Journal of Intelligent & Robotic Systems (2023) 109:18
https://doi.org/10.1007/s10846-023-01950-y

REGULAR PAPER

Deep Reinforcement Learning with Heuristic Corrections for UGV Navigation

Changyun Wei[1] · Yajun Li[1] · Yongping Ouyang[1] · Ze Ji[2]

Received: 13 March 2023 / Accepted: 7 August 2023

Abstract
Mapless navigation for mobile Unmanned Ground Vehicles (UGVs) using Deep Reinforcement Learning (DRL) has attracted significantly rising attention in robotic and related research communities. Collision avoidance from dynamic obstacles in unstructured environments, such as pedestrians and other vehicles, is one of the key challenges for mapless navigation. This paper proposes a DRL algorithm based on heuristic correction learning for autonomous navigation of a UGV in mapless configuration. We use a 24-dimensional lidar sensor, and merge the target position information and the speed information of the UGV as the input of the reinforcement learning agent. The actions of the UGV are produced as the output of the agent. Our proposed algorithm has been trained and evaluated in both static and dynamic environments. The experimental result shows that our proposed algorithm can reach the target in less time with shorter distances under the premise of ensuring safety than other algorithms. Especially, the success rate of our proposed algorithm is 2.05 times higher than the second effective algorithm and the trajectory efficiency is improved by 24% in the dynamic environment. Finally, our proposed algorithm is deployed on a real robot in the real-world environment to validate and evaluate the algorithm performance. Experimental results show that our proposed algorithm can be directly applied to real robots robustly.

Keywords Collision avoidance · Deep reinforcement learning · Heuristic correction learning

1 Introduction

Unmanned Ground Vehicles (UGVs) have been extensively applied in the society for a large range of scenarios, such as patrolling robots in public areas, logistics robots in smart warehouses, service robots in shopping malls, and even assistive robots in the medical field. In particular scenarios, UGVs are required to operate in dynamic and unstructured environments, such as food delivery in a restaurant or luggage handling in an airport surrounded by pedestrian crowds. UGVs that perform these tasks automatically need to be able to find optimal paths for navigation and avoid dynamic obstacles safely, when people are surrounding or approaching the robot.

Many recent studies have addressed the autonomous navigation and collision avoidance problem in the robot control field in the past decades. In order to allow more generalisability for autonomous navigation in multiple different environments, various path planning algorithms have been proposed by scholars. These path planning algorithms can be categorized into map-known environment and mapless environment depending on the availability of prior map information of environment. Classical methods for robot autonomous navigation in map-known environments are usually map-based algorithms, such as Dijkstra, A* [1] and RRT* [2] that require pre-built maps using techniques, such as simultaneous localization and mapping (SLAM) [3], so that path planning can be carried out based on pre-known environmental information.

However, in complex scenarios, such as large smart factories, UGVs need to not only observe the information of the

✉ Ze Ji
jiz1@cardiff.ac.uk

Changyun Wei
c.wei@hhu.edu.cn

Yajun Li
liyajun0908@gmail.com

Yongping Ouyang
yongping_ouyang@outlook.com

[1] College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China

[2] School of Engineering, Cardiff University, Cardiff, Wales, UK

Published online: 06 September 2023

surrounding environment, but also avoid factory employees or pedestrians. It requires the UGV to be able to update the map environment in real time. Considering the interaction of social awareness between factory employees or pedestrians and UGVs in indoor environment, the work [4] presents a motion planner with social awareness based on deep reinforcement learning. The experimental results show that UGV can autonomously navigate and avoid obstacles in indoor environment, but it realizes navigation and dynamic obstacle avoidance in indoor environment on the premise of establishing a global map. However, UGVs are unable to create accurate global maps of the environment in many application scenarios. When the environment is unknown, UGV cannot construct the environment in advance and can only move according to the relative position information of the target and its onboard measurement of the local surroundings. Therefore, one of the major challenges for robot autonomous navigation is to develop a safe and robust collision avoidance policy for the robot navigating from its starting position to its destination.

Since conventional approaches heavily rely on global maps, specific rules and heuristics, they often necessitate human intervention and are difficult to adapt to complex and uncertain environments. Deep reinforcement learning (DRL) has emerged as a promising approach to combining deep learning and reinforcement learning techniques. Consequently, DRL allows robots to learn navigation and obstacle avoidance policies through trial-and-error interactions with unknown environments. Moreover, DRL allows robots to sense unknown environments with collected data and learn how to plan paths based on the measured data. Deep neural networks can serve as function approximators that can generalize to different environments. By interacting with the environment, the robots can gradually improve the performance of navigation and obstacle avoidance, and ultimately learn the optimal policy to resolve this problem. DRL algorithms have gained remarkable achievements for various complex tasks, including the most notable match with AlphaGo [5] and many video games [6, 7]. The emergence of DRL algorithms provides another avenue for UGV navigation in unknown scenes that would allow a robot to plan its path without a pre-built map. On the other hand, many DRL algorithms have been introduced in recent years, such as Asynchronous Advantage Actor Critic (A3C) [8], Deep Deterministic Policy Gradient (DDPG) [9], Trust Region Policy Optimization (TRPO) [10], and Proximal Policy Optimization (PPO) [11], and those algorithms have attracted rising attention for solving robot control problems and dynamic obstacle avoidance. DRL-based mapless navigation algorithms [12, 13] are also proposed with stronger adaptability and robustness to unknown environments. One notable early work of mapless navigation [12] deploys the algorithm named ADDPG [14], which, however, does not take dynamic environments into consideration, and all experiments are carried out in static environments.

Despite the great success of DRL algorithms in different domains, DRL also faces several challenges related to its efficiency and practicality, such as Sim-to-Real to transfer and simulation-trained agents to real-world environments [9, 15]. Moreover, reward functions for DRL need to be defined for specific tasks, where empirically determined constant values are employed. In other words, the rewards usually need to be pre-designed and regulated with fine-tuned parameters. The rewards for UGV navigation tasks are usually related to environmental states, e.g., the distance to the goal and the obstacles surrounding the agent, which can be used to construct a unified reward or two individual rewards. For example, methods, such as [4, 13], combine the rewards together as a single value function. However, the rewards of a single valued function is not efficient for training the model of DRL. Despite the final convergence of the training process, the learning speed is usually very slow. On the other hand, raw measurement data of the surrounding environments obtained by the sensors are directly used as the state for reinforcement learning algorithms, and this usually results in a long training time and poor training performance.

In this paper, we propose a novel reward value function with a multi-objective constraint and a correction policy based on sensor information to address the above issues. Considering that the reward of each target task is processed separately, corresponding constraint is added to each task to ensure that the robot can reach the target quickly and accurately. On the one hand, not only do we consider the above rewards related to navigation planning and collision avoidance, we also consider the reward related to the linear and angular velocities of the UGV to regulate the motion. If the speed of the UGV is lower than a threshold, we give it a negative reward value. This would encourage the UGV to learn to move faster for better efficiency. Similarly, a penalty is given when the angular velocity is higher than a threshold. As shown in Fig. 1, we propose a deep reinforcement learning algorithm based on a Heuristic Correction Policy (HCP) to improve the performance of obstacle avoidance. In order to learn a better obstacle avoidance policy, we perform secondary processing on the lidar data by introducing a potential field-based method. When the UGV performs reinforcement learning to output the action command, we correct the actions through the processed lidar data, which enables fast convergence of reinforcement learning during the training process. Finally, various simulation environments and real-world experiments have been performed to verify the algorithm.

The main contributions of this work are summarised as follows.

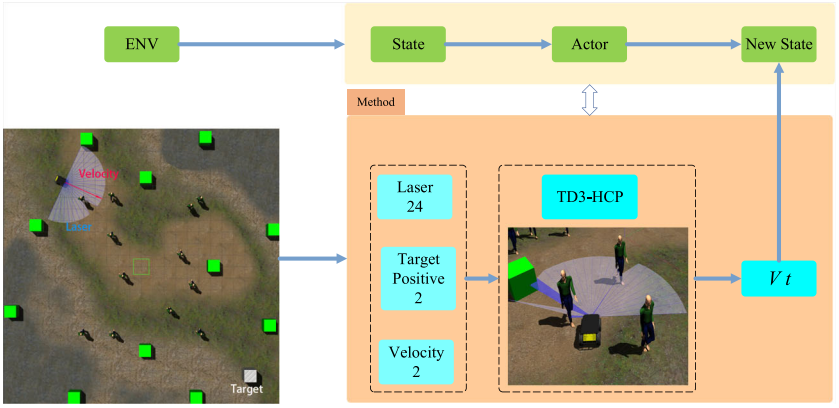


Fig. 1 The framework of our method. The mobile UGVs interact with the environment and the sensors collect the surrounding information, which will be processed and passed to a neural network. Then, based on the current state, the TD3-HCP algorithm outputs an optimal action, which is executed by the mobile UGVs to avoid obstacles

1. The methods described in the studies [1–3] typically require pre-built maps to plan collision-free paths based on the global information. However, building an accurate map can be expensive due to the need of sensor fusion. In contrast, our proposed method can allow the robot to navigate effectively and efficiently in the absence of such global maps.
2. The methods presented in the studies [12–14] mainly address the navigation and collision avoidance mechanism for static obstacles. However, in many practical applications, the environment can be dynamic, such as workers moving in a factory. In this work, we propose heuristic corrections based on local lidar measurement to cope with a sudden potential collision state.
3. In conventional reinforcement learning methods described in the studies [8–11], the optimal control policy has to be trained through excessive interactions with the environment, and exploring all the states in dynamic situations is unaffordable in practical robotic domains. In this work, in order to cope with dynamic uncertain obstacles, the DRL based TD3 network combines with the heuristic corrections to compete the best action to perform, which can significantly improve the success rate, completion time and trajectory costs.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Achieving the mapless navigation and collision avoidance with our method is detailed in Section 3, and the experiments and results are provided in Section 4. Finally, we conclude this work in Section 5.

