Map-less End-to-end Navigation of Mobile Robots via Deep Reinforcement Learning

Haiyue Ma, Siqi Wang, Shouwu Zhang, Song Ren, Heng Wang
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
hengwang@ustb.edu.cn

2023 IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC) | 979-8-3503-3172-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICEIEC58029.2023.10200799

Abstract—**A deep reinforcement learning model with a unique long-term memory capability is proposed in this paper, which addresses the map-less navigation of mobile robots in dynamic environments. Based on the recurrent neural network, the proposed model takes continuous historical states as input and better handles dynamic obstacles. Furthermore, a novel reward function is designed to ensure smooth navigation trajectories and satisfactory navigation results in dynamic environments. The proposed approach is evaluated on the Gazebo simulation platform, and higher navigation success rates in dynamic environments are achieved.**

Keywords-deep reinforcement learning; map-less; navigation; ROS; LSTM

I. INTRODUCTION

Navigation is a fundamental challenge in robotics, particularly when it involves achieving end-to-end navigation in complex and unknown environments. Traditional methods for unmanned vehicle navigation rely on SLAM [1] algorithms for map building, path planning algorithms such as A* [2] and D* [3] for global path planning, and tracking controllers for implementation. However, these methods have several limitations. Visual SLAM [4] algorithms heavily rely on objective factors such as camera parameters and lighting, while laser SLAM [5] requires expensive laser radar sensors, and the algorithm's accuracy heavily relies on sensor precision. Moreover, global path planning algorithms require static maps and struggle to handle complex dynamic environments [6]. On the other hand, local path planning methods [7] require a significant amount of computation, making it challenging to guarantee real-time planning.

Deep Reinforcement Learning (DRL) [8], a powerful combination of Deep Learning (DL) and Reinforcement Learning (RL), has emerged as a promising solution to these challenges. DRL methods have shown effectiveness in training unmanned vehicles to navigate through unknown environments in simulation without relying on pre-existing maps or path planning algorithms [9]. Recently, various DRL-based navigation methods [10] have been proposed for different scenarios. For instance, an asynchronous DDPG algorithm using sparse LiDAR data to achieve map-free end-to-end navigation in simple static scenes was proposed by Shi et al. [11]. A navigation strategy is proposed by Semnani et al. [12], which introduces an LSTM network structure that can use an arbitrary number of states as network inputs to generate appropriate collision-free paths even when the number of dynamic obstacles in the environment changes. Furthermore, an enhanced TD3 with two stream network structure by introducing spatial change information was proposed by Zhang et al. [13], which has shown improved reward and training success rates, as well as reduced training time. These studies show that DRL-based navigation methods have the potential to address the challenges of autonomous navigation in complex and dynamic environments, making significant contributions to the field of robotics and autonomous systems.

However, existing DRL-based navigation methods still have limitations, especially when dealing with high-dimensional sensory inputs and predicting the motion of dynamic obstacles in complex environments. To address these issues, we propose the Long-Term Twin Delayed Deterministic Deep Reinforcement Learning (LTD3) algorithm for end-to-end navigation in complex and dynamic environments. This novel approach integrates Long Short-Term Memory (LSTM) networks [14] into the DRL model and utilizes laser scanning data, which has superior generalization capabilities compared to images, as inputs. This approach allows the model to store long-term navigation experience and predict the movement of dynamic obstacles, resulting in more effective navigation strategies. Our method addresses the limitations of existing DRL-based navigation approaches and brings significant improvements in autonomous navigation in complex and dynamic environments. Our model was validated in a Gazebo environment based on the ROS interface and produced promising results.

The contributions of this paper are three folds:

- The LTD3 algorithm is proposed for end-to-end navigation in complex and dynamic environments, overcoming some limitations of existing DRL-based navigation methods.
- A novel reward mechanism is introduced, which considers multiple factors of the navigation task to overcome the challenge of reward sparsity.
- The proposed neural network structure combines LSTM and Fully-Connected (FC) layers in parallel, improving model performance.

This work was supported by the National Natural Science Foundation of China (62173029, 62273033) and the State Key Laboratory of Automotive Safety and Energy under Project No. KFY2214.

979-8-3503-3172-1/23/$31.00©2023 IEEE

The rest of the paper is structured as follows: Section II defines the problem and outlines the DRL navigation setup, Section III describes the experiments, and Section IV concludes the paper.

II. DRL NAVIGATION SETUP

A new DRL model, called LSTM-TD3 (LTD3), is proposed in this paper, which builds upon the TD3 algorithm. Our approach addresses two key challenges that can hinder the performance of existing navigation systems: the TD3 algorithm's inability to store historical state information and the model's ability to handle dynamic obstacles. With these limitations being addressed, higher navigation success rates in complex and dynamic environments are aimed to be achieved.

