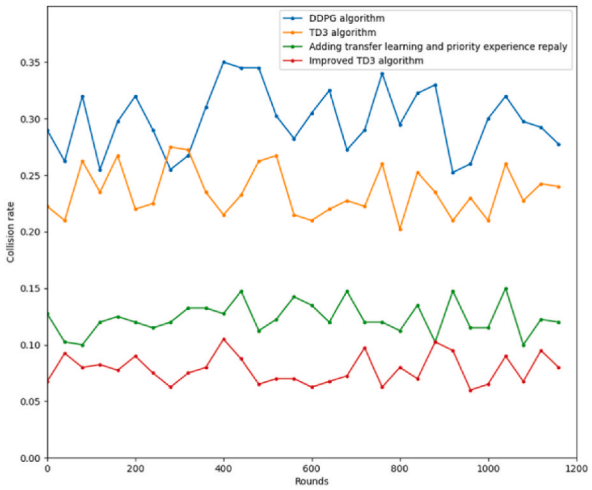



**Fig. 10.** Collision rate of various algorithms.

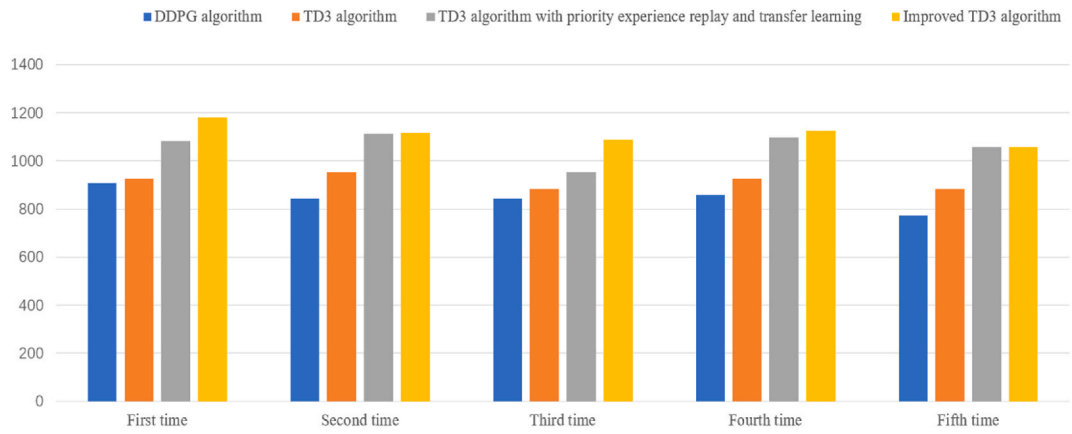



**Fig. 11.** Collision rate of various algorithms.

**Table 3**
Comparison of algorithm performance in dynamic environment.

| | Success rate (%) | Collision rate (%) | Model training duration (h) |
|---|---|---|---|
| DDPG algorithm | 70.4 % | 28.2 % | 10.5 |
| TD3 algorithm | 76.2 % | 22.5 % | 11.5 |
| TD3 algorithm with priority experience replay and transfer learning | 88.4 % | 10.9 % | 9 |
| Improved TD3 algorithm | 92.8 % | 7.1 % | 8.5 |

under test mode. To guarantee the rationality of the consequence, we take the average of the test results from the last 100 rounds of each algorithm for analysis. By analyzing the figures in the table, it is evident that the DDPG algorithm requires approximately 83 s and a step size of approximately 238 to reach the target point in mobile robot path planning in the dynamic environment mentioned in this article. After using the TD3 algorithm, the time was reduced to 71s and the step size was reduced to 215. On the basis of TD3 algorithm, priority experience replay and transfer learning are added, and time is greatly reduced to 66s and the step size is reduced to 203. After further refinement and improvement of the TD3 algorithm, new progress was made in testing, with the lead time reaching 52 s and the step size significantly reduced to 178. Through comparison of test data, the improved TD3 algorithm has been optimized in terms of navigation time and step size in dynamic obstacle environments based on the original environment, further verifying the effectiveness of the algorithm.