2 Related Work

In recent years, the applications of unmanned systems have gained significant attention because of its potential advantages in various scenarios [16–18]. For example, in the manufacturing industry, Unmanned Aerial Vehicles (UAVs) and UGVs are used for inventory tracking and monitoring of equipment. In addition, unmanned systems are also applied in fields such as healthcare [19] and sports-assisted training [20]. However, with the widespread application of unmanned systems, a series of challenges and issues have emerged. Some scholars have conducted extensive research on the problem of communication interference [21–24] and data privacy [25, 26]. In these cases, unmanned systems need to communicate and collaborate with other devices to achieve more efficient and intelligent performance. However, when unmanned systems perform complicated and practical tasks, in addition to communication issues, they also need navigation and obstacle avoidance capabilities. Some studies are dedicated to improving the localization accuracy during the motion process of robots [27, 28]. The work [29] presents a robot visual positioning method based on iterative Kalman particle filtering, which achieves global localization and thus improves the accuracy of trajectory tracking.

With the recent advancement of computational hardware technologies, deep neural networks (DNN) have shown great potential in solving complex tasks, such as computer vision and navigation problems. Learning-based collision avoidance techniques have been extensively studied for various robotics fields, such as monocular images [30], depth images [31], as well as for static collision avoidance. In [30], steering angles are generated directly by training a 6-layer convolutional neural network from raw pixels in the static environment, which would take a long time to train the network model. Image semantic information extracted by DNN can be used for autonomous navigation of UGVs towards a target [32]. However, the work [32] is limited that it produces only discrete actions as the output for vehicle control, which are not desirable for navigation and collision avoidance tasks of UGVs. Behavioural models of other agents can be learned [33], where Hamiltonian Markov Chain Monte Carlo sampling is used to produce behaviours that meet social requirements for UGV navigation. An end-to-end motion planning method is proposed to train a navigation model only by acquiring lidar information and target locations [34];

however, it would take significantly long time to collect data. A mapless motion planner is proposed in [12], based on end-to-end learning by using lidar data and relative target information as input data. This policy outputs command data directly and demonstrates its efficiency in unknown simulation and real scenarios. A long-term path planning algorithm is proposed [35] for agent navigation in unexplored environments by combining the deep reinforcement learning model with a greedy strategy.

Deep Q-network (DQN) is used to train the robot navigation policy based on depth images acquired by the vision sensor via successor feature reinforcement learning [36]. A strategy with long short-term memory (LSTM) is proposed to enable autonomous agents to navigate through an arbitrary number of agents in [37], extending previous work with a fixed number of agents. A deep neural network based on the Actor-Critic algorithm is introduced to train a deterministic policy in a model-free manner [9], which is designed for continuous space using raw pixels. A continuous variant of DQN is introduced by deriving Q learning algorithm, which calls the normalized advantage functions (NAF) [38]. The introduction of the experience replay policy greatly improves the success rate of deep reinforcement learning methods. A DRL-based time-efficient navigation policy is proposed with the aim to train an agent to comply with social norms in [4]. It simulates human behaviours and navigation rules for robot navigation, such as overtaking, crossing, and overtaking. Crowd-aware navigation is introduced by incorporating the self-attention mechanism and deep reinforcement learning algorithms for crowd-robot interaction, enabling agents to learn from interpersonal interactions in dense crowds [39]. It turns out that an agent could eventually predict human behaviour and navigate in a crowd. The framework of compound reinforcement learning (CRL) is deployed to promote DRL training in the real environment by combining prior knowledge into the system [40]. In this framework, robots learn appropriate social navigation through sensor input and reward updates based on human feedback.

Despite many advantages of DRL-based planning, such as generalizability, there remain many challenges with DRL methods for mapless navigation. A robot control model based on DDPG is proposed in [13], which constructs a navigation policy that maps lidar measurement and position information. However, the convergence rate of the algorithm is slow and the training efficiency is low. An asynchronous version of the normalized advantage functions for offline policy training based on deep Q-functions is proposed in [14], which can be extended to complex 3D manipulation tasks and can be trained on real robots. The asynchronous DDPG (ADDPG) algorithm is proposed in [41], which uses multiple robots in the same environment to improve the efficiency of experience collection and shorten the algorithm training time. However, it does not consider the navigation collision avoidance rules of UGVs. In this work, we consider that combining traditional collision avoidance algorithms and DRL algorithms would be a promising approach to achieve efficient dynamic obstacle avoidance for mapless navigation.

In this paper, we propose a DRL algorithm based on a heuristic correction policy. Our proposed method not only reduces the training time, but also improves the performance of the final trained model in terms of the success rate and efficiency. In the evaluation experiments, the results show that the model trained using our proposed method significantly outperforms other models in terms of the mission success rate, the completion time, and trajectory efficiency. Finally, the trained model is transplanted to the real environment for further validation. The experimental results show that the proposed algorithm can be directly transplanted to the real robot for experiments with highly promising performance in collision avoidance.

3 Proposed Navigation Approach

In this section, we will introduce the DRL based approach with heuristic corrections for UGV navigation. To this end, we first present the implementation of DRL for UGV navigation, and then detail the network framework of our proposed approach. Afterwards, the improved Prioritized Experience Replay (PER) strategy is described, followed by our proposed approach for UGV navigation with collision avoidance capabilities.

3.1 Sensor-Level Navigation Problem

In this work, collision avoidance for UGVs is defined in the context of an autonomous agent moving on the Euclidean plane. At each time step t, the robot can acquire observation information state s_t from the surrounding environment by sensors, and then the robot outputs an action command a_t that encourages the robot to maneuver towards its goal. The policy is represented by π. The relative distance from the UGV to the goal is denoted by g_t. Instead of having a perfect observation of the whole environment, we assume that the UGV has only partial observation s_t within the range of the lidar sensor and the relative goal position. This assumption makes our method more applicable and realistic for the real world environment. The action a_{t-1} of the previous time step is also considered. Therefore, the problem can be formulated as to find the policy function,

$$a_t \sim \pi_\theta(s_t, a_{t-1}) \tag{1}$$

where s_t includes the lidar range data l_t at time step t and the relative distance g_t. The UGV executes action commands, sampled from the policy function given the partial observation s_t.

3.2 DRL for UGV Navigation

Robot navigation problems require making correct judgements about dynamic environments in real time to avoid collisions with obstacles. Although discrete action space can be used for such problems, in our work, we only consider continuous actions to be generated by DRL agents for interacting with the environment. Several DRL algorithms that support continuous action space can be considered as the base to construct our approach, such as the Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO). In particular, we consider the Twin Delayed Deep Deterministic Policy Gradient (TD3) [42] as the main base for modification. The TD3 algorithm is based on the actor-critic architecture, which is suitable for agents with continuous observation in action space. TD3 combines DDPG and the double Q learning algorithm [43], and achieves promising performance in many continuous control tasks. As an important part of the TD3 algorithm, experience replay has proved to be effective in solving various robot applications. TD3 accumulates environmental information and other data collected by the robot during the task execution into the memory as experience samples, which will be extracted from the memory buffer for training during the learning process.

Our approach is implemented using Python with the TensorFlow and PyTorch frameworks. The UGV with a lidar sensor is simulated in Gazebo, as shown in Fig. 2. The blue lines visualize the lidar scanning beams around the UGV. The simulation process of the UGV subscribes from the ROS sensor node to obtain the lidar measurement, which has a 180° field of view and its range is (0.12m, 3.5m). In this paper, we use 24 lidar rays from $(-90°, 90°)$ to collect data around the robot.

Here we will detail the DRL-based framework for the UGV navigation problem. The following four important design specifications are introduced respectively: 1) observation space, 2) action space, 3) reward design and 4) actor and critic neural network structure.

1) Observation space: The observation state s_t consists of lidar ranger data l_t, velocity v_t and relative target position g_t. Lidar data l_t are pre-filtered and normalized by its maximum range. Velocity v_t includes both the linear speed and the angular speed of the agent. The relative target position g_t is calculated as the relative distance and angle with respect to the current position.

2) Action space: There are two action values in our setting: i) linear velocity l_v, and ii) rotational velocity a_v. The range of velocities are limited to $l_v \in (0, 1)$ m/s and $a_v \in (-\pi/2, \pi/2)$ rad/s to reflect the motion constraints of the robot. The neural network outputs continuous velocity commands to control the robot.

