TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/15 pp674–691
DOI: 10.26599/TST.2021.9010012
Volume 26, Number 5, October 2021

Deep Reinforcement Learning Based Mobile Robot Navigation: A Review

Kai Zhu and Tao Zhang*

Abstract: Navigation is a fundamental problem of mobile robots, for which Deep Reinforcement Learning (DRL) has received significant attention because of its strong representation and experience learning abilities. There is a growing trend of applying DRL to mobile robot navigation. In this paper, we review DRL methods and DRL-based navigation frameworks. Then we systematically compare and analyze the relationship and differences between four typical application scenarios: local obstacle avoidance, indoor navigation, multi-robot navigation, and social navigation. Next, we describe the development of DRL-based navigation. Last, we discuss the challenges and some possible solutions regarding DRL-based navigation.

Key words: mobile robot navigation; obstacle avoidance; deep reinforcement learning

1 Introduction

The navigation capability is a fundamental problem of mobile robots, which include unmanned vehicles, aerial vehicles, and ships. The general aim of navigation is to identify an optimal or suboptimal path from a starting point to a target point in a Two-Dimensional (2D) or Three-Dimensional (3D) environment while avoiding obstacles. Delivery robots, warehouse automated guided vehicles, and indoor service robots require robust robot navigation systems in their dynamic environments.

In the past two decades, researchers from all over the world have been focused on solving the navigation problem. One popular approach is combining a series of different algorithms. As shown in Fig. 1, the traditional navigation framework uses Simultaneous Localization and Mapping (SLAM) to construct a map of the unknown environment, then uses a localization algorithm to determine the current position of the robot and moves it to its destination using a path planning module[1].

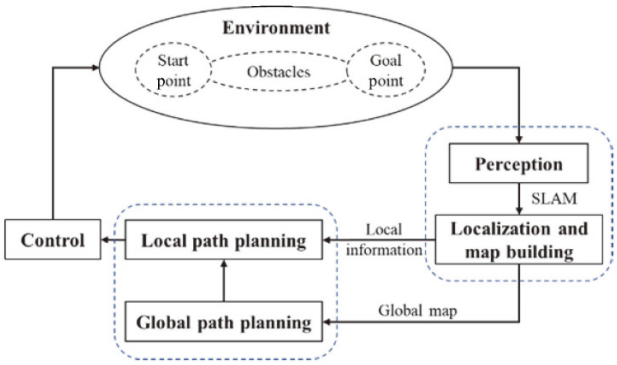

Fig. 1 Traditional robot navigation framework.

SLAM algorithms can be divided into visual and laser SLAMs. The visual SLAM algorithm extracts artificial image features, estimates the pose of the robot and camera based on multi-view geometry theory, and builds an obstacle map. Classical visual SLAM methods, such as LSD-SLAM[2] and ORB-SLAM[3], face two main challenges: (1) designing effective image features to express image information and (2) possible failure of the algorithm in cases of object movement, camera parameter change, illumination change, and single environments that lack texture. The laser SLAM algorithm directly constructs an obstacle map of the environment based on the dense laser ranging results of algorithms such as GMapping[4] and Hector SLAM[5].

• Kai Zhu and Tao Zhang are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: zhuk19@mails.tsinghua.edu.cn; taozhang@tsinghua.edu.cn.
• Tao Zhang is also with the Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China.
* To whom correspondence should be addressed.
Manuscript received: 2021-02-05; accepted: 2021-02-22

© The author(s) 2021. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

The challenges of laser SLAM include (1) the time-consuming establishment and update of the obstacle map and (2) the need for a dense laser sensor because the algorithm performance strongly depends on the sensor accuracy.

Path planning is another key module in the traditional navigation framework. Based on different amounts of environmental information obtained, this module can be divided into global and local path planning. Global path planning involves selecting a complete path based on a known environmental map. Commonly used methods include the A-star, ant colony optimization, and rapid-exploration random tree[6], which rely on known static maps and are therefore difficult to use in dynamic environments. Local path planning methods, such as the Artificial Potential Field (APF) and dynamic window approach[7], are used to deal with dynamic changes in the environment and replanning local paths. The bottlenecks encountered by traditional path planning algorithms include (1) the contradiction of grid-based map representation between its accuracy and memory requirements and (2) the intensive calculations required for real-time replanning of the navigation path in a dynamic environment, which limits its reactivity to some extent.

As mentioned above, each aspect of the traditional navigation framework represents a challenging research topic, and their integration often leads to large computational errors. These calculation errors gradually accumulate along the pipeline from mapping, to positioning, to the path planning algorithm, which leads to poor performance of all these algorithms in practical applications. The traditional navigation framework relies on a high-precision global map that is very sensitive to sensor noise, resulting in limitations in the ability to manage an unknown or dynamic environment.

With the powerful representation capabilities of deep-learning technology, new ideas have been introduced for using reinforcement learning frameworks that can directly learn navigation strategies from raw sensor inputs. In 2013, Mnih et al.[8] were the first to propose the concept of Deep Reinforcement Learning (DRL). They proposed the Deep Q Network (DQN), which can learn to play Atari 2600 games at a level beyond that of human experts based only on image input. Since then, researchers have proposed numerous methods that use DRL algorithms for handling autonomous navigation tasks. These methods describe navigation as a Markov Decision Process (MDP), with sensor observations as the state and a goal of maximizing the expected return of the action. By interacting with the environment, the DRL method finds the optimal policy of guiding the robot to the target position. DRL-based navigation has the advantages of being mapless and having a strong learning ability and low dependence on sensor accuracy. Since 2016, the trend of applying DRL to mobile robot navigation has increased, achieving great success[9].

Researchers have published several surveys on mobile robot navigation[10, 11], which mainly introduce path planning and obstacle avoidance methods under the traditional navigation framework. The contents of the DRL technology are not comprehensive. In 2020, Nguyen et al.[12] investigated multi-agent DRL, which uses DRL to solve multi-agent cooperation problems. Zeng et al.[13] published a survey on the use of DRL for the visual navigation of artificial agents, which mainly focused on visual navigation tasks and the division of DRL methods into five categories for review. In contrast to their research, because we believe that different navigation scenarios have similar characteristics, here, we focus more on the intrinsic enhancement that DRL brings to a variety of navigation tasks, and the use of state-of-the-art techniques in dealing with mobile robot navigation problems.

In this paper, we present a comprehensive and systematic review of DRL-based mobile robot navigation from 2016 to 2020. The application scenarios, current challenges, and possible solutions to the challenges of DRL-based navigation are discussed in detail to guide researchers for further improvement of current research results and their deployment to real systems.

In Section 2, we present the background knowledge of DRL. In Section 3, we present the framework and key elements of the DRL-based navigation problem. In Section 4, we divide the application scenarios of DRL-based navigation into four categories, and describe in detail the developments and approaches used in each scenario. We present the current challenges and available solutions in Section 5. Finally, we draw our conclusions in Section 6. The architecture of this paper is shown in Fig. 2.

2 Background: Deep Reinforcement Learning

2.1 Preliminary

Reinforcement Learning (RL), inspired by animal learning in psychology, learns optimal decision-making

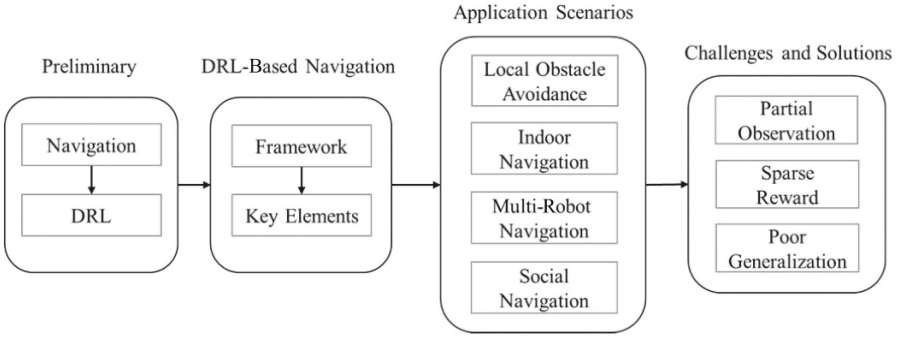

Fig. 2 Architecture of this paper.

strategies from experience. RL defines any decision maker as an agent and everything outside the agent as the environment. The agent aims to maximize the accumulated reward and obtains a reward value as a feedback signal for training through interaction with the environment. The interaction process between the agent and environment can be modeled as an MDP comprising the essential elements S, A, R, and P; S is the state of the environment, A is the action taken by the agent, R is the reward value obtained, and P is the state transition probability. The agent's policy π is the mapping from state space to action space. When the state $s_t \in S$, the agent takes action $a_t \in A$, and then transfers to the next state s_{t+1} according to the state transition probability P, while receiving reward value feedback $r_t \in R$ from the environment.