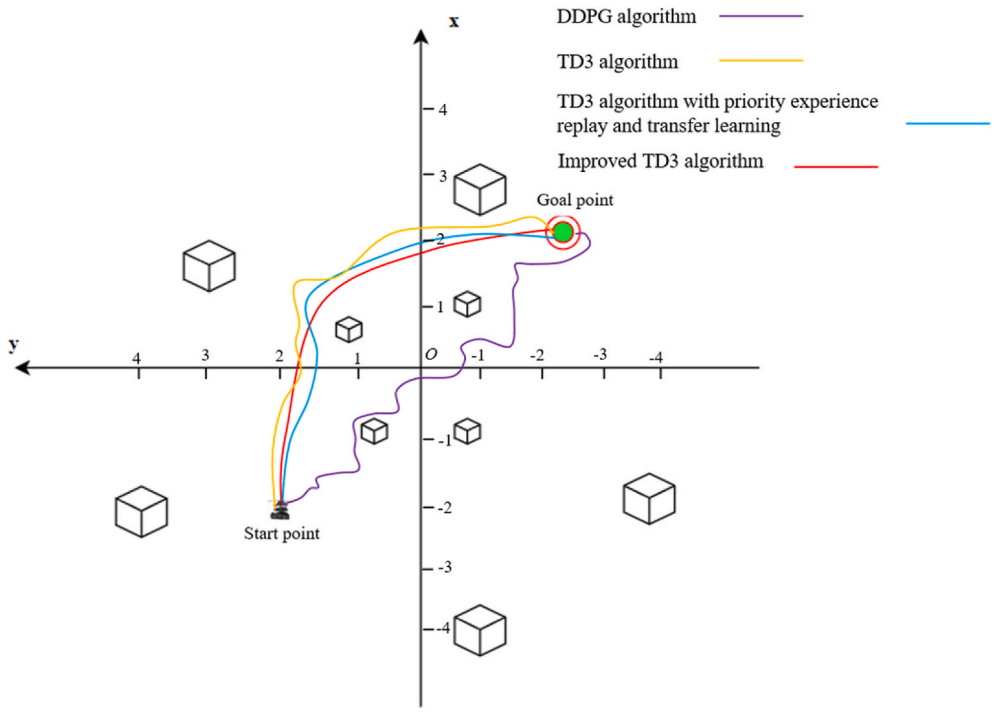



**Fig. 12.** Trajectories of various algorithms.

**Table 4**
Comparison of path effects.

| | Time (s) | Step length (steps) |
|---|---|---|
| DDPG algorithm | 83 | 238 |
| TD3 algorithm | 71 | 215 |
| TD3 algorithm with priority experience replay and transfer learning | 66 | 203 |
| Improved TD3 algorithm | 52 | 178 |

## 5. Experiment on the generalization ability of improved TD3 algorithm in dynamic environment for mobile robot path planning

Generalization ability experiment of the algorithm must be implemented for verifying the adaptability of improved TD3 algorithm, where applying the improved TD3 algorithm to different simulation environments. It proves that the algorithm has strong generalization ability if the algorithm can maintain good navigation performance in different or more complex environments.

### *5.1. Establishment of simulation environment for robot generalization ability experiment in dynamic environment*

To validate the generalization ability of the trained mobile robot model to adapt to new samples, this paper builds a test environment similar to the original training environment but more complex, so as to provide new test samples for the robotic system, and applies the trained model to this test environment after relatively fewer rounds of training optimization to observe whether the robot can realize the function of reaching the target point without collision in this new environment. The new dynamic simulation environment was established as shown in Figs. 13– 15.

The dynamic obstacle simulation environment in Fig. 13 generalization ability experiment is more complex, comparing to the dynamic obstacle environment in chapter 5 of this paper. Similarly, the test environment was built on the Gazebo 3D simulation platform. The simulation area has been expanded to 12 m × 12 m, where, 4 dynamic obstacles were added to reach 12, including 4 cubic obstacles with a side length of 0.3 m in the middle, and 4 cubic obstacles with 1 m side length in middle and outermost layers respectively. In the setup, each obstacle moves along its specified path at a random speed of (0.1–0.5) m/s, and the motion trajectories of each obstacle do not affect each other.