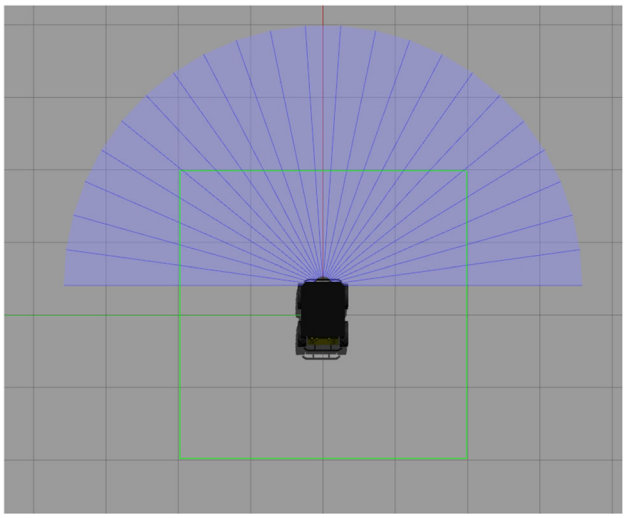


Fig. 2 The simulated husky mobile robot with lidar scanning beams in Gazebo

3) Reward Design: The reward function is constructed by four main parts in our work, encouraging the robot to move towards the goal while also avoiding collisions, as defined below

$$r = r_d + r_c + r_{va} + r_{vl}, \tag{2}$$

where r stands for the total reward, r_d denotes the distance reward, r_c represents the collision reward, and r_{va} and r_{vl} represent the angular velocity reward and linear velocity reward, correspondingly. r_d is calculated using

$$r_d = \begin{cases} r_{ar} & \text{if } d_g < d_{gmin} \\ \triangle d_g & \text{else,} \end{cases} \tag{3}$$

where d_g denotes the distance to the goal. If d_g is less than the threshold d_{gmin}, the robot will be considered to arrive at the goal and it will receive a arrival reward. $\triangle d_g$ is the difference between the distance at the last time step (d_{t-1}) and the distance at the current time step (d_t). The collision reward r_c can be calculated by

$$r_c = \begin{cases} -e^{-k_l(l_{min}-o_l)/l_{max}} & \text{if } l_{min} < k_d \\ 0 & \text{else,} \end{cases} \tag{4}$$

where $l_{min} = \min(l_1, l_2, \ldots l_{24})$ is the minimum lidar range value and k_l and o_l are the gain and distance offset used to determine the curvature of the reward. l_{min} is normalized by its maximum range (l_{max}). If $l_{min} < k_d$, r_c will be calculated

by Eq. 4. Rewards r_{va} and r_{vl} can be calculated by

$$r_{va} = \begin{cases} r_a \text{ if } |a_v| > k_{va}|a_{vmax}| \\ 0 \text{ else,} \end{cases} \tag{5}$$

$$r_{vl} = \begin{cases} r_l \text{ if } l_v < l_{vmin} \\ 0 \text{ else,} \end{cases} \tag{6}$$

where a_{vmax} represents the maximum angular velocity and l_{vmin} denotes the minimum linear velocity. If the angular velocity $|a_v|$ of the UGV is larger than $k_{va}|a_{vmax}|$, we will give it a reward r_a. If the linear velocity l_v of the UGV is lower than l_{vmin}, a reward r_l will be given. We make the rewards r_a and r_l negative, so that the velocities will be constrained to desirable ranges, where slower rotations are more encouraged and larger linear velocities are more desired.

3.3 Neural Network Architecture

In this work, the TD3 algorithm is applied as the base for the modification of the UGV navigation approach. As shown in Fig. 3, the framework of the TD3 algorithm is adapted from the traditional Actor-Critic framework, i,e., the DDPG algorithm, which includes three main parts:

1) Critic network: Similar to the double-Q network, the deployed method can overcome the overestimation problem of the standard DDPG algorithm that tends to overestimate the Q values of actions by the critic network;

2) Delayed update: Similar to the target network update method [9], after the critic network is updated for n times, we delay the update of the actor network, which can effectively improve the stability of the training of the actor network;

3) Noise: To enhance the stability of the algorithm, we apply the same method as with the TD3 algorithm by adding noises to the actor target network.

Figure 4 depicts the architecture of the actor network. The input of the actor network is a multi-dimensional data vector, including a 24-dimensional lidar ranger measurement, a 2-dimensional relative goal position vector and a 2-dimensional velocity vector. The input data are fed into three fully connected layers, and the actor network produces the linear velocity and angular velocity by a sigmoid function and a hyperbolic tangent function, respectively.

Figure 5 shows the architecture of the critic network, where the state input of the critic network is the same as the input of the actor network, except that the two additional velocity commands, produced by the actor network, are also included as part of the input of the critic network. The output of the actor network needs to be connected to the second fully connected layer. Finally, the critic network generates the Q-value through a linear activation function.

At the beginning of each episode, the states of the robot s_t, along with its previous states s_{t-1}, actions a_{t-1}, and rewards of the action r_{t-1}, will be sent to the PER buffer for storing the transitions and the controller to generate the subsequent action a_t. Once the capacity of the PER buffer exceeds the batch size of the samples, while learning from the PER buffer, the actor neural network π_θ will start to produce actions for

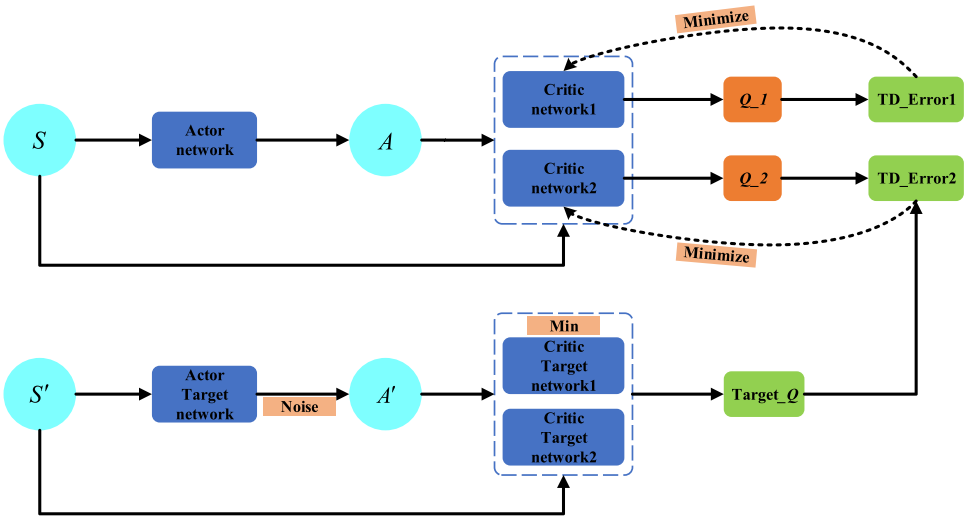


Fig. 3 The architecture of the TD3 algorithm. The goal of the Actor network is to learn an optimal policy that maximizes the long-term cumulative reward from the current state. The Critic network has two outputs representing two Q-value functions, and its objective is to learn an accurate Q-value function to evaluate the actions generated by the Actor network. When calculating the target Q-value in the TD3 algorithm, the minimum Q-value among the two target Critic networks is used as the target Q-value to reduce the risk of overestimation. Then, this target Q-value can be used to calculate the loss function of the Critic network, and update the parameters of the Critic network accordingly

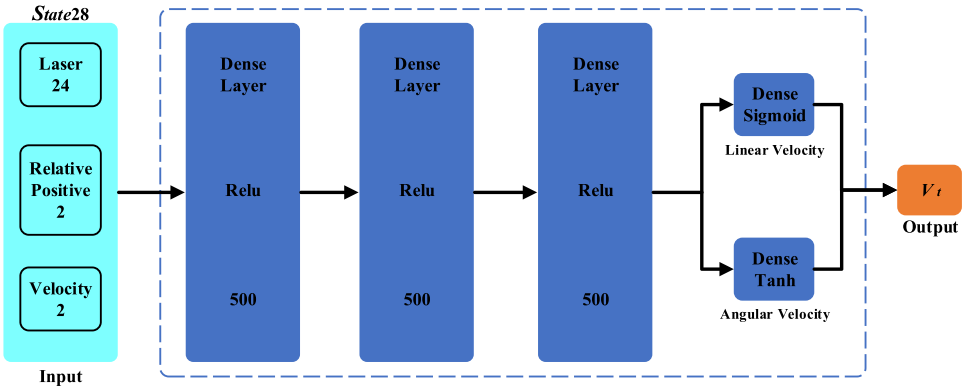


Fig. 4 The architecture of the actor network

the robot. At the same time, when new state transitions are added to the buffer, those transitions with lower probabilities will tend to be replaced.

3.4 Improved Prioritized Experience Replay

Usually, experience transitions are uniformly sampled from a memory buffer without considering their significance that would lead to low efficiency in experience sampling. Instead of uniform sampling of historical data, in order to accelerate training, the PER algorithm [44] is applied to ensure important historical data to be sampled more frequently. Conventionally, the PER algorithm is used as a sampling method aiming to improve the effectiveness in reducing the loss of the critic network in the model.

