



Deep Reinforcement Learning of Map-Based Obstacle Avoidance for Mobile Robot Navigation

Guangda Chen¹ · Lifan Pan¹ · Yu'an Chen¹ · Pei Xu² · Zhiqiang Wang¹ · Peichen Wu¹ · Jianmin Ji¹ · Xiaoping Chen¹

Received: 21 November 2020 / Accepted: 11 August 2021 / Published online: 18 August 2021
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2021

Abstract

Autonomous and safe navigation in complex environments without collisions is particularly important for mobile robots. In this paper, we propose an end-to-end deep reinforcement learning method for mobile robot navigation with map-based obstacle avoidance. Using the experience collected in the simulation environment, a convolutional neural network is trained to predict the proper steering operation of the robot based on its egocentric local grid maps, which can accommodate various sensors and fusion algorithms. We use dueling double DQN with prioritized experienced replay technology to update parameters of the network and integrate curriculum learning techniques to enhance its performance. The trained deep neural network is then transferred and executed on a real-world mobile robot to guide it to avoid local obstacles for long-range navigation. The qualitative and quantitative evaluations of the new approach were performed in simulations and real robot experiments. The results show that the end-to-end map-based obstacle avoidance model is easy to deploy, without any fine-tuning, robust to sensor noise, compatible with different sensors, and better than other related DRL-based models in many evaluation indicators.

Keywords Robot navigation · Obstacle avoidance · Deep reinforcement learning · Grid map

Introduction

Robot navigation is the key and essential technology for autonomous robots and is widely used in industrial, service, or field applications [1]. One of the main challenges

of mobile robot navigation is to develop a safe and reliable collision avoidance strategy to navigate from the starting position to the desired target position without colliding with obstacles and pedestrians in unknown complicated environments. Although numerous methods have been proposed to solve this practical problem [2, 3], conventional methods are usually based on a set of assumptions that may not be met in practice [4], and may require a lot of computing needs [5]. Besides, conventional algorithms usually involve many parameters that need to be adjusted manually [6] rather than

This work is partially supported by the 2030 National Key AI Program of China 2018AAA0100500, the National Natural Science Foundation of China (No. 61573386), and Guangdong Province Science and Technology Plan Projects (No. 2017B010110011).

✉ Jianmin Ji
jianmin@ustc.edu.cn

Guangda Chen
cgdsss@mail.ustc.edu.cn

Lifan Pan
lifanpan@mail.ustc.edu.cn

Yu'an Chen
an11099@mail.ustc.edu.cn

Pei Xu
xp816@mail.ustc.edu.cn

Zhiqiang Wang
tt1248163264@mail.ustc.edu.cn

Peichen Wu
wpc16@mail.ustc.edu.cn

Xiaoping Chen
xpchen@ustc.edu.cn

- ¹ School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, Anhui, People's Republic of China
- ² School of Data Science, University of Science and Technology of China, Hefei 230026, Anhui, People's Republic of China

being able to learn automatically from past experience [7]. These methods are difficult to generalize well to unpredictable situations.

Recently, several supervised and self-supervised deep learning methods have been applied to robot navigation. However, these methods have some limitations that it is difficult to widely use in real robot environments. For example, the training of the supervised learning approaches requires a massive manually labeled dataset. On the other hand, deep reinforcement learning (DRL) methods have achieved significant success in many challenging tasks, such as Go game [8], video games [9], and robotics [10]. Unlike previous supervised learning methods, DRL-based methods learn from a large number of trials and corresponding feedback (rewards), rather than from labeled data. To learn sophisticated control strategies through reinforcement learning, the robot needs to interact with the training environment for a long time to accumulate experience about the consequences of taking different actions in different states. Collecting such interactive data in the real world is very expensive, time-consuming, and sometimes impossible due to security issues [11]. For example, Kahn et al. [7] proposed a generalized computational graph that includes model-based methods and value-based model-free methods and then instantiated the graph to form a navigation model that is learned from the original image and is highly sample efficient. However, it takes tens of hours of destructive self-supervised training to navigate only tens of meters without collision in an indoor environment. Because learning good policies requires a lot of trials, training in simulation worlds is more appropriate than gaining experience from the real world. The learned policies can be transferred according to the close correspondence between the simulator and the real world. And there are different transfer abilities to the real world between different types of input data.

According to the type of input perception data, the existing obstacle avoidance methods of mobile robots based on deep reinforcement learning can be roughly divided into two categories: agent-level inputs and sensor-level inputs. In particular, an agent-level method takes into account positions and the movement data, like velocities, accelerations and paths, of obstacles or other agents. Although the agent-level methods have the disadvantages of requiring precise and complex front-end perception processing modules, they still have the advantages of sensor type independence (which can be adapted to different front-end perception modules), easy-to-design training simulation environment, and easy migration to the real environment. However, a sensor-level method uses the sensor data directly. Compared with agent-level methods, sensor-level methods do not require a complex and time-consuming front-end perception module. However, because sensor data are not abstracted, such methods are generally not compatible with other sensors, or even multi-sensors carried by robots. Currently, more and

more robots are equipped with different and complementary sensors and capable of navigating in complex environments with high autonomy and safety guarantee [12]. But all existing sensor-level works depend on specific sensor types and configurations.

In this paper, we propose an end-to-end map-based deep reinforcement learning algorithm to improve the performance of mobile robot decision-making in complicated environments, which directly maps local probabilistic grid maps with target position and robot velocity to an agent's steering commands. Compared with previous works on DRL-based obstacle avoidance, our motion planner is based on egocentric local grid maps to represent observed surroundings, which enables the learned collision avoidance policy to handle different types of sensor input efficiently, such as the multi-sensor information from 2D/3D range scan finders or RGB-D cameras. And our trained CNN-based policy is easily transferred and executed on a real-world mobile robot to guide it to avoid local obstacles for long-range navigation and robust to sensor noise. We evaluate our DRL agents both in simulation and on robot qualitatively and quantitatively. Our results show the improvement in multiple indicators over the DRL-based obstacle avoidance policy.

Our main contributions are summarized as follows:

- Formulate the obstacle avoidance of mobile robots as a deep reinforcement learning problem based on the generated grid map, which can be directly transferred to real-world environments without adjusting any parameters, compatible with different sensors and robust to sensor noise.
- Integrate curriculum learning technique to enhance the performance of dueling double DQN with a prioritized experienced replay.
- Conduct a variety of simulation and real-world experiments to demonstrate the high adaptability of our model to different sensor configurations and environments.

The rest of this article is organized as follows. “[Related Work](#)” introduces our related work. “[System Structure](#)” describes the structure of the DRL-based navigation system. “[Map-Based Obstacle Avoidance](#)” introduces the deep reinforcement learning algorithm for obstacle avoidance based on egocentric local grid maps. “[Experiments](#)” presents the experimental results and “[Conclusion](#)” gives the conclusions.

This paper is a significantly expanded version of our previous conference paper [13].

