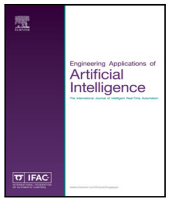
Engineering Applications of Artificial Intelligence 123 (2023) 106197

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Event-triggered reconfigurable reinforcement learning motion-planning approach for mobile robot in unknown dynamic environments

Huihui Sun [a,b], Changchun Zhang [a,c,d], Chunhe Hu [a,c,d,*], Junguo Zhang [a,c,d,**]

[a] School of Technology, Beijing Forestry University, Beijing, 100083, China
[b] College of mechanical and electrical engineering, North China institute of science and technology, Langfang, 065201, China
[c] Research Center for Biodiversity Intelligent Monitoring, Beijing Forestry University, Beijing 100083, China
[d] State Key Laboratory of Efficient Production of Forest Resources

ARTICLE INFO

Keywords:
Mobile robot
Reinforcement learning
Actor–critic
Reconfigurable structure
Sample pretreatment
Event-triggered

ABSTRACT

Deep reinforcement learning (DRL) is an essential technique for autonomous motion planning of mobile robots in dynamic and uncertain environments. In attempting to acquire a satisfactory DRL-based motion planning strategy, the mobile robots encountered several difficulties, including poor convergence, insufficient sample information, and low learning efficiency. These problems not only consume plenty of training time, but also bring a negative impact on motion planning performance. One promising research direction is to provide a more effective network framework for DRL-based policies. Along this line of thinking, our paper presents a novel DRL-based motion planning approach called Reconfigurable Structure of Deep Deterministic Policy Gradient (RS-DDPG) for mobile robots. To account for the poor convergence, the proposed approach first introduces an event-triggered reconfigurable actor–critic network framework for motion policy that adaptively changes its network structure to suppress the overestimation of action value. Then, the time convergence of the motion policy can be enhanced based on the value actions with minor valuation deviation. Afterwards, an adaptive reward mechanism is designed for reconfigurable networks to compensate for the lack of sample information. To deal with the problem of low learning efficiency, we developed a sample pretreatment method for the experience samples, which employs three novel techniques to improve the sample utilization, including a double experience memory buffer, a variable proportional sampling principle, and a similarity judgment mechanism. In extensive experiments, the proposed method outperforms the compared approaches.

1. Introduction

Based on their rich autonomous motion capabilities, mobile robots have achieved remarkable success in exploration and transportation (Samsonov et al., 2022; Qian et al., 2020). The technology is necessary for mobile robots to face complex task scenarios with excellent motion planning, which has attracted extensive attentions from researchers. Existing motion planning methods (Zhang et al., 2018; Bayat et al., 2018) have achieved relatively high performance in familiar task scenarios with sufficient prior knowledge. In a dynamic indoor environment, He et al. (2022) adopted a polystage humanlike motion planning approach to ensure predictability and flexibility. When encountering scattered obstacles, Rostami et al. (2019) proposed an optimal method based on potential field theory to obtain a robust motion planning solution. However, these methods can be severely weakened when encountering unpredictable and unmapped scenarios (Zi et al., 2015). Therefore, it remains an open question how to effectively improve the adaptability and self-learning ability of motion planning strategies in unknown environments.

Deep Reinforcement Learning (DRL) provides a set of data-driven tools for mobile robots to address such problems (Fahad et al., 2020; Nguyen et al., 2020; Konar et al., 2021; Kober et al., 2013), and it provides a natural way to design end-to-end mappings from sensed data to execution. In addition, they have been shown to handle complex complications such as communication interference and unpredictable variable scenarios. Generally, DRL-based approaches consist of a deep learning (DL) module and a reinforcement learning (RL) module (Nguyen et al., 2020; Jia and Ma, 2021). DL modules have been applied to deal with specific problems not typically addressed by the computer vision community (Sünderhauf et al., 2018) and to provide guidelines for generating optimal training strategies and model structures for robots (Károly et al., 2020; Morales et al., 2021). RL modules are responsible for developing and optimizing motion strategies via trial-and-error interaction with the environment. This process rarely depends on prior knowledge and scenario modeling (Mnih et al., 2015; de Jesus et al., 2021a). Based on the trained model, the DRL method extracts abstract representations directly from the

* Correspondence to: No. 35, Tsinghua Esatern Road, Haidian District, Beijing, 100083, China.
** Corresponding author at: School of Technology, Beijing Forestry University, Beijing, 100083, China.
E-mail addresses: huchunhe@bjfu.edu.cn (C. Hu), zhangjunguo@bjfu.edu.cn (J. Zhang).

https://doi.org/10.1016/j.engappai.2023.106197
Received 10 June 2022; Received in revised form 23 February 2023; Accepted 21 March 2023
Available online xxxx
0952-1976/

high-dimensional sensor data of the mobile robot and matches the action space with specific state parameters (including coordinates, displacement, and orientation), which will greatly reduce the cumulative planning error and improve the self-adaptive capability of the mobile robot. With these advantages, the DRL method has been considered as an efficient motion planning solution for most unknown and complex environments in the last decade (Huang et al., 2021; Samsani and Muhammad, 2021; Li et al., 2021b; Lei et al., 2022).

According to the updating methods of motion policy, DRL-based approaches can be divided into a value-based algorithms and a policy-based algorithms. However, the application of value-based algorithms is severely limited in the field of robot motion planning, because value-based methods can only output discrete action space vectors, which is not suitable for robots with continuous action spaces. Policy-based methods can solve the problem, but they face the problem of poor convergence. To address this obstacle, the Actor–Critic network framework for RL is proposed by combining the strengths of value-based and policy-based methods. Zhu et al. (2021), Yoo et al. (2021). Actors are considered as policy networks, while critics are evaluator networks. Hybrid frameworks can be effective in obtaining motion strategies for mobile robots. These advantages have inspired many interesting methods. Among the various DRL approaches to Actor–Critic framework, the deep deterministic policy gradient (DDPG) has recently attracted attention (Li et al., 2021a; de Jesus et al., 2021a; Shen and Xu, 2019). DDPG uses the Actor–Critic network framework and deterministic policy gradient technique to improve the time convergence performs relatively better in continuous motion space. It also introduces a target network and an experience memory buffer in the Actor–Critic framework, whose core idea is derived from double deep Q network algorithm (Pan et al., 2018). Specifically, DDPG typically plays an additional target neural network to restrain the overestimation bias of action values and adopts a memory buffer technique to break the correlation of experience samples that can update the policy network in off-line mode (Xie et al., 2020). This training mode can reduce the consumption of motion exploration and improve the training efficiency of the network model (Guo et al., 2021).

Although DDPG methods have achieved significant results, they still suffer from two bottlenecks: First, despite the target network is well trained to suppress the bias of action values, the motion strategy is still unreliable in face of complex navigation assignments. Secondly, the training process is always expensive as the samples are not utilized efficiently enough. These problems have been noticed by several researchers. For instance, Zhou et al. (2021), Fujimoto et al. (2018) proposed a double Q-learning network to estimate action values by selecting the minimum action value. Then, Schaul et al. (2015) designed an empirical framework for prioritizing memory buffers to distinguish the importance of training samples by calculating the Temporal-Difference. In order to reduce training time, Pfeiffer et al. (2018) combined expert argumentation and reinforcement learning (RL) for pre-training and achieved at least five times more efficient training and the same performance. Although these solutions have achieved some results, the sparse rewards and insufficient feedback in the absence of maps further increase the difficulty of strategy optimization. To resolve this problem, curiosity-driven (Lei et al., 2022; Chen et al., 2021) methods incorporates additional rewards into Actor–Critic network framework, which facilitates the exploration of enhanced movement strategies. In addition, a double path structure is deployed to generate supplementary reward information in an unstructured environment, which makes action value assessment more efficacious (Mirowski et al., 2018).

These improved DRL-based approaches alleviate the above problems to some extent, but the problems of inefficient learning and inflexible network structure still exist. To further work along this direction, a novel event-triggered reconfigurable structure based on deep deterministic policy gradient (RS-DDPG) is proposed. Compared with the rigid network structure of DDPG, RS-DDPG adopts a more flexible reconfigurable Actor–Critic framework, which can adaptively transform the critic network structure based on the current valuation deviation triggered by events. The event-triggered reconfigurable structure can improve training efficiency and enhance time convergence and robustness. In addition, instead of a single memory buffer in DDPG, RS-DDPG introduces an auxiliary experience memory buffer to optimize mechanism of the sample set. With the support of the double experience memory buffer, the sample utilization is improved via developing a sample pretreatment mechanism. Finally, to compensate for the lack of reward information in the current DRL-based method, an adaptive reward mechanism based on the state potential function is designed, and the fruitless motion exploration of the mobile robot is significantly reduced. The comparison experiments show that the RS-DDPG gives good results over the compared methods.

In summary, the main contributions of this paper are as follows:

- Firstly, a novel event-triggered reconfigurable structure based on deep deterministic policy gradient (RS-DDPG) is proposed for mobile robot motion planning. Compared to the rigid network structure in current DRL-based methods, the reconfigurable actor–critic can adaptively transform its structure to suppress the overestimation of action values and improve the time convergence of the motion policy according to the optimal valuation deviation of action values;
- Then, instead of the single memory buffer in previous DRL-based methods, we develop a sample pretreatment mechanism for the experience samples, which employs three novel techniques to improve the sample utilization, including the double experience memory buffer, the variable proportional sampling principle, and the similarity judgment mechanism;
- To address the problem of insufficient sample information in the current method, an adaptive reward mechanism is designed for RS-DDPG; the mechanism constructs an integrated reward based on three different feedback signals and employs a state potential reward to compensate for the insufficient sample information;
- Extensive experiments in both simulated and physical scenarios have been carried out, and the motion planning performance of the proposed approach is shown to outperform the comparison approaches.

The rest of this paper is organized as follows. Section 2 presents the work related to the learning-based motion planning method. The statement of robot motion planning is presented in Section 3. Section 4 describes the design process of the RS-DDPG approach. Then, we evaluate the performance of our method with the existing algorithms in Section 5. Section 6 concludes the paper.

2. Related work

2.1. Motion planning based on deep-learning

Deep Learning (DL) has a great potential for solving robot motion planning problems using raw observation information. In most previous work, a deep neural network was trained as a controller to perform navigation mission tasks by extracting identifiable feature representations from high dimensional observation states (Justesen et al., 2019; Giusti et al., 2015). For example, a hierarchical hybrid multi-layer neural network was used to construct a decision-making system to explore unknown targets of a mobile robot. The decision system is trained based on expert demonstrations and generates a target-oriented motion planning model for the robot platform (Pfeiffer et al., 2017). Target-oriented models are capable of generating a series of action commands for a robot based on visual RGB-D sensors or 2D-laser data (Do et al., 2018; Tai et al., 2017). Further, existing motion planning tasks typically embed target detection techniques into deep convolutional neural networks. Garcia et al. (2019) focused on transforming video streams into specific features by target detectors, and generating policy models based on bounding boxes of identified objects. Networks with

target detectors are particularly well-suited for indoor navigation and exploration missions. Although deep neural networks can provide end-to-end motion strategies for mobile robots, they require large amounts of labeled observations and lack self-learning capabilities in unknown task environments. Therefore, we focus on deep reinforcement learning methods with powerful decision-making abilities and autonomous learning capabilities.

2.2. Motion planning based on deep reinforcement learning

Deep reinforcement learning (Mnih et al., 2015; Yu et al., 2020; Shi et al., 2019) methods typically employ a critic network to generate current actions and optimize motion policies according to evaluation values (Pan et al., 2018; Shen and Xu, 2019). Recent studies have shown that trained control models can easily outperform professional manual operators in the field of Atari games and intelligent robots. As a result, these approaches with powerful self-learning capabilities have been rapidly developed to deal with motion planning problems in unmapped environments (Shi et al., 2019).