Although the agent receives instant reward feedback at every time step, the goal of RL is obtaining the largest long-term cumulative reward value rather than short-term rewards. By introducing the discount factor $\gamma \in [0, 1)$, we can express the return value as follows:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1}$$

The value function of the state s_t is $V_\pi(s)$ and action value function of the state-action pair (s, a) is $Q_\pi(s, a)$. These values are used to evaluate the long-term reward that the agent can expect by the use of policy π.

$$V_\pi(s) = E_\pi[R_t | s_t = s] \tag{2}$$

$$Q_\pi(s, a) = E_\pi[R_t | s_t = s, a_t = a] \tag{3}$$

Using Eq. (1), $V_\pi(s)$ and $Q_\pi(s, a)$ can be expressed in a recursive form to establish the relationship between the states $s = s_t$ and $s' = s_{t+1}$:

$$V_\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_\pi(s') \right] \tag{4}$$

$$Q_\pi(s, a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \sum_{a'} Q_\pi(s', a') \right] \tag{5}$$

where $P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$, $R^a_{ss'} = E[r_{t+1} | s_t = s, s_{t+1} = s', a_t = a]$. Equations (4) and (5) are known as Bellman equations. The approximate solution to a Bellman equation is obtained via dynamic programming to obtain the current value function. The agent then continuously improves policy π by optimizing the value function.

As dynamic programming requires complete dynamic information and massive memory consumption, which are not feasible, researchers have proposed and developed two learning methods: Monte Carlo and Temporal-Difference (TD) learning. In 1989, Watkins[14] proposed the Q-learning algorithm, which combines theories, including the Bellman equations and MDP, with TD learning. Since then, RL research has made huge breakthroughs, and RL algorithms have been used to solve a range of practical problems.

However, in high-dimensional problems, the traditional RL algorithm faces what is referred to as the curse of dimensionality, whereby the amount of computation sharply increases with the increase in the number of inputs. Thus, finding a good policy in a large state space is difficult using RL. The deep-learning approach approximates any nonlinear function by training deep neural networks and learns the inherent laws and essential characteristics of the input data. Its powerful representation ability enabled another breakthrough for RL by its integration with deep neural networks to constitute DRL.

DRL can be divided into two approaches: value-based and policy-based methods. Value-based DRL indirectly obtains the agent's policy by iteratively updating the value function. When the value function reaches an optimal value, the agent's optimal policy is obtained via the optimal value function. The policy-based method directly uses the function approximation method to establish a policy network, selects actions through the policy network to obtain the reward value, and

optimizes the policy network parameters along the gradient direction for obtaining an optimized policy that maximizes the reward value.

2.2 Value-based DRL methods

2.2.1 Deep Q network

Mnih et al. published DQN-related work in *Nature* in 2015, reporting that the trained network could reach a level equivalent to that of humans after playing 49 games[15]. DQN, which is based on Q learning, uses a convolutional neural network (a deep neural network) to represent the action value function, and the network is trained based on reward feedback from the game. The main features of DQN are as follows:

(1) The target network is set to deal with the TD error in the time-difference algorithm separately. The parameter θ_i of the current Q-network $Q(s,a;\theta_i)$ is copied to θ_i^- of the target Q-network $Q(s',a';\theta_i^-)$ every n time steps, which prevents instability of the target Q network from the changes made in the current Q network during training.

(2) The experience pool $U(D)$ is used to store and manage samples (s,a,r,s'), and an experience replay mechanism is used to select the samples. These samples are stored in the experience pool, from which batch samples are randomly selected to train the Q network. The experience replay mechanism helps to eliminate the correlation between samples so that the samples used in the training approximately realize independent and identical distributions.

The parameters of the neural network are updated by gradient descent. The loss function of the DQN is denoted as:

$$L(\theta_i)=E\left[\left(r+\gamma\max_{a'}Q(s',a';\theta_i^-)-Q(s,a;\theta_i)\right)^2\right] \tag{6}$$

2.2.2 Double DQN (DDQN)

The emergence of DQN promoted widespread use of DRL, but DQN has a number of shortcomings, one of which is the overestimation of the action value function. Van Hasselt et al.[16] pointed out that when the DQN calculates the TD error, using the same Q network to select actions and calculate value functions leads to overestimation of the value function. Thus, the authors proposed the DDQN algorithm.

DDQN uses a dual network structure in the target Q function, whereby the optimal action is selected based on the current Q network, and the target Q network evaluates the selected optimal action. Two sets of parameters separate the action selection and policy evaluation tasks, which reduces the overestimation risk.

The loss function of the DDQN is denoted as

$$L(\theta_i)=E\left[\left(r+\gamma Q(s',\arg\max_{a'}Q(s',a';\theta_i);\theta_i^-)-Q(s,a;\theta_i)\right)^2\right] \tag{7}$$

The experimental results in 57 Atari games show that the normalized performance of DDQN without adjustment is twice that of DQN, and three times that of DQN when adjusted.

2.3 Policy-based DRL methods

2.3.1 Deep Deterministic Policy Gradient (DDPG)

Value-based DRL methods (DQN and its variants) solve problems with a high-dimensional observation space but can only handle discrete and low-dimensional action spaces. However, several practical tasks, especially physical control tasks, have continuous and high-dimensional action spaces. To address this issue, the action space can be discretized but will inevitably face the curse of dimensionality, i.e., the number of actions will increase exponentially with the increase in degree of freedom.

Lillicrap et al.[17] proposed the DDPG, which uses a method based on the policy gradient to directly optimize the policy, which can be used for problems with a continuous action space. Unlike the random strategy represented by the probability distribution function $a_t\sim\pi_\theta(s_t|\theta^\pi)$, DDPG uses a deterministic policy function $a_t=\mu(s_t|\theta^\mu)$. It also uses a convolutional neural network to simulate the policy and Q functions and learns from the experience replay and target network in the DQN to stabilize the training and ensure high sample utilization efficiency. K samples in the experience pool are randomly selected, and the Q network is gently updated by gradient ascent. The loss function of the Q network is defined as follows:

$$L=\frac{1}{K}\sum_i\left(y_i-Q(s_i,a_i|\theta^Q)\right)^2 \tag{8}$$

where $y_i=r_i+\gamma Q'(s_{i+1},\mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ indicates the expected value.

The unbiased estimate of the policy network gradient is obtained as follows:

$$\nabla_{\theta^\mu}J\approx\frac{1}{K}\sum_i\nabla_aQ(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \tag{9}$$

The experimental results show that the DDPG is suitable for solving more than 20 continuous control tasks such as robotic arm control.

2.3.2 Asynchronous Advantage Actor-Critic (A3C)

The A3C algorithm proposed by Mnih et al.[18] is a representative Actor-Critic (AC) method. The classic policy gradient algorithm directly optimizes the agent's policy; it must collect a series of complete sequence data to update the policy. In DRL, collecting sequence data is often challenging and large variances can be introduced. The AC structure that combines the value function with the policy gradient method is receiving much attention.

In the AC structure, the actor selects actions using the policy gradient method, and the critic evaluates those actions using the value function method. During training, the parameters of the actor and critic are alternately updated. The advantage of the AC structure is that it changes the sequence update in the policy gradient to a single-step update, without the need to wait for the sequence to end before evaluating and improving the policy. This ability reduces both the difficulty of data collection and the variance experienced by the policy gradient algorithm.

Based on the AC structure, A3C makes the following improvements:

(1) Parallel agents: The A3C algorithm creates multiple parallel environments, thereby enabling multiple agents with secondary structures to simultaneously update the parameters of the main structure in these parallel environments. Multiple actors are used to explore the environment.

(2) *N*-step return: Although other algorithms typically use a one-step return of the instant reward calculation function obtained in the sample, the value function of A3C's critic is updated based on the multi-step cumulative return. Calculation of the N-step return improves the iterative update propagation and convergence speed.

A3C can run on a multi-core CPU, and its computational cost is lower than methods like DQN. The experimental results show that, despite the problems of hyperparameter adjustment and low sampling efficiency, A3C has achieved success in tasks such as the continuous control of a robotic arm and maze navigation.

2.3.3 Proximal Policy Optimization (PPO)

Traditional policy gradient methods adopt an on-policy strategy in which the sampled minibatch can only be used for one update epoch, and the minibatch must be resampled to implement the next policy update. Schulman et al.[19] proposed the PPO algorithm, which can perform multiple epochs of minibatch updates, thereby improving the sample utilization efficiency.

The PPO algorithm uses an alternative goal to optimize the new policy using the old policy:

$$L(\theta) = \hat{E}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] \tag{10}$$

where $\hat{A}_t$ is an estimation of the advantage function at time t. $\dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \triangleq r_t(\theta)$ is the probability ratio of the new policy π_θ to the old policy $\pi_{\theta_{\text{old}}}$.