Configuring the mobile robot designed in chapter 2 and the saved neural network models to the launch file of the dynamic environment, including the starting position and target point position of the mobile robot, and finally obtain the complete dynamic environment as shown in Fig. 14. For preventing the serendipity of a single generalization experiment, the dynamic simulation

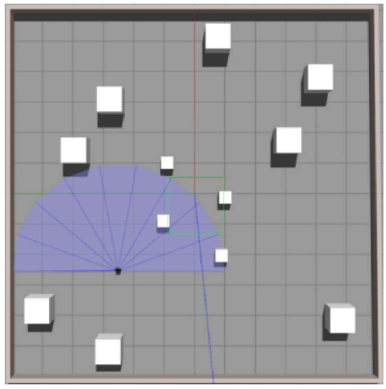


**Fig. 13.** Dynamic simulation environment for generalization ability experiment.

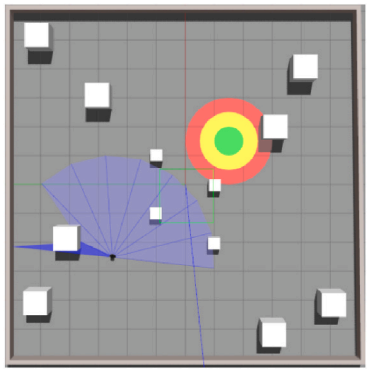


**Fig. 14.** Complete dynamic environment simulation model for generalization ability experiment.

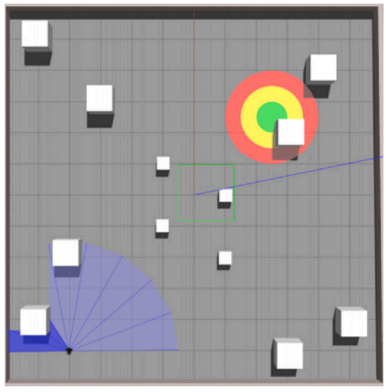


**Fig. 15.** Dynamic environment simulation model of new starting point and target point.

environment of the original generalization experiment was adjusted again by setting the position coordinates of the mobile robot to (−5, 4) and the position coordinates of the center of the target point to (2.5, −2.5), as shown in Fig. 15.

### *5.2. Experimental results of robot generalization ability training and testing in dynamic environment*

Firstly, training the robot in a new dynamic simulation environment. To verify the generalization ability, this training needs to be based on the improved TD3 algorithm trained robots in chapter 5. We will use the weight parameters trained in chapter 5 directly to train the generalization ability in the dynamic environment of the experiment. The essence of this is also a kind of transfer learning, and the training situation can intuitively reflect the generalization ability of the algorithm. After a single transfer learning, the convergence of the reward value is shown in Figs. 16 and 17.

As can be seen from Fig. 16, the mobile robot model trained in Chapter 5 using the improved TD3 algorithm performs well even under more complex dynamic environments. In the first 300 training rounds, the reward value fluctuates greatly between 0 and 2700. After 300 rounds, the algorithm shows a trend of convergence, and the number of rounds reaching around 2700 rewards gradually increases. After 400 rounds of training, the degree of convergence of the algorithm greatly improves, and the reward value can reach around 2700 in most of the training rounds. This suggests that in the later stages of training, the success rate of the algorithm convergence is high, which preliminarily confirms the strong generalization ability of the improved TD3 algorithm. Similarly, Fig. 17 demonstrates a good convergence capability after 400 rounds.

Subsequently, the robot trained using the improved TD3 algorithm underwent 1200 rounds of independent navigation tests in a new dynamic environment. The motion process and trajectory of the robot during navigation are shown in Figs. 18 and 19. The mobile robot starts from the start point Fig. 18 (1), arrives the target point Fig. 18 (2), forms a complete motion trajectory Fig. 18 (3). Then, setting new starting point, the mobile robot starts from the start point Fig. 19 (1), arrives the target point Fig. 19 (2), forms a complete motion trajectory Fig. 19 (3).