Related Work

In recent years, deep neural networks have been widely applied in the supervised learning paradigm to train a collision avoidance policy that maps sensor input to the robot's control commands to navigate a robot in environments with static obstacles. Learning-based obstacle avoidance approaches try to optimize a parameterized policy for collision avoidance from experiences in various environments. Giusti et al. [14] used deep neural networks to classify the input color images to determine which action would keep the quadrotor on the trail. To obtain a large number of training samples, they equipped the hikers with three head-mounted cameras. The hikers then glanced ahead and quickly walked on the mountain path to collect the training data. Gandhi et al. [15] created tens of thousands of crashes to make one of the biggest drone crash datasets for training unmanned aerial vehicles to fly. Tai et al. [16] proposed a hierarchical structure that combines a decision-making process with a convolutional neural network, which takes the original depth image as the input. They use expert teaching methods to remotely control the robot to avoid obstacles and collect experience data manually. Although the trouble of manually collecting labeled data can be alleviated by self-supervised learning, the performance of learned models is largely limited by the strategy of generating training labels. Pfeiffer et al. [17] proposed a model capable of learning complex mappings from raw 2D laser range results to the steering commands of a mobile indoor robot. During the generation of training data, the robot is driven to a randomly selected target position on the training map. The simulated 2D laser data, relative target positions and, steering velocities generated by conventional obstacle avoidance methods (dynamic window method [18]) are recorded. The performance of the learned models is largely limited by the generating process of training data (steering commands) from conventional methods, and thus the robot is just capable of passing some simple static obstacles in the very empty corridor. Liu et al. [19] proposed a deep imitation learning method for obstacle avoidance based on local grid maps. To improve the robustness of the policy, they expanded the training data set with the artificially generated map, which effectively alleviates the shortage of training samples in normal demonstrations. However, the robot using their method is only tested in an empty and static environment, and thus the method proposed by Liu et al. [19] is difficult to be extended to complicated environments.

On the other hand, obstacle avoidance algorithms based on deep reinforcement learning have attracted widespread attention. The existing DRL-based methods are roughly divided into two categories: agent-level inputs and

sensor-level inputs. In particular, an agent-level method takes into account positions and the movement data, like velocities, accelerations and paths, of obstacles or other agents. However, a sensor-level method uses the sensor data directly. As representatives of agent-level methods, Chen et al. [20] trained an agent-level collision avoidance policy of mobile robots using deep reinforcement learning. They learned a value function for two agents, which explicitly maps the agent's own state and the states of its neighbors into a collision-free action. And then they generalize the policy of two agents to the policy of multiple agents. However, their method requires perfect perception data, and the network input has dimensional constraints. In their follow-up work [21], multiple sensors were deployed to estimate the states of nearby pedestrians and obstacles. Such a complex perception and decision-making system not only requires expensive online calculations, but also makes the entire system less robust to perceptual uncertainty. Although the agent-level methods have the disadvantages of requiring precise and complex front-end perception processing modules, they still have the advantages of sensor type independence (which can be adapted to different front-end perception modules), easy to design simulation environment training, and easy migration to the real environment.

As for sensor-level input, the types of sensor data used in DRL-based obstacle avoidance mainly include 2D laser ranges, depth images and color images. Tai et al. [22] proposed a learning-based map-free motion planner by taking sparse 10-dimensional range findings and a target position relative to the mobile robot coordinate system as inputs and taking steering commands as outputs. Long et al. [23] proposed a decentralized sensor-level collision avoidance policy for multi-robot systems, which directly maps the measurement values of the original 2D laser sensor to the steering commands of the agents. Xie et al. [11] proposed a novel assisted reinforcement learning framework in which classic controllers are used as an alternative and switchable strategy to speed up DRL training. Furthermore, their neural network also includes 1D convolutional network architectures, which is used to process raw perception data captured by the 2D laser scanner. The 2D laser-based method is competitive in being portable to the real world because the difference between the laser data in the simulator and the real world is very small. However, 2D sensing data cannot cope with complex 3D environments. Instead, vision sensors can provide 3D sensing information. However, during the training process, RGB images will encounter significant deviations from the actual situation and the simulated environment, which leads to a very limited generalization of various situations. In [24], models trained in the simulator are migrated through the conversion of actual and simulated images, but their experimental environment is a single football field and

cannot be generalized to all real environments. Compared with RGB images, the depth image is easier to transfer the trained model to actual deployment, because the depth image is agnostic to texture variation and thus has a better visual fidelity, thus greatly reducing the burden of transferring the trained model to actual deployment [25]. Zhang et al. [26] proposed a deep reinforcement learning algorithm based on depth images, which can learn to transfer knowledge from previously mastered obstacle avoidance tasks to new problem instances. Their algorithm greatly reduces the learning time required after solving the first task instance, which makes it easy to adapt to the changing environment. Currently, all existing sensor-level work depends on specific sensor types and configurations. However, equipped with different and complementary sensors, the robots are capable of navigating in complex environments with high autonomy and safety guarantee [12]. Compared with agent-level methods, sensor-level methods do not require a complex and time-consuming front-end perception module. However, because sensor data are not abstracted, such methods are generally not compatible with other sensors, or even multi-sensors carried by robots.

System Structure

Our proposed DRL-based mobile robot navigation system contains six modules. As shown in Fig. 1, the simultaneous localization and mapping (SLAM) module builds an environment map based on sensor data and can estimate the robot's position and speed in the map at the same time. When the target location is received, the global path planner module generates a path or a series of local target points from the current location to the target location based on a pre-built map from the SLAM module. To deal with dynamic and complex environments, a safe and robust collision avoidance module is needed in an unknown cluttered environment. In addition to the local target points from the path planning module and the robot speed provided by the SLAM positioning module, our local collision avoidance

module also needs the surrounding environment information, which is represented as the egocentric grid map generated by the grid map generator module based on various sensor data. In general, our DRL-based local planner module needs to input the speed of the robot generated by the SLAM module, the location of the local target generated by the global path planner, and the grid map of the map generator module that can fuse multi-sensor information, and outputs the control commands of the robot: linear velocity v and angular velocity w . Finally, the output speed command is executed by the base controller module which maps the control speed to the instructions of the wheel motor based on the specific kinematics of the robot.

Map-Based Obstacle Avoidance

We begin this section by introducing the mathematical definition of the local obstacle avoidance problem. Next, we describe the DQN reinforcement learning algorithm with improvement methods (including double DQN, dueling network and prioritized experienced replay) and key ingredients (including the observation space, action space, reward function and network architecture). Finally, the curriculum learning technique is introduced.

Problem Formulation

At each timestamp t , given a frame of sensing data s_t , a local target position \mathbf{p}_g in the robot coordinate system and the linear velocity v_t , angular velocity ω_t of the robot, the proposed map-based local obstacle avoidance policy can provide a velocity action command a_t as follows:

$$\mathbf{M}_t = f_\lambda(s_t), \quad (1)$$

$$a_t = \pi_\theta(\mathbf{M}_t, \mathbf{p}_g, v_t, \omega_t), \quad (2)$$

where λ and θ are the parameters of grid map generator and policy model, respectively. Specifically, the grid map \mathbf{M}_t is constructed as a collection of robot configurations and the perceived surrounding obstacles, which will be explained below.