Equation (10) is used to improve the actions generated by the new policy relative to those generated by the old policy. However, a large-scale improvement by the new policy will lead to instability of the training algorithm. The PPO algorithm improves the objective function to obtain the following new clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \hat{E}\left[\min(r_t(\theta), \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon))\hat{A}_t\right] \tag{11}$$

where $\text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)$ denotes the restriction of the probability ratio to the interval of $[1-\varepsilon, 1+\varepsilon]$. The clipped surrogate objective keeps the change of the new policy within a certain range and improves the algorithm's stability.

By achieving a good balance between sample complexity, simplicity, and time effectiveness, PPO outperforms A3C and other on-policy gradient methods.

3 DRL-Based Navigation

3.1 Framework

Mobile robots include unmanned vehicles, aerial vehicles, and ships that move in two or three dimensions. Their navigation involves searching for an optimal or suboptimal path from the starting point to a target point while avoiding obstacles. To simplify this challenge, most researches have focused only on the navigation problem in 2D space.

In essence, the mobile robot navigation task constitutes Point-To-Point (P2P) movement and obstacle avoidance: (1) The P2P task requires the position of the goal point relative to the start point, which can be directly obtained via GPS or ultra-wideband localization[20], or indirectly through a target perspective image. (2) Obstacles include those that are statically, dynamically, and structurally continuous, which can be sensed by laser range finding, ultrasonic finding, cameras, or other sensors. In this paper, a structurally continuous obstacle

refers to an inherent structure in the environment, such as a corridor or wall. These obstacles constitute an indoor or maze-like environment.

The purpose of using a DRL algorithm in an autonomous navigation task is to find the optimal policy for guiding the robot to its target position through interaction with the environment. Many well-known DRL algorithms, such as DQN, DDPG, PPO, and their variants, have been extended to realize a DRL-based navigation system. These methods describe the navigation process as an MDP that uses sensor observation as the state with the goal of maximizing the expected revenue of the action. As mentioned above, DRL-based navigation has the advantages of being mapless and having a strong learning ability and low dependence on sensor accuracy. As RL is a trial-and-error learning technology, the physical training process inevitably leads to collisions of the robot with environmental obstacles, which is prohibited. Generally, the deep neural network is trained in a simulation environment before being deployed in a real robot for real-time navigation decision making.

DRL-based navigation has been used to replace or be integrated into the traditional navigation framework. Figure 3 shows the interaction process between the agent and environment of the DRL-based navigation system. The DRL agent replaces the localization and map building module as well as the local path planning module of the traditional navigation framework, moving toward the target point while avoiding static, dynamic, and simple structurally continuous obstacles. However, in an environment where structurally continuous obstacles are too complex, the agent may fall into a local trap. In this case, DRL requires additional

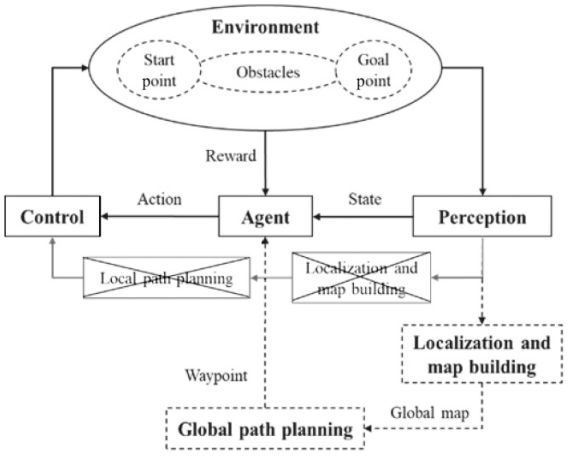

Fig. 3 DRL-based navigation system.

global information provided by the traditional navigation technique[21]. As shown in Fig. 3, the global path planning module generates a series of waypoints as intermediate goal points for DRL-based navigation, which enables the integrated navigation system to realize long-distance navigation in a complex structural environment.

3.2 Key elements

The DRL-based navigation system contains three key elements that directly determine the application scenarios and performance of the DRL algorithm: the state space S, the action space A, and the reward function R.

3.2.1 State space

The most often used state-space settings include the start point, goal point, and obstacles. (1) The start point and goal point are represented by the current and destination coordinates of the mobile robot, respectively. Several researchers convert global Cartesian coordinates into local polar coordinates and use the direction and distance relative to the robot for expressing the target point position. (2) The obstacle state is represented by the speed, position, and size of the moving obstacle (agent level) or treats sensor data directly as a sensor-level state, i.e., lidar/ultrasonic ranging data, monocular camera image, or depth camera data.

3.2.2 Action space

In DRL-based navigation research, there are three kinds of actions, i.e., (1) discrete moving actions: moving forward, moving backward, turning left, turning right, and so on; (2) continuous velocity commands: the linear velocity and angular velocity of the mobile robot; and (3) motor speed commands: the desired speeds of the left and right motor. In general, discrete moving actions and continuous velocity commands require a Proportional–Integral–Derivative (PID) or other low-level motion controllers to output motion control instructions and control the mobile robot to achieve the desired motion. Motor speed commands can realize end-to-end control using the sensor-level state, but the associated training is much more difficult.

3.2.3 Reward function

The reward function is used to train the RL agent to complete a task. In the navigation task, positive or negative rewards are only given when reaching the target or colliding with obstacles, which means this reward is very sparse. Sparse rewards are not conducive to

rapid convergence by the agent. To improve training efficiency, dense reward-shaping methods are used in most studies: (1) goal rewards include positive rewards given for arriving at goals and movement close to these goals; (2) a collision penalty is a negative reward given following a collision with an obstacle, or movement too close to an obstacle; and (3) a time step penalty is a negative reward given at each time step to encourage the robot to move faster on its way to the target.

4 Application Scenario

In the past five years, several studies have been conducted on DRL-based navigation, but the classification of DRL-based navigation remains confusing. For example, when using lightweight localization solutions, such as GPS and Wifi, a DRL-based navigation system can obtain the relative position of a goal point without global map information, which several researchers refer to as "mapless" navigation. In other research, the DRL method preprocesses the sensor's local observation data into the form of a local map, which is called a "map-based" method, and global map information is not used. Moreover, some studies refer to "visual navigation" as the use of a first-person-view Red-Green-Blue (RGB) image as the target, whereas other studies refer to navigation based on visual sensors.

We found that although researchers use similar DRL algorithms for essentially solving the same problem (Fig. 4), different researchers have conducted specific research and added expert knowledge for different application scenarios. These different approaches have occurred because in the current state of the art, if the DRL navigation policy space is set too large, it is difficult to converge. Currently, to reduce the difficulty of DRL training, agents usually learn navigation capabilities in a specific scene, which are then generalized to similar scenes.

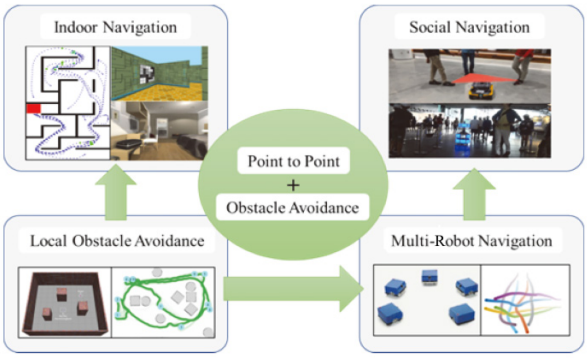

Fig. 4 Four application scenarios of DRL-based navigation.

In this review, we divide the application scenarios of DRL navigation into four categories: local obstacle avoidance, indoor navigation, multi-robot navigation, and social navigation. A simple comparison of these scenarios is shown in Table 1. Each scenario has the same basic navigation tasks but features different emphases and details. The local obstacle-avoidance scenario emphasizes dynamic changes in the simple structural environment, whereas indoor navigation focuses on the complexity of the indoor structural environment. The multi-robot navigation scenario involves an environment with multiple high-speed mobile robots. Social navigation focuses on moving through pedestrian-rich environments.

4.1 Local obstacle avoidance

4.1.1 Feature

The local obstacle-avoidance scenario, which is the most common application scenario of the DRL-based navigation system, is the basis of the other scenarios and can be extended to more complex navigation tasks. In traditional navigation frameworks, reactive methods are typically used to solve this type of problem, such as the APF or velocity-based methods. One of the biggest problems of reactive methods is the need for a good sensor system that can generate accurate position coordinates for any local obstacle. DRL methods implicitly process sensor data through neural networks, which overcome the shortcomings of traditional obstacle-avoidance methods.

4.1.2 Development

In 2016, Duguleana and Mogan[22] studied autonomous navigation in environments containing static and dynamic obstacles; they combined the neural network

Table 1 Simple comparison of different DRL-based navigation scenarios.