From the motion process and trajectory in Fig. 18, it can be seen that in the tests under this complex dynamic environment, the mobile robot can successfully reach the target point in the vast majority of test rounds. Its motion path is roughly to move upwards for a while, then bypass two small dynamic obstacles on the left and upper movement routes, turn right, and find a path to the target point between the movement routes of the small dynamic obstacle on the upper side and the large dynamic obstacle in the middle layer.

Upon conducting a statistical analysis of the 1200 rounds of independent navigation tests on the mobile robot, such as the reward values obtained from each round, the boolean values of whether the destination was reached, as well as the time required for navigation and the step length required for the travel path in the rounds where the destination was successfully reached, the following conclusions can be drawn.

(1) Based on the distribution of the reward values received by the mobile robot in each round, it can be inferred that the total return value for reaching the target point is stable at around 2700.
(2) By summing up the boolean values of whether the destination was reached in the 1200 rounds of navigation tests for the robot trained with the improved TD3 algorithm, we obtain a value of 914. This means that the mobile robot successfully drove to the target point 914 times in this complex generalization capability dynamic simulation environment, within the limited step length and without colliding with obstacles. By calculating the ratio with the 1200 rounds, we can deduce that its success rate is approximately 75.7 %.
(3) In the successful navigation round, the navigation time is approximately 58 s and the step size is 195.

The dynamic simulation environment with new starting point and target point has similar conclusion. The navigation process and experimental data of improving the generalization ability of the TD3 algorithm show that mobile robots can still maintain good navigation performance in more complex environments, further verifying the strong generalization ability of the improved TD3 algorithm in robot path planning problems in dynamic environments.

## 6. Conclusion

An improved TD3 algorithm for path planning of mobile robot in dynamic environments is proposed in the paper. A dynamic delay update strategy is devised to improve TD3 algorithm, and priority experience replay, transfer learning and OU noise are introduced. This algorithm can flexibly change the delay update step size of the Actor Network to adapt to the dynamic changes of Critic updates, effectively improving training efficiency and convergence speed. Then, a dynamic environment was established in the Gazebo simulation environment based on the ROS system, and which was trained and tested, then test results were analyzed, and finally, the