In general, there are two solutions based on deep reinforcement learning. One is the value-based method, and the other is the policy-based method. For instance, deep Q networks (DQN) (Pan et al., 2018; Li et al., 2022; Pfeiffer et al., 2017) are one of the most widely used methods in the field of value-based motion planning. Mobile robots can constantly optimize their motion planning strategies by finding the maximum action values in the action-state space via the DQN method. However, the Q values are always overestimated and prone to unreliable strategies because the single-critique network is not accurate enough. To solve this bottleneck, a double DQN (Li et al., 2022) architecture with a duple critic network is proposed to curb the overestimation of action values, and this method also improves the performance of motion planning strategies to some extent. Unfortunately, the DQN method can only be applied to mobile robots with discrete action spaces, while most robotic systems require continuous action policies to meet the demand for precise motions. To address the aforementioned problem, a policy-based method has been proposed to learn continuous motion policies consisting of normalized probability distributions. Along this line, combining value-based and policy-based approaches, many advanced variants of actor–critic have also been proposed, such as the deep deterministic policy gradient (DDPG) (Xie et al., 2020), the proximal policy optimization (PPO) algorithm (Vanvuchelen et al., 2020), the soft actor critic (SAC) algorithm (Wang et al., 2020), and the twin delayed deep deterministic policy gradient (TD3) algorithm (Zhou et al., 2021). PPO is an online update strategy, which is insensitive to hyper parameters and step size. The performance of SAC algorithm and TD3 are based on updated versions of DDPG but their network structures are more efficient. Compared to DDPG, the SAC algorithm is an off-line reinforcement learning method based on maximum entropy and performs better in the fields of autonomous obstacle avoidance for mobile robots and walking skill learning for foot robots (Hwangbo et al., 2019). (de Jesus et al., 2021b) studied the application of actor–critic algorithms for robot navigation in two simulated environments, and the comparisons proved the superior performance of the SAC algorithm. The TD3 algorithm can control a bipedal robot to learn to walk and acquire a human-like movement strategy (Cui et al., 2021; Khoi et al., 2021). A comprehensive attempt proves the effectiveness of the DRL-based method for motion planning tasks, particularly for complex and unknown scenarios.

Unlike previous studies, we extend an event-triggered reconfigurable RL structure in the motion planning tasks, and carefully design auxiliary experience memory buffers and sample pretreatment mechanism to improve sample utilization. We employ state potentials to build an adaptive reward signal function that strengthens the self-exploration capability of motion planning.

Table 1
Parameters for Markov decision process.

Item	DRL	Robot
Agent	An object	Mobile Robot
State, S	Observation of agent	$S_t = (x_t, y_t, \sigma_t, D, d, \alpha_t)$
Action, a	Output of agent	$a_t\,(v, w)$
Probability, P	State transition distribution	$P\left(A_t = a_t \mid S_t = s_t\right)$
Reward, r	Immediate feedback	$r_t\left(s_t, a_t\right)$
Discount factor, γ	Discounts for single step	Constant, $\gamma \in [0, 1]$

3. Statement of the problem

3.1. Motion description

In this section, we use a mobile robot to perform obstacle avoidance and navigation tasks in an unpredictable environment. As shown in Fig. 1, the robot is driven by a two-wheel differential mechanism. The robots with blue and orange colors in Fig. 1 represent the same robot at different instants.

The global coordinate system is (O, X, Y), and the local coordinate of the robot is (o', x', y'). The linear velocity v of the mobile robot is denoted as: $v = \left(v_l + v_r\right)/2$, where v_l and v_r denote the velocities of the left and right wheels, respectively. The location coordinate of the goal is $\left(x_g, y_g\right)$. The angular velocity ω of the robot is denoted as: $\omega = \left(v_l - v_r\right)/L$, where L denotes the wheel pitch of the robot. Body coordination is described as $W_t = \left[x_t, y_t, \sigma_t\right]^T$ at time instant t, where x_t and y_t represent the position coordinates, and σ_t denotes the heading angle. α_t is the deflection angle of the target position. O_c is the ICR (instantaneous center of rotation) of the mobile robot. R_0 is the instantaneous radius of rotation. As shown in Fig. 1, the angular velocity ω of the robot can be calculated by Eq. (1):

$$\omega = \frac{v_l}{R_0 - \frac{L}{2}} = \frac{v}{R_0} = \frac{v_r}{R_0 + \frac{L}{2}} \tag{1}$$

Then, the positive kinematics equation of the wheeled mobile robot (WMR) can be described by the following equation (Deng et al., 2010):

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ -1/L & 1/L \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix} \tag{2}$$

According to Eq. (2), the inverse kinematics equation of the WMR can be expressed as (Sekiguchi et al., 2006):

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} 1 & -L/2 \\ 1 & L/2 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{3}$$

In the simulator, the robot has a two-wheel differential structure with the driving parameters of the left and right wheel velocities: $\left(v_l, v_r\right)$. However, the output of the proposed DRL-based strategy are the linear and angular velocities of the robot: (v, ω), which cannot drive the robot directly. Therefore, it is necessary to calculate the $\left(v_l, v_r\right)$ of the robot according to Eq. (3).

3.2. MDP and DRL

To handle the motion planning task, a Markov Decision Process (MDP) was formulated to describe the trial-and-error interactions of the robot. The MDP contains five elements, which can be represented as (S, A, P, R, γ). Where, S denotes the input state observation and A denotes the action of the robot. P represents the state transition probability distribution, and R is the immediate reward. $\gamma \in [0, 1]$ is the discount factor. A detailed description of the MDP elements is listed in Table 1. Where, D represents the sensor data information and d denotes the distance to the target position.

The target of MDP is to learn a motion planning strategy to guide the robot to reach a specific destination and to avoid various obstacles. Firstly, the robot obtains the current state s_t by the state sensor and

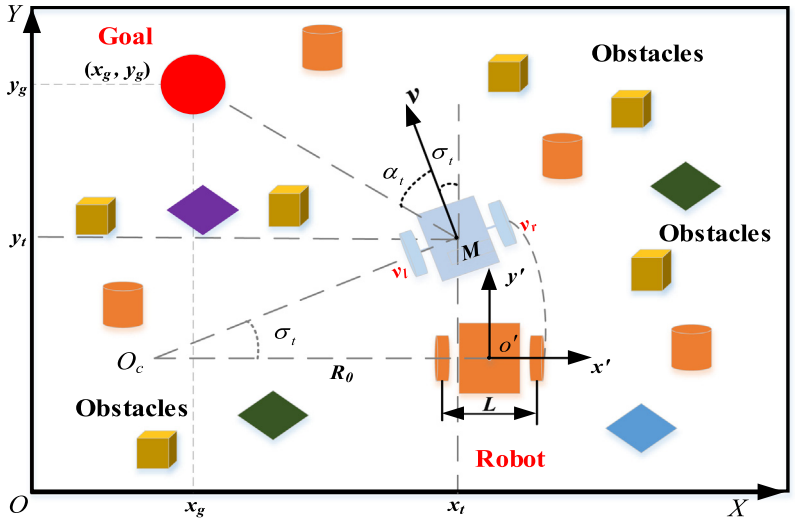


Fig. 1. Motion diagram of the pioneer mobile robot.

selects the motion action a_t based on the state transition probability $P\left(A_t = a_t \mid S_t = s_t\right)$. Then, the robot executes the selected action a_t and reaches the next state s_{t+1}. At the same time, immediate rewards $r_t(s_t, a_t)$ can be obtained based on the reward function. Finally, the motion parameters (s_t, a_t, r_t, s_{t+1}) are stored as experience samples to a memory buffer. During the strategy optimization phase, these samples are randomly sampled to form a data episode that is used to train the strategy network at the next moment.

However, since the state transition probability distribution P is unpredictable, the training process of policy network is not always efficient. To cope with this issue, we attempt to find a motion policy π_w to replace the probability distribution P. The motion policy π_w is optimized by maximizing the expected reward. Therefore, an action value function is constructed to calculate the expected reward, which can be expressed as:

$$Q_\pi\left(s_t, a_t\right) = \mathrm{E}_\pi\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid S_t = s_t, A_t = a_t\right) \tag{4}$$

where $r_{t+1}, r_{t+2} \cdots$ are the immediate future rewards, $Q_\pi\left(s_t, a_t\right)$ denotes the action value function.

Then, the optimal motion planning policy π_w can be obtained by maximizing the action values $Q_\pi\left(s_t, a_t\right)$:

$$\pi_w\left(a_t \mid s_t\right) = \arg\max \mathrm{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t, a_t\right] = \arg\max\left[Q\left(s_t, a_t\right)\right] \tag{5}$$

where, w is the weight of the policy network.

In the process of strategy optimization, the motion policy π_w is formulated by using a neural networks whose weight parameters w can be updated by an episode of date selected from an experience memory buffer. To optimize the policy network π_w, an objective function $J\left(\pi_w\right)$ is established as:

$$J\left(\pi_w\right) = \mathrm{E}_{s\sim\rho_\pi}\left[Q\left(s_t, a_t \mid \theta\right)\Big|_{s=s_t, a=\pi\left(s_t \mid w\right)}\right] \tag{6}$$

where ρ_π represents the state distribution of the policy, and θ is the weight of action value network.

The target of optimization problem is to maximize the objective function $J\left(\pi_w\right)$, which can be obtained by calculating the gradient of $J\left(\pi_w\right)$.

4. Methodology

In this section, we show a specific instruction of the proposed event-triggered actor–critic reconfigurable structure of the deep deterministic policy gradient method (RS-DDPG). As shown in Fig. 2, our proposed RS-DDPG method consists of three main components: an event-triggered reconfigurable network, a sample pretreatment mechanism, and an adaptive reward mechanism.

Firstly, we propose an event-triggered reconfigurable network framework to replace the single actor–critic network module of traditional DRL-based method, which can adaptively transform the network structure according to the current valuation deviation to improve time convergence and robustness.

Then, we construct an auxiliary experience memory buffer to optimize how the experience sample is stored and develop an efficient sample pretreatment mechanism to improve the sample utilization. The sample pretreatment mechanism has employed the principles of variable proportional sampling and similarity judgment.

Based on this, we design an effective adaptive reward mechanism for the problem of sparse rewards in the unmapped scenario to provide more accurate and adequate rewards for the robot motion planning tasks, thus compensating for the deficiency of sparse rewards and enhancing the self-exploration capability.

4.1. Event-triggered reconfigurable actor–critic network

In our approach, the DRL-based motion network is represented as an event-triggered reconfigurable framework, which contains a pair of actor networks and multiple pairs of variable critic networks. In contrast to previous works, we adopt a flexible structure with an event-triggered critic network instead of the single critic network. As shown in Fig. 3, the event-triggered reconfigurable framework consists of n sets of eval-critic networks and n sets of target-critic networks. Depending on different task scenarios, n can be updated adaptively according to the parameters of event-triggered.

In particular, the eval-critic network is used to estimate the current action value $Q_{\theta_i}\left(s_t, a_t\right)$, while the target-critic network is responsible for calculating the target action value $Q_{\theta_i'}\left(s_{t+1}, a_{t+1}\right)$ for the next moment. Their weight parameters are θ_i and θ_i', respectively. On the other hand, the actor network is responsible for generating an action policy, which consists of an eval-policy π_w and a target policy $\pi_{w'}$. In the actor network, the Rectified Linear Unit (ReLU) is considered as the activation function, while the Adam is considered as the optimizer.

As shown in Fig. 2, actions a_t can be generated based on the eval-policy π_ω and the current state s_t. In the early stage of training process, the policy is not effective enough and the actions it generates may be low-reward. Therefore, to improve the exploratory ability of the policy,

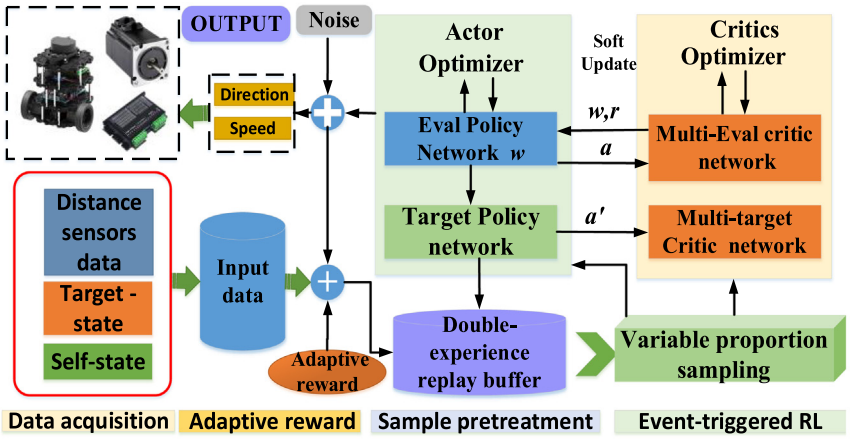


Fig. 2. Motion planning network architecture of RS-DDPG.

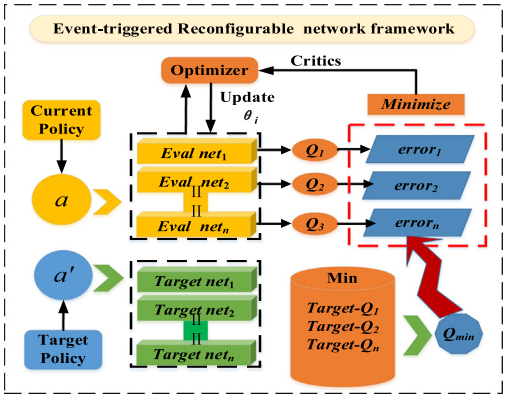


Fig. 3. Reconfigurable actor–critic network framework.

we added random noise N_o to make the agent strive to explore more beneficial action policies.

$$a_t = \pi_w\left(s_t\right) + N_O \tag{7}$$

where N_O is random noise conforming to the Gaussian distribution. The mean of N_O is 0, and the standard deviation N_O is $V_a \in [1,3]$. As the training process increases, the policy model becomes efficient, and it can obtain high reward on the action a_t. At this point, the effect of random noise N_O needs to be reduced. Therefore, we will continue to attenuate the standard deviation V_a of the noise. That is, $V_a' = \max(0.995^{\bar{n}} V_a, 0.01)$. $\bar{n}$ is the frequency of training episode. When $V_a' = 0.01$, the executed action will be slightly affected by the noise N_O. The minimum standard deviation $V_a' = 0.01$ allows the agent to maintain a certain degree of exploration throughout the training phase.