Therefore, the robot collision avoidance problem can be simply expressed as a sequential decision problem. The sequential decision problem involving observing observations $\mathbf{o}_t \sim [\mathbf{M}_t, \mathbf{p}_g, v_t, \omega_t]$ and the actions (velocities) $a_t \sim [v_{t+1}, \omega_{t+1}]$ ($t = 0 : t_g$) can be regarded as a trajectory l from starting position \mathbf{p}_s to desired target \mathbf{p}_g , where t_g is the travel time. Our goal is to minimize the expectation of arrival time and take into account that the robot will not collide with other objects in the environment, which is defined as:

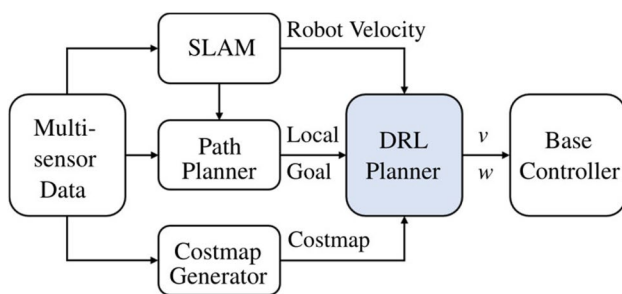


Fig. 1 The DRL-based navigation system for autonomous robots

$$\begin{aligned} & \underset{\pi_\theta}{\operatorname{argmin}} \mathbb{E}[t_g | a_t = \pi_\theta(\mathbf{o}_t), \\ & \mathbf{p}_t = \mathbf{p}_{t-1} + a_t \cdot \Delta t, \\ & \forall k \in [1, N] : (\mathbf{p}_o)_k \notin \Omega(\mathbf{p}_t)], \end{aligned} \tag{3}$$

where \mathbf{p}_o is the position of obstacle and Ω indicates the size and shape of the robot, $(\mathbf{p}_o)_k \notin \Omega(\mathbf{p}_t)$ represents that the robot with Ω shape does not collide with any obstacle.

Dueling DDQN with Prioritized Experienced Replay

The Markov Decision Process (MDP) provides a mathematical formulation and framework for modeling stochastic planning and decision-making problems under uncertainty. MDP is a five-tuple $M = (S, A, P, R, \gamma)$, where S represents the state space, A indicates the action space, and P is the transition function, which describes the probability distribution of the next state when the action a is taken in the current state s , R represents the reward function, and γ indicates the discount factor. In the MDP problem, the policy $\pi(a|s)$ represents the basis on which an agent takes action, the agent will choose an action based on policy $\pi(a|s)$. The most common policy expression is a conditional probability distribution, which is the probability of taking action a in state s , i.e. $\pi(a|s) = P(A_t = a | S_t = s)$. At this time, the action with a high probability is selected by an agent with a higher probability. The quality of policy π can be evaluated by the action-value function (Q -value) defined as:

$$Q^\pi(s, a) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right]. \tag{4}$$

That is, the Q -value function is the expectation of the sum of discounted rewards. The goal of the agent is to maximize the expected sum of future discounted rewards, which can be achieved by using the Q-learning algorithm that iteratively approximates the optimal Q -value function by the Bellman equation:

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}). \tag{5}$$

Combined with deep a neural network, the Deep Q Network (DQN) enables conventional reinforcement learning to cope with complex high-dimensional decision-making problems. Generally, the DQN algorithm contains two deep neural networks, including an online deep neural network with parameters θ which are constantly updated by minimizing the loss function $(y_t - Q(s_t, a_t; \theta'))^2$, and a separate target deep neural network with parameters θ' , which are fixed for generating Temporal-Difference (TD) targets and regularly assigned by those of the online deep neural network. The y_t in the loss function is calculated as follows:

$$y_t = \begin{cases} r_t & \text{if episode ends} \\ r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta') & \text{otherwise} \end{cases}. \tag{6}$$

Due to the maximization step in Eq. (6), traditional Q-learning is affected by overestimation bias, which can hurt the learning process. Double Q-learning [27] solves this overestimation problem by decoupling by choosing actions from its evaluation in the maximization performed for the bootstrap target. Therefore, if the episode does not end, y_t in the above formula is rewritten as follows:

$$y_t = r_t + \gamma Q(s_{t+1}, \underset{a_{t+1}}{\operatorname{argmax}} Q(s_{t+1}, a_{t+1}; \theta); \theta'). \tag{7}$$

In this work, dueling networks [28] and prioritized experienced replay [29] are also deployed for more reliable estimation of Q -values and sampling efficiency of replay buffer, respectively. Dueling DQN proposes a new network architecture. They divided the Q -value estimation into two parts: $V(s)$ and $Adv(s, a)$, one is to estimate how good it is to be in state s , and the other is to estimate the advantage of taking action a in that state. The two parts are based on the same front-end perception module of the neural network and finally fused to represent the final Q -value. The result of this architecture is that state values can be learned independently without being confused by the effects of action advantages. This in particular leads to the identification of state information where actions have no effect on.

The idea of Prioritized Experienced Replay is to prioritize experiences that contain more important information than others. Each experience is stored with an additional priority value, so the higher priority experiences have a higher sampling probability, and have the opportunity to stay in the buffer longer than other experiences. As importance measure, the TD error can be used. It can be expected that if the TD error is high, the agent can learn more from the corresponding experience, because the agent's performance is better or worse than expected. Prioritized Experienced Replay can speed up the learning process.

In the following, we describe the key ingredients of the DQN reinforcement learning algorithm, including the details of the observation and action space, the reward function and the network architecture.

Observation Space

As mentioned in “[Problem Formulation](#)”, the observation \mathbf{o}_t includes the generated grid maps \mathbf{M}_t which is compatible with multiple sensors and easy to simulate and migrate to real robots, the relative goal position \mathbf{p}'_g and the current velocity of the robot \mathbf{v}_t . Specifically, \mathbf{M}_t represents the grid map images with the size and shape of robot generated from a 2D laser scanner or other sensors. The relative target

position \mathbf{p}'_g is a two-dimensional vector representing the target coordinates relative to the current position of the robot. The observation \mathbf{v}_t includes the current linear velocity and angular velocity of the differentially driven robot.

We use layered grid maps [30] to represent the environmental information perceived by multiple sensors. Then, through the map generator module, we obtain the state maps \mathbf{M}_t by drawing the robot configuration (size and shape) into the layered grid maps. Figure 2b shows an example of a grid map generated from a single 2D laser scanner.

Action Space

In this work, the action space of the robot agent is a set of allowable velocity in discrete space. The action of the differential robot only consists of two parts: translation velocity and rotation velocity, i.e. $\mathbf{a}_t = [v_t, w_t]$. In the process of implementation, considering the kinematics and practical application of the robot, we set the range of the rotation velocity $\omega \in [-0.9, -0.6, -0.3, 0.0, 0.3, 0.6, 0.9]$ (in radians per second) and the translation velocity $v \in [0.0, 0.2, 0.4, 0.6]$ (in meters per second). It is worth mentioning that since the laser rangefinder carried by our robot cannot cover the back of the robot, it is not allowed to move backward (i.e. $v < 0.0$).

Reward

The objective in reinforcement learning is to maximize expected discounted long-term return and the reward function in reinforcement learning implicitly specifies what the agent is encouraged to do. In our long-distance obstacle avoidance task, the reward signal is often weak and sparse unless the target point is reached or a collision occurs. In this work, the reward shaping technique [31] is used to solve this problem by adding an extra reward signal $(r^g)_t$ based on prior knowledge. The value of $(r^g)_t$ indicates that the target point is attractive to the agent. Specifically, the closer to the target

point, the larger the value of $(r^g)_t$, the further away from the target point, the smaller the value of $(r^g)_t$. That is

$$(r^g)_t = \epsilon(\|\mathbf{p}_{t-1} - \mathbf{p}_g\| - \|\mathbf{p}_t - \mathbf{p}_g\|). \quad (8)$$

In addition, our objective is to avoid collisions during navigation and minimize the average arrival time of the robot. The reward r at time step t is a sum of four terms: r^g , r^a , r^c and r^s . That is

$$r_t = (r^g)_t + (r^a)_t + (r^c)_t + (r^s)_t. \quad (9)$$

In particular, when the robot reaches the target point, the robot is awarded by $(r^a)_t$:

$$(r^a)_t = \begin{cases} r_{\text{arr}} & \text{if } \|\mathbf{p}_t - \mathbf{p}_g\| < 0.2 \\ 0 & \text{otherwise} \end{cases}. \quad (10)$$

If the robot collides with other obstacles, it will be punished by $(r^c)_t$:

$$(r^c)_t = \begin{cases} r_{\text{col}} & \text{if collision} \\ 0 & \text{otherwise} \end{cases}. \quad (11)$$

And we also give robots a small fixed penalty r^s at each step. We set $r_{\text{arr}} = 500$, $\epsilon = 10$, $r_{\text{col}} = -500$ and $r^s = -5$ in the training procedure.