When a transition is selected for training, the Temporal Difference (TD) error of the transition can be calculated using Eq. 7, which will be recorded as the importance sampling weight of the transition. Along with the experience buffer, priority information of each transition is calculated by Eq. 8 and is stored in the priority tree.

$$L_{oss} = \mathbb{E}[Q^*(s_t, a_t) - Q(s_t, a_t)]^2, \tag{7}$$

$$P(i) = \frac{{p_i}^{\alpha}}{\Sigma_k {p_k}^{\alpha}}, \tag{8}$$

where $Q^*(s_t, a_t)$ denotes the Q-value output of the target Critic network, while $Q(s_t, a_t)$ represents the Q-value output of the current Critic network. L_{oss} indicates the TD error, p_i denotes the sampling probability of the i^{th} transition, and α is an indicator of the priority for the corresponding transition, where uniform sampling is applied when $\alpha = 0$ and greedy strategy sampling is applied when $\alpha = 1$. p_i stands for the priority of a set of transition data, formulated in Eq. 9.

$$p_i = |L_{oss}| + \epsilon, \tag{9}$$

where ϵ represents a small positive sampling probability for each transition, which ensures that all buffered transition experience can still be sampled. However, priority sampling will also compromise the sampling diversity.

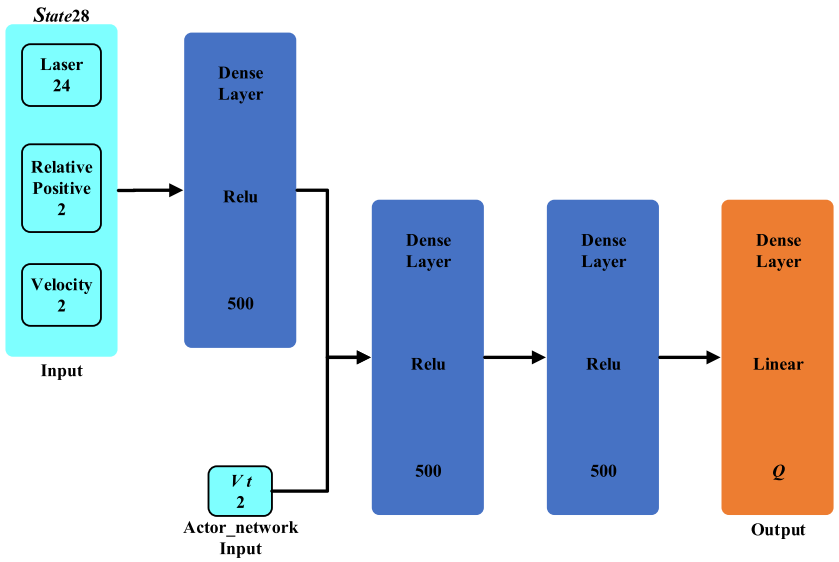


Fig. 5 The architecture of the critic network

In order to deal with the above issue, in our work, the selection process uses stochastic priorities. We consider that the information level contained in each transition should be an indicator for deciding the sampling probability. The more information the transition contains, the more likely it should be selected. In the current implementation (Eq. 9), the informative transitions are not necessarily always selected, as they may not always have high sampling probabilities to be selected. In this regard, in order to consider the information of transition, we optimize Eq. 9 by adding the loss of the actor network, so that the new priority can be calculated by,

$$p_i = L_{oss} + \lambda |A_{loss}|^n + \epsilon \tag{10}$$

where the second term A_{loss} denotes the loss of the actor network, weighted by λ, and n is a bias. Each transition is evaluated by,

$$\omega_i = (\frac{1}{B} \cdot \frac{1}{P(i)})^{\beta} \tag{11}$$

where ω_i denotes the sampling weight for updating the network, indicating the importance of each transition data, B stands for the batch size, β is an coefficient, and $\beta \in [0, 1]$. It should be noted that β affects the degree to which PER cancels out on the convergence. Therefore, we address the diversity loss problem by increasing the information of the transition and sampling weight.

3.5 Algorithm of TD3-HCP

In this work, we propose a local collision avoidance policy based on local lidar measurement, named the Heuristic Correction Policy (HCP). The dynamic obstacle avoidance policy of the HCP is to process the information obtained by the current lidar sensor, when the UGV enters a certain state. In such cases, motion instructions for the UGV will be selected based on a heuristic policy to avoid obstacles. First, we use the distance information obtained by the lidar sensor to define the risk factor f_{risk}, which determines whether the UGV has entered a dangerous state. The risk factors f_{risk} can be calculated by

$$f^i_{risk} = e^{-\left[k_2 - (\frac{l_i}{l_{max}} - k_1)^2\right]}, \tag{12}$$

where l_i denotes the value of the i-th lidar ranger, l_{max} represents the maximum measurable distance of the lidar sensor, k_1 denotes the offset of the risk factor, and k_2 is a bias. As shown in Fig. 6, we apply the clipping operation on f^i_{risk} to limit the risk factors to fall in the range of [0, 1]. When $f^i_{risk} = 1$, it means that the UGV is facing a dangerous state and needs to perform emergency collision avoidance. In this case, the distance from the robot to the nearest target is defined as the safe distance threshold d_{min} and any distance below the threshold is considered a dangerous area.

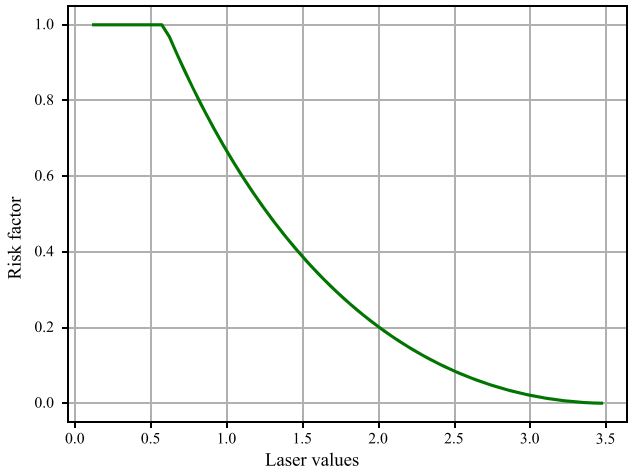


Fig. 6 The risk factor of laser values

When the UGV enters a dangerous state, due to the nature of randomness of DRL, the action output by the DRL agent cannot be always optimal and stable. In such safety-critical situations, it is costly for a robot to rely on a DRL policy. Therefore, we propose the heuristic correction policy that will be executed when the UGV enters an emergency collision avoidance area, i.e., distance below than d_{min}.

When the UGV enters the emergency collision avoidance area, we build a potential field around the UGV to characterize the local environment information. The attractive and repulsive potential fields can be calculated by

$$U_{att} = \frac{1}{2}\zeta(\overrightarrow{d}_{goal} - \overrightarrow{d}_{robot})^m, \tag{13}$$

$$U_{rep} = \begin{cases} \frac{1}{2}\eta(\frac{1}{l_i} - \frac{1}{d_{min}})^n & \text{if } l_i < d_{min} \\ 0 & \text{else,} \end{cases} \tag{14}$$

where U_{att} denotes the attractive potential fields, ζ means the attractive factor, $\overrightarrow{d}_{goal}$ and $\overrightarrow{d}_{robot}$ indicate the goal (X_{goal}, Y_{goal}) and the UGV location (X_{robot}, Y_{robot}) in the world coordinate system, respectively. m and n are adjustable parameters, and l_i denotes the distance to the nearest obstacle. U_{rep} represents the repulsive potential fields, and η denotes repulsive factor.

In situations of more obstacles, the potential field for the robot can be complicated. When a robot moves to a position between the obstacles and the goal, the attractiveness and the repulsive potential forces of the robot may cancel each other out. In this case, the robot may encounter a local minimum problem. Therefore, we improve Eq. 14 to calculate the new

repulsive potential field, as below,

$$U_{rep} = \begin{cases} \frac{1}{2}\eta(\frac{1}{l_i} - \frac{1}{d_{min}})^n(\overrightarrow{d}_{goal} - \overrightarrow{d}_{robot})^k & \text{if } l_i < d_{min} \\ 0 & \text{else.} \end{cases} \tag{15}$$

The new formula has an extra component, $(\overrightarrow{d}_{goal} - \overrightarrow{d}_{robot})^k$, which is used to encourage the robot to move towards the goal by assigning a variable weight to the original repulsive force defined in Eq. 14. Intuitively, when a robot moves toward the goal, while also trying to avoid obstacles, the repulsive force potential field will be decreased accordingly. This enables the robot to approach the target point faster when it gets closer to its destination.

The direction of the potential field for a robot is calculated in the robot's coordinate system. Therefore, before calculating the motion direction, we first need to get the position of the goal relative to the robot, which can be calculated by

$$\begin{pmatrix} x_{goal} \\ y_{goal} \end{pmatrix} = R_T \begin{pmatrix} X_{goal} - X_{robot} \\ Y_{goal} - Y_{robot} \end{pmatrix}, \tag{16}$$

where (x_{goal}, y_{goal}) denotes the position of the goal relative to the robot and R_T is the coordinate transformation matrix. The angle θ_{goal} of the goal relative to the robot can be calculated by

$$\theta_{goal} = \arctan \frac{y_{goal}}{x_{goal}}, \tag{17}$$

Then, we can calculate the direction of the resultant force based on the established potential field. First, the magnitudes of the gradients of both the attractive and repulsive potential fields can be calculated by

$$\overrightarrow{F} = \nabla U_{att} - \sum \nabla U_{rep}, \tag{18}$$

where $\overrightarrow{F}$ denotes the magnitude of the resultant force. The angle θ_H of the resultant force can be obtained by

$$\theta_H = \arctan \frac{F_y}{F_x}, \tag{19}$$

where F_x and F_y represent the resultant forces along the x and y directions, respectively. Finally, we normalize θ_H by Eq. 20 to get the radians of rotation of the robot.

$$\theta_{RH} = \begin{cases} 1 - e^{-\frac{(\theta_H - \xi)^2}{2\sigma^2}} & \text{if } \theta_H > 0 \\ e^{-\frac{(\theta_H - \xi)^2}{2\sigma^2}} - 1 & \text{else,} \end{cases} \tag{20}$$

where θ_{RH} denotes the radian of the next movement direction. We can also find the specific values of θ_{RH} in Fig. 7.