Then, the mobile robot executes the optimal action a_t in the task environment and reaches the next state s_{t+1}. The target action a_{t+1} for the next moment can be generated according to the target policy $\pi_{w'}$:

$$a_{t+1} = \pi_{w'}\left(s_{t+1}\right) \tag{8}$$

Since the target action a_{t+1} is only used to calculate the target action value $Q_{\theta_i'}\left(s_{t+1}, a_{t+1}\right)$, it will not be actually executed by the robot. Therefore, there is no need to add another random noise to a_{t+1}.

In our proposed reconfigurable actor–critic framework, a multiple-critic network is constructed to solve the problem of overestimating action values. Based on the reconfigurable framework, we invariably select the minimum action value from the n episodes of action values. That is:

$$\begin{cases} \overline{Q}_\theta\left(s_t, \pi_{w'}\left(s_t\right)\right) = \min_{i=1,2\cdots n} Q_{\theta_i}\left(s_t, \pi_{w'}\left(s_t\right)\right) \\ \overline{Q}_{\theta'}\left(s_{t+1}, \pi_{w'}\left(s_{t+1}\right)\right) = \min_{i=1,2\cdots n} Q_{\theta_i'}\left(s_{t+1}, \pi_{w'}\left(s_{t+1}\right)\right) \end{cases} \tag{9}$$

According to Eq. (9), the minimum target action value $y_{\min}$ can be denoted as:

$$y_{\min} = r_t + \gamma \overline{Q}_{\theta_i'}\left(s_{t+1}, \pi_{w'}\left(s_{t+1}\right)\right) \tag{10}$$

In addition, the Temporal-Difference error (TD-error) between the target and the current action values can be calculated:

$$\delta_i = y_{\min} - \overline{Q}_{\theta_i}\left(s_t, \pi_w\left(s_t\right)\right) \tag{11}$$

The TD-error δ_i represents the deviation of the estimated action value. Based on this deviation, it can be determined whether the Q value is severely overestimated. An overestimated Q value will negatively affect the convergence of the motion strategy. Therefore, the structure of critic network needs to be updated automatically to suppress the overestimation bias. The proposed RS-DDPG method provides an event-triggered mechanism to transform the network structure. Specifically, the triggering event is defined as follows: $\Delta = \frac{1}{n}\sum_n \left|\delta_i / y_{\min}\right|$. According to Eqs. (10) and (11), we can get $0 < \left|\delta_i / y_{\min}\right| < 1$. Therefore, the range of Δ is: $\Delta \in [0,1]$.

Then, we define the threshold of event-triggering as κ, which can be set in advance according to the tested environment. In our experiments, we define $\kappa = 0.5$. If $\Delta \geq \kappa$, the events of the transformed network will be triggered. This indicates that the estimation deviation exceeds 50%, and the network structure will be adaptively transformed. Then, the number n of auxiliary critic networks will be increased to restrain the overestimated Q value. That is: $n' = n + 1$.

On the other hand, the actor network will optimize its weight parameters based on the current action values $Q_{\theta_i}\left(s_t, \pi_{w'}\left(s_t\right)\right)$, and its objective function can be reconstructed based on the minimized current action values:

$$J\left(\pi_w\right) = \int_S \rho_\pi(s) \overline{Q}_\theta\left(s_t, \pi_{w'}\left(s_t\right)\right) ds = E_{s\sim\rho_\pi}\left[\overline{Q}_\theta\left(s_t, \pi_{w'}\left(s_t\right)\right)\right] \tag{12}$$

Finally, to update the policy network weights w, we use stochastic gradient ascent (SGD) to calculate the policy gradient of the objective function:

$$\nabla_w J(w) = \frac{1}{m}\sum_{j=1}^{m}[\nabla_a \overline{Q}_\theta^j(s_t^j, \pi_{w'}(s_t^j))\big|_{a=\pi_{w'}(s_t^j)} \nabla_\theta \pi_{w'}(s_t^j)] \tag{13}$$

where m denotes the batch size of the selected samples.

To sum up, the weights of all target networks, including the target Actor network w' and target Critic networks θ_i' are updated in a soft update mode based on eval-networks :

$$\begin{cases} w' \leftarrow \tau w + (1-\tau) w' \\ \theta_i' \leftarrow \tau \theta_i + (1-\tau)\theta_i' \end{cases} \tag{14}$$

4.2. Adaptive reward mechanism

Recently studies have shown that inadequate reward signals have a significant negative impact on the convergence of motion strategies. To mitigate this influence, we innovatively designed an adaptive reward mechanism based on the state potential function, which provides more accurate reward feedback information for the current action a_t and status s_t. In summary, the adaptive reward function can be written as follows:

$$R\left(s_t, a_t, s_{t+1}\right) = R_T\left(s_{t+1}\right) + R_C\left(a_t\right) + R_F\left(a_t, s_{t+1}\right) \tag{15}$$

where $R_T\left(s_{t+1}\right)$ indicates the termination reward for each episode, $R_C\left(a_t\right)$ is the action smooth reward, and $R_F\left(a_t, s_{t+1}\right)$ denotes the adaptive reward based on the state potential function.

Specifically, the state potential reward R_F is calculated by using the potential reward for the current state and the potential reward for the state deviation.

$$R_F\left(s_t, s_{t+1}\right) = \varphi\left(s_t\right) r_g + \left(\varphi\left(s_t\right) - \lambda\varphi\left(s_{t+1}\right)\right) r_j \tag{16}$$

where $\varphi\left(s_t\right)$ is the potential function of the current state, and $\varphi\left(s_{t+1}\right)$ denotes the potential function of the target state at the next moment; $\left(\varphi\left(s_t\right) - \lambda\varphi\left(s_{t+1}\right)\right) r_j$ is the potential reward of state deviation, and r_g and r_j are the reward coefficients. $\lambda \in [0, 1]$ indicates the decay factor of the state potential reward.

Moreover, the current state potential function $\varphi\left(s_t\right)$ can be calculated based on the current state parameters:

$$\varphi\left(s_t\right) = \frac{e_1}{1 + \xi_1 d_t^2} + \frac{e_2}{1 + \xi_2 \alpha_t^2} - \sum_o \frac{e_3}{1 + \xi_3 d_o^2} \tag{17}$$

where d_t indicates the distance to the target, and d_o represents the distance to the obstacle; α_t denotes the heading angle $\alpha_t = \sqrt{\left[\arctan(y_t - y_0 / x_t - x_0) - \sigma\right]^2}$; e_1, e_2, e_3 denote the potential constant, and ξ_1, ξ_2, ξ_3 are the state constants.

The current state potential function $\varphi\left(s_t\right)$ contains information about the robot and the environment, and the reward can be dynamically adjusted according to the robot's state potential function. Compared to previous reward principles, the adaptive potential reward function can autonomously obtain more adequate and accurate reward information.

4.3. Sample pretreatment mechanism

The utilization of the experience samples in the memory buffer has a critical effect on the training efficiency of the strategy model. To improve the utilization of samples, we need extract higher-quality experience samples from the memory buffer. However, in previous methods, all the experience samples are stored in the corporate experience buffer in an unordered manner, and most of the extracted experience samples are ineffective for strategy upgrading. Therefore, we design a sample pretreatment mechanism to improve the utilization of samples. The structure of the sample pretreatment mechanism is shown in Fig. 4. Firstly, a double-experience memory buffer is built to store and distinguish the successful and failed samples separately. During interactive trial-and-error period, if the robot is deviating from the goal or encounters any obstacles, the interaction process will be considered as a failed exploration. Then, the relevant samples would be stored into the failed experience memory buffer $C_{Failure}$. On the contrary, if the robot is approaching the goal and successfully reaches the target position, the associate experience samples would be stored in the successful experience memory buffer, which is shown as follows:

$$\begin{cases} \left(s_t^T, a_t^T, r_t^T, s_{t+1}^T\right) \rightarrow C_{Success} \\ \left(s_t^F, a_t^F, r_t^F, s_{t+1}^F\right) \rightarrow C_{Failure} \end{cases} \tag{18}$$

After enough experience samples are stored, these samples need to be extracted from the experience memory buffer to update the Actor–Critic network. The previous sampling process uses a random sampling manner that does not take into account the value and quality of the experience samples, and the selected samples are not necessarily valuable enough. Therefore, we designed a more intelligent sampling approach based on the double-experience memory buffer, i.e., the variable proportion sampling principle. In the initial stage, we extract a larger proportion of experience samples from the successful memory buffer. At the end of training process, the sampling probability of failed samples is increased to improve the exploration ability of the robot. The sampling principle is described as follows:

$$C_{smaple} \leftarrow p_1 * batchsize\left(s_i, a_i, r_i, s_{i+1}\right) + p_2 * batchsize\left(s_i', a_i', r_i', s_{i+1}'\right) \tag{19}$$

where p_1 represents the proportion of successful experience samples extracted from the success buffer, and p_2 denotes the proportion of failed experience samples drawn from the failure buffer. As the number of training sessions increased, p_1 represents the proportion of failed experience samples drawn from the failure buffer, and p_2 denotes the proportion of successful experience samples extracted from the success buffer; As training proceeds, p_1 gradually decreases until the minimum value. That is, $p_1 = \max(0.99^{\widehat{n}}, 0.1), p_2 = 1 - p_1$, and $\widehat{n}$ is the number of training episodes. The *batchsize* is the total number of samples selected in per batch.

In addition, to further improve the quality of the selected samples, we used a similarity judgment mechanism, which can remove similar samples from the batch by calculating the euclidean distance:

$$d\left(c_i, c_j\right) = \sqrt{\left(S_t^i - S_t^j\right)^2 + \left(R_t^i - R_t^j\right)^2 + \left(A_t^i - A_t^j\right)^2 + \left(S_{t+1}^i - S_{t+1}^j\right)^2} \tag{20}$$

where c_i, c_j are selected experience samples, $c_i = \left(S_t^i, R_t^i, A_t^i, S_{t+1}^i\right)$, $c_j = \left(S_t^j, R_t^j, A_t^j, S_{t+1}^j\right)$.

We set the Euclidean distance threshold to ζ. If $d\left(c_i, c_j\right) < \zeta$, the selected experience samples will be removed and sent back to the experience memory buffer.

The learning efficiency and convergence speed of actor–critic network can be improved by the variable proportion sampling mechanism and similarity judgment principle.

4.4. Approach principle

The detailed principle of RS-DDPG approach for the robot is shown in Algorithm 1

In summary, RS-DDPG proposed three main improvements:

(1) An event-triggered reconfigurable Actor–Critic network structure is constructed to deal with the poor convergence problem.
(2) An adaptive reward mechanism is introduced to compensate for the lack of sample information.
(3) A sample pretreatment mechanism is designed to improve the learning efficiency.

5. Experiment and result

5.1. Experiment's setting

We evaluate our method on a simulation platform with a mobile robot (Bogaerts et al., 2020). As shown in Fig. 5, the static task scenario has a size of 5*5 m and is divided into four regions. The scenario includes a two-wheeled robot and a random target. There are about 18 irregular obstacles and other related objects. The action space of the mobile robot is continuous and it has a maximum linear speed

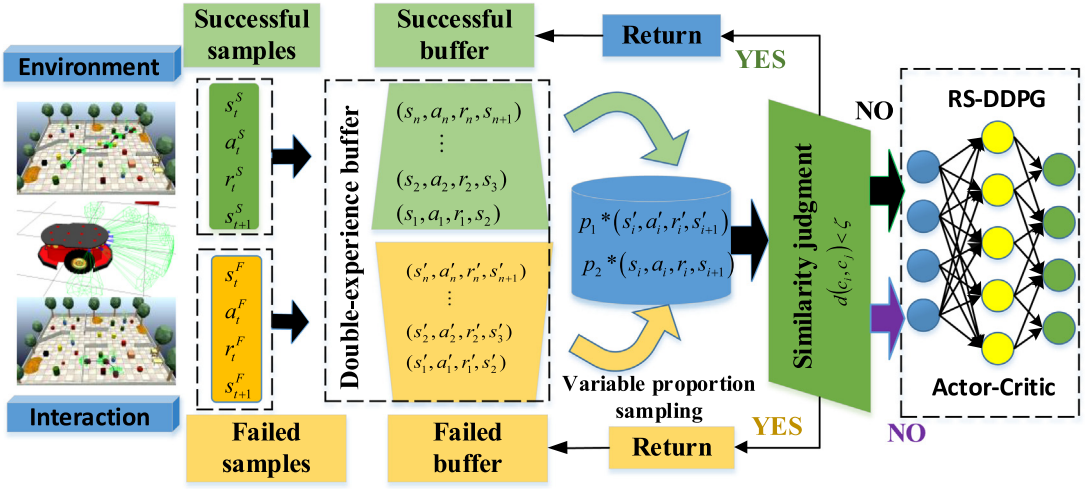


Fig. 4. Sample pretreatment mechanism.