Network Architecture

We define a deep convolutional neural network that represents the action-value function for each discrete action. The input of the neural network has three local maps \mathbf{M} , which have $60 \times 60 \times 3$ gray pixels and a four-dimensional vector with local target \mathbf{p}'_g and robot speed \mathbf{v} . The output of the network is the Q values of each discrete action. The architecture of the deep Q value network is shown in Fig. 3. The input grid maps are fed into an 8×8 convolution layer with stride 4, then followed by a 4×4 convolution layer with stride 2 and a 3×3 convolution layer with stride 1. The local target and the robot speed form a four-dimensional vector, which is processed by a fully connected layer, and then the output is tiled to a special dimension and added to the response map of the processed grid maps. The results are then processed by three 3×3 convolution layers with stride 1 and two fully connected layers with 512, 512 units, respectively, and then fed into the dueling network architecture, which divides the Q value estimation into two parts: $V(s)$ and $Adv(s, a)$, one is to estimate how good it is to be in state s , and the other is to estimate the advantage of taking action a in that state. The two parts are based on the same front-end perception module of neural network and finally fused together to represent the final Q value of 28 discrete actions. Once trained the network can predict Q values for different actions in the input

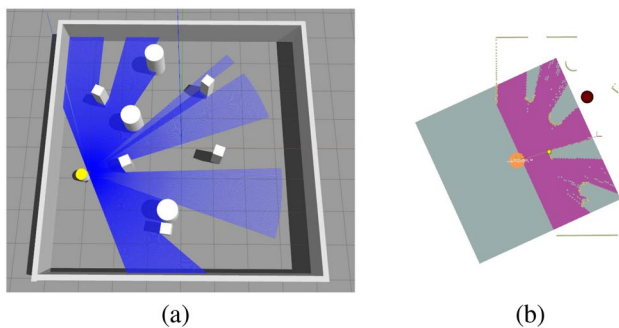


Fig. 2 Gazebo training environments (left) and corresponding grid map displayed by Rviz (right)

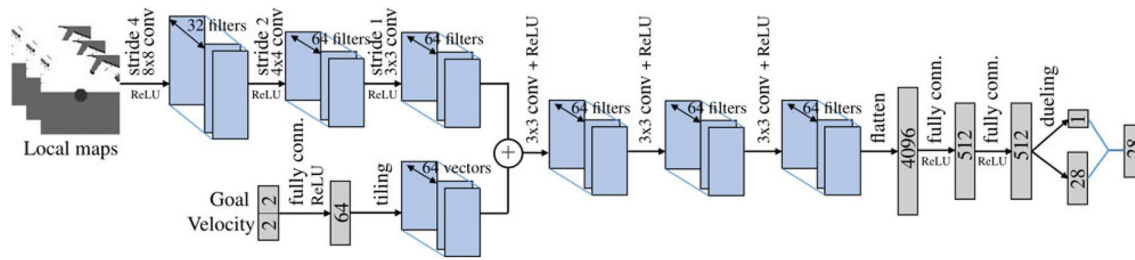


Fig. 3 The architecture of our CNN-based dueling DQN network. This network takes three local maps and a vector with a local goal and robot velocity as input and outputs the Q values of 28 discrete actions

state (three local maps \mathbf{M} , local goal \mathbf{p}'_g and robot velocity \mathbf{v}). The action with the maximum Q value will be selected and executed by the obstacle avoidance module.

Curriculum Learning

Curriculum learning [32] aims to improve learning performance by designing appropriate curriculums for progressive learners from simple to difficult. Elman et al. [33] put forward the idea that a curriculum of progressively harder tasks could significantly accelerate a neural network's training. And curriculum learning has recently become prevalent in the machine learning field, which assumes that the curriculum learning can improve the convergence speed of the training process and find a better local minimum value than the existing solvers. In this work, we use Gazebo simulator [34] to build a training world with multiple obstacles. As the training progresses, we gradually increase the number of obstacles in the environment and also increase the distance from the starting point of the target point gradually. This makes our strategy training from easy situation to difficult ones. At the same time, the position of each obstacle and the start and endpoints of the robot are automatically random during all training episodes. One training scene is shown in Fig. 2a.

Experiments

In this section, we will introduce the experimental setup and evaluation in simulation and real environments. We quantitatively and comparatively evaluated the DQN-based obstacle avoidance policy to show that it is superior to other related methods on multiple indicators, robust to sensor noise and compatible with different sensors. Moreover, we also performed qualitative tests on real robots, including static obstacles of cardboard boxes and dynamic pedestrian scenarios, and also integrated our obstacle avoidance policy into the navigation framework for long-range real-world navigation testing. It should be noted that we only use global path planning to provide local target points for our local DRL-based

obstacle avoidance module in the long-range navigation experiment at the end of “Navigation in real world”. In other experiments, we only use the local obstacle avoidance module to control the robot, that is, there is only one final target point.

Reinforcement Learning Setup

The training experiments were performed using a simulated differentially driven robot in a simulated virtual environment using Gazebo. As shown in Fig. 2a, a 180-degree 2D laser scanner is installed on the front of the differential drive robot. Based on performance and our computing resource constraints, the system parameters are empirically determined as shown in Table 1.

The implementation of our deep neural network is in TensorFlow, and we trained the deep Q network in terms of the objective function (Eq. 7) with Adam optimizer [35]. The training hardware is a computer with an i9-9900k CPU and a single NVIDIA GeForce 2080 Ti GPU. The entire training process (including exploration and training time) takes about 10 hours to allow the policy to converge to robust performance.