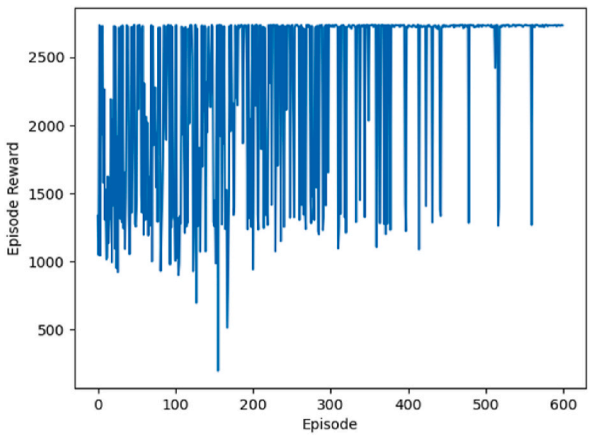



**Fig. 16.** Reward value for generalization ability experiment training.

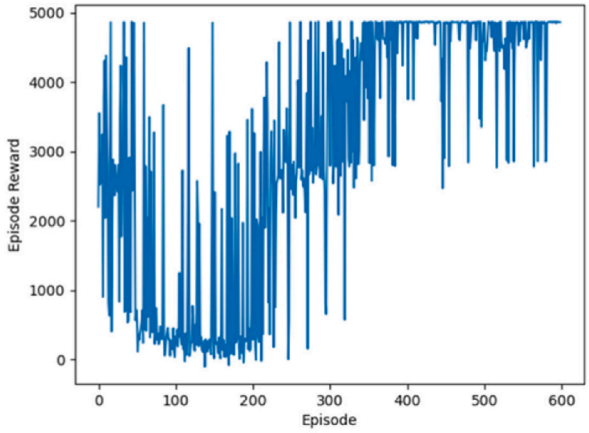



**Fig. 17.** Reward value of new starting point and target point.

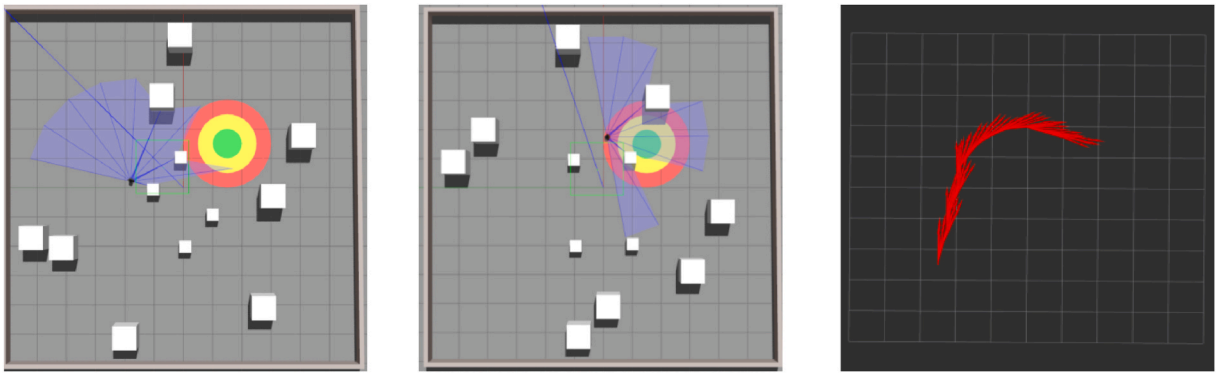



**Fig. 18.** Generalization ability experiment of mobile robot motion process and trajectory.

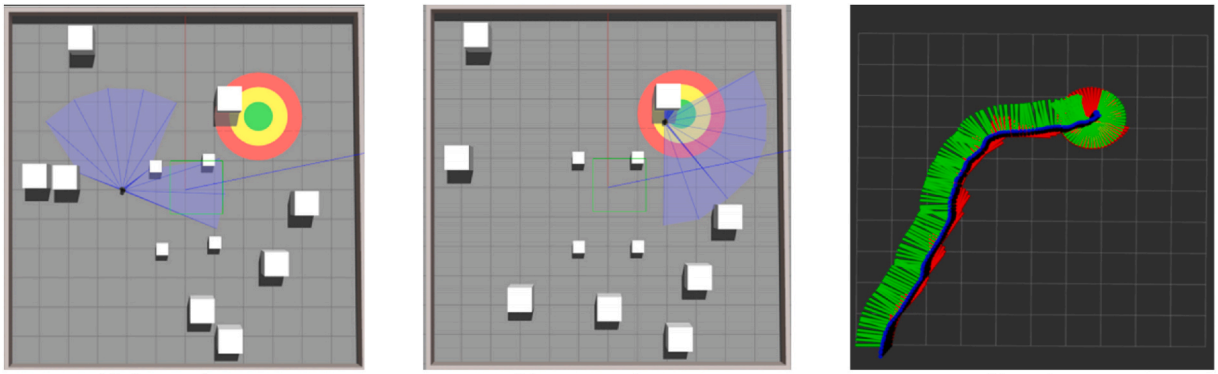



**Fig. 19.** Mobile robot motion process and trajectory with new starting point and target point.

generalization ability of the algorithm was verified through testing experiments. The research results indicate that in dynamic environment the improved TD3 algorithm can enable mobile robots reach the target area quickly and safely. In addition, compared to DDPG and the original TD3 algorithm, the improved TD3 algorithm has superior navigation performance and generalization ability in dynamic environments.