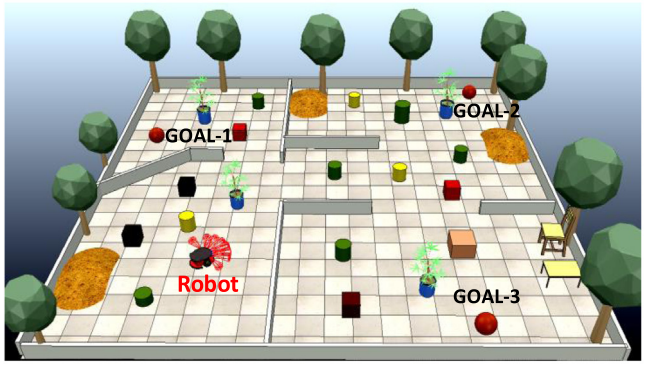


Fig. 5. Simulated environment for mobile robot.

of 2 m/s and a maximum angular speed of 0.5 rad/s. The robot is uniformly equipped with eight sets of distance sensors to detect barriers around the robot. At the same time, the collected state information is transmitted to the motion strategy immediately. More specifically, starting points appear randomly in four specific regions and the mobile robot needs to learn how to reach random target positions and avoid barriers without any prior knowledge or manual rules. During the trial-and-error interaction of the robot with the environment, the pioneer robot needs to continuously explore unknown areas to obtain the corresponding experience samples. When the robot reaches the target position, the episode will be ended and the associated samples will be stored into success memory buffer. The scenario will then also be reset for the next episode. Once the robot accidentally encounters a barrier, the robot's reward will be reduced. Unsuccessful experience samples will be stored in the failure memory buffer. In the static mission experiments, we conducted 3,000 episodes training experiments with a maximum training step of 600 per episode. Other specific parameters of RS-DDPG are shown in Table 2.

As shown in Table 2, lr_1 and lr_1 represent the learning rates of the policy network and the critic network respectively. They are used to determine the step size of the policy gradient. A higher learning rate can improve the update efficiency of the network, but it will increase the oscillation amplitude of the loss function, which will easily cause the overfitting problem of the action value network. Therefore, we choose a minimum value as the learning rate to slow down the update speed of the network weights so that the loss function converges with smaller fluctuations.

$C_{Success}$ and $C_{Failure}$ indicate the capacity of the success experience replay buffer and the failure experience replay buffer. The experience replay buffer is mainly used to store experience samples of interactions between the robot and the environment. Different proportions of samples will be extracted from the experience pool to efficiently update the strategy network. The *Batch size* represents the number of samples extracted from the experience pool per episode. In general, the capacity of the experience replay buffer is about 10 times the *Batch size*. Considering the storage space and running speed, the capacity of experience replay buffer is defined in the simulation experiments as follows: $C_{Success} = C_{Failure} = 6400$. Then the *Batch size* is set as: *Batchsize* = 64.

γ is the discount factor, which represents the concern for future rewards. In general, its value is between [0.9~1]. The robot motion planning policy should focus not only on the immediate rewards of current actions, but also on the effects of future rewards, so that the robot can avoid being trapped in a local optimization dilemma. Therefore, the discount factor γ is set to a larger value: $\gamma = 0.9995$.

τ_1 and τ_2 are the soft update factors of target network. To suppress the overestimation of the target action values, the weights of the target network learn only a small fraction of each episode from the current network. Therefore, the soft update factor is set to a very small value: $\tau_1 = \tau_2 = 0.001$, which slows down the update rate of the policy network.

5.2. Comparative experiments

In the simulation experiment, compared with previous deterministic strategy gradient methods, the proposed method has the following main innovations: First, we design an event-triggered reconfigurable RL structure based on deep deterministic policy gradient to restrain the overestimation bias of Q values and enhance the time convergence of the strategy. RS-DDPG can adaptively provide the optimal actor–critic network framework for different mission scenarios. Then, an adaptive reward mechanism is proposed to generate accurate reward signals and reduce fruitless motion exploration. Facing the problems of low quality and low utilization of training samples, RS-DDPG innovatively constructs an auxiliary experience memory buffer, optimizes the storage manner, and develops the principles of variable scale sampling and similarity judgment to extract higher-quality samples and shorten the strategy training period.

Based on our proposed method, we conducted adequate comparative experiments. Fig. 6 shows the successful interaction episodes between the robot and task scenario. It can be found that the starting point and the target position are located in two different regions, and

Algorithm 1: Event-triggered reconfigurable structure

Input: robot position, obstacle sensing state, target position

Output: motion planning policy π_w and action a_t

1. Initialize RS-DPPG parameters: discount factor γ, batch B, maximum volume of double experience replay pool $C_{Success}$, $C_{Failure}$. Maximum training episodes Ne, maximum step N_P, random strategy parameter N_O, learning rate lr_1 and lr_2

2 Initialize the $Critic_1$, $Critic_2$, $Critic_n$ and $Actor$ with the parameters θ_1, θ_2.....θ_n, w

$$\theta_1^{'} \leftarrow \theta_{1,}, \theta_2^{'} \leftarrow \theta_2, \cdots \theta_n^{'} \leftarrow \theta_n, w' \leftarrow w$$

3. For 1 to N_E

Initialize the environment

For 1 to Np

a) Initialize s_t as the original state and obtain the eigenvector s_t

b) Select action for the robot: $a_t = \pi_w(s_t) + N_o$, N_o is a random process for action exploration.

c) Execute a_t, get new state s_{t+1} and reward r_t.

d) State judgment : *Success or Failure*

e) Store $\{s_t, a_t, r_t, s_{t+1}\}$ in memory buffers $C_{Success}$ or $C_{Failure}$

$$\begin{cases} \left(s_t^T, a_t^T, r_t^T, s_{t+1}^T\right) \rightarrow C_{Success} \\ \left(s_t^F, a_t^F, r_t^F, s_{t+1}^F\right) \rightarrow C_{Failure} \end{cases}$$

f) Status update: $s_t = s_{t+1}$

g) Variable proportion sampling:

Choose p_1 proportion of samples from $C_{Success}$, Choose p_2 proportion of samples from $C_{Failure}$.

$$C_{smaple} \leftarrow p_1 * batchsize\left(s_i, a_i, r_i, s_{i+1}\right) + p_2 * batchsize\left(s_i', a_i', r_i', s_{i+1}'\right), batchsize = m$$

h) Similarity judgment:

Getting euclidean distance:

$$d\left(c_i, c_j\right) = \sqrt{\left(S_t^i - S_t^j\right)^2 + \left(R_t^i - R_t^j\right)^2 + \left(A_t^i - A_t^j\right)^2 + \left(S_{t+1}^i - S_{t+1}^j\right)^2}$$

If $d\left(c_i, c_j\right) < \zeta$, then: Return: step *g*)

i) Calculate the target action value y_{min}:

$$y_{\min} = r_t + \gamma \overline{Q}_{\theta_i'}\left(s_{t+1}, \pi_{w'}\left(s_{t+1}\right)\right)$$

j) Calculate loss function:

$$L = \frac{1}{m}\sum_{j=1}^{m}(y_{\min} - Q_{\theta_i}^j(s_t^j, a_t^j))^2$$

k) Event-triggered judgment:

Define trigger event as: $\Delta = \frac{1}{n}\sum_{n}\left|\delta_i / y_{\min}\right|, \kappa = 0.5$

If $\Delta \geq \kappa$, then: $n' = n + 1$

l) Update parameters for eval-Critic networks:

$$\theta_i \leftarrow \arg\min_{\theta_i}\left(\frac{1}{m}\sum_{j=1}^{m}(y_{\min} - Q_{\theta_i}(s_t^j, a_t^j))^2\right), i = 1, 2, \cdots n$$

m) Update parameter for eval-Actor network:

$$\nabla_w J(w) = \frac{1}{m}\sum_{j=1}^{m}[\nabla_a \overline{Q}_\theta^j(s_t^j, \pi_{w'}(s_t^j))\Big|_{a=\pi_{w'}(s_t^j)}) \nabla_\theta \pi_{w'}(s_t^j)]$$

n) Update parameters for target Critic and Actor networks:

$$\theta_1^{'} \leftarrow \tau\theta_1 + (1-\tau)\theta_1^{'};\ \theta_2^{'} \leftarrow \tau\theta_2 + (1-\tau)\theta_2^{'}$$
$$\theta_n^{'} \leftarrow \tau\theta_n + (1-\tau)\theta_n^{'};\ w^{'} \leftarrow \tau w + (1-\tau)w^{'}$$

o) Terminal judgment:*is_end*

if *is_end* = *True*, then: break

End for

End for

Table 2
Description of the parameters used by RS-DDPG.

Parameter	Instructions	Value
lr_1 and lr_2	Learning rate of networks	0.0001
τ_1	Soft update factor of Actor network	0.001
τ_2	Soft update factor of Critic network	0.001
Batch size	The size of samples chosen from the experience buffer per batch	64
γ	Discount factor	0.9995
$C_{Success}$	Capacity of the success experience replay buffer	6400
$C_{Failure}$	Capacity of the failed experience replay buffer	6400

the pioneer robot can reach the GOAL successfully while avoiding obstacles and passing through narrow passages. Then, the robot will receive a huge positive reward to reinforce the current motion planning strategy.

On the contrary, as shown in Fig. 7, the mission failed because it did not succeed in reaching the target position. One main reason is that the mobile robot always tries to directly choose the shortest path to get more rewards and ignores avoiding obstacles. Another reason is that the

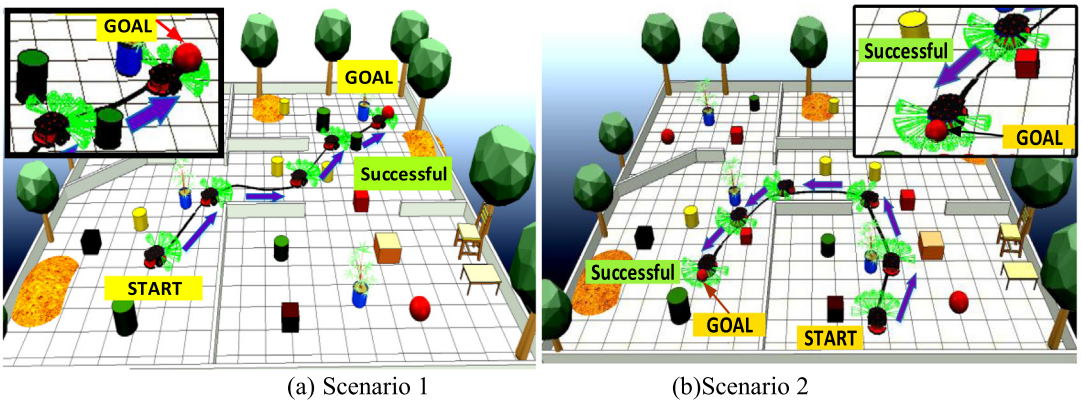


Fig. 6. Successful motion planning process.

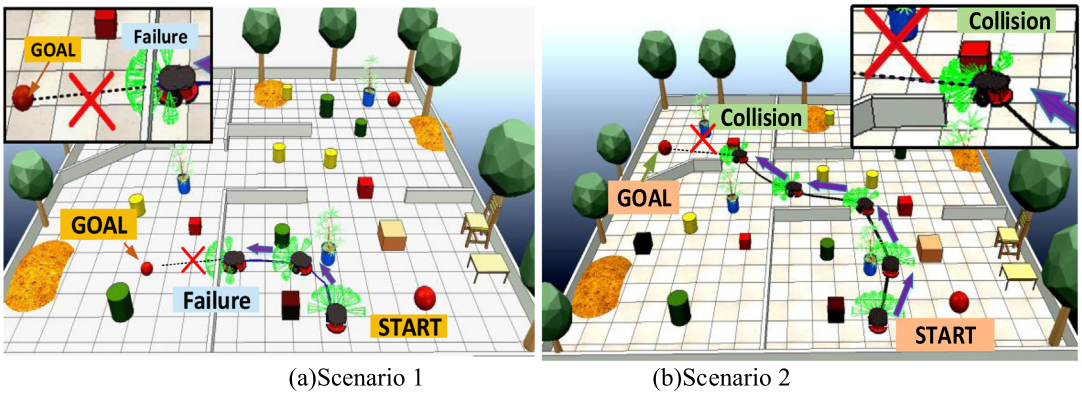


Fig. 7. Failed motion planning training process.

traversable paths are too narrow for the robot to avoid obstacles, which indicates that the motion planning strategy is not high-performance enough.