Experiments on Simulation Scenarios

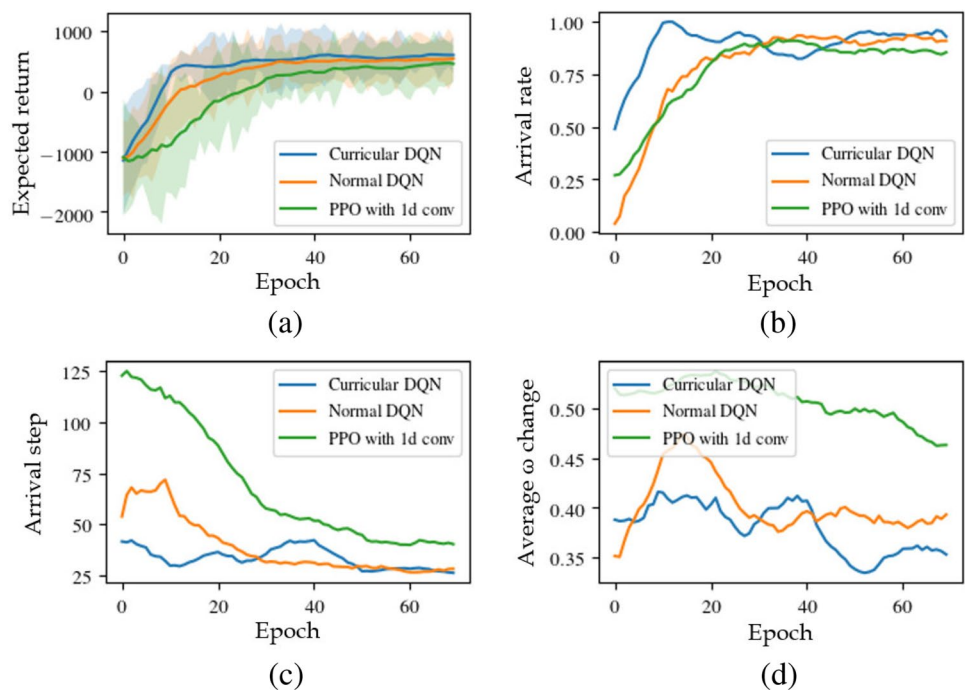
Comparative Experiments

To quantitatively compare the performance of our approach with other related methods in various test cases, we define the following performance indicators for obstacle avoidance behavior.

- *Expected return* E_r is the average value of the sum of each episode's rewards.
- *Success rate* $\bar{\pi}$ is the probability of the episodes in which the robot reaching the goals within a limited step without any collisions.
- *Arrival step* \bar{s} is the average number of steps required to successfully reach the target point without any collisions.

Table 1 System parameters

Hyperparameter	Value	Description
Learning rate	5×10^{-4}	The learning rate of optimizer
Discount factor	0.99	Discount factor gamma for updating Q-learning
Minibatch size	1024	The number of training cases needed for each random gradient descent update was calculated
Replay buffer size	2×10^5	The random gradient descent updates are sampled from this number of the buffer
Image size	60×60	Length and width of the map image used for training
History length	3	Number of latest frames required for the input of the Q-network
Episode length	300	The longest trajectory length per episode
Initial exploration	1	Initial ϵ in ϵ -greedy exploration
Final exploration	0.1	Final ϵ in ϵ -greedy exploration
Control frequency	5	Control frequency of robot speed command

Fig. 4 Indicator curves of different methods during training. Our DQN-based policy has significant improvement over PPO policy with 1D convolution network on multiple indicators

- *Average angular velocity change* $\nabla\omega$ is the average value of the angular velocity changes for each step, which reflects the smoothness of the trajectory.

In the training test, we compared our curricular DQN policy with normal (non-curricular) DQN policy and PPO with a one-dimensional convolutional network [23]. As shown in Fig. 4, our DQN-based obstacle avoidance policy has significant improvements over the PPO policy in terms of expected return, success rate, arrival step, and average angular velocity change, and the curricular DQN policy also has slight improvements in multiple indicators. In the tests with more obstacles, all trained agents have been tested in 10 random worlds with 12 random obstacles. Each trained agent

Table 2 Experimental indicators' values of different methods

Evaluating	PPO with	Normal	Curricular
Indicator	1D conv	DQN	DQN
E_r	467.87	547.43	617.04
$\bar{\pi}$	0.85	0.91	0.94
\bar{s}	40.1	27.76	26.13
$\nabla\omega$	0.46	0.39	0.35

had to solve 200 static tasks per world. the specific indicators values of various methods are shown in the Table 2. It shows that our map-based DQN algorithm is better than PPO algorithm with a one-dimensional convolution network in

various indicators, and the use of curricular learning technology will further improve the result. Figure 5 shows a test case of our curricular DQN policy in a test scenario.

Robustness to Noise

Figure 6 depicts the performance of our DQN-based policy and the traditional vector field histogram (VFH) method [36] in the environment with 12 random obstacles as the noise error of the laser sensor data. Sensor noise is based on Gaussian distribution with a fixed variance. When the variance is zero, it means that there is no noise. The results show that our DQN-based policy is resilient to noise, and the laser noise seriously affects the arrival rate of the traditional obstacle avoidance algorithm VFH. This is to be expected because traditional obstacle avoidance methods use obstacle clearance to calculate their objective function, and this greedy method usually guides the robot to a local minimum. Most of the time the robot is stuck somewhere instead of colliding with an obstacle. More importantly, the learned policy (the DQN policy with noise in the Fig. 6) will work better when using the same noise variance as the test environment during training, which enables the neural network to learn the ability to deal with sensor noise.

Qualitative Experiment

To evaluate the performance of our proposed method in more difficult situations, we simulated some test environments in the Gazebo simulator. As shown in Fig. 7, it includes a simple obstacles environment, random obstacles environment, right-angle environment, *s*-shaped environment, and *u*-shaped environment. From the motion trajectory formed by robot footprints in the figures, we can see that our learned policy can output relatively successful actions in different complex environments. The experimental results show that our dueling DDQN model can obtain effective and proper steering commands in more

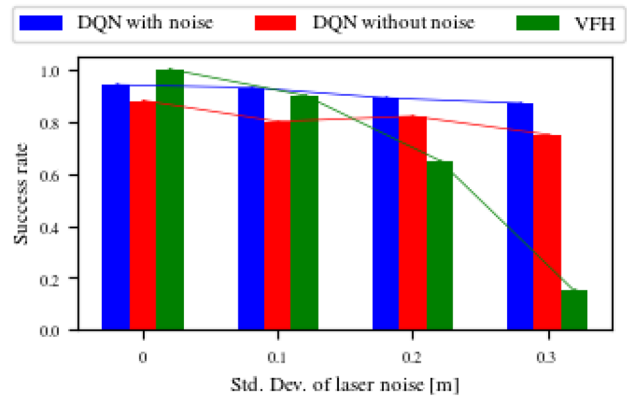
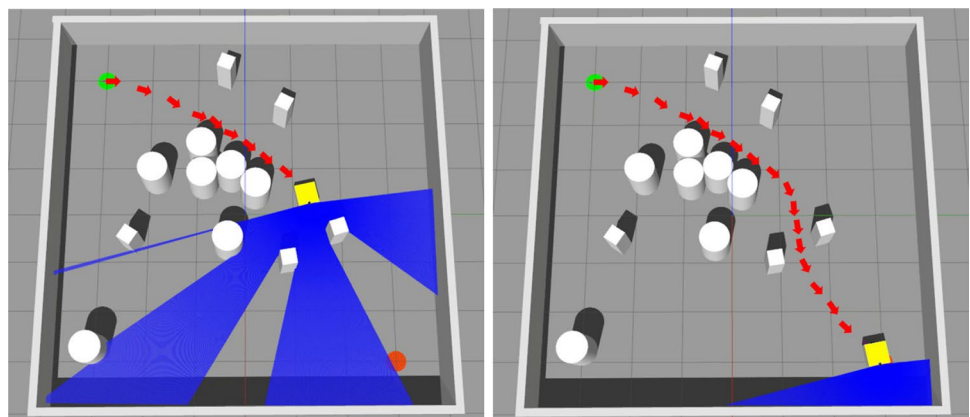


Fig. 6 Success rate over laser noise. DQN policy trained with sensor noise compared to policy without sensor noise and traditional VFH method

difficult situations, which proves its high transferability and generalization capability.