Navigation scenario	Static obstacle	Dynamic obstacle	Structure continuous obstacle	Obstacle scale	Obstacle velocity	Cooperation	Randomness
Local obstacle avoidance	Y	Y	N	Low	Low	–	–
Indoor navigation	Y	N	**Y**	Low	–	–	–
Multi-robot navigation	Y	Y	N	High	High	**Y**	Low
Social navigation	Y	Y	N	High	High	N	**High**

Pose-Net and a 30-20-3 multi-layer perceptron with the famous RL method Q learning. By dividing the surrounding obstacle environment into eight angular regions, they reduced the number of states. Pose-Net can output three discrete actions, i.e., moving forward, turning left, and turning right. This early research realized effective obstacle avoidance in simple physical environments.

Subsequently, DRL solutions, such as DQN, were rapidly developed, receiving widespread attention. In numerous works, mature DRL methods have been applied to local obstacle-avoidance scenarios. Feng et al.[23] used DDQN to train the agent in a simulation environment to avoid collisions with a wall without using a target point. Most actual local obstacle-avoidance tasks must be simultaneously performed with P2P tasks. For example, Kato et al.[24] developed a navigation system that combines DDQN with a topological-map-based global planning method. The topology map node stores the topology map, plans the global path, and selects the next waypoint. The local navigation node uses the control commands learned through the DDQN to move between waypoints. The authors further improved the system using a real-time kinematic global navigation satellite system that provides waypoints[25]; this system does not perform global path planning and was verified in an outdoor obstacle environment. Wang et al.[26] proposed the Fast Rent Deterministic Policy Gradient (Fast-RDPG) algorithm, which improved the DDPG performance on Unmanned Aerial Vehicles (UAVs). By assuming that a virtual UAV only flies at a fixed altitude and speed, they simplified the flying task to two dimensions. The input of the Fast-RDPG includes five-dimensional ranging data, the distance and angle of the target point provided by the GPS signal, and the direction angle of the UAV. The control profile only includes turning left or right. The authors then extended their work to the 3D environment[27]. In the study of Ma et al.[28], a single camera was used to avoid flight obstacles in 3D space. The authors used an improved saliency detection method based on a Convolutional Neural Network (CNN) to extract monocular visual cues; an AC RL module receives states from the obstacle detection module to adjust the position and altitude of the UAV.

In addition, Woo and Kim[29] and Wu et al.[30] combined international regulations for the prevention of collisions at sea[31] to study the collision avoidance problem of the unmanned surface vehicle based on the feature extraction ability of the CNN and learning ability of DRL.

4.1.3 Sim-to-real

Like most scientific research, the purpose of DRL-based navigation research is its application to real systems. In the RL training process, high-speed collisions, even during training, can damage the robot. Kahn et al.[32] focused on this problem and proposed a learning algorithm based on an uncertainty model that uses a neural network to estimate the probability of collision. The algorithm naturally chooses to proceed cautiously in unfamiliar environments and increases the robot's velocity in environments where it has high confidence.

However, training in a physical environment is very time-consuming and dangerous. Researchers usually train agents in a simulation environment and then transfer this learning to the physical environment. Sparse laser ranging data can reduce the large difference between simulation and reality[33]. Tai et al.[34] trained an asynchronous DDPG algorithm in a simulation environment, took only the 10-dimensional sparse range findings and the target position as input, and output continuous linear and angular velocities. They were able to directly transfer the network to a real incomplete differential robot platform without any fine-tuning. Yokoyama and Morioka[35] proposed a system that uses the pre-trained Struct2depth model to estimate depth data from a monocular camera image. They then converted the depth data into 2D distance data and trained the DDQN agent in a 2D-lidar simulation environment. The proposed system realized navigation in the local environment using only a monocular camera.

For expanding the state-space distribution of the pool of samples and enabling the agent to adapt to more new situations, Zhang et al.[36] adopted the approach of randomly changing the starting and target points in each epoch. They set up four different maze environments for training. Lei et al.[37] introduced a radius constraint on the initialization by randomly setting the start point in a circle with the target point as the center and a radius of L_r. The initial value of L_r is small, and as the neural network is updated, the value of L_r gradually increases, thereby increasing the success probability and ensuring a positive incentive in the sample space. However, expanding the sample space causes the DRL algorithm training to become very time-consuming. Several researchers have combined methods, such as expert demonstrations[38], artificial potential fields[39], and non-expert helpers[40], to improve training efficiency, which have been tested in

local obstacle-avoidance scenarios.

4.1.4 Sensor robustness

In most studies, DRL-based navigation applications must be deployed using the same sensors as those used in the training environment. When faced with different hardware conditions, the system may fail. Aznar et al.[41] designed a navigation policy specifically for fault tolerance, whereby the proposed system can continue to work normally under sensor failure conditions and shows several advantages in its robustness, scalability, and practicality. Choi et al.[42] studied the limited Field Of View (FOV) problem. They transformed depth data obtained by a D435 depth camera with a FOV of 90° into distance data and proposed a Long Short-Term Memory (LSTM) agent with a Local-Map Critic (LSTM-LMC) that has memory ability. The proposed method improves the robustness of DRL agents in the real world by introducing a dynamics randomization technique, including scan noise, velocity randomization, and time-scale randomization. In addition, the sensors on the robot can be replaced, or the parameters can be changed (e.g., the resolution and range of the lidar can be changed). Using another approach to solve the sensor robustness problem, Leiva and Ruiz-del-Solar[43] incorporated the corresponding angle of the range measurement as part of the observation, which allows the learning of features that are independent of the geometric distribution of the readings. The authors used variably sized 2D point clouds as the agent's 2D observations, thereby improving robustness and extensibility to the real world.

4.2 Indoor navigation

4.2.1 Feature

Here, the indoor navigation scenario refers to a house with multiple rooms or a 3D maze-like environment with numerous walls and corridors. Although the local obstacle-avoidance scenarios in the previous section may also be indoors, their structures are often relatively simple. In this section, we focus on indoor navigation scenarios with a complex structural environment. Research on indoor navigation originated from a 3D maze navigation game, which was established at the DeepMind Lab[44]. Because they can be very large and provide very complex structures, 3D maze environments are usually considered to be test platforms for DRL algorithms. Navigational abilities can emerge as a byproduct of an agent learning a policy that maximizes reward in a 3D maze environment. Moreover, because image sensors more easily perceive the features of a 3D environment, the indoor navigation task is highly related to the visual navigation task. Classic visual navigation tasks typically use an RGB image with a first-person view as the target[45], rather than relying on the goal-point coordinates provided by other localization systems, which is very different from a local obstacle-avoidance task.

4.2.2 Development

Early DRL work focused on learning target-specific models. In 2016, Oh et al.[46] proposed a new set of RL tasks in the Minecraft 3D block world. Their DRL method can process complex scene images and generate correct actions. However, the target must be fixed, which is not applicable to actual navigation because targets may change in different environments. Similarly, Brunner et al.[47] used DRL to learn to read a global map and find the shortest path out of a random maze, but this study was also designed for specific tasks, so retraining is required after changing targets.

The models described above deal with single tasks on an individual basis. Training occurs in the same scenario to learn a spatial structure, with the target implicitly embedded in the model parameters. This kind of training set is inflexible when the mission objectives change. In 2017, Zhu et al.[45] proposed a target-driven concept, which takes the task goal (i.e., the goal of navigation) as the model input. They projected the current image and the first-person-view image into the same embedding space using a deep siamese AC network based on a pre-trained ResNet. Models trained in different scenarios can generalize across targets and scenarios.

Zhu et al.[45] also proposed the AI2-THOR framework, an interactive 3D room environment for visual artificial intelligence. Two other well-known 3D environment simulation frameworks include the DeepMind Lab and House3D frameworks. In the DeepMind Lab, an agent is able to move and collect objects in maze environments. House3D[48], a rich, scalable, and efficient indoor environment based on the SUNCG[49] dataset, contains many changes in color, texture, objects, and scenario layout. These excellent simulation frameworks have made great contributions to the training and verification of state-of-the-art DRL algorithms.

In 2017, Mirowski et al.[44] adjusted the network structure of A3C and proposed the Nav A3C algorithm, in which two LSTM layers are added between the CNN and the output layer. This algorithm was trained and verified on the 3D maze environment of the DeepMind Lab platform. The authors set random

goals and used sparse "fruit" rewards to encourage exploration. Two auxiliary tasks, including depth prediction and loop-closure prediction, were proposed to solve the sparse reward problem. However, NAV A3C was found to have an unstable policy, poor data efficiency, and poor robustness in a complex environment. Zeng and Wang[50] used the monotonic policy improvement advantage of PPO and proposed the appoNav (asynchronous PPO) algorithm to solve the visual navigation problem. Kulhánek et al.[51] developed auxiliary tasks including pixel control, reward prediction, a depth map, segmentation, and target segmentation. They conducted experiments on the DeepMind Lab, AI2-THOR, and House3D frameworks and verified the performance improvement of the visual navigation algorithm.