After 3,000 episodes of interactive training in the simulated scenarios, we compared the results of RS-DDPG to a baseline, including DDPG, TD3 and SAC. The performance indexes are shown in Fig. 8, including Episode Reward, Success Rate, Actor-loss and Critic-loss.

Episode Reward: We plot the average episodes rewards and the number of completed episodes in Fig. 8(a). Due to our proposed adaptive reward mechanism, the pioneer robot can earn more rewards in each episode. The average reward curve of RS-DDPG rises rapidly, reaching more than 20 after 500 episodes. However, the average episode rewards of DDPG and SAC are lower than 20, which leads to a decrease in motion planning performance over time. The maximum episode rewards of TD3 are close to those of RS-DDPG, but its reward curve rises slowly.

Success Rate: Fig. 8(b) shows the probability of successfully reaching the target in every 50 episodes. In other words, the navigation success rate. The success rate is a comprehensive index of the motion planning task, which not only represents the performance of the motion strategy, but also reflects the advantage of the reward mechanism. A reliable strategy and accurate immediate rewards can successfully guide the robot to the target position. As can be seen in Fig. 8(b), the RS-DDPG method achieves a high success rate rapidly in the early learning phase. After 500 episodes, the maximum success rate has reached 98%, and the average success rate is 93%. The success rate curves of both TD3 and SAC are both lower than those of RS-DDPG, which remain at 70% and 85%, respectively. The minimum success rate of DDPG is only around 60%. From the comparison, it is clear that the reconfigurable multi-critic structure improves the motion performance of the strategy and the adaptive reward mechanism provides an essential reference for action selection. The pioneer robot was able to navigate to the target with the highest success rate.

Loss function: Fig. 8(c) and 8(d) show the variation trends of Actor-loss and Critic-loss. The loss function curves indicate the convergence speed of the motion planning policy. From Fig. 8 we can find that the Actor-loss and Critic-loss of RS-DDPG begin to converge within 500 episodes, while the DDPG method gradually converges after the 1500*th* episode. The SAC and TD3 algorithms converge relatively faster than DDPG, but they are still lower than RS-DDPG. The fastest convergence speed indicates that RS-DDPG has the most valuable samples and the highest sample update efficiency. These advantages are mainly attributed to the essential role of the double-experience memory buffer and the variable proportion sampling principle.

The double-experience memory buffer and variable proportion sampling principle can provide higher-quality sample data for RS-DDPG and improve the training efficiency of the motion strategy. As a result, compared with other comparative methods, the proposed method shows better time convergence and training efficiency, and the navigation success rate of the robot is significantly improved. In terms of loss function, the variation trend of RS-DDPG is small and uniform, and RS-DDPG can provide a reliable motion planning policy for mobile robots in the absence of maps. It is shown that the multi-critic network framework of RS-DDPG can effectively suppress the overestimation bias of action values to a large extent.

5.3. Performance evaluation

To verify the positive effect of reconfigurable critic networks, we compared the standard deviations of Q-value for different network structures, including single-critic networks, dual-critic networks and triple-critic networks. After 3,000 episodes interactions training, the

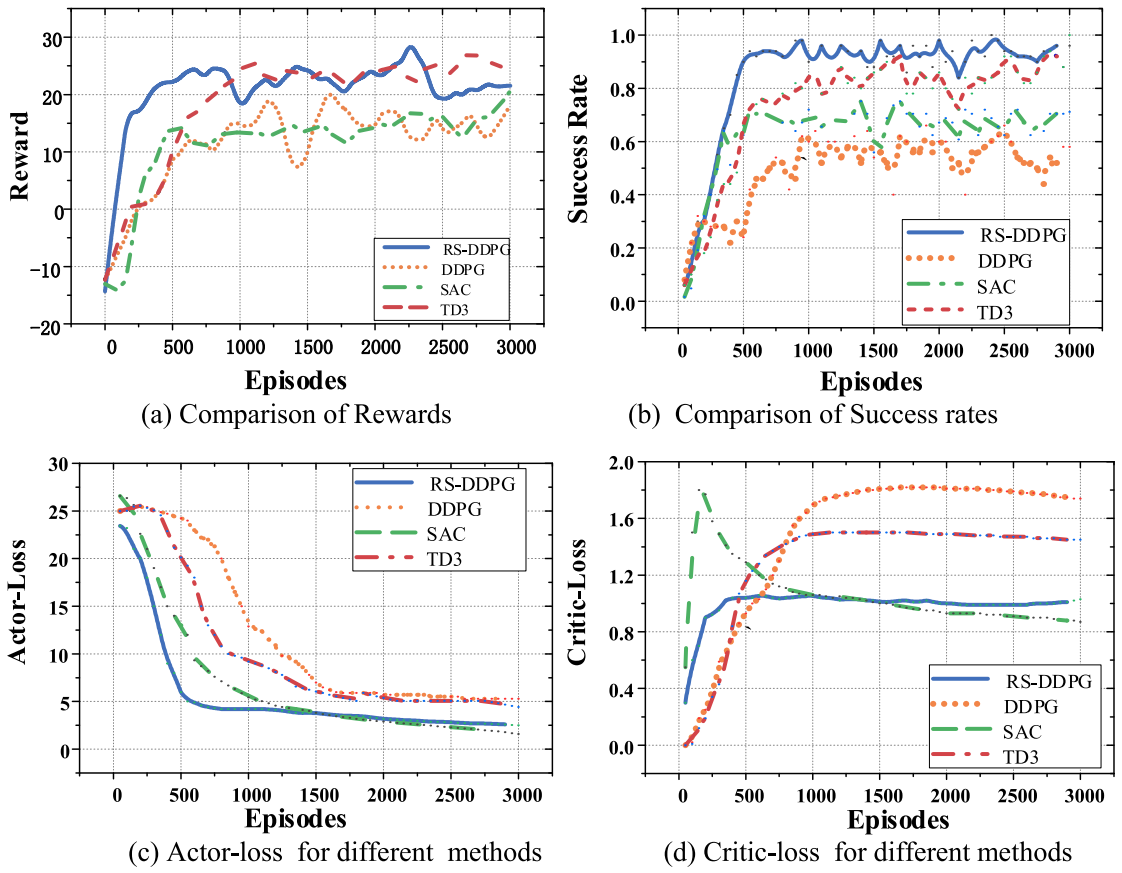


Fig. 8. Motion planning performance analysis.

variation trends of the standard deviations are shown in Fig. 9. Compared to the other two structures, the standard deviation of the Q-value with triple-critic networks performs smaller fluctuation and faster convergence speed. After 2,000 episodes, the Q-value standard deviations reach the minimum values. These advantages indicate that the reconfigurable multi-network structure plays an influential role in suppressing the overestimation deviation of Q value.

Then, an adaptive reward mechanism is proposed to generate accurate reward signals and reduce fruitless motion exploration. It can be verified by the speed of rise and the average value of the reward function. As shown in Fig. 8(a), the reward of our RS-DDPG is compared with other approaches. Considering our proposed adaptive reward mechanism, the robot can obtain more significant reward signals in each episode, which reaches a higher level than other comparative algorithms. At the same time, the number of invalid explorations is greatly reduced, and the average reward curve of RS-DDPG gets the fastest rising speed.

Finally, in order to illustrate the advantages of the double-experience memory buffer and the variable proportion sampling principle, we analyze the quality of the extracted samples. The quality of the samples can be measured by the sample-reward values $\bar{r}$. The average reward value of all extracted samples is:

$$\bar{r} = \frac{1}{mp_1}\sum_{i=1}^{mp_1} r_i + \frac{1}{mp_2}\sum_{j=1}^{mp_2} r_j \tag{21}$$

where p_1 is the percentage of successful samples and p_2 is the percentage of failed samples.

Therefore, we recorded the average reward value of the selected samples during the interaction and compared it with the samples selected from the single-experience memory buffer. The results are shown in Fig. 10.

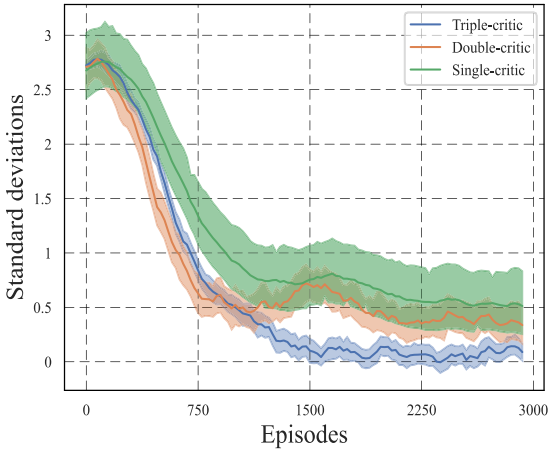


Fig. 9. Comparison of Q-value standard deviations.

In Fig. 10, the sample-reward values of the variable proportion sampling mechanism are higher at the beginning of the update process, which indicates that the robot is moving toward the target and avoiding obstacles successfully. Based on these successful experience samples, the strategy can converge to the optimal direction at a faster rate, which is also verified through the variation of Actor–Critic loss in Fig. 8(c) and 8(d). In the later stage of training, the sample-reward value of RS-DDPG begins to decrease because we need to extract more failure samples to prevent the motion strategy from entering the local optimum. By learning the failed experience samples, the robot can avoid barriers in time and obtain higher navigation success rate, which is also illustrated in Fig. 8(b).

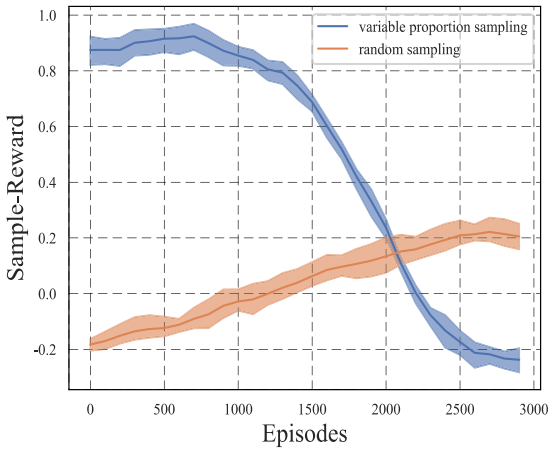


Fig. 10. Comparison of the average rewards.

Table 3
Performance indicators compare of different algorithms.

Algorithms	Average reward	Average steps	Convergence episodes	Success rates(%)
RS-DDPG	25.9	70.2	480	97.5
DDPG	14.1	103	1500	60.2
SAC	18.7	81.9	1400	75.6
TD3	20.1	85.4	1000	86.8

Moreover, Table 3 shows specific comparisons of average rewards, episode steps, convergence episodes and success rates.

It can be found that RS-DDPG has the maximum average reward and the minimum episode number of steps in each training period compared to the other three baselines, which indicates that the robot reaches the target position at the fastest speed. The comparison of the convergence episodes shows that RS-DDPG has the least convergence episodes and at least doubles the training efficiency compared to the other comparative baselines. In addition, the performance of motion strategy is mainly reflected in the loss function and the motion planning success rate of the mobile robot. The navigation success rate of RS-DDPG has reached 97.5% when the strategy converges, which is significantly higher than other comparison methods. The improvement of the motion planning performance is mainly due to the suppression of Q-value deviation by using a multi-critic network.

To sum up, the proposed event-triggered reconfigurable multi-critic network, adaptive reward mechanism and double-experience replay principle play a significant role in the convergence of the motion strategy and improve the success rates of the robot in complex environments.

5.4. Complex scenarios experiments

To further discuss how the proposed method works in the presence of moving obstacles and non-feasibility problems, we built more challenging environments for the mobile robot, as shown in Fig. 11(a) and 11(b).

The size of the dynamic scenario is still 5*5 m, which contains various dynamic obstacles and multiple barrier walls. The initial and target positions appear randomly in the scenario, and the dynamic obstacles move back and forth freely. The maximum speed of dynamic obstacles is 0.8 m/s, and their motion trajectories are marked with dotted lines in Fig. 11. Similarly, we performed 3,000 episodes interactions in the dynamic scenarios where the robot had to avoid various obstacles to reach the random target position.

In the dynamic comparison experiment, we added the AI2-THOR (Zhu et al., 2017) and DS-RNN (Liu et al., 2021) algorithms as auxiliary comparison baselines. These two DRL-based algorithms were developed for the motion planning missions and showed relatively good performance on the mobile robot platform. After extensive experiments, the consequences are shown in Figs. 12, 13 and Table 4.