Because obstacle avoidance algorithms usually only use local perception information, the robot is easy to fall into a difficult area (such as a *u*-shaped environment), which is called a local minimum problem [37]. As shown in Fig. 8, we also tested the ability of the VFH obstacle avoidance algorithm to solve the local minimum problem. Especially in the more difficult *u*-shaped environment (Fig. 8c), the robot cannot reach the target point by swinging left and right. In practical applications, the local obstacle avoidance algorithm will cooperate with global path planning to improve its ability to get out of trouble. Recently, some researchers [38] began to explore how traditional planning algorithms can guide the DRL-based local obstacle avoidance algorithm to better navigate. The design of other modules of the navigation system can also alleviate the local minimum problem caused by the local obstacle avoidance algorithm to a certain extent.

Fig. 5 A test result of our curricular DQN policy in a test scenario, the green and red dot represents the starting and target position, respectively, and the robot's trajectory is marked with red arrows



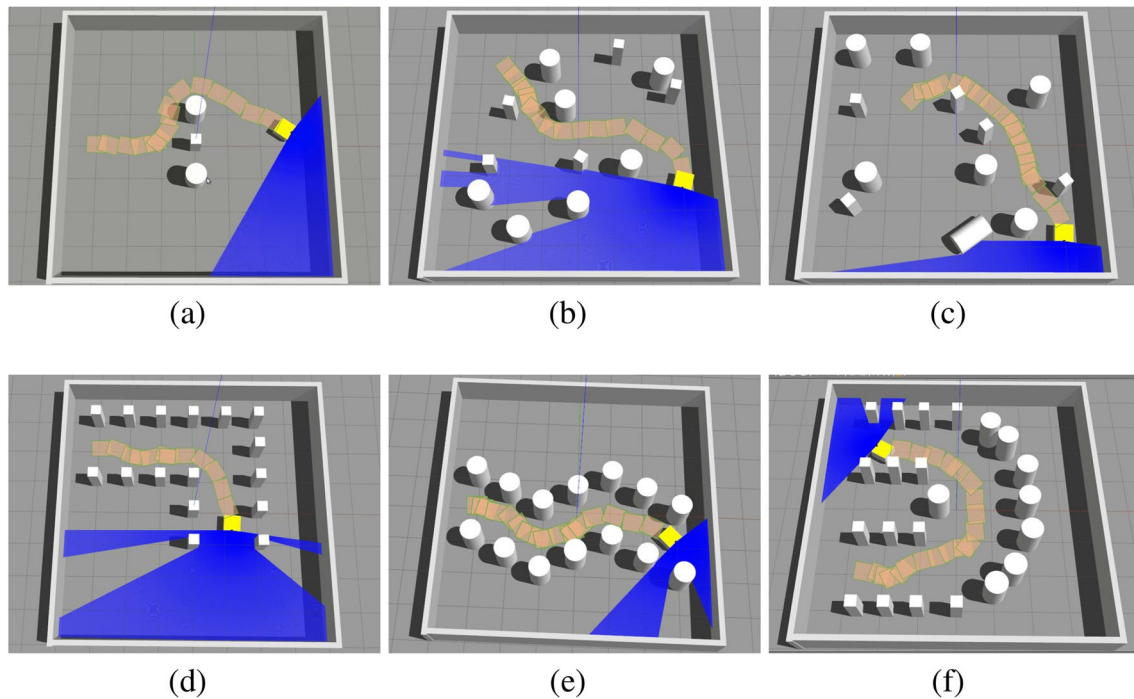


Fig. 7 Some simulation experiments, including simple obstacles environment (a), random obstacles environment (b, c), right-angle environment (d), s-shaped environment (e), and u-shaped environ-

ment (f). The orange rectangles represent the footprints of the robot, and the interval between each footprint is 1 s

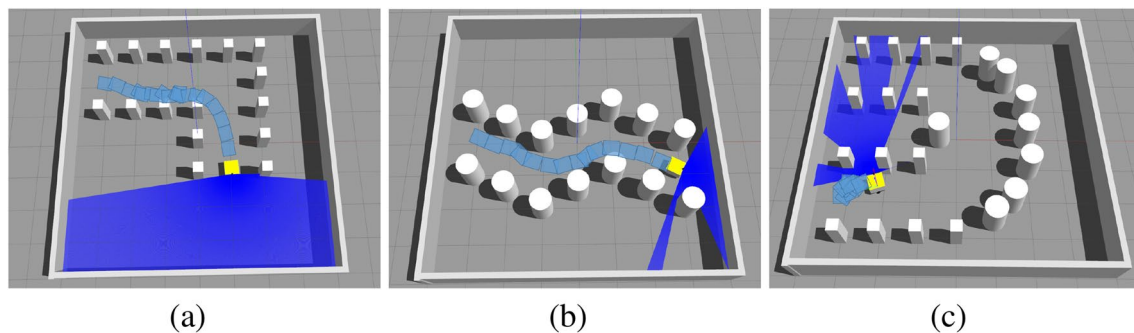


Fig. 8 Trajectories of robots executing VFH policy in the right-angle environment (a), s-shaped environment (b), and u-shaped environment (c), respectively. The blue rectangles represent the footprints of the robot, and the interval between each footprint is one second

Compatible with Different Sensors

To prove that our method is compatible with different sensors, we demonstrated a simulation experiment of obstacle avoidance based on depth cameras. We simulated a depth camera on the robot according to the parameters of Kinect sensors. Its horizontal viewing angle is 60 degrees, the closest measurement distance is 0.8 m, the longest measurement distance is 4 m, and the resolution of the depth image is 640×480 . We first need to convert the point cloud data into our grid map format. We downsample the point cloud according to the map resolution, and at the same time filter

out the point cloud on the ground according to the height, and then press down from the top to a two-dimensional flat grid map. An example of generating a grid map from point cloud data is shown in Fig. 9. Using the grid map generated by the point cloud as input, we use our map-based deep reinforcement learning method to learn obstacle avoidance policies in an environment with seven random obstacles. The curve of expected returns rise with the epoch during the training process is shown in Fig. 10a. And Fig. 10b shows that the robot with a depth camera in the test scene successfully avoided the obstacle and reached the endpoint. However, due to the narrow field of view of the depth camera, the

Fig. 9 Point cloud data (white) of the simulated Kinect RGB-D sensor (a) and the grid map generated from the point cloud (b)

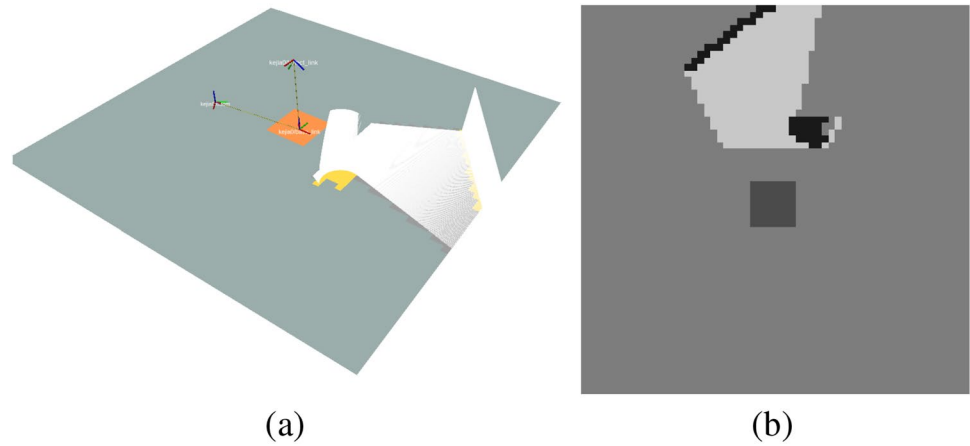
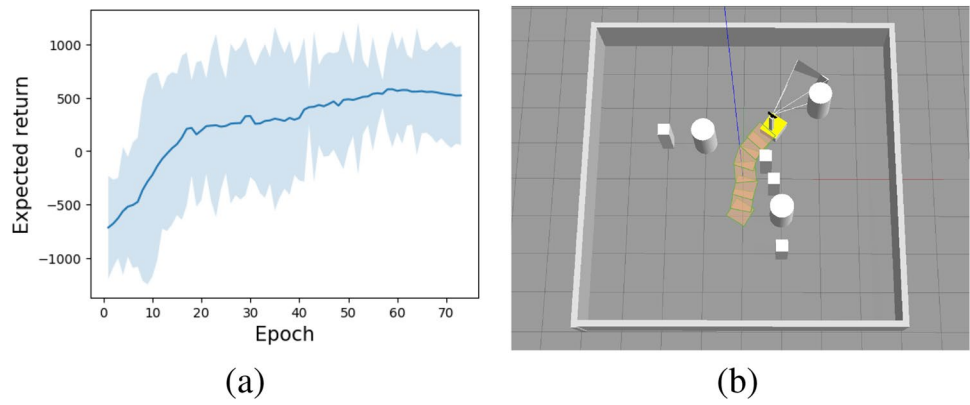