The practicality of mobile robots depends on their autonomous navigation planning capability, obtaining fast, high-quality and collision-free path planning is one of the most fundamental directions for their development research. With the expansion of

application scenarios, the states of environment and the dimensionality of state space is increasingly complex, robots with the ability to both process high-dimensional environmental information and make autonomous decisions to carry out the corresponding planning tasks show particularly great potential. However, real-world application environments are faced with more mobile robot dynamics constraints, dynamic obstacles interference, terrain states, etc. So the use of deep reinforcement learning algorithms to solve planning and control problems for mobile robots has become a trend, whereas with urgent need to solve algorithms' intrinsic problems such as slow training speed, low learning quality and generalized adaptation ability. On the one hand, traditional path planning algorithms can be combined with deep reinforcement learning algorithms to realize the combination of global planning advantages and local obstacle avoidance advantages. On the other hand, the deep reinforcement learning algorithm itself is improved from the direction of reward function, sample efficiency, and environment dynamic complexity. In the dynamic delayed updates method of this paper, taking the segmentation function with the update magnitude as the independent variable as the basis for setting the next update step is not smooth enough, and the strategy can be optimized by adopting the method of rounding the linear function with the update magnitude as the independent variable.

## Data availability statement

Data used in this article will be made available on request.

## CRediT authorship contribution statement

**Peng Li:** Writing – review & editing, Writing – original draft, Supervision, Methodology. **Donghui Chen:** Writing – review & editing, Writing – original draft, Validation, Software. **Yuchen Wang:** Validation, Software, Data curation. **Lanyong Zhang:** Writing – review & editing, Supervision, Funding acquisition. **Shiquan Zhao:** Writing – review & editing, Validation, Methodology.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

This research was supported by State Key Laboratory of Robotics and Systems (HIT) (SKLRS-2023-KF-17), Heilongjiang Provincial Youth Reserve Talent Program (CXRC20231117219), Central Universities Funding (3072024LJ0401).

## References

[1] S. Tian, Y. Li, Y. Kang, et al., Multi-robot path planning in wireless sensor networks based on jump mechanism PSO and safety gap obstacle avoidance, Future Generat. Comput. Syst. 118 (2021) 37–47.
[2] T. Oktay, Harun Celik, I. Turkmen, Maximizing autonomous performance of fixed-wing unmanned aerial vehicle to reduce motion blur in taken images, Proc. IME J. Syst. Control Eng. 232 (2018) 857–868.
[3] S. Enrico, W. Francesco, T. Sandro, et al., Autonomous underwater intervention: experimental results of the MARIS Project, IEEE J. Ocean. Eng. 43 (3) (2018) 620–639.
[4] R. Cupek, M. Drewniak, M. Fojcik, et al., Autonomous guided vehicles for smart industries – the state-of-the-art and research challenges, Computational Science - ICCS 2020 (12141) (2020) 330–343.
[5] L. Parungao, F. Hein, W. Lim, Dijkstra algorithm based intelligent path planning with topological map and wireless communication, ARPN J. Eng. Appl. Sci. 13 (8) (2018) 2753–2763.
[6] S. Erke, D. Bin, N. Yiming, et al., An improved A-star based path planning algorithm for autonomous land vehicles, Int. J. Adv. Rob. Syst. 17 (5) (2020) 591–617.
[7] D. Jonathan, S. Siddhartha, D. Timothy, Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic [C], in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997–3004.
[8] C. Lamini, S. Benhlima, A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, Procedia Comput. Sci. 127 (2018) 180–189.
[9] V. Roberge, M. Tarbouchi, G. Labonté, Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, IEEE Trans. Ind. Inf. 9 (1) (2012) 132–141.
[10] K. Akka, F. Khaber, Mobile robot path planning using an improved ant colony optimization, Int. J. Adv. Rob. Syst. 15 (3) (2018) 1–7.
[11] H. Zhang, L.Z. Liu, H. Xie, et al., Deep learning-based robot vision: high-end tools for smart manufacturing, IEEE Instrum. Meas. Mag. 25 (2) (2022) 27–35.
[12] W. Ning, G. Jiahui, Multi-task dispatch of shared autonomous electric vehicles for Mobility-on-Demand services – combination of deep reinforcement learning and combinatorial optimization method, Heliyon 8 (11) (2022) 11319, 2022.
[13] J.L. Gao, W. Ye, J. Guo, et al., Deep reinforcement learning for indoor mobile robot path planning, Sensors 20 (19) (2020) 5493.
[14] N. Saito, T. Oda, A. Hirata, et al., A movement adjustment method for DQN-based autonomous aerial vehicle, Advances in Intelligent Networking and Collaborative Systems 312 (2022) 136–148.
[15] John Schulman, Sergey Levine, Philipp Moritz, et al., Trust region policy optimization [C], Proceedings of the 32nd International Conference on International Conference on Machine Learning 37 (2015) 1889–1897.
[16] L. Engstrom, A. Ilyas, S. Sabturkar, et al., Implementation matters in deep RL: a case study on PPO and TRPO [C], in: International Conference on Learning Representations, 2020.
[17] T. Haarnoja, A. Zhou, P. Abbeel, et al., Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor [C], Proceedings of the 35th International Conference on Machine Learning 80 (2018) 1861–1870.
[18] C. Nica, M. Oprea, A.C. Stan, On the development of a mobile TurtleBot3 burger multi-robot system for manufacturing environment monitorization[C], Proceedings of Emerging Trends and Technologies on Intelligent Systems 1371 (2021) 323–337. Singapore.
[19] M. Missura, M. Bennewitz, Predictive collision avoidance for the dynamic Window approach [C], IEEE International Conference on Robotics and Automation ICRA (2019) 8620–8626.