Table 4
Performance indicators comparison in dynamic scenario.

Algorithms	Average rewards	Average steps	Success rates(%)
RS-DDPG	30.8	89.5	97.6
DDPG	18.6	110.4	67.3
SAC	25.3	107.6	82.5
TD3	26.9	98.6	91.2
AI2-THOR	25.8	97.2	88.9
DS-RNN	27.1	92.5	90.3

Fig. 12 shows the average reward comparison per episode, and Fig. 13 shows the navigation success rate of the mobile robot based on different approaches. From the result figures, it can be seen that after 3,000 episodes, the reward values and success rates of all approaches almost reached their maximum values, and the policy network models reached convergence. The curve comparison shows that the reward value curve of RS-DDPG fluctuates less and the overall convergence speed is the fastest. The DS-RNN method is slightly lower than RS-DDPG, and the reward value of AI2-THOR is between SAC and TD3.

On the other hand, it can be observed from Fig. 13 that the ranking of navigation success rate is similar to the ranking of the reward value in a complex dynamic environment. RS-DDPG also has a considerable advantage over the basic algorithm and the improved DRL-based methods in terms of the success rate of the navigation metrics.

In addition, all methods are almost not affected by dynamic obstacles and local minima traps, and they show at least the same level of motion planning capacity as in the static scenarios. Overall, these results suggest that DRL-based motion planning algorithms have relatively good environmental understanding.

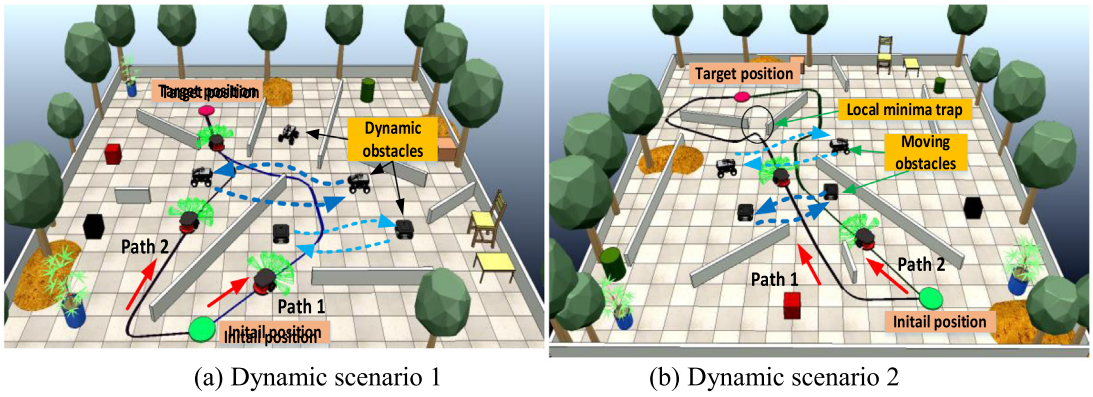
Table 4 shows a more intuitive comparison of the performance indicators. In addition to the average rewards and navigation success rates, the RS-DDPG strategy still takes the fewest steps to reach the target position, which also illustrates its relatively efficient motion planning capability in dynamic complex environments.

Finally, to analyze the impact of the reconfigurable structure on the training time requirements, we compared the time requirements of the different algorithms for each training episode. The results are shown in Fig. 14.

In particular, it can be seen from Fig. 14 that the training time requirement is almost the same for all algorithms, about 6 s per episode. This is because the emulator spends most of its time calculating the dynamic features of the robot and rendering the scenarios, while the reconfigurable network structure consumes less time. In each episode, the time consumption of our proposed RS-DDPG method increases by only 0.4 s compared to the fastest DDPG method. In the whole training process, we spend only 20 min longer than DDPG. In addition, the time consumption of TD3, RS-DDPG and DS-RNN basically reached the same level. Therefore, although the structural modification brings an increase in time demand, the additional cost is acceptable for the robot platform.

To discuss the task timeout problem in the simulation, we performed additional experiments in both static and dynamic scenarios. The results are shown in Fig. 15.

It should be noted that during the simulation, we used the maximum number of training steps $N_P = 600$ per episode as a judgment index for task timeout. If the robot does not reach the goal within 600 steps, the task is considered a failed attempt. Fig. 15(a) shows the comparison of task failure rates in static simulation scenarios. It can be found that in the static environment, the timeout rate of the DDPG algorithm reaches 23.6%, which is significantly higher than the collision rate. The timeout rate of our RS-DDPG method is only 1.2%, which is at the lowest level among the compared methods. In the dynamic task scenario, as shown in Fig. 15(b), the distribution of the task timeout rate is similar to



(a) Dynamic scenario 1 (b) Dynamic scenario 2

Fig. 11. Complex scenario with moving obstacles.

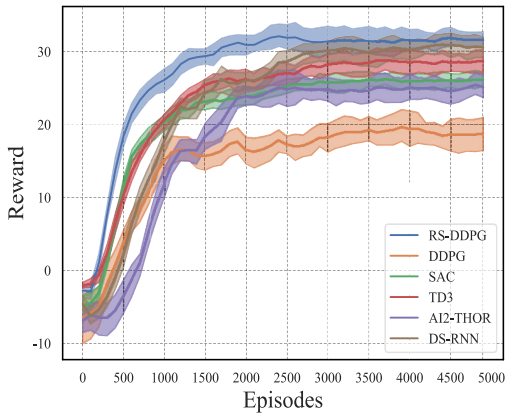


Fig. 12. Comparison of Rewards.

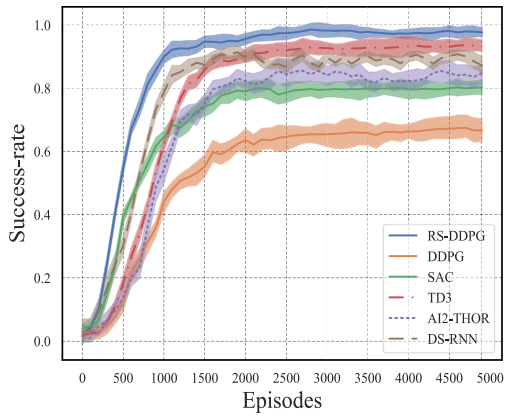


Fig. 13. Comparison of success rates.

that in the static scenario. The task timeout rates of the AI2-THOR and DS-RNN algorithms is about 3%, which is lower than that of the SAC algorithm. The collision rate and timeout rate of RS-DDPG are also significantly lower than those of the other compared methods, which indicates that the robot has efficient motion planning performance in the simulation scenarios.

By analyzing and comparing all the experiment results, it can be verified that the motion planning performance of the proposed method outperforms the comparison results.

Table 5
Performance indicators comparison in physical scenarios.

Algorithms	DDPG	SAC	TD3	RS-DDPG	AI2-THOR	DS-RNN
Success rates (%)	61	79	87	95	83	90
Collision rates (%)	30	11	7	3	10	6
Timeout rates (%)	11	10	6	2	7	4
Average time (s)	38.5	30.2	28.6	25.1	27.7	26.4

5.5. Physical experiments

To further illustrate the applicability of the proposed method in real-world scenarios, we conducted physical experiments using the experimental setup of the BINGGO robot. As shown in Fig. 16, the experimental platform was equipped with a Jetson-Nano (1.43 GHz, 4.0 GB memory) control board and the ROS Melodic operating system.

When we transfer the motion policy model from simulation to the real world, the motion planning capability of this robot will be compromised by parameter uncertainty and measurements noise. The task success rates will show a slight decreasing trend compared to that in the simulation environment.

In the physical experiments, if the robot fails to successfully reach the target position within 60 s, the mission is considered to have timed out. If the robot unexpectedly encounters an obstacle, it is also regarded as a failure episode. During the physics experiments, we conducted 100 rounds of testing based on each motion planning method. At the same time, the average navigation time for each episode was also calculated as an evaluation index. The specific results are shown in Table 5.

Similar to the results in the simulation environment, it can be found from Table 5 that the navigation success rate of our proposed RS-DDPG method is 95%, which is at least 5% higher than the comparison algorithm DS-RNN. The navigation success rate of DDPG, SAC, AI2-THOR, and TD3 is lower than that of DS-RNN. In terms of collision rates, the RS-DDPG method encounters only two accidental collisions, which means that RS-DDPG has relatively good obstacle avoidance capacity. During the experimental test, if the motion strategy falls into local optimization, the mobile robot will continue to explore in a small range, which then leads to a task timeout. The proposed method shows the minimum timeout rates and reflects better time convergence of motion strategy. The last performance index is the average time consumption of the robot to navigate to the target position. The average time requirement of our proposed method in the physical environment is about 25.1 s, which is 13 s faster than DDPG method. The reason is that RS-DDPG always selects superior actions for the mobile robot during the experiments, and the mobile robot can find and access the target position faster than other algorithms. On the other hand, all success rates are slightly lower than those in the simulated environment

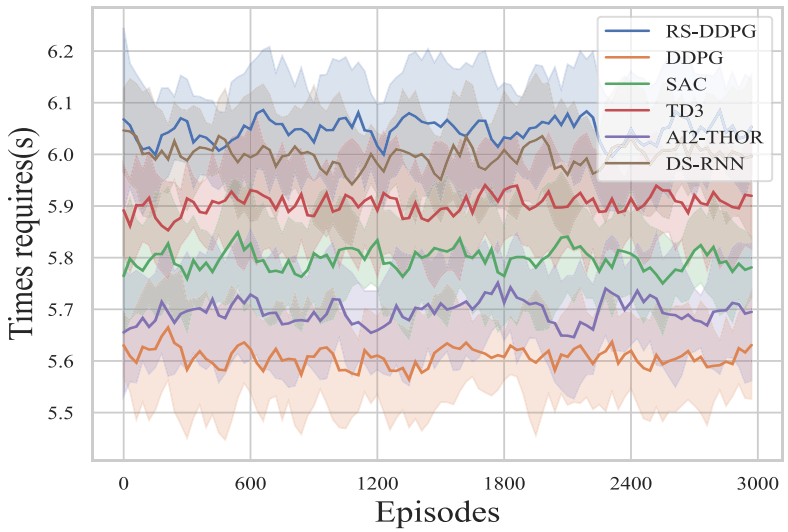


Fig. 14. Comparison of time requires in complex scenario.

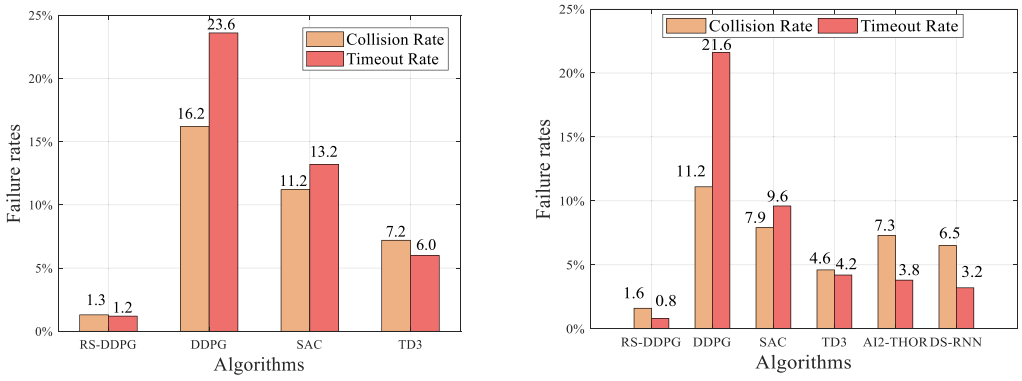


Fig. 15. Failure rates analysis for different simulation scenarios.

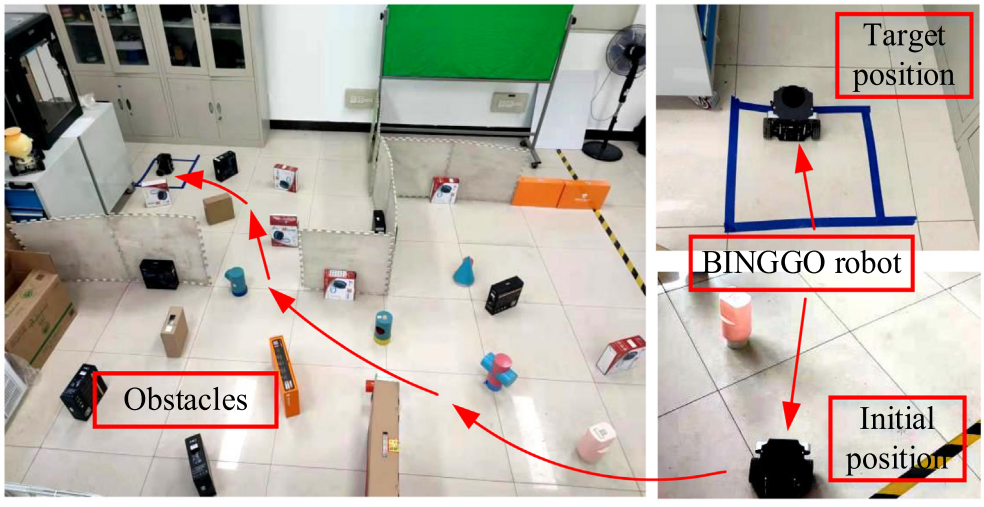


Fig. 16. Physical experiment environment.