In previous research[45], the DRL algorithm was found to be generalizable to new scenarios, but at the expense of a decrease in performance and the need to fine-tune the network. To improve the generalization ability of the visual navigation algorithm, Devo et al.[52] proposed the importance weighted actor–learner architecture, a new framework comprising object localization and navigation networks. The object localization network takes the target image and current frame as input, and outputs a six-dimensional vector that represents the position of the target in the current frame. The vector and the current frame are then input into the navigation network, and an action decision is generated. The agent only uses sparse rewards, which can be transferred to real environments and real objects without fine-tuning. However, the authors noted that the navigation performance fluctuated, and with some combinations of light and textures, the agent could not achieve satisfactory results. Thus, further research is required.

In addition to using a first-view image to express the target location, Devo et al.[53] studied the navigation task that follows natural language instruction input. Hsu et al.[54] divided the complex indoor environment into different local areas, and generated navigation actions based on the scene image and target location. The latest research interests also include hierarchical RL[55] and the graph structure neural network[56]. Indoor navigation is becoming increasingly practical.

4.3 Multi-robot navigation

4.3.1 Feature

Multi-robot navigation is an extension of the local obstacle-avoidance task performed by a single robot. Scenarios with multiple high-speed mobile robots, such as a warehouse, feature stronger dynamics that bring new challenges to DRL-based navigation. Traditional solutions for multi-robot navigation can be divided into centralized and distributed solutions. Centralized methods typically provide a central server, which requires communication between robots, collects all relevant information, and then determines the actions of each robot using an optimization algorithm. In the classical distributed method[57], each robot makes independent decisions, but must obtain the accurate motion data of other robots, such as speed, acceleration, and path. The DRL method provides a new solution for distributed obstacle avoidance and the cooperation of multiple robots under non-communication conditions.

4.3.2 Development

In 2016, Chen et al.[58] first proposed multi-robot Collision Avoidance with Deep RL (CADRL) in the decentralized non-communicating condition, in which the expensive online calculations of traditional methods were converted into the offline training processes of value networks. The optimal reciprocal collision avoidance method was used to generate the training set. Then, given an agent's joint configuration (position vector, velocity vector, and robot radius) with its neighbors, the value network encoded the estimated time to reach the target. The authors refined the policy using RL and simulated multiple situations.

However, the agent's joint configuration cannot be obtained directly, and the calculation is time-consuming. To simplify the decision-making process, raw sensor data can be directly used as input. Ding et al.[59] studied the hierarchical RL method using lidar data as input. A high-level evaluation module is responsible for perceiving the overall environmental risks, and a low-level control module is responsible for making action decisions. Long et al.[60] have also done a lot of work on the multi-robot collision avoidance problem. They chose 512-dimensional lidar data with the 180° FOV, target position, and current robot velocity as the state space, and continuous translational and rotational velocities as the action space. To improve the training efficiency, they made the artificially designed reward function dense. This artificially designed dense reward function and multi-scenario multi-stage training framework were used to improve the training efficiency of the parallel PPO algorithm, but the authors found that the RL policy still could not produce consistently perfect behavior. Therefore, they combined the classic PID

controller with the DRL policy and proposed a hybrid-RL framework[61, 62], classified the scenarios faced by a robot into three categories, and designed separate control sub-policies for each category.

In addition to avoiding collisions during navigation, multi-robots must often perform cooperative tasks such as maintaining formation. Chen et al.[63] studied the problem of multi-robot formation. In their parallel DDPG method, multiple agents share experience memory data and navigation strategies. Reward-shaping techniques are used to adjust the reward function of single-robot navigation tasks into a multi-robot version, and curriculum learning techniques are used to train robots to complete a series of increasingly difficult tasks. Lin et al.[64] proposed a distributed navigation method based on PPO, which takes the geometric center of the mass of the robot group as the target, and achieves the goal point while avoiding collisions and maintaining connectivity. The authors verified this algorithm in a real system. Recently, Sartoretti et al.[65] combined RL and imitation learning to achieve distributed path planning for thousands of robots in a large-scale environment. Ma et al.[66] successfully applied DRL to a multi-robot hunting problem.

4.4 Social navigation

4.4.1 Feature

Social navigation refers to the movement of a mobile robot in pedestrian-rich environments such as airports and shopping malls. This scenario focuses on obstacle-avoidance capabilities in the complex environments of human society. Social and multi-robot navigation are similar, because we can treat pedestrians as non-cooperative mobile robots. Most local obstacle avoidance and distributed multi-robot navigation methods can be extended to social navigation scenarios, but the density of dynamic obstacles is much higher in crowded environments. In addition, because of the randomness of human behavior, navigation that meets social standards remains difficult to quantify, which widens the difference between simulated and actual environments. Ensuring safe crowd navigation requires further research.

4.4.2 Development

In 2017, Chen et al.[67] extended their previous work CADRL to social navigation scenarios, and proposed the socially aware CADRL algorithm. Various sensors, such as lidar, a depth camera, and a camera in differential drive vehicles, are used to detect pedestrians[68]. The speed, velocity, and size (radius) of a pedestrian are estimated by clustering point-cloud data. The authors considered complex normative motion patterns to possibly be the result of simple local interactions, and they proposed a way to induce behavior that respects the norms of human society. To induce a particular norm, they introduced a small bias to the RL training process in favor of one set of behaviors over others. They conducted an experiment in a real environment with plenty of pedestrians. After further research, they found that their assumptions regarding social norms deviated from reality as the number of agents in the environment increased. Previous research had required fixed-size observations. In 2018, the GA3C-CADRL[69] algorithm combined with LSTM was proposed to avoid collisions between various types of dynamic agents without assuming that they follow any specific behavioral rules. In terms of the reward function, Ciou et al.[70] used reward updates from human feedback to learn appropriate social navigation. Sun et al.[71] calculated the collision probability between the subject and a dynamic obstacle, whereby the higher the collision probability is, the greater the penalty is received by the agent.

Unlike agent-level states such as coordinates and velocity, Sasaki et al.[72] used a local map to represent the status of dynamic crowds. The advantage of this approach is that it can simultaneously identify multiple moving targets and track all targets based on lidar data. Each map is generated from the current step to several previous steps. They designed a unique geometric transformation method, wherein the map contains target location information. They evaluated the A3C algorithm using pedestrian data collected in the real world, but at the time of publication of this review, they had not yet deployed it in a real system.

Recently, Sathyamoorthy et al.[73] studied the freezing robot problem that arises when a robot navigates through dense scenarios and crowds, using a DRL-based hybrid approach to handle crowds of varying densities. Chen et al.[74] focused on the problem of DRL performance degradation when the number of people increases and combined it with a state-of-the-art graph convolutional network technique for training the agent to pay attention to the most critical person in the crowd. Their research efforts will facilitate the development of social navigation. Simple comparison of relevant references is shown in Table 2.

Table 2 Simple comparison of relevant references.

Application scenario	Reference	Algorithm	Perception type	Action space	Reward setting	Simulation	Real system	Year
Local obstacle avoidance	[32]	Uncertainty-aware RL	C	Con, 2	–	Y	Y	2017
	[24]	DDQN	A	Dis, 3	Dense	Y	Y	2017
	[34]	ADDPG (Asynchronous DDPG)	A	Con, 2	Dense	Y	Y	2017
	[26]	Fast-RDPG (Fast Recurrent DPG)	B	Con, 1	Dense	Y	N	2017
	[36]	SF-RL (Successor Feature RL)	D	Dis, 4	Dense	Y	Y	2017
	[37]	DDQN	A	Dis, 8	Dense	Y	Y	2018
	[28]	Salient region detection + AC	C	Con, 2	–	Y	N	2018
	[38]	IL (Imitation Learning) + CPO	A	Con, 2	Dense	Y	Y	2018
	[39]	DDPG	A	Con, 2	Dense	Y	Y	2018
	[42]	SAC (Soft AC)	D	Con, 2	Dense	Y	Y	2019
	[25]	DDQN	A	Dis, 3	Dense	Y	Y	2019
	[27]	Fast-RDPG (Fast Recurrent DPG)	B	Con, 1	Dense	Y	N	2019
	[43]	DDPG	A	Con, 2	Dense	Y	N	2020
	[33]	ICM A3C (Intrinstic Curiosity)	A/C	Dis, –	Dense	Y	Y	2020
	[40]	Improved A3C	A	Con, 2	Sparse	Y	N	2020
	[35]	DDQN	C	Dis, 3	Dense	Y	Y	2020
Indoor navigation	[44]	Nav A3C (A3C + LSTM)	C	Dis, 8	Sparse	Y	N	2017
	[45]	AI2-THOR (Deep siamese AC network)	C	Dis, 4	Dense	Y	Y	2017
	[54]	LSTM + DRL	C	Dis, 3	Dense	Y	Y	2018
	[51]	A2CAT-VN	C	Dis, 8	Dense	Y	N	2019
	[52]	IMPALA	C	Dis, 3	Sparse	Y	Y	2020
	[55]	HISNav framework	D	–	–	Y	Y	2020
	[50]	AppoNav (LSTM+PPO)	C	Dis, 8	Sparse	Y	N	2020
Multi-robot navigation	[58]	CADRL (Coll. Avoidance with Deep RL)	E	Con, 2	Dense	Y	N	2016
	[60]	Parallel PPO	A	Con, 2	Dense	Y	N	2018
	[61]	Hybrid-RL (Parallel PPO)	A	Con, 2	Dense	Y	Y	2018
	[63]	PDDPG (Parallel DDPG)	A	Con, 2	Dense	Y	N	2019
	[64]	PPO	A	Con, 2	Dense	Y	Y	2019
	[65]	PRIMAL (IL + A3C + LSTM)	E	Dis, 5	Dense	Y	Y	2019
Social navigation	[67]	SA-CADRL	E	Con, 2	Dense	Y	Y	2017
	[69]	GA3C-CADRL	E	Dis, 11	Dense	Y	Y	2018
	[72]	A3C	–	Dis, –	Dense	Y	Y	2019
	[71]	PPO + LSTM + collision prediction	E	Con, 2	Dense	Y	N	2019
	[73]	Frozone + DRL	E	Con, 2	Dense	Y	Y	2020