Since DRL algorithms also output action commands according to the current state, to decide on better actions for the robot, we compare the two action instructions as follows. We input the two angle values into a critic network respectively to get the value of the immediate reward so as to judge the optimality of the corresponding angle. The larger the reward value is, the more favourable it is for the robot to avoid obstacles and approach the goal. Figure 8 depicts the network architecture of our proposed TD3-HCP algorithm for the UGV navigation problem. To be specific, if the danger factor in the robot's environment is less than 1, the TD3 network outputs the action. When the danger factor is equal to 1, the heuristic correction policy outputs an angle, which is used to generate a Q-value by inputting the action into the Critic network. The Q-value generated by the heuristic correction policy is compared with the Q-value generated by the Actor network's output action, and the action with the higher Q-value is selected as the robot's action instruction.

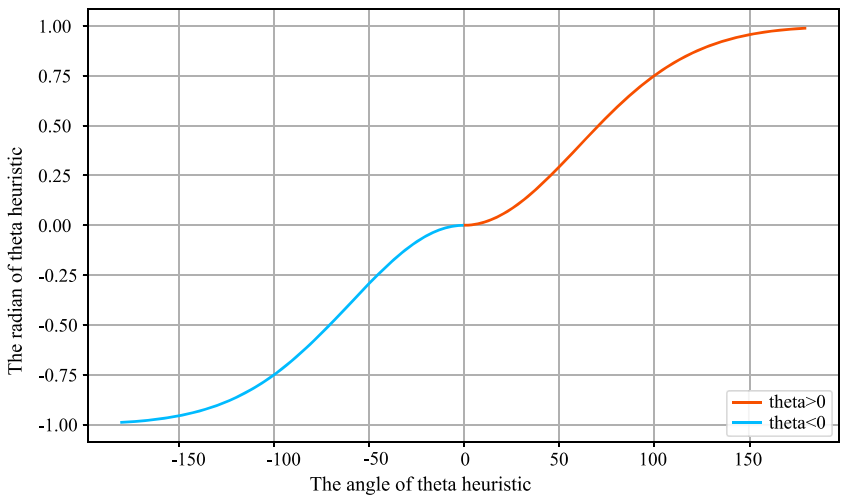


Fig. 7 The radian of the next movement direction (θ_{RH})

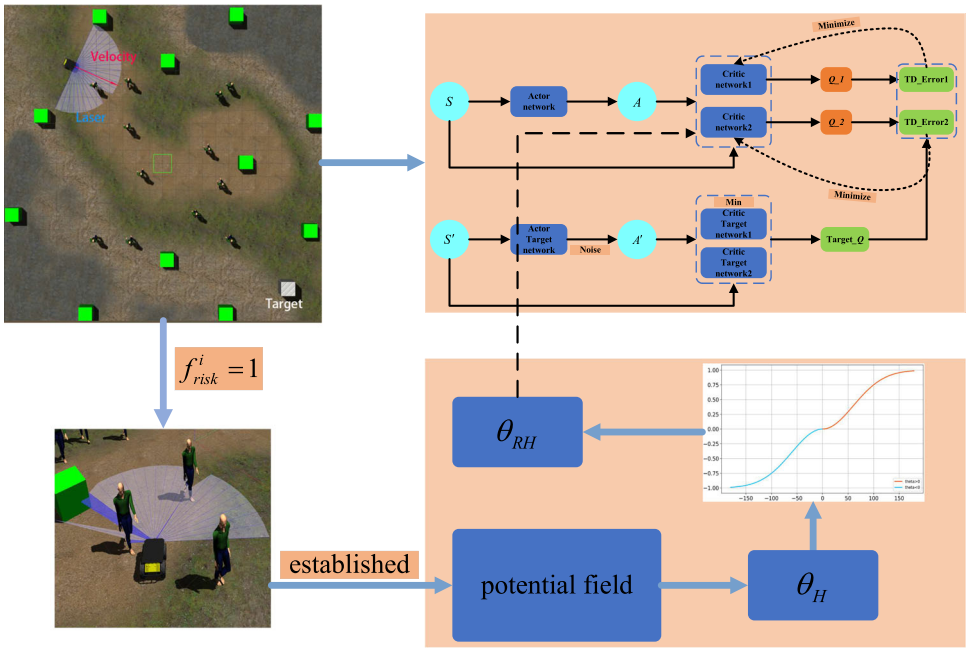


Fig. 8 The proposed TD3-HCP algorithm consists of the TD3 network and a heuristic correction policy

4 Experiments and Results

In this section, we describe the experiments carried out in the work, including both simulation and real-world experiments. We quantitatively compare our method with other methods for performance benchmarking in the Gazebo simulation environments. Finally, real robot experiments are also performed to verify the transferability of the proposed approach.

4.1 Training Configuration

The training process is performed on a computer with Intel Xeon(R) Platinum 8251 CPU and NVIDIA GeForce RTX 3060 GPU. The training process is undertaken in the Gazebo environment on Ubuntu 18.04 as shown in Fig. 9 that simulates the dynamics of the UGV [45]. We expect that the learned policy from the simulations can be transferred to real-world experiments using a real robot with minimal effort.

In the simulations, the green cubes are static obstacles, and the white cube represents the target, which is initialized randomly in the environment for each episode. If the robot hits an obstacle or reaches the target area, the white cube will reset its position to start a new training round.

4.2 Experiment in Static Environments

As shown in Fig. 9, during the training phase, the robot is placed in the center of the map, and the position of the target is randomly initialized to the diagonal area of the environment. We use this static environment to evaluate the performance of training and testing phases.

4.2.1 Evaluation During the Training Phase

The reward of our proposed method and other algorithms in the training phase are recorded for comparison, as shown in

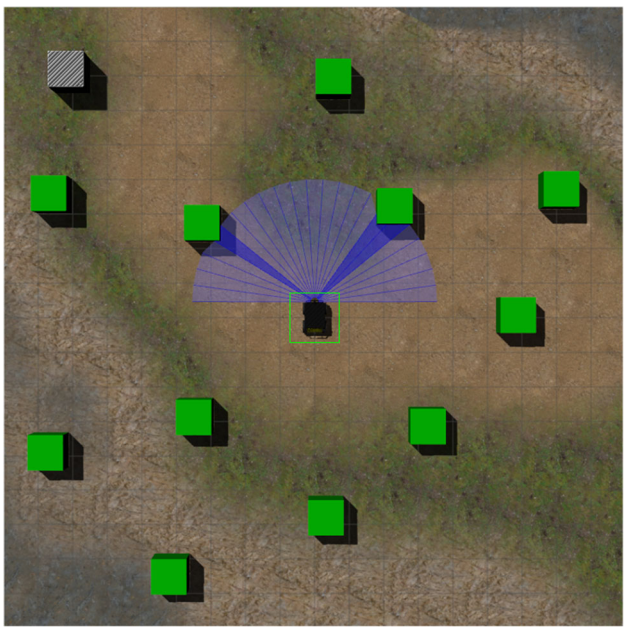


Fig. 9 The Gazebo simulation environment for the UGV navigation problem

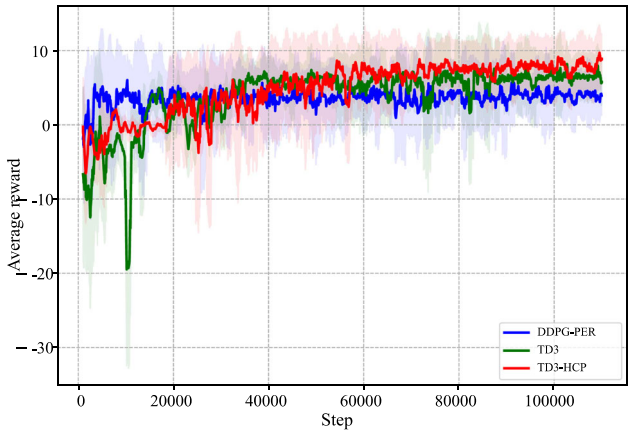


Fig. 10 The reward comparison of the proposed method with other methods

Fig. 10. The average reward of 1000 continuous steps are applied to reduce the effect of the random mission problem. In our work, three algorithms are tested using the same reward function, including the standard Deep Deterministic Policy Gradient (DDPG) with PER (named DDPG-PER), TD3, and our proposed method, i.e., TD3-HCP. We count the average reward value obtained by the three algorithms over 110,000 steps. It can be seen that the average reward value obtained by DDPG-PER training begins to converge after about 10,000 steps, and the reward reaches a stable range of about [4, 6]. The average reward of TD3 algorithms remains between 6 and 8 after only about 40000 steps. In contrast, the average reward of our proposed method TD3-HCP is slower but consistently growing to exceed both the other two algorithm, at about the 60000-th step, reaching the range of from 8 to 10. It is clear that the reward is following an upward trend. Because of the existence of the PER algorithm, DDPG-PER converges quickly compared to other algorithms, but the average reward value it obtains is not as high as ours. As mentioned, overall, our proposed algorithm is slower than other algorithms to a certain extent at the beginning of training, however, the final results are considerably better than other algorithms.