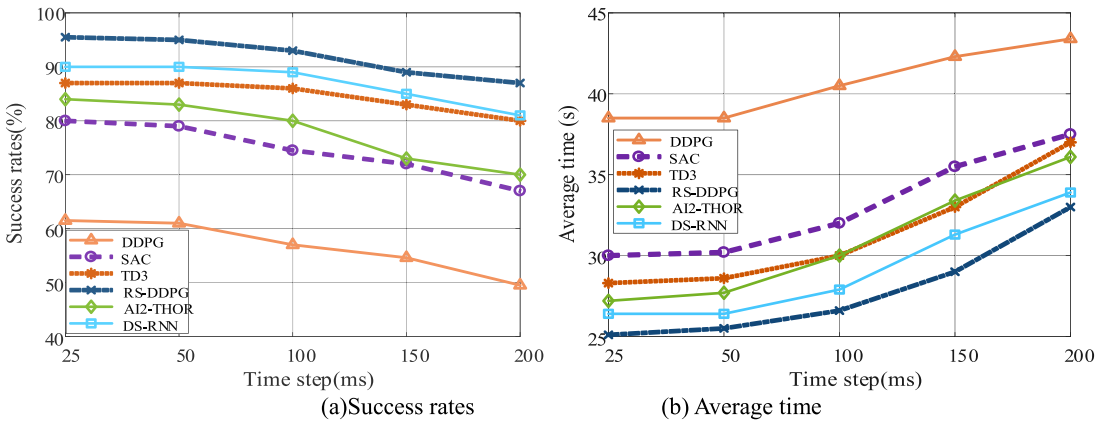


Fig. 17. Success rates and average time based on different time steps.

Table A.1
Description of symbols.

Parameters	Illustration	Parameters	Illustration
$Q_\pi(s_t, a_t)$	Action value	w	Weight of policy network
π_w	Motion policy	θ_i	Weight of critic network
$J(\pi_w)$	Objective function	y_{min}	Minimum target action value
$R_F(a_t, s_{t+1})$	state potential reward	δ_i	TD error
$R_T(s_{t+1})$	Termination reward	$\varphi(s_t)$	State potential function
$R_C(a_t)$	Smooth reward	ξ_1, ξ_2, ξ_3	Reward state constants
$d(c_i, c_j)$	Euclidean distance	p_1,p_2	Sampling Coefficient
$f(u)$	Compression mapping	$V_\pi(s)$	State value
χ	Metric space	β	Lipschitz constant

due to uncertain interference and data transmission delays in the physical environment.

To discuss the effect of time steps Δt on the performance of motion planning, we conducted experiments based on different time steps Δt. The experiment results are shown in Fig. 17.

As can be seen in Fig. 17, we selected different time steps for the physical experiments, including 25 ms, 50 ms, 100 ms, 150 ms and 200 ms. As the time steps increases, the success rates of the robot's navigation based on different methods shows a decreasing trend. When the time step was over 100 ms, all the success rates decreased greatly. However, the success rates of navigation at 50 ms and 25 ms were almost the same. In terms of the average time consumption per episode, we found that the average time increased constantly as the time step increases. Similarly, when the time step is 50 ms, the increase in time consumption is not significant compared to 25 ms. Therefore, it is relatively reasonable to choose 50 ms as the standard time step in the physical scenarios. At the same time, we also adopt $\Delta t = 50$ms as the standard time step in the simulation scenarios.

Overall, the proposed method RS-DDPG shows relatively good motion planning capability in the physical environment and outperforms other comparative methods compared to other algorithms.

6. Conclusion and future work

In this paper, we propose a novel learning-based motion planning approach, called event-triggered reconfigurable structure based on deep deterministic policy gradient (RS-DDPG) for mobile robots in unknown dynamic environments. Compared with other DRL-based motion planning approaches, the proposed approach has addressed the problems of poor convergence, insufficient sample information and low learning efficiency.

Specifically, a reconfigurable actor–critic network framework is constructed for RS-DDPG based on the event-triggered condition, which automatically generates optimal network structures to improve convergence speed and suppress overestimation of action values. Then, an adaptive reward mechanism via a state potential function to compensate for the insufficient sample information is innovatively introduced, so that the robot receives adequate reward feedback during the interactive training process. In addition, to address the problem of inefficient learning, a sample pretreatment mechanism is developed for RS-DDPG, and the number of training episodes is reduced by utilizing triple techniques such as double-experience memory buffers, variable proportion sampling principle and similarity judgment mechanism.

Finally, extensive experiments are carried out in both simulated and physical scenarios. The results show that the proposed method has the largest reward value and the fastest convergence rate compared to other comparative algorithms. The robot achieves a task success rate of over 97% in the simulation environment, while it remains around 95% in the real environment. Our method also outperforms other comparative methods in terms of other motion performance metrics, including collision rates, timeout rates and average time.

In future improvements, we aim to extend our RS-DDPG method to multi-agent motion planning systems subject to complex kinematic constraints, and investigate how to ensure motion safety in threatening environments.

CRediT authorship contribution statement

Huihui Sun: Data curation, Writing – original draft, Visualization. **Changchun Zhang:** Writing – review & editing, Investigation, Formal analysis. **Chunhe Hu:** Supervision, Project administration, Funding acquisition. **Junguo Zhang:** Methodology, Conceptualization, Validation.

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.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61703047), and the Science and Technology Project of Hebei Education Department, China (No. QN2021312).

Appendix A

See Table A.1.

Appendix B

Convergence proof

The reconfigurable structure of the deep deterministic policy gradient (RS-DDPG) can be regarded as a Markov decided process (MDP), which can be described as: $\Gamma \triangleq (s, a, P, r, \gamma)$. In this section, the convergence of the RS-DDPG will be proved by using the Banach's fixed point theorem (Gordji et al., 2017).

Banach's fixed point theorem: Let $\left(\chi, \widetilde{d}\right)$ be a complete metric space where a function $f : \chi \to \chi$ is a contractor, then there is a unique fixed point $\chi *$ that causes the sequence $f(f(f(\cdots f(\chi))))$ to converge to $\chi *$ (Ran and Reurings, 2004; Koczka et al., 2009).

Suppose that the sequence of contractor composed of set χ converges to two different values x_1^* and x_2^*, then:

$$\widetilde{d}\left(f\left(x_1^*\right), f\left(x_2^*\right)\right) = \widetilde{d}\left(x_1^*, x_2^*\right) \tag{B.1}$$

Because the function f is a contractor, it can be obtained:

$$\widetilde{d}\left(f\left(x_1^*\right), f\left(x_2^*\right)\right) \le \beta\widetilde{d}\left(x_1^*, x_2^*\right) \tag{B.2}$$

As $\beta < 1$, we can obtain according to Eq. (B.2):

$$\widetilde{d}\left(f\left(x_1^*\right), f\left(x_2^*\right)\right) \le \beta\widetilde{d}\left(x_1^*, x_2^*\right) < \widetilde{d}\left(x_1^*, x_2^*\right) \tag{B.3}$$

It can be found that Eq. (B.3) contradicts with Eq. (B.1). That is, there cannot be two or more convergent values. So the convergence value $x *$ of χ is unique.

According to Banach's fixed point theorem, if RS-DDPG satisfies the existence conditions of the fixed point theorem, for any finite MDP, there exists an optimal policy $\pi *$ that is better than or equal to all other possible policies π. In other words, if our method holds in the fixed point theorem, our method must be convergent. Therefore, the RS-DDPG requires the following two conditions to be met.

Condition 1. There exists a complete metric space $\left(\chi, \widetilde{d}\right)$ in the Markov game process, and for each subset that can form a Cauchy sequence, the Cauchy sequence also converges within that set χ.

Proof. For any finite MDP, the metric space can be defined as $\left(V(s), \widetilde{d}\right)$ based on the state value function $V_\pi(s)$. The distance of the metric space $\widetilde{d}$ can be expressed as an infinite norm:

$$\widetilde{d} = \|\chi\|_\infty = \left\|V_i(s) - V_j(s)\right\|_\infty \tag{B.4}$$

Then, we take a subset on the metric space as a Cauchy sequence: $\Lambda\left\{V_1, V_2, V_3, \ldots, V_n\right\}$.

By the definition of the state value function $V_\pi(s)$, the metric space belongs to the sequence of real numbers: $\chi : V(s) \in R$. It can be easily deduced that a subset of the Cauchy sequence $\Lambda\left\{V_1, V_2, V_3, \ldots, V_n\right\}$ will also converge to a sequence of real numbers: $\Lambda \to R$.

Therefore, the conditions for forming a complete metric space $\chi : V(s)$ are met, and MDP is able to construct a complete metric space $(V(s), L - \text{infinity})$.

Condition 2. There is a contraction mapping function f in the metric space $\left(\chi, \widetilde{d}\right)$. For any x_1, x_2 in χ, it is possible to find a Lipschitz constant $\beta \in [0, 1]$ satisfying the inequality: $\widetilde{d}\left(f\left(x_1\right), f\left(x_2\right)\right) \le \beta\widetilde{d}\left(x_1, x_2\right)$.

Proof. Define the action value function $Q_\pi(s, a)$ as a compressed mapping f on the complete measure space $(V(s), L - \text{infinity})$: $Q_\pi(s, a) = f(V(s))$. set any value function $V_i(s') = \widetilde{u}$, $V_j(s') = \widetilde{v}$. Then, their compression mappings are: $f\left(\widetilde{u}\right), f\left(\widetilde{v}\right)$. According to Eq. (B.4), the distance d of the metric space of the compressed mapping can be rewritten as:

$$\widetilde{d}\left(f\left(\widetilde{u}\right) - f\left(\widetilde{v}\right)\right) = \left\|Q_i^\pi(s, a) - Q_j^\pi(s, a')\right\|_\infty \tag{B.5}$$

The optimization process of the RS-DDPG-based motion strategy π is to find the maximum action value function $Q_\pi(s, a)$ (Melo, 2001). Meanwhile, according to the Behrman equation, Eq. (B.5) can be expressed as:

$$\widetilde{d}\left(f\left(\widetilde{u}\right) - f\left(\widetilde{v}\right)\right) = \left\| \begin{array}{l} \max\left(\sum_{s',r} P\left(s' \mid s, a\right)\left[r(a, s) + \gamma V_i\left(s'\right)\right]\right) \\ -\max\left(\sum_{s',r} P\left(s' \mid s, a'\right)\left[r\left(a', s'\right) + \gamma V_j\left(s'\right)\right]\right) \end{array} \right\|_\infty \tag{B.6}$$

where $P\left(s' \mid s, a\right)$ denotes the state transition probability distribution,

Then, we introduce the inequality relation of Eq. (B.5):

$$\begin{aligned} &\widetilde{d}(f(u) - f(v)) = \left\|Q_1^\pi(s, a) - Q_2^\pi(s, a')\right\|_\infty \\ &\le \beta \max\left\|\sum_{s',r} P\left(s' \mid s, a\right) V_i\left(s'\right) - \sum_{s',r} P\left(s' \mid s, a\right) V_j\left(s'\right)\right\|_\infty \\ &\le \beta\left\|V_i\left(s'\right) - V_j\left(s'\right)\right\|_\infty \max\sum_{s',r} P\left(s', r \mid s, a\right) \\ &\le \beta\left\|V_i\left(s'\right) - V_j\left(s'\right)\right\|_\infty = \beta\widetilde{d}\left(\widetilde{u} - \widetilde{v}\right) \end{aligned} \tag{B.7}$$

Therefore, Eq. (B.7) is consistent with Condition 2. In the RS-DDPG-based complete metric space $(V(s), L - \text{infinity})$, there exists a compression mapping f and a Lipschitz constant $\gamma \in [0, 1]$, which satisfies the inequality:

$$\widetilde{d}\left(f\left(\widetilde{u}\right), f\left(\widetilde{v}\right)\right) \le \beta\widetilde{d}\left(\widetilde{u}, \widetilde{v}\right) \tag{B.8}$$