[20] F. Bounini, D. Gingras, H. Pollart, et al., Modified artificial potential field method for online path planning applications [C], in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 180–185.
[21] R.R. Hete, K.M. Sanjoy, D. Ritesh, et al., Analysis of DFIG-STATCOM P2P control action using simulated annealing technique, Heliyon 8 (3) (2022) e09008, 9008.
[22] I. Sung, B. Choi, P. Nielsen, On the training of a neural network for online path planning with offline path planning algorithms, Int. J. Inf. Manag. 57 (2021) 102142.
[23] V.H. Hado, G. Arthur, S. David, Deep reinforcement learning with double Q-Learning [C], in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16), 2016, pp. 2094–2100.
[24] S. Han, W.B. Zhou, J.Y. Lu, et al., NROWAN-DQN: a stable noisy network with noise reduction and online weight adjustment for exploration, Expert Syst. Appl. 203 (2022) 117343.
[25] J. Peters, S. Vijayakumar, S. Schaal, Natural actor-critic [C]. 16th European conference on machine learning (ECML)/9th European conference on principles and practice of knowledge discovery in databases (PKDD) 3720 (2005) 280–291.
[26] M. Volodymyr, P.B. Adrià, M. Mehdi, et al., Asynchronous methods for deep reinforcement learning [C], Proceedings of the 33rd International Conference on International Conference on Machine Learning 48 (2016) 1928–1937.
[27] J. Zheng, S. Mao, Z. Wu, P. Kong, et al., Improved path planning for indoor patrol robot based on deep reinforcement learning, Symmetry 14 (1) (2022) 132.
[28] M. Bodi, Z.B. Liu, Q.Q. Dang, et al., Deep reinforcement learning of UAV tracking control under wind disturbances environments, IEEE Trans. Instrum. Meas. 72 (2023) 1–13.
[29] Y. Chen, F. Ying, X. Li, et al., Deep reinforcement learning in maximum entropy framework with automatic adjustment of mixed temperature parameters for path planning [C], in: 2023 7th International Conference on Robotics, Control and Automation (ICRCA), 2023, pp. 78–82, 10(1109).
[30] J. Fan, Z. Wang, J. Ren, et al., UAV Online Path Planning Technology Based on Deep Reinforcement Learning [C], 2020 Chinese Automation Congress (CAC), 2020, pp. 5382–5386.
[31] J. Gao, W. Ye, J. Guo, et al., Deep reinforcement learning for indoor mobile robot path planning, Sensors 20 (19) (2020) 5493.
[32] M. Kim, D.-K. Han, J.-H. Park, et al., Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay, Appl. Sci. 10 (2) (2020) 575.
[33] J. Filar, K. Vrieze, Competitive Markov Decision Processes[M], Springer Science & Business Media, 1997.
[34] X. Tao, A.S. Hafid, DeepSensing: a novel mobile crowdsensing framework with double deep Q-network and prioritized experience replay, IEEE Internet Things J. 7 (12) (2020) 11547–11558.
[35] F. Zhuang, Z. Qi, K. Duan, et al., A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2020) 43–76.