Fig. 10 Training curve of obstacle avoidance based on depth camera (a). A test result of our DQN policy with a depth camera (b), the robot with a depth camera in the test scene successfully avoided the obstacle and reached the endpoint



success rate of the test in the environment shown in Fig. 10b is only 0.83. In future work, we will use the recurrent neural network architecture and increase the state history length to increase the success rate of obstacle avoidance using a depth camera.

Navigation in Real World

A variety of real-world experiments using our robot chassis are conducted as well to verify the proposed map-based DQN model. As shown in Fig. 11, our robot platform is a differential wheel robot with a Hokuyo UTM-30LX scanning laser rangefinder and a laptop with an i7-8750H CPU and a NVIDIA 1060 GPU. The location and velocity of the robot are provided by a state estimator based on particle filter. The egocentric local map is constructed by laser measurement and cropped to a fixed size of 6.0×6.0 m with a resolution of 0.1 m. It is worth mentioning that our dueling DDQN model trained in the simulated environment can be directly transferred to the real-world environment without adjusting any parameters, and well adapt to the unseen scenarios by adopting egocentric local maps as the inputs to the network.

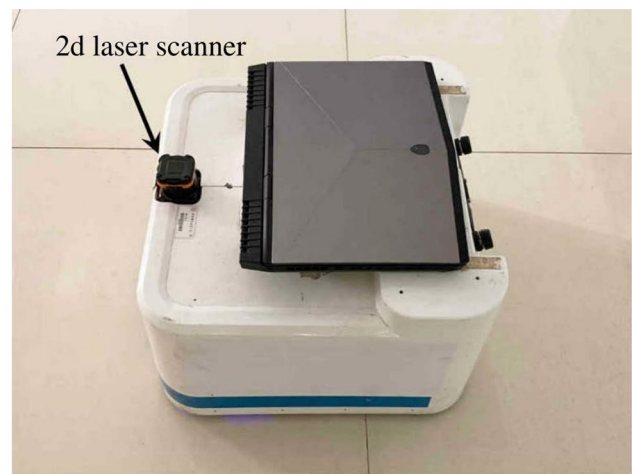


Fig. 11 Our robot chassis with a laptop and a Hokuyo UTM-30LX laser scanner

To illustrate the effectiveness of the proposed map-based obstacle avoidance algorithm in a more intuitive way, Fig. 12 shows eight scene cases before and after the motion of the robot, which is controlled based on the output of the DQN model. The second frame of each group of images is the



Fig. 12 Robot test of obstacle avoidance module, including right-angle (a) and s-shaped environment (b), discrete random obstacles environment (c, d). And the robot successfully avoided passing pedestrians (e, f, g) and sudden still pedestrians (h)

scene after the robot has taken action against the first scene. In the first and second case, the robot successfully avoided the right-angle and s-shaped static obstacle environment built by cardboard boxes. In the much more constrained space shown in Fig. 12c and d, when the robot encounters obstacles, the trained policy successfully provided reactive action commands and keep the robot away from the obstacles. Figure 12e–g show that the robot successfully avoided passing pedestrians. Finally in Fig. 12h, two pedestrians suddenly appeared and stayed in front of the robot, and the robot successfully changed the original route and bypassed them, which indicates the high transferability and effectiveness of our proposed obstacle avoidance model.

To further verify the effectiveness of the entire DRL-based navigation system for long-range navigation, the entire navigation system with DRL-based obstacle avoidance is also evaluated in large area environments, including both long corridor and big hall scenarios. The short-range DRL-based motion planner directly maps egocentric local grid maps to robot's control commands in terms of intermediate goals provided by the global A* path planner. In the navigation procedure, each local goal position is selected as a position that is 3 m away from the robot on the global planned path. Five examples in corridor environments are described in Fig. 13a–e. The robot set off from one end of the corridor, successfully avoided static carton obstacles

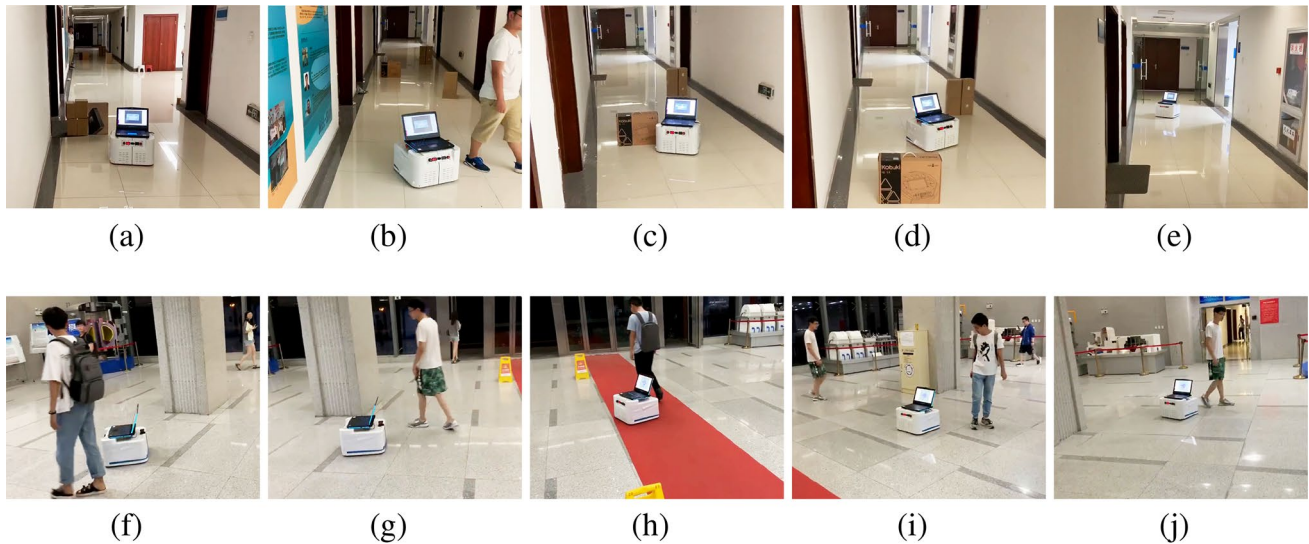


Fig. 13 Long-range experiments in the corridor (a–e) and lobby (f–j). The robot successfully avoided obstacles and pedestrians and reached a target point tens of meters away

and dynamic pedestrians, and reached the target point at the other end of the corridor. Similarly, in a dynamic and open lobby environment (Fig. 13f–j), the robot took the action output by the DQN network and successfully reached the endpoint several tens of meters away.