Let $\widetilde{u} = x_m, \widetilde{v} = x_n$ be two elements in the set χ, and $m \gg n$. According to the characteristic of triangular inequality in metric space $\widetilde{d}$, we have:

$$\begin{aligned} &\widetilde{d}\left(x_m, x_n\right) \le \widetilde{d}\left(x_m, x_{m-1}\right) + \widetilde{d}\left(x_{m-1}, x_n\right) \\ &\le \widetilde{d}\left(x_m, x_{m-1}\right) + \cdots + \widetilde{d}\left(x_{n+1}, x_n\right) \\ &\le \widetilde{d}\left(f^m\left(x_0\right), f^{m-1}\left(x_0\right)\right) + \cdots + \widetilde{d}\left(f^{n+1}\left(x_0\right), f^n\left(x_0\right)\right) \end{aligned} \tag{B.9}$$

Since the f is a contractor, then:

$$\begin{aligned} \widetilde{d}\left(f^n\left(x_1\right), f^n\left(x_2\right)\right) &\le \beta\widetilde{d}\left(f^{n-1}\left(x_1\right), f^{n-1}\left(x_2\right)\right) \\ &\le \beta^{n-1}\widetilde{d}\left(f\left(x_1\right), f\left(x_2\right)\right) \le \beta^n\widetilde{d}\left(x_1, x_2\right) \end{aligned} \tag{B.10}$$

Combining Eqs. (B.9) and (B.10), and then, Eq. (B.9) can be rewritten as:

$$\begin{aligned} &\widetilde{d}\left(x_m, x_n\right) \le \beta^{m-1}\widetilde{d}\left(f\left(x_0\right), x_0\right) + \cdots + \beta^n\widetilde{d}\left(f\left(x_0\right), x_0\right) \\ &\le \widetilde{d}\left(f\left(x_0\right), x_0\right)\sum_{i=n}^{m-1}\beta^i \le \beta^n\widetilde{d}\left(f\left(x_0\right), x_0\right)\sum_{i=0}^{m-n-1}\beta^i \\ &\le \frac{\beta^n}{1-\beta}\widetilde{d}\left(f\left(x_0\right), x_0\right) < \epsilon \end{aligned} \tag{B.11}$$

That is, for $\forall\epsilon > 0, \exists N = \left[\log_\beta \frac{\epsilon(1-\beta)}{d(f(x_0), x_0)}\right]$. Then, for $\forall m, n > N$, we can get $\widetilde{d}\left(x_m, x_n\right) \le \epsilon$.

To sum up, the optimization process of RS-DDPG satisfies the conditions of the fixed point theorem. That is, the state value $V(s)$ will converge to optimal value V^*, and the $Q_\pi(s, a)$ will also converge to the optimal value Q^*. The robot acts according to the optimal action value Q^* at each step, then we will generate the optimal motion planning strategy $\pi *$.

References

Bayat, F., Najafinia, S., Aliyari, M., 2018. Mobile robots path planning: Electrostatic potential field approach. Expert Syst. Appl. 100, 68–78.

Bogaerts, B., Sels, S., Vanlanduit, S., et al., 2020. Connecting the CoppeliaSim robotics simulator to virtual reality. SoftwareX 11, 100426.

Chen, W., Shi, H., Li, J., et al., 2021. A fuzzy curiosity-driven mechanism for multi-agent reinforcement learning. Int. J. Fuzzy Syst. 23 (5), 1222–1233.

Cui, Q., Kim, G., Weng, Y., 2021. Twin-delayed deep deterministic policy gradient for low-frequency oscillation damping control. Energies 14 (20), 6695.

de Jesus, J.C., Bottega, J.A., Cuadros, M.A.D.S., et al., 2021a. Deep deterministic policy gradient for navigation of mobile robots. J. Intell. Fuzzy Systems 40 (1), 349–361.

de Jesus, J.C., Kich, V.A., Kolling, A.H., et al., 2021b. Soft actor-critic for navigation of mobile robots. J. Intell. Robot. Syst. 102 (2), 1–11.

Deng, M., Inoue, A., Sekiguchi, K., et al., 2010. Two-wheeled mobile robot motion control in dynamic environments. Robot. Comput.-Integr. Manuf. 26 (3), 268–272.

Do, T., Duong, M., Dang, Q., et al., 2018. Real-Time Self-Driving Car Navigation using Deep Neural Network. IEEE, pp. 7–12. http://dx.doi.org/10.1109/GTSD.
Fahad, M., Yang, G., Guo, Y., 2020. Learning human navigation behavior using measured human trajectories in crowded spaces. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 11154–11160.
Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596.
Garcia, A., Mittal, S.S., Kiewra, E., et al., 2019. A convolutional neural network feature detection approach to autonomous quadrotor indoor navigation. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 74–81.
Giusti, A., Guzzi, J., Cireşan, D.C., et al., 2015. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics Autom. Lett. 1 (2), 661–667.
Gordji, M.E., Ramezani, M., De La Sen, M., et al., 2017. On orthogonal sets and Banach fixed point theorem. Fixed Point Theory 18 (2), 569–578.
Guo, S., Zhang, X., Du, Y., et al., 2021. Path planning of coastal ships based on optimized DQN reward function. J. Mar. Sci. Eng. 9 (2), 210.
He, J., Wang, Y., Liang, Y., et al., 2022. Learning-based airborne sensor task assignment in unknown dynamic environments. Eng. Appl. Artif. Intell. 111, 104747.
Huang, X., Deng, H., Zhang, W., et al., 2021. Towards multi-modal perception-based navigation: A deep reinforcement learning method. IEEE Robotics Autom. Lett. 6 (3), 4986–4993.
Hwangbo, J., Lee, J., Dosovitskiy, A., et al., 2019. Learning agile and dynamic motor skills for legged robots. Science Robotics 4 (26), u5872.
Jia, Y., Ma, S., 2021. A coach-based Bayesian reinforcement learning method for snake robot control. IEEE Robotics Autom. Lett. 6 (2), 2319–2326.
Justesen, N., Bontrager, P., Togelius, J., et al., 2019. Deep learning for video game playing. IEEE Trans. Games 12 (1), 1–20.
Károly, A.I., Galambos, P., Kuti, J., et al., 2020. Deep learning in robotics: Survey on model structures and training strategies. IEEE Trans. Syst. Man Cybern. 51 (1), 266–279.
Khoi, P.B., Giang, N.T., van Tan, H., 2021. Control and simulation of a 6-DOF biped robot based on twin delayed deep deterministic policy gradient algorithm. Indian J. Sci. Technol. 14, 2460–2471.
Kober, J., Bagnell, J.A., Peters, J., 2013. Reinforcement learning in robotics: A survey. Int. J. Robot. Res. 32 (11), 1238–1274.
Koczka, G., Auberhofer, S., Biro, O., et al., 2009. Optimal convergence of the fixed-point method for nonlinear eddy current problems. IEEE Trans. Magn. 45 (3), 948–951.
Konar, A., Baghi, B.H., Dudek, G., 2021. Learning goal conditioned socially compliant navigation from demonstration using risk-based features. IEEE Robotics Autom. Lett. 6 (2), 651–658.
Lei, D., Cui, Z., Li, M., 2022. A dynamical artificial bee colony for vehicle routing problem with drones. Eng. Appl. Artif. Intell. 107, 104510.
Li, J., Chen, Y., Zhao, X., et al., 2022. An improved DQN path planning algorithm. J. Supercomput. 78 (1), 616–639.
Li, B., Yang, Z., Chen, D., et al., 2021a. Maneuvering target tracking of UAV based on MN-DDPG and transfer learning. Def. Technol. 17 (2), 457–466.
Li, J., Yu, T., Zhang, X., 2021b. Emergency fault affected wide-area automatic generation control via large-scale deep reinforcement learning. Eng. Appl. Artif. Intell. 106, 104500.
Liu, S., Chang, P., Liang, W., et al., 2021. Decentralized structural-RNN for robot crowd navigation with deep reinforcement learning. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 3517–3524.
Melo, F.S., 2001. Convergence of Q-Learning: A Simple Proof. Tech. Rep., Institute Of Systems and Robotics, pp. 1–4.
Mirowski, P., Grimes, M., Malinowski, M., et al., 2018. Learning to navigate in cities without a map. Adv. Neural Inf. Process. Syst. 31, 1–8.
Mnih, V., Kavukcuoglu, K., Silver, D., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Morales, E.F., Murrieta-Cid, R., Becerra, I., et al., 2021. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. Intell. Serv. Robotic 14 (5), 773–805.
Nguyen, T.T., Nguyen, N.D., Vamplew, P., et al., 2020. A multi-objective deep reinforcement learning framework. Eng. Appl. Artif. Intell. 96, 103915.
Pan, J., Wang, X., Cheng, Y., et al., 2018. Multisource transfer double DQN based on actor learning. IEEE Trans. Neural Netw. Learn. Syst. 29 (6), 2227–2238.
Pfeiffer, M., Schaeuble, M., Nieto, J., et al., 2017. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In: 2017 IEEE International Conference on Robotics and Automation. icra, IEEE, pp. 1527–1533.
Pfeiffer, M., Shukla, S., Turchetta, M., Cadena, C., Krause, A., Siegwart, R., Nieto, J., 2018. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. IEEE Robotics Autom. Lett. 3 (4), 4423–4430.
Qian, S., Bao, K., Zi, B., et al., 2020. Dynamic trajectory planning for a three degrees-of-freedom cable-driven parallel robot using quintic B-splines. J. Mech. Des. 142 (7), 73301.
Ran, A.C.M., Reurings, M.C.B., 2004. A fixed point theorem in partially ordered sets and some applications to matrix equations. In: Proceedings of the American Mathematical Society. pp. 1435–1443.
Rostami, S.M.H., Sangaiah, A.K., Wang, J., et al., 2019. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. EURASIP J. Wireless Commun. Networking 2019 (1), 1–19.
Samsani, S.S., Muhammad, M.S., 2021. Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. IEEE Robotics Autom. Lett. 6 (3), 5223–5230.
Samsonov, V., Ben Hicham, K., Meisen, T., 2022. Reinforcement learning in manufacturing control: Baselines, challenges and ways forward. Eng. Appl. Artif. Intell. 112, 104868.
Schaul, T., Quan, J., Antonoglou, I., et al., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952.
Sekiguchi, K., Deng, M., Inoue, A., 2006. Obstacle avoidance and two wheeled mobile robot control using potential function[C]. In: 2006 IEEE International Conference on Industrial Technology. IEEE, pp. 2314–2319.
Shen, D., Xu, J., 2019. An iterative learning control algorithm with gain adaptation for stochastic systems. IEEE Trans. Automat. Control 65 (3), 1280–1287.
Shi, H., Shi, L., Xu, M., et al., 2019. End-to-end navigation strategy with deep reinforcement learning for mobile robots. IEEE Trans. Ind. Inform. 16 (4), 2393–2402.
Sünderhauf, N., Brock, O., Scheirer, W., et al., 2018. The limits and potentials of deep learning for robotics. Int. J. Robot. Res. 37 (4–5), 405–420.
Tai, L., Li, S., Liu, M., 2017. Autonomous exploration of mobile robots through deep neural networks. Int. J. Adv. Robot. Syst. 14 (4), 1734860947.
Vanvuchelen, N., Gijsbrechts, J., Boute, R., 2020. Use of proximal policy optimization for the joint replenishment problem. Comput. Ind. 119, 103239.
Wang, S., Diao, R., Xu, C., et al., 2020. On multi-event co-calibration of dynamic model parameters using soft actor-critic. IEEE Trans. Power Syst. 36 (1), 521–524.
Xie, L., Miao, Y., Wang, S., et al., 2020. Learning with stochastic guidance for robot navigation. IEEE Trans. Neural Netw. Learn. Syst. 32 (1), 166–176.
Yoo, H., Kim, B., Kim, J.W., et al., 2021. Reinforcement learning based optimal control of batch processes using Monte–Carlo deep deterministic policy gradient with phase segmentation. Comput. Chem. Eng. 144, 107133.
Yu, J., Su, Y., Liao, Y., 2020. The path planning of mobile robot by neural networks and hierarchical reinforcement learning. Front. Neurorobotics 63.
Zhang, X., Wang, J., Fang, Y., et al., 2018. Multilevel humanlike motion planning for mobile robots in complex indoor environments. IEEE Trans. Autom. Sci. Eng. 16 (3), 1244–1258.
Zhou, J., Xue, S., Xue, Y., et al., 2021. A novel energy management strategy of hybrid electric vehicle via an improved TD3 deep reinforcement learning. Energy 224, 120118.
Zhu, Y., Mottaghi, R., Kolve, E., et al., 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 3357–3364.
Zhu, P., Zhang, J., Xiao, Y., et al., 2021. Deep reinforcement learning-based radio function deployment for secure and resource-efficient NG-RAN slicing. Eng. Appl. Artif. Intell. 106, 104490.
Zi, B., Lin, J., S., Qian, 2015. Localization, obstacle avoidance planning and control of a cooperative cable parallel robot for multiple mobile cranes. Robot. Comput.-Integr. Manuf. 34, 105–123.