Table 1 Comparison of navigation performance in static environments

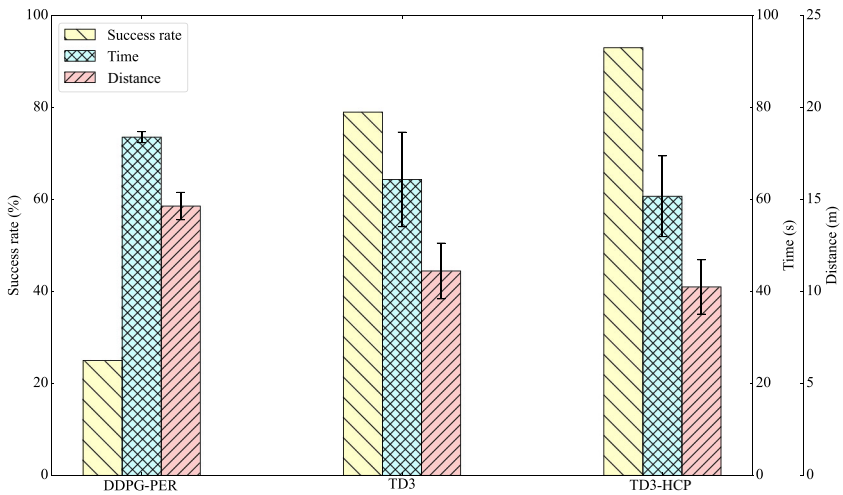
Method	Success Rate	Completion Time(s)	Trajectory Efficiency(m)
DDPG-PER	25%	73.55±1.20	14.64±0.75
TD3	79%	64.36±10.29	11.11±1.52
HCP-TD3	93%	60.70±8.78	10.25±1.48

4.2.2 Evaluation During the Testing Phase

After training the three DRL policies, we also carry out evaluation experiments for further testing. Figure 11 and Table 1 show the performance of those algorithms in the static environment. We consider three performance metrics in our work, including i) success rate, ii) completion time, and iii) trajectory efficiency. The success rate refers to the percentage of total rounds in which the mobile robot successfully can reach the target point without any collisions. The completion time and trajectory efficiency refer to the average time and distance without collision by a mobile robot to reach the target location, respectively. Thus, task success rate is an indicator of the robustness of the controller, while task completion time and trajectory efficiency could represent the performance optimality of each training strategy. The mean and standard deviation of each indicator are calculated for a total of 100 runs.

As detailed in Table 1, in terms of success rate, our method scores the highest rate (93%), while DDPG-PER and TD3 score far lower than our method, at 25% and 79%, respectively. In terms of the completion time, our method consumes the least time (60.70±8.78s) in comparison with

Fig. 11 Comparison of navigation performance in static environments



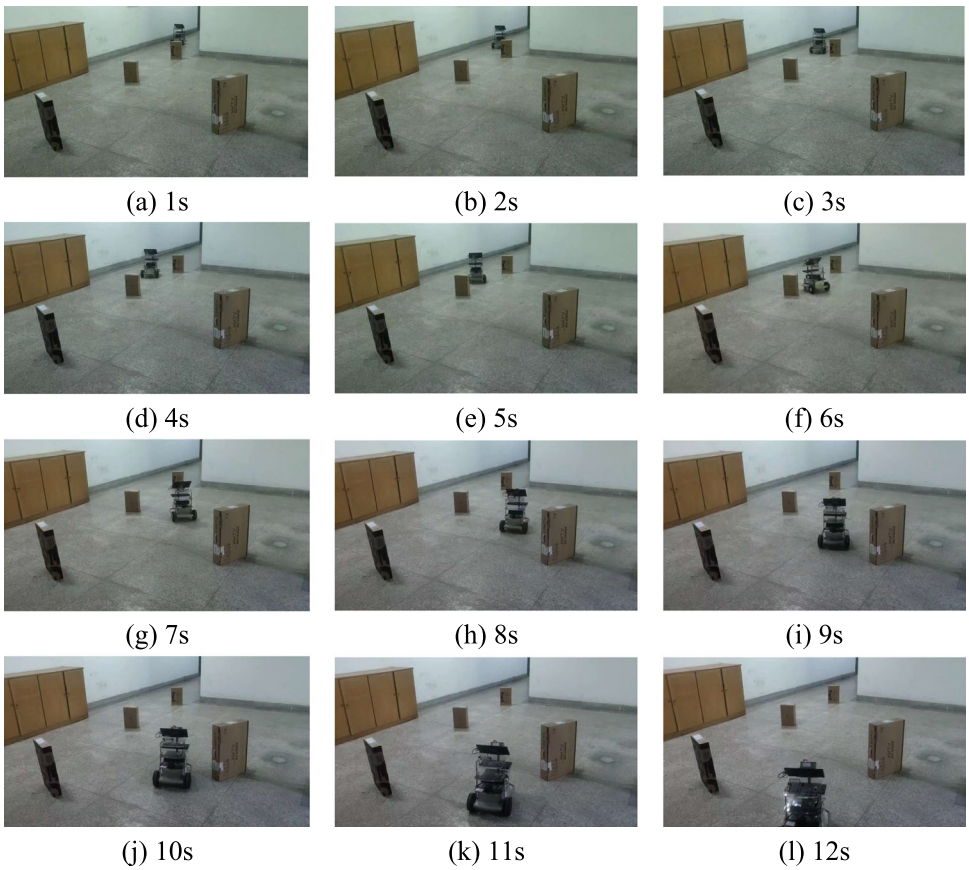


Fig. 12 Real robot testing of collision avoidance in a static environment

TD3 (64.36±10.29s) and DDPG-PER (73.55±1.20s). With regard to the trajectory efficiency, our method shows a lower mean value (10.25±1.48m) than both TD3 (11.11±1.52m) and DDPG-PER (14.64±0.75m). Therefore, we can conclude that our method can ensure the shortest driving path and the highest success rate under the premise of avoiding potential collisions.

4.2.3 Real Robot Testing in Static Environments

After testing the trained models in the static simulation environment, we have migrated the models trained in the virtual environment to real robots to verify the robustness of the proposed approach in real-world scenarios. The results are shown in Fig. 12.

The conducted experiment is carried out in an indoor setting, featuring a corridor containing four immovable obstacles. Upon receiving the assigned target location, the robot enters the testing area through the corridor entrance in one second. Subsequently, within 2nd and 4th second, the first obstacle is detected within the robot's field of view, prompting it to shift leftward to circumvent the obstacle. At about the 5th second, another obstacle is detected on the robot's left side, causing it to change direction and move rightward. This pattern repeats once more at about the 8th second, where yet another obstacle is detected and evaded. Ultimately, in approximately 12th second, the robot safely reaches the intended destination, having successfully navigated around all of the static obstacles. The experimental outcomes demonstrate that our proposed approach is capable of transferring trained models from simulations to real-world robots and effectually accomplishing navigation missions in static environments.

4.3 Experiment in Dynamic Environments

We have also modified the original static environment for simulations, and changed the original map size of $15 \times 15\, m^2$ to a map of $24 \times 24\, m^2$. As depicted in Fig. 13, in the dynamic environment, the white cube still represents the target goal, which is placed above the ground in the air, and the green cubes indicate static obstacles. In addition, 12 randomly moving pedestrians are added to the map as dynamic obstacles. During the algorithm testing in dynamic environments, the robot is positioned randomly on the diagonal of the environment at the start of each episode. The target location, represented by a white cube, is then placed at a different position also on the diagonal. This protocol for initializing both the robot and the target ensures that the algorithm must

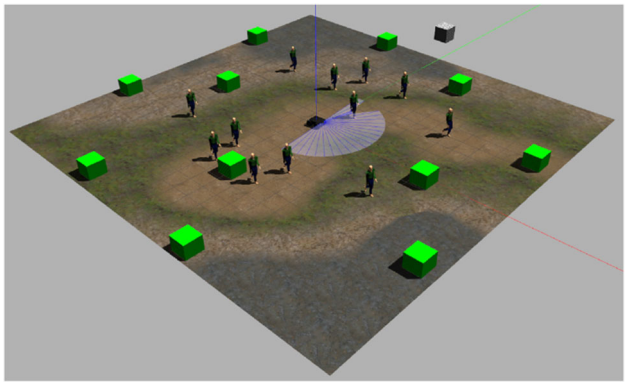


Fig. 13 The dynamic environment with randomly moving pedestrians

adapt to new scenarios and conditions in every episode, thus enhancing its robustness and versatility.

4.3.1 Evaluation in Dynamic Simulated Environment

Figure 14 illustrates the performance metrics of three algorithms. Detailed results are listed in Table 2, the success rate of our method reaches 74%, which is considerably higher than DDPG-PER (18%) and TD3 (36%). The results show that our method can more effectively avoid obstacles in complex dynamic environments. In other words, our method led to score 2.05 times higher with regard to the success rate than the TD3 and 4.1 times than the DDPG-PER.