Note: Perception type: "A" denotes the laser range finder, "B" denotes ultrasonic sonar or other range finders, "C" denotes the monocular camera, "D" denotes the depth camera, and "E" denotes agent-level data provided by the system. Action space: "Con" denotes continuous action, "Dis" denotes discrete action, and the number denotes dimension.

5 Current Challenge and Solution

5.1 Partial observation

5.1.1 Challenge

In robotics, RL is typically applied to motion control tasks such as manipulator control, because the state space is considered to be fully observable. However, because of the limited FOV and the uncertainty regarding the state of other subjects, the mobile robot navigation task is partially observable. This causes two problems: (1) the agent cannot uniquely distinguish its state based on individual current observations and (2) the agent can only learn suboptimal strategies.

The MDP assumes that the state is completely observable, and it cannot capture the complex structure of an environment. Many researchers model using the Partially Observable Markov Decision Process (POMDP), an extension of MDP that can obtain more information from historical trajectories. Formally, a POMDP consists of six tuples (S, A, R, P, Ω, O), where Ω is the observation space ($o_t \in \Omega$) and O is the observation probability distribution given the

system state ($o_t \sim O(s_t)$). At each time step t, the agent observes o_t, executes an action a_t according to the policy $\pi(a_t|o_t)$, receives a reward r_t, and transfers into a new state s_{t+1} according to the distribution $P(s_{t+1}|s_t, a_t)$.

5.1.2 Solution

Two main methods are used to solve the partial observation problem.

(1) Expansion of network input

Considering that the agent cannot uniquely distinguish its state based on current observations, the simplest solution is to add several previous observation frames as network inputs to improve its ability to distinguish among states[36, 51, 72, 75, 76]. In addition, previous rewards and actions also contain state information, so some studies have input previous rewards and actions to the network[33, 44, 63, 77]. Another input expansion technique is the two-stream Q network proposed by Wang et al.[78], which adds the difference between two frames of laser scanning data.

(2) Addition of memory ability

Compared with the manual selection of the previous observation frame as input, having the agent automatically memorize and process the previous observation is more attractive[79]. The Recurrent Neural Network (RNN), a classic network structure in deep-learning research, uses sequence data as input and has memory capabilities. Previous observations have an impact on the current output of the RNN. Some scholars have integrated an RNN into the DRL framework and proposed RDPG to solve the POMDP. Wang et al.[26, 27] successfully applied this approach to DRL-based navigation tasks.

However, RNNs have a long-term dependencies problem, and the LSTM network was proposed by deep-learning researchers to improve this problem. Numerous DRL-based navigation studies have dealt with POMDP by adding several LSTM layers to the network structure[44, 50, 54, 65, 71]. As a variant of LSTM, the Gated Recurrent Unit (GRU) is easier to train. Zeng et al.[80] built a GRU-based memory neural network and proved that it could improve performance in complex navigation tasks.

5.2 Sparse reward

5.2.1 Challenge

As noted above, the mobile robot navigation task comprises P2P and obstacle avoidance. The agent receives a positive reward when it reaches the target point and a negative reward when it collides with an obstacle. The reward is generated only at the end of each epoch, which means it is very sparse. As we know, effective RL depends on collecting useful reward signals, so agents must identify a series of corrective actions to achieve the goal of generating a sparse reward. The probability of finding a sparse reward signal by a random search is very small. When the surrounding environment becomes more complex and dynamic, the sample space rapidly expands. Sparse reward can aggravate data inefficacy and result in poor training convergence and long training times.

5.2.2 Solution

The well-known experience replay mechanism, exploration, and utilization techniques in RL research can be directly applied to DRL-based navigation. In addition, three techniques can be used to solve the sparse reward problem.

(1) Reward shaping

An intuitive approach is to design dense rewards manually to make the problem easier to learn. By introducing expert knowledge, the agent can earn a reward at each step in which it executes an action and interacts with the environment. Most methods, especially those that can be verified in real systems, adopt reward-shaping techniques.

Generally, based on the target distance, moving closer to a target will generate a small positive reward[24, 34, 58, 60], and moving closer to an obstacle will generate a small negative reward[69]. A small penalty is also imposed at each time step to encourage the robot to reach the target faster[26, 36, 37, 45, 54]. Direction-related rewards can also encourage agents to move toward a target. Sampedro et al.[39] adopted an APF formulation to design their reward function. Other researchers have also designed special reward functions to promote learning[31, 42, 67, 72, 81].

However, manually designed reward functions have two disadvantages: (1) they are closely related to the task scene, and over-fitting a certain scene leads to its non-universality, and (2) an inappropriate reward sometimes leads to the wrong guidance for learning. To solve these problems, Chiang et al.[75] proposed the AutoRL algorithm, which automates the search for both the shaped reward and the neural network architecture. Zhang et al.[82] introduced a general reward function based on a matching network. In addition, expert demonstration can provide an auxiliary experience for DRL, which can be used to address the sparse reward problem[38, 83].

(2) Auxiliary task

In the case of a sparse reward, when it is difficult to complete the original task, we can set auxiliary tasks to accelerate learning. In essence, auxiliary tasks provide extra dense pseudo-rewards for RL[84]. Auxiliary tasks are supervised, so the associated auxiliary loss can be used to adjust the network parameters. The original and auxiliary tasks share part of the network structure, which helps to build up the model representation.

The auxiliary task technique is often used in indoor visual navigation research. Mirowski et al.[44] added two auxiliary tasks to the Nav A3C framework, including depth and loop-closure predictions. They also applied the auxiliary heading prediction task[85] to an urban navigation problem, that is, to predict the angle between the current direction and due north. Hsu et al.[54] trained their local region model with an auxiliary task that can speed up the convergence. Kulhánek et al.[51] further developed the auxiliary task technique for visual navigation by extending the batched A2C algorithm with pixel-control, reward prediction, depth-map prediction, and image segmentation tasks.

(3) Curriculum learning

Curriculum learning refers to the design of appropriate curricula for progressive learners from simple to complex, for example, moving a target from nearby to far away in navigation. The probability of completing a simple task and receiving a reward is higher than the probability of completing the original task, which helps to speed up the neural network training. The complexity of the course is gradually increased.

This technique has been applied in DRL-based navigation task. Chen et al.[76] introduced a two-stage training process for curriculum learning. In the first stage, they trained the policy in a random scenario with eight robots; and in the second stage, they trained the policy in both random and circular scenarios with 16 robots. Similar to curriculum learning[63], multi-stage learning with policy evolution can also improve the DRL training efficiency under the sparse reward condition[40, 86].

5.3 Poor generalization

5.3.1 Challenge

Poor generalization is another challenge in DRL-based navigation. Because training in a physical environment is very time-consuming and dangerous, researchers typically train the agent in a simulation environment and then transfer it to the entity environment. There are two types of generalization:

(1) The agent is transferred from one simulation environment to another. The DRL algorithm essentially trains a reactive strategy to select the action with the maximum cumulative return in its training environment. The data distributions in different environments are very different, so it is not easy to transfer the navigation model trained in one environment to another environment.

(2) The agent is transferred from a simulation environment to a real environment. Generally, the real environment is more complex and dynamic. The reality gap between the virtual and real worlds is the core challenge of deploying a trained model directly into a real robot.

5.3.2 Solution

We can address the generalization problem by either expanding the sample space or reducing the state space.

(1) Expansion of the sample space

Adding randomness to the simulation environment to expand the sample space is a technique commonly used to solve the generalization problem. Random sensor noise, a random target position, random obstacles, and other changes help expand the data distribution of various scenarios and reduce the difficulty of transfer between simulation environments.

To cause the agent to adapt to the new situation, Lei et al.[37] randomly set the target position in an unoccupied area to expand the state-space distribution of the experience pool. Zhang et al.[36] had the agent start from a random position and used four different maze environments for training. Long et al.[60] presented a multi-scenario training framework to learn an optimal policy, which is simultaneously trained using a large number of robots in rich, complex environments. The research of Zhu et al.[45] showed that a model trained in 16 scenes could be applied to new scenes without extra training. The model can then be transferred from a simulation environment to a real one with only a small amount of fine-tuning.