A video of simulated and real-world navigation experiments can be found at https://youtu.be/Eq4AjsFH_cU. The experimental results show that our proposed map-based DQN model trained in the virtual environment can be directly transferred to various simulated and real-world unseen environments without adjusting any parameters.

Conclusion

In this paper, we propose an end-to-end deep reinforcement learning method for mobile robot navigation with map-based obstacle avoidance, which directly maps egocentric local grid maps to an agent’s steering commands in terms of the target position and movement velocity. Our approach is mainly based on dueling double DQN with the prioritized experienced replay, and integrate curriculum learning techniques to further enhance our performance. It is worth mentioning that our proposed map-based DQN model trained in the simulated environment can be directly transferred to various real-world unseen environments without adjusting any parameters due to the high fidelity of the local grid maps.

Our proposed model was evaluated in multiple virtual environments and compared with related works. The experimental results show the outstanding performance of our proposed method in terms of expected return, success rate, arrival step as well as average angular velocity

change. Furthermore, many real-world scenarios were built to evaluate our proposed model, including static obstacles of cardboard boxes and dynamic pedestrian scenarios. The experimental results show that our dueling DDQN model can effectively output proper steering commands in static and dynamic environments, which indicates the high transferability and effectiveness of our proposed obstacle avoidance model. Finally, we integrated our obstacle avoidance policy into the navigation framework for long-range navigation testing. Our mobile robot successfully avoided obstacles and pedestrians and reached a target point tens of meters away in the corridor and lobby scenarios.

Our future work intends to extend our map-based deep reinforcement learning approach for multi-robot collision avoidance in a distributed and communication-free environment, use other multi-sensor fusion solutions to take advantage of our map-based method, and utilize other positioning methods (such as UWB technology or visual SLAM) to solve the error-prone problem of laser positioning in dense environments.

Funding This research was supported by 2030 National Key AI Program of China (Grant 2018AAA0100500), National Natural Science Foundation of China (CN) (Grant 61573386) and Science and Technology Planning Project of Guangdong Province (Grant 2017B010110011).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

- Ingrand F, Ghallab M. Deliberation for autonomous robots: a survey. *Artif Intell.* 2017;247:10–44.
- Minguez J, Lamiraux F, Laumond J-P. Motion planning and obstacle avoidance. In: Springer handbook of robotics. Springer, 2016;1177–202.
- Mohanan M, Salgoankar A. A survey of robotic motion planning in dynamic environments. *Robot Auton Syst.* 2018;100:171–85.
- Zhang W, Wei S, Teng Y, Zhang J, Wang X, Yan Z. Dynamic obstacle avoidance for unmanned underwater vehicles based on an improved velocity obstacle method. *Sensors.* 2017;17(12):2742.
- Zhou D, Wang Z, Bandyopadhyay S, Schwager M. Fast, on-line collision avoidance for dynamic vehicles using buffered Voronoi cells. *IEEE Robot Autom Lett.* 2017;2(2):1047–54.
- Rösmann C, Hoffmann F, Bertram T. Integrated online trajectory planning and optimization in distinctive topologies. *Robot Auton Syst.* 2017;88:142–53.
- Kahn G, Villafior A, Ding B, Abbeel P, Levine S. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), 2018;1–8.
- Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. Mastering the game of go without human knowledge. *Nature.* 2017;550(7676):354–9.
- Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, et al. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature.* 2019;575(7782):350–4.
- Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int J Robot Res.* 2018;37(4–5):421–36.
- Xie L, Wang S, Rosa S, Markham A, Trigoni N. Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), 2018;6276–83.
- Chen G, Cui G, Jin Z, Wu F, Chen X. Accurate intrinsic and extrinsic calibration of RGB-D cameras with GP-based depth correction. *IEEE Sens J.* 2018;19(7):2685–94.
- Chen G, Pan L, Chen Y, Xu P, Wang Z, Wu P, Ji J, Chen X. Robot navigation with map-based deep reinforcement learning. In: Proceedings of the 2020 IEEE international conference on networking, sensing and control (ICNSC), 2020;1–6.
- Giusti A, Guzzi J, Cireşan DC, He F-L, Rodríguez JP, Fontana F, Faessler M, Forster C, Schmidhuber J, Di Caro G, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot Autom Lett.* 2015;1(2):661–7.
- Gandhi D, Pinto L, Gupta A. Learning to fly by crashing. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2017;3948–55.
- Tai L, Li S, Liu M. A deep-network solution towards model-less obstacle avoidance. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2016;2759–64.
- Pfeiffer M, Schaeuble M, Nieto J, Siegart R, Cadena C. From perception to decision: a data-driven approach to end-to-end motion planning for autonomous ground robots. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), 2017;1527–33.
- Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. *IEEE Robot Autom Mag.* 1997;4(1):23–33.
- Liu Y, Xu A, Chen Z. Map-based deep imitation learning for obstacle avoidance. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2018;8644–9.
- Chen YF, Liu M, Everett M, How JP. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), 2017;285–92.
- Chen YF, Everett M, Liu M, How JP. “Socially aware motion planning with deep reinforcement learning.” In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017;1343–1350.
- Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2017;31–6.
- Long P, Fanl T, Liao X, Liu W, Zhang H, Pan J. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), 2018;6252–9.
- Lobos-Tsunekawa K, Leiva F, Ruiz-del Solar J. Visual navigation for biped humanoid robots using deep reinforcement learning. *IEEE Robot Autom Lett.* 2018;3(4):3247–54.
- Wu K, Abolfazli Esfahani M, Yuan S, Wang H. Learn to steer through deep reinforcement learning. *Sensors.* 2018;18(11):3650.
- Zhang J, Springenberg JT, Boedecker J, Burgard W. Deep reinforcement learning with successor features for navigation across similar environments. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2017;2371–8.
- Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proceedings of the thirtieth AAAI conference on artificial intelligence (AAAI), 2016.
- Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd international conference on machine learning (ICML), 2016;1995–2003.
- Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. In 2016.
- Lu DV, Hershberger D, Smart WD. Layered costmaps for context-sensitive navigation. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), 2014;709–15.
- Ng AY, Harada D, Russell S. Policy invariance under reward transformations: theory and application to reward shaping. *ICML.* 1999;99:278–87.
- Bengio Y, Louradour J, Collobert R, Weston J. Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning (ICML). ACM, 2009;41–8.
- Elman JL. Learning and development in neural networks: the importance of starting small. *Cognition.* 1993;48(1):71–99.
- Koenig N, Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator. *Proc IEEE/RSJ Int Confe Intell Robot Syst.* 2004;3:2149–54.
- Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980), 2014.
- Ulrich I, Borenstein J. Vfh/sup*: local obstacle avoidance with look-ahead verification. *Proc IEEE Int Conf Robot Autom.* 2000;3:2505–11.
- Koren Y, Borenstein J, et al. Potential field methods and their inherent limitations for mobile robot navigation. *Proc IEEE Int Conf Robot Autom.* 1991;2:1398–404.
- Brito B, Everett M, How JP, Alonso-Mora J. Where to go next: learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robot Autom Lett.* 2021;6(3):4616–23.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.