With respect to the completion time to reach the target area, our method consumes substantially less time (112.33±4.72s) than DDPG-PER (198.44±10.35s) and TD3 (192.85±21.73s). In terms of trajectory efficiency, our method results in a considerably lower mean travel distance (19.45±1.65m) than that of DDPG-PER (30.44±9.34m) and TD3 (25.35±2.61m). This result clearly demonstrates that the our proposed method produces more efficient trajectories in less time, while also ensuring path safety. Furthermore, our method has a much lower standard deviations than other

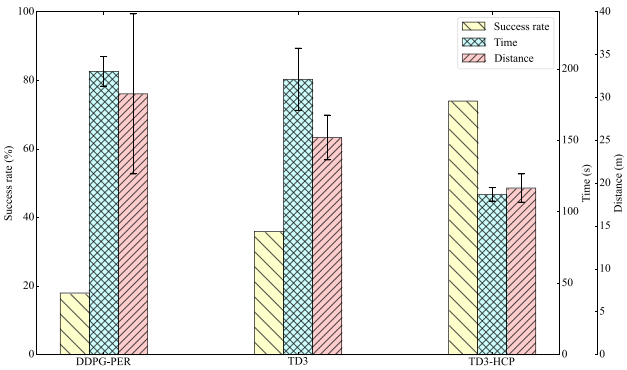


Fig. 14 Performance comparison in the dynamic simulated environment

Table 2 Performance comparison in the dynamic simulated environment

Method	Success Rate	Completion Time(s)	Trajectory Efficiency(m)
DDPG-PER	18%	198.44±10.35	30.44±9.34
TD3	36%	192.85±21.73	25.35±2.61
HCP-TD3	74%	112.33±4.72	19.15±1.65

methods in terms of task completion time and trajectory efficiency. This clearly evidences that the model trained using our method is more robust.

4.3.2 Evaluation in Dynamic Factory-Like Environment

In order to further verify the generalization of the proposed approach, we conduct a second dynamic scenario in a factory-like environment, as shown in Fig. 15. This environment has more complicated settings, which includes obstacles of various shapes, such as shelves, and moving obstacles, such as pedestrians. In each round, the robot's initial position, target location, and obstacles in the environment will be set randomly. Each method has been tested for 100 rounds.

Figure 16 and Table 3 show the performance of our proposed method in comparison with other two methods in the factory-like environment. We can find that the success rate of our method is 82%; In contrast, the DDPG-PER (32%) and TD3 (48%) are considerably lower than our method. Moreover, our method consumes the least completion time to reach the target area (103.34±0.73s) among all three methods, where the DDPG-PER takes 121.81±1.33s and the TD3 needs 104.72±1.70s. In terms of trajectory efficiency, our method results in a considerably lower mean travel distance (33.18±1.92m) than that of DDPG-PER (42.87±3.02m) and TD3 (36.73±2.54m).

Fig. 15 The factory-like environment with complicated configurations

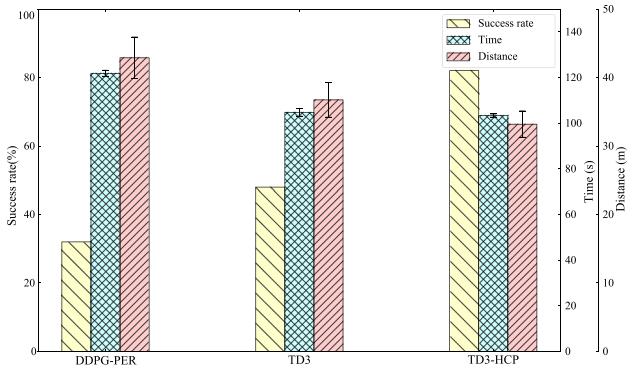


Fig. 16 Algorithm performance comparison in dynamic environment

4.3.3 Real Robot Testing in Dynamic Environments

We further test the performance of the model trained from the virtual environment on a real robot in a dynamic environment so as to verify the robustness of our proposed algorithm. As shown in Fig. 17, compared with the static environment, two pedestrians will suddenly appear to block the robot's path in this dynamic environment. Of course, there are also multiple static obstacles on the floor.

The experimental results reveal that, around the three-second mark, the robot detects both a static obstacle on its right and a dynamic obstacle ahead. To avoid collision, between the 4th and 5th second, the robot executes turning and reversing manoeuvres. At the seven-second mark, the robot detects another dynamic obstacle and comes to a halt while navigating through a turn. The robot's avoidance strategy successfully circumvents all obstacles, with the task being completed around the nine-second mark. Particularly noteworthy is that the robot has demonstrated distinct behaviours while encountering different types of obstacles. Ultimately, the robot successfully accomplishes the assigned task within 12 seconds.

Table 3 A performance compare between algorithms in smart factory

Method	Success Rate	Completion Time(s)	Trajectory Efficiency(m)
DDPG-PER	32%	121.81±1.33	42.87±3.02
TD3	48%	104.72±1.70	36.73±2.54
HCP-TD3	82%	103.34±0.73	33.18±1.92

4.4 Discussion

Overall, the conducted experiment showcases that the proposed TD3-HCP approach offers promising performance. Specifically, it not only considerably reduces the mission completion time for the robot to reach the target area, but also enables routes free of collision and shorter paths while maintaining high success rates. Compared to the other DRL-based algorithms, our method has the highest average reward, indicating its better convergence in the training period. Addi-

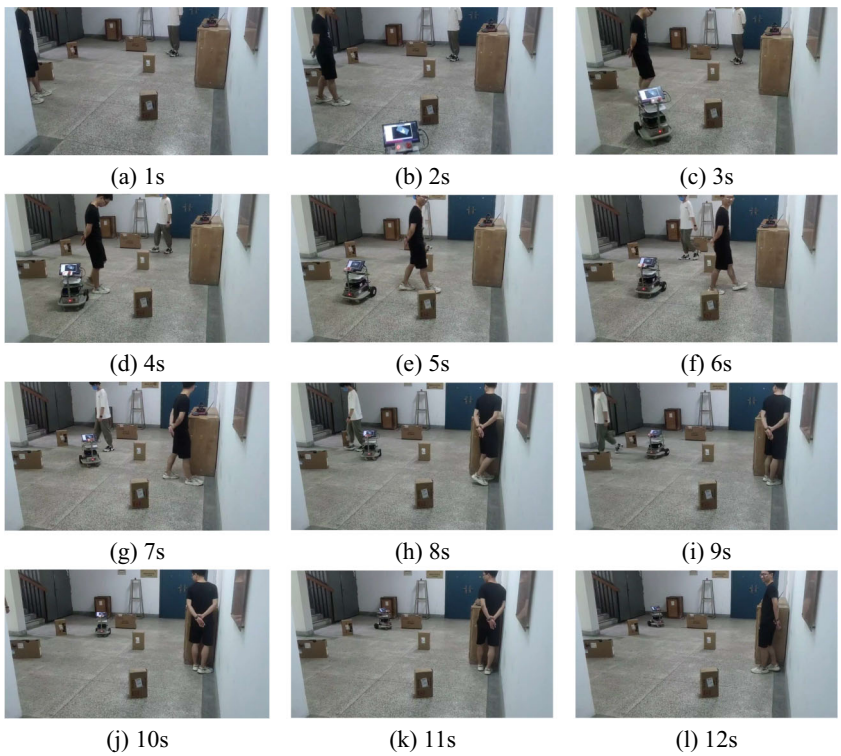


Fig. 17 Real robot testing in a dynamic environment with two pedestrians that will suddenly appear to block the robot's path

tionally, the results also show that the trained model from virtual simulations can be directly transferred to real-world scenarios, and the robot can demonstrate impressive ability to avoid obstacles in a dynamic environment, even in response to sudden obstructions.

Nevertheless, our proposed method has some limitations. First of all, although our method does not require an accurate global map for path planning and collision avoidance, the robot still needs to know the precise location of the destination or estimate the distance from its current location to the destination when designing the reward function. Secondly, since our method only employs 2D lidar for environmental perception, we do not consider the roughness of the ground during planning efficient collision-free paths.

5 Conclusion and Future Work

In this paper, we propose a novel DRL-based approach to address the problem of mapless navigation. The results show that the proposed TD3-HCP approach effectively reduces the training time compared to two poplar DRL-based methods, namely DDPG-PER and TD3. We have tested the navigation performance in both static and dynamic simulations, in which our TD3-HCP demonstrates a significant superior performance in reducing the mission completion time for the robot to reach the goal consistently than the baselines. In the dynamic environment scenarios, the success rate is 2.05 times higher than TD3 and the trajectory efficiency is improved by 24%, keeping the completion time to the minimum level. Moreover, the results also reveal that our TD3-HCP requires the shortest time and distance of the traversing trajectory while also ensuring a high success rate and safety of collision avoidance. In addition, we migrate the control model trained in the virtual environment to a real robot, further verifying the generalization and robustness of the proposed method.

In future studies, it is recommended to further investigate the convergence speed while exploring multi-sensor information fusion. Specifically, combining vision with lidar range sensor may elevate the robustness of the method in practical applications. Moreover, we can allow multiple agents to train asynchronously in various scenarios so as to improve the learning efficiency.