(2) Reduction of the state space

Expanding the sample space is time-consuming and requires subsequent fine-tuning in practical applications. Reducing the state-space dimensions can achieve similar generalization ability at less computational cost. Using the image as a part of the state space has the advantage of abundant information, but there are often details such as illumination, texture, and color changes, which may not be helpful or may even be harmful to the learning navigation ability. Using sparse laser ranging as input

focuses the DRL model on the mapping relationship between the distances among the learning obstacles and the motion instructions.

Most DRL-based navigation research that can be transferred to a real environment has used range finders as the sensor. Tai et al.[34] used only the 10-dimensional sparse laser range findings and related target information as the state space. Their model can be extended to a real mobile robot platform without any fine-tuning. Shi et al.[33] also reported that sparse laser range findings could reduce the reality gap. The system proposed by Yokoyama and Morioka[35] converts the depth images that are estimated from a monocular camera to 2D range data, rather than directly inputting the image.

In addition to reducing the state input size, Devo et al.[52] designed a two-network architecture comprising an object localization network and a navigation network to solve the generalization problem. This architecture reduces the state-space dimension of the navigation network by preprocessing images through the object localization network.

6 Conclusion

Navigation is a core ability of the mobile robot. Although the DRL policy cannot generate perfect behavior all the time and the DRL-based navigation performance fluctuates, DRL continues to show the most promise for achieving breakthroughs in navigation capability.

This paper provided a comprehensive and systematic review of DRL-based mobile robot navigation research. The application scenarios, current challenges, and possible solutions were discussed. We anticipate that this paper will help researchers further improve the current research results and apply them to real systems to realize human-level navigation intelligence.

References

[1] W. Rone and P. Ben-Tzvi, Mapping, localization and motion planning in mobile multi-robotic systems, *Robotica*, vol. 31, no. 1, pp. 1–23, 2013.

[2] J. Engel, T. Schöps, and D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds. Zurich, Switzerland: Springer International Publishing, 2014, pp. 834–849.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, ORB-SLAM: A versatile and accurate monocular SLAM system, *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] G. Grisetti, C. Stachniss, and W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[5] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, A flexible and scalable SLAM system with full 3D motion estimation, presented at 2011 IEEE Int. Symp. Safety, Security, and Rescue Robotics, Kyoto, Japan, 2011, pp. 155–160.

[6] M. Elbanhawi and M. Simic, Sampling-based robot motion planning: A review, *IEEE Access*, vol. 2, pp. 56–77, 2014.

[7] D. Fox, W. Burgard, and S. Thrun, The dynamic window approach to collision avoidance, *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602, 2013.

[9] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, et al., Vector-based navigation using grid-like representations in artificial agents, *Nature*, vol. 557, no. 7705, pp. 429–433, 2018.

[10] A. Pandey, S. Pandey, and D. R. Parhi, Mobile robot navigation and obstacle avoidance techniques: A review, *International Robotics & Automation Journal*, vol. 2, no. 3, pp. 96–105, 2017.

[11] F. Kamil, S. H. Tang, W. Khaksar, Zulkifli N, and S. A. Ahmad, A review on motion planning and obstacle avoidance approaches in dynamic environments, *Advances in Robotics & Automation*, vol. 4, no. 2, p. 1000134, 2015.

[12] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[13] F. Y. Zeng, C. Wang, and S. S. Ge, A survey on visual navigation for artificial agents with deep reinforcement learning, *IEEE Access*, vol. 8, pp. 135 426–135 442, 2020.

[14] C. J. C. H. Watkins, Learning from delayed rewards, PhD dissertation, University of Cambridge, Cambridge, England, 1989.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] H. Van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double Q-learning, in *Proc. 30th AAAI Conf. Artificial Intelligence*, Phoenix, AZ, USA, 2016, pp. 2094–2100.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971, 2015.

[18] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, arXiv preprint arXiv:1602.01783, 2016.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347, 2017.

[20] Q. Shi, S. Zhao, X. W. Cui, M. Q. Lu, and M. D. Jia, Anchor self-localization algorithm based on UWB ranging and inertial measurements, *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 728–737, 2019.

[21] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning, in *Proc. 2018 IEEE Int. Conf. Robotics and Automation*, Brisbane, Australia, 2018, pp. 5113–5120.

[22] M. Duguleana and G. Mogan, Neural networks based reinforcement learning for mobile robots obstacle avoidance, *Expert Systems with Applications*, vol. 62, pp. 104–115, 2016.

[23] S. M. Feng, H. L. Ren, X. R. Wang, and P. Ben-Tzvi, and Asme, Mobile robot obstacle avoidance based on deep reinforcement learning, in *Proc. ASME 2019 Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conf.*, Anaheim, CA, USA, 2019.

[24] Y. Kato, K. Kamiyama, and K. Morioka, Autonomous robot navigation system with learning based on deep Q-network and topological maps, in *Proc. 2017 IEEE/SICE Int. Symp. System Integration*, Taipei, China, 2017, pp. 1040–1046.

[25] Y. Kato and K. Morioka, Autonomous robot navigation system without grid maps based on double deep Q-network and RTK-GNSS localization in outdoor environments, in *Proc. 2019 IEEE/SICE Int. Symp. System Integration*, Paris, France, 2019, pp. 346–351.

[26] C. Wang, J. Wang, X. D. Zhang, and X. Zhang, Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning, in *Proc. 2017 IEEE Global Conf. Signal and Information Processing*, Montreal, Canada, 2017, pp. 858–862.

[27] C. Wang, J. Wang, Y. Shen, and X. D. Zhang, Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach, *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, 2019.

[28] Z. W. Ma, C. Wang, Y. F. Niu, X. K. Wang, and L. C. Shen, A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles, *Robotics and Autonomous Systems*, vol. 100, pp. 108–118, 2018.

[29] J. Woo and N. Kim, Collision avoidance for an unmanned surface vehicle using deep reinforcement learning, *Ocean Eng.*, vol. 199, p. 107001, 2020.

[30] X. Wu, H. L. Chen, C. G. Chen, M. Y. Zhong, S. R. Xie, Y. K. Guo, and H. Fujita, The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method, *Knowl.-Based Syst.*, vol. 196, p. 105201, 2020.

[31] X. Y. Zhang, C. B. Wang, Y. C. Liu, and X. Chen, Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning, *Sensors*, vol. 19, no. 18, p. 4055, 2019.

[32] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, Uncertainty-aware reinforcement learning for collision avoidance, arXiv preprint arXiv:1702.01182, 2017.

[33] H. B. Shi, L. Shi, M. Xu, and K. S. Hwang, End-to-end navigation strategy with deep reinforcement learning for mobile robots, *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2393–2402, 2020.

[34] L. Tai, G. Paolo, and M. Liu, Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation, in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vancouver, Canada, 2017, pp. 31–36.

[35] K. Yokoyama and K. Morioka, Autonomous mobile robot with simple navigation system based on deep reinforcement learning and a monocular camera, in *Proc. 2020 IEEE/SICE Int. Symp. System Integration*, Honolulu, HI, USA, 2020, pp. 525–530.

[36] J. W. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, Deep reinforcement learning with successor features for navigation across similar environments, in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vancouver, Canada, 2017, pp. 2371–2378.

[37] X. Y. Lei, Z. Zhang, and P. F. Dong, Dynamic path planning of unknown environment based on deep reinforcement learning, *Journal of Robotics*, vol. 2018, p. 5781591, 2018.

[38] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations, *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[39] C. Sampedro, H. Bavle, A. Rodriguez-Ramos, P. de la Puente, and P. Campoy, Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning, in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Madrid, Spain, 2018, pp. 1024–1031.

[40] C. Wang, J. Wang, J. J. Wang, and X. D. Zhang, Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards, *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6180–6190, 2020.

[41] F. Aznar, M. Pujol, and R. Rizo, Obtaining fault tolerance avoidance behavior using deep reinforcement learning, *Neurocomputing*, vol. 345, pp. 77–91, 2019.

[42] J. Choi, K. Park, M. Kim, and S. Seok, Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view, in *Proc. 2019 Int. Conf. Robotics and Automation*, Montreal, Canada, 2019, pp. 5993–6000.

[43] F. Leiva and J. Ruiz-del-Solar, Robust RL-based map-less local planning: Using 2D point clouds as observations, *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5787–5794, 2020.

[44] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al., Learning to navigate in complex environments, arXiv preprint arXiv:1611.03673, 2017.

[45] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in *Proc. 2017 IEEE Int. Conf. Robotics and Automation*, Singapore,

2017, pp. 3357–3364.