Author Contributions All authors contributed to the study conception and design. Changyun Wei and Yongping Ouyang performed material preparation, data collection and analysis. Changyun Wei and Yongping Ouyang wrote the first draft of the manuscript. Yajun Li and Ze Ji revised the manuscript based on previous versions. All authors read and approved the final manuscript.

Funding This work was supported in part by the National Natural Science Foundation of China under Grant 61703138.

Code Availability The simulation data will be available upon request.

Declarations

Ethical approval Not applicable. Our manuscript does not report results of studies involving humans or animals.

Consent to participate Not applicable. Our manuscript does not report results of studies involving humans or animals.

Consent for Publication All authors have approved and consented to publish the manuscript.

Conflicts of interest The authors have no relevant financial or nonfinancial interests to disclose.

References

1. Likhachev, M., Ferguson, D.I., Gordon, G.J., Stentz, A., Thrun, S.: Anytime dynamic a*: An anytime, replanning algorithm. In: International Conference on Automated Planning and Scheduling (ICAPS), vol. 5, pp. 262–271 (2005)
2. Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., Muhammad, M.S.: Rrt*-smart: A rapid convergence implementation of rrt. Int J Adv Robot Syst **10**(7), 1651–1656 (2013)
3. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. IEEE Robot Autom Mag **13**(2), 99–110 (2006)
4. Chen, Y.F., Everett, M., Liu, M., How, J.P.: Socially aware motion planning with deep reinforcement learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1343–1350 (2017). IEEE
5. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
6. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning (2013). arXiv:1312.5602
7. Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., et al.: Mastering complex control in moba games with deep reinforcement learning. In: AAAI Conference on Artificial Intelligence, vol. 34, pp. 6672–6679 (2020)
8. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning (ICML), pp. 1928–1937 (2016). PMLR

9. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015). arXiv:1509.02971
10. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning (ICML), pp. 1889–1897 (2015). PMLR
11. Wang, Y., He, H., Tan, X.: Truly proximal policy optimization. In: Uncertainty in Artificial Intelligence, pp. 113–122 (2020). PMLR
12. Tai, L., Paolo, G., Liu, M.: Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 31–36 (2017). IEEE
13. Zhang, P., Wei, C., Cai, B., Ouyang, Y.: Mapless navigation for autonomous robots: A deep reinforcement learning approach. In: Chinese Automation Congress (CAC), pp. 3141–3146 (2019). IEEE
14. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396 (2017). IEEE
15. Chaffre, T., Moras, J., Chan-Hon-Tong, A., Marzat, J.: Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation (2020). arXiv:2004.14684
16. Pakrooh, R., Bohlooli, A.: A survey on unmanned aerial vehicles-assisted internet of things: A service-oriented classification. Wirel Pers Commun **119**, 1541–1575 (2021)
17. Alam, T.: Blockchain-enabled deep reinforcement learning approach for performance optimization on the internet of things. Wirel Pers Commun **126**(2), 995–1011 (2022)
18. Swarup, A., Gopal, M.: Control strategies for robot manipulators-a review. IETE J Res **35**(4), 198–207 (1989)
19. An, X., Wang, Y.: Smart wearable medical devices for isometric contraction of muscles and joint tracking with gyro sensors for elderly people. J Ambient Intell Hum Comput, 1–12 (2021)
20. Ding, H.: Motion path planning of soccer training auxiliary robot based on genetic algorithm in fixed-point rotation environment. J Ambient Intell Hum Comput **11**, 6261–6270 (2020)
21. Pawar, P., Yadav, S.M., Trivedi, A.: Performance study of dual unmanned aerial vehicles with underlaid device-to-device communications. Wirel Pers Commun **105**, 1111–1132 (2019)
22. Alimi, I.A., Teixeira, A.L., Monteiro, P.P.: Effects of correlated multivariate fso channel on outage performance of space-air-ground integrated network (sagin). Wirel Pers Commun **106**(1), 7–25 (2019)
23. Li, H., Luo, J., Li, J.: Reinforcement learning based full-duplex cognitive anti-jamming using improved energy detector. Wirel Pers Commun **111**, 2107–2127 (2020)
24. Li, L., Mao, Y.: Autonomously coordinating multiple unmanned vehicles for data communication between two stations. Wirel Pers Commun **97**, 3793–3810 (2017)
25. Praise, J.J., Raj, R.J.S., Benifa, J.B.: Development of reinforcement learning and pattern matching (rlpm) based firewall for secured cloud infrastructure. Wirel Pers Commun **115**, 993–1018 (2020)
26. Tasgaonkar, P.P., Garg, R.D., Garg, P.K.: Vehicle detection and traffic estimation with sensors technologies for intelligent transportation systems. Sens & Imaging **21**, 1–28 (2020)
27. Annepu, V., Rajesh, A.: Implementation of an efficient artificial bee colony algorithm for node localization in unmanned aerial vehicle assisted wireless sensor networks. Wirel Pers Commun **114**, 2663–2680 (2020)
28. Kumar, A.: Real-time performance comparison of vision-based autonomous landing of quadcopter on a ground moving target. IETE J Res, 1–18 (2021)
29. Li, X.: Robot target localization and interactive multi-mode motion trajectory tracking based on adaptive iterative learning. J Ambient Intell Hum Comput **11**, 6271–6282 (2020)
30. Muller, U., Ben, J., Cosatto, E., Flepp, B., Cun, Y.: Off-road obstacle avoidance through end-to-end learning. Adv Neural Inf Process Syst **18**, 739–746 (2005). (**MIT Press**)
31. Tai, L., Li, S., Liu, M.: A deep-network solution towards model-less obstacle avoidance. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2759–2764 (2016). IEEE
32. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: IEEE International Conference on Computer Vision, pp. 2722–2730 (2015). IEEE
33. Kretzschmar, H., Spies, M., Sprunk, C., Burgard, W.: Socially compliant mobile robot navigation via inverse reinforcement learning. Int J Robot Res **35**(11), 1289–1307 (2016)
34. Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., Cadena, C.: From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1527–1533 (2017). IEEE
35. Zhu, D., Li, T., Ho, D., Wang, C., Meng, M.Q.-H.: Deep reinforcement learning supervised autonomous exploration in office environments. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 7548–7555 (2018). IEEE
36. Zhang, J., Springenberg, J.T., Boedecker, J., Burgard, W.: Deep reinforcement learning with successor features for navigation across similar environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2371–2378 (2017). IEEE
37. Everett, M., Chen, Y.F., How, J.P.: Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3052–3059 (2018). IEEE
38. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous deep q-learning with model-based acceleration. In: International Conference on Machine Learning (ICML), pp. 2829–2838 (2016). PMLR
39. Chen, C., Liu, Y., Kreiss, S., Alahi, A.: Crowd-robot interaction: Crowdaware robot navigation with attention-based deep reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 6015–6022 (2019). IEEE
40. Ciou, P.-H., Hsiao, Y.-T., Wu, Z.-Z., Tseng, S.-H., Fu, L.-C.: Composite reinforcement learning for social robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2553–2558 (2018). IEEE
41. Chen, Y.F., Liu, M., Everett, M., How, J.P.: Decentralized noncommunicating multiagent collision avoidance with deep reinforcement learning. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 285–292 (2017). IEEE
42. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning (ICML), pp. 1587–1596 (2018). PMLR
43. Hasselt, H.: Double q-learning. Adv Neural Inf Process Syst **23**, 2613–2621 (2010)
44. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay (2015). arXiv:1511.05952
45. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an opensource multi-robot simulator. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2149–2154 (2004). IEEE

Changyun Wei received the B.Eng. degree in mechanical engineering and automation and the master's degree in mechanical engineering from Hohai University, Nanjing, China, in 2008 and 2010, respectively, and the Ph.D. degree in artificial intelligence (AI) from the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, in 2015. He is an Associate Professor with the College of Mechanical and Electrical Engineering, Hohai University. His present research interests include autonomous unmanned systems, intelligent robots, and multirobot systems.

Yajun Li received his B.Eng. degree in vehicle engineering from Ningbo University of Technology, Ningbo, China, in 2019. He is currently pursuing the master degree with the College of Mechanical and Electrical Engineering, Hohai University, Nanjing, China. His research interests include intelligent autonomous unmanned systems and distributed control.

Yongping Ouyang received the B.Eng. degree in mechanical engineering from Hohai University, Nanjing, China, in 2020, where he is currently pursuing the master degree with the College of Mechanical and Electrical Engineering. His research interests include intelligent autonomous unmanned systems and multi-robot collaboration.

Ze Ji received the B.Eng. degree from Jilin University, Changchun, China, in 2001, the M.Sc. degree from the University of Birmingham, Birmingham, U.K., in 2003, and the Ph.D. degree from Cardiff University, Cardiff, U.K., in 2007. He is a Senior Lecturer (Associate Professor) with the School of Engineering, Cardiff University, U.K. Prior to his current position, he was working in industry (Dyson, Lenovo, etc) on autonomous robotics. His research interests are cross-disciplinary, including autonomous robot navigation, robot manipulation, robot learning, simultaneous localization and mapping (SLAM), acoustic localization, and tactile sensing.