[46] J. Oh, V. Chockalingam, S. Singh, and H. Lee, Control of memory, active perception, and action in minecraft, arXiv preprint arXiv:1605.09128, 2016.

[47] G. Brunner, O. Richter, Y. Y. Wang, and R. Wattenhofer, Teaching a machine to read maps with deep reinforcement learning, arXiv preprint arXiv:1711.07479, 2017.

[48] Y. Wu, Y. X. Wu, G. Gkioxari, and Y. D. Tian, Building generalizable agents with a realistic and rich 3D environment, arXiv preprint arXiv:1801.02209, 2018.

[49] S. R. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, Semantic scene completion from a single depth image, arXiv preprint arXiv:1611.08974, 2016.

[50] F. Y. Zeng and C. Wang, Visual navigation with asynchronous proximal policy optimization in artificial agents, *Journal of Robotics*, vol. 2020, p. 8702962, 2020.

[51] J. Kulhánek, E. Derner, T. de Bruin, and R. Babuška, Vision-based navigation using deep reinforcement learning, arXiv preprint arXiv:1908.03627, 2019.

[52] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi, Towards generalization in target-driven visual navigation by using deep reinforcement learning, *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1546–1561, 2020.

[53] A. Devo, G. Costante, and P. Valigi, Deep reinforcement learning for instruction following visual navigation in 3D maze-like environments, *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1175–1182, 2020.

[54] S. H. Hsu, S. H. Chan, P. T. Wu, K. Xiao, and L. C. Fu, Distributed deep reinforcement learning based indoor visual navigation, in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Madrid, Spain, 2018, pp. 2532–2537.

[55] A. Staroverov, D. A. Yudin, I. Belkin, V. Adeshkin, Y. K. Solomentsev, and A. I. Panov, Real-time object navigation with deep neural networks and hierarchical reinforcement learning, *IEEE Access*, vol. 8, pp. 195 608–195 621, 2020.

[56] Y. Lu, Y. R. Chen, D. B. Zhao, and D. Li, MGRL: Graph neural network based inference in a Markov network with reinforcement learning for visual navigation, *Neurocomputing*, vol. 421, pp. 140–150, 2021.

[57] Z. Fan, G. S. Pereira, and V. Kumar, Cooperative localization and tracking in distributed robot-sensor networks, *Tsinghua Science and Technology*, vol. 10, no. 1, pp. 91–101, 2005.

[58] Y. F. Chen, M. Liu, M. Everett, and J. P. How, Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning, arXiv preprint arXiv:1609.07845, 2016.

[59] W. H. Ding, S. J. Li, H. H. Qian, and Y. Q. Chen, Hierarchical reinforcement learning framework towards multi-agent navigation, in *Proc. 2018 IEEE Int. Conf. Robotics and Biomimetics*, Kuala Lumpur, Malaysia, 2018, pp. 237–242.

[60] P. X. Long, T. X. Fan, X. Y. Liao, W. X. Liu, H. Zhang, and J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, arXiv preprint arXiv:1709.10082, 2018.

[61] T. X. Fan, P. X. Long, W. X. Liu, and J. Pan, Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios, arXiv preprint arXiv:1808.03841, 2018.

[62] T. X. Fan, P. X. Long, W. X. Liu, and J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios, *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, 2020.

[63] W. Z. Chen, S. Z. Zhou, Z. S. Pan, H. X. Zheng, and Y. Liu, Mapless collaborative navigation for a multi-robot system based on the deep reinforcement learning, *Applied Sciences*, vol. 9, no. 20, p. 4198, 2019.

[64] J. T. Lin, X. Y. Yang, P. W. Zheng, and H. Cheng, End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning, in *Proc. 2019 IEEE Int. Conf. Mechatronics and Automation*, Tianjin, China, 2019, pp. 2493–2500.

[65] G. Sartoretti, J. Kerr, Y. F. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning, *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[66] J. C. Ma, H. M. Lu, J. H. Xiao, Z. W. Zeng, and Z. Q. Zheng, Multi-robot target encirclement control with collision avoidance via deep reinforcement learning, *Journal of Intelligent & Robotic Systems*, vol. 99, no. 2, pp. 371–386, 2020.

[67] Y. F. Chen, M. Everett, M. Liu, and J. P. How, Socially aware motion planning with deep reinforcement learning, in *Proc. 2017 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vancouver, Canada, 2017, pp. 1343–1350.

[68] L. Chen, N. Ma, P. Wang, J. H. Li, P. F. Wang, G. L. Pang, and X. J. Shi, Survey of pedestrian action recognition techniques for autonomous driving, *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 458–470, 2020.

[69] M. Everett, Y. F. Chen, and J. P. How, Motion planning among dynamic, decision-making agents with deep reinforcement learning, in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Madrid, Spain, 2018, pp. 3052–3059.

[70] P. H. Ciou, Y. T. Hsiao, Z. Z. Wu, S. H. Tseng, and L. C. Fu, Composite reinforcement learning for social robot navigation, in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Madrid, Spain, 2018, pp. 2553–2558.

[71] L. B. Sun, J. F. Zhai, and W. H. Qin, Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning, *IEEE Access*, vol. 7, pp. 109 544–109 554, 2019.

[72] Y. Sasaki, S. Matsuo, A. Kanezaki, and H. Takemura, A3C based motion learning for an autonomous mobile robot in crowds, in *Proc. 2019 IEEE Int. Conf. Systems, Man and Cybernetics*, Bari, Italy, 2019, pp. 1036–1042.

[73] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, Frozone: Freezing-free, pedestrian-friendly navigation in human crowds, *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[74] Y. Y. Chen, C. C. Liu, B. E. Shi, and M. Liu, Robot navigation in crowds by graph convolutional networks with attention learned from human gaze, *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.

[75] H. T. L. Chiang, A. Faust, M. Fiser, and A. Francis, Learning navigation behaviors end-to-end with AutoRL, *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[76] G. D. Chen, S. Y. Yao, J. Ma, L. F. Pan, Y. A. Chen, P. Xu, J. M. Ji, and X. P. Chen, Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning, *Sensors*, vol. 20, no. 17, p. 4836, 2020.

[77] V. J. Hodge, R. Hawkins, and R. Alexander, Deep reinforcement learning for drone navigation using sensor data, *Neural Computing and Applications*, vol. 33, no. 6, pp. 2015–2033, 2021.

[78] Y. D. Wang, H. B. He, and C. Y. Sun, Learning to navigate through complex dynamic environment with modular deep reinforcement learning, *IEEE Transactions on Games*, vol. 10, no. 4, pp. 400–412, 2018.

[79] E. Parisotto and R. Salakhutdinov, Neural map: Structured memory for deep reinforcement learning, arXiv preprint arXiv:1702.08360, 2017.

[80] J. J. Zeng, R. S. Ju, L. Qin, Y. Hu, Q. J. Yin, and C. Hu, Navigation in unknown dynamic environments based on deep reinforcement learning, *Sensors*, vol. 19, no. 18, p. 3837, 2019.

[81] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-del-Solar, Visual navigation for biped humanoid robots using deep reinforcement learning, *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247–3254, 2018.

[82] Q. C. Zhang, M. Q. Zhu, L. Zou, M. Li, and Y. Zhang, Learning reward function with matching network for mapless navigation, *Sensors*, vol. 20, no. 13, p. 3664, 2020.

[83] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne, Deep imitation learning for 3D navigation tasks, *Neural Computing and Applications*, vol. 29, no. 7, pp. 389–404, 2018.

[84] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, Reinforcement learning with unsupervised auxiliary tasks, arXiv preprint arXiv:1611.05397, 2016.

[85] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, Learning to navigate in cities without a map, arXiv preprint arXiv:1804.00168, 2019.

[86] D. W. Wang, T. X. Fan, T. Han, and J. Pan, A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing, *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3098–3105, 2020.

Tao Zhang received the BS, MS, and PhD degrees in control science and engineering from Tsinghua University, Beijing, China in 1993, 1995, and 1999, respectively, and the second PhD degree from Saga University, Saga, Japan in 2002. He was a visiting associate professor with Saga University, from 1999 to 2003. From 2003 to 2006, he worked as a researcher with the National Institute of Informatics, Tokyo, Japan.

He is currently a professor and serves as the dean of the Department of Automation, School of Information Science and Technology, Tsinghua University, Beijing. He has authored or coauthored more than 200 papers and eight books. He is a fellow of IET, and a member of the American Institute of Aeronautics and Astronautics and Institute of Electronics, Information and Communication Engineers. He currently serves as the editorial board member and technical editor for *IEEE/ASME Transactions on Mechatronics*. His current research includes robotics, image processing, control theory, artificial intelligent, navigation, and control of spacecraft.

Kai Zhu received the BS degree in mechanical engineering from Tsinghua University, Beijing, China in 2019. He is currently pursuring the PhD degree at the Department of Automation, Tsinghua University. His research interests include path planning, autonomous navigation, deep reinforcement learning, and their applications in multi-robot collaboration.