A. Problem Definition

A robust DRL model for mobile ground robots is provided in this paper, such a model can be mapped as a translation function:

$$v_t = f_t(s_t, p_t, v_{t-1}) \tag{1}$$

where s_t are the information observed from the original sensor information for the last three-time steps, respectively, p_t is the position of the target in the Cartesian coordinate system, and v_{t-1} is the velocity of the agent in the last time step. The parameters of the model can be regarded as the current state of the agent, and the model can map this state directly to next velocity v_t of the agent, effectively serving as a motion planner for the system. To ensure that the agent can respond accurately and efficiently to new observations, the model must provide real-time control.

B. Algorithm

LTD3 uses a strategy network, similar to the Actor network in the TD3, to update the strategy based on deterministic policy parameters that output the deterministic action value. Additionally, a value network is used to fit the value function and also provide gradient information to the update of the value network, similar to the Critic network in the TD3

The Actor network $\mu(s;\theta)$i.e., the strategy network, updates parameters θ. The strategy network gradient is updated as follows:

Actor network

$$\nabla_\theta J = E_s\left[\nabla_\varphi Q(s,a;\varphi)|_{a=\mu(s,\theta)} \nabla_\theta \mu(s;\theta)\right] \tag{2}$$

where $a = \mu(s,\theta)$ is the action generated in the current state, s is the current state and $\nabla_\theta \mu(s;\theta)$ is the target Actor network.

The Critic network $Q(s,a;\varphi)$i.e., the value network updates parameters φ. The value network gradient is updated as follows:

Critic Network

$$\begin{cases} \nabla_\varphi J = E_{s,a,r,s'}\left[(y - Q(s,a;\varphi'))\nabla_\varphi Q(s,a;\varphi)\right] \\ y = r + \nabla_\varphi \gamma Q'(s, \mu'(s;\theta')) \end{cases} \tag{3}$$

Q(s, a) represents a state of affairs Q-value (Q-value) in a DRL model represents the long-term cumulative reward that can be obtained by performing an action a in a state s, In this formulas, the y = Q(s, a) is a estimated value

where s, a, r, s' is the current moment state, action, reward, and next moment state; γ is the decay coefficient and y is the target Critic network.

One effective technique for training deep neural networks with value-based methods is the soft update method. This method mitigates the issue of instability in training by making the parameters of the target network change less, which in turn stabilizes the gradients of the online network, facilitating convergence. The soft update is performed by gradually updating the target network parameters based on a weighted average of the current target network parameters and the current online network parameters. Specifically, at each time step, the target network parameters are updated as follows:

$$\begin{cases} \theta' \leftarrow \tau\theta + (1-\tau)\theta' \\ \varphi' \leftarrow \tau\varphi + (1-\tau)\varphi' \end{cases} \tag{4}$$

where θ' and θ are the parameters of the target and current actor networks, φ' and φ are the parameters of the target and current critic networks, respectively, and τ is the soft update factor.

In unmanned vehicle navigation, the training process involves exploring the environment to obtain the optimal trajectory. To simulate real-world scenarios, we add Gaussian noise to the output of the policy network. This approach enhances environment exploration efficiency, reduces trial and error attempts, and improves the model's generalization and robustness across scenarios. In this paper, the noise function is added as follows:

$$a_t = \begin{cases} \pi(s'_t) + \epsilon, \epsilon \sim clip(\mathcal{N}(0,\sigma), -c, c) & \text{if } \rho < \varepsilon_t \\ \pi(s'_t) & \text{if } \rho \geq \varepsilon_t \end{cases} \tag{5}$$

where $\pi(s'_t)$ is the action output by the actor network, ϵ is the noise we designed, $\mathcal{N}(0,\sigma)$ is a positive terrestrial distribution with σ as the parameter, c is the noise limit, ρ is a random number between 0 and 1, ε_t is the noise threshold at time t.

C. Reward Function Definition

The reward function used in this study includes five different components, namely, r_s, r_c, r_t, r_g, and r_d. The reward function in the algorithm of this paper is designed as follows:

$$r(s_t, a_t) = r_s(\omega_t, v_t) + r_c(d) + r_t + r_g + r_d(d_\Delta) \tag{6}$$

The reward system adopted in this study includes a smooth reward design for the angular velocity, aimed at ensuring the smoothness of the trajectory. To this end, penalties of $r_{unsmooth}$ are imposed per step for excessive angular velocity or insufficient linear speed to prevent the intelligent body from turning excessively or decelerating excessively, resulting in a non-smooth trajectory.

$$r_s(\omega_t, v_t) = r_{unsmooth} \quad \text{if } |\omega_t| > max_\omega \text{ or } v_t < max_v \tag{7}$$

A collision reward r_c is also implemented in the study to ensure the safety of the agent during navigation. When the obstacle distance is detected by the agent's sensor to be less than $d_{collision}$, a larger penalty is applied, and the round is terminated. If the distance d is between $d_{collision}$ and d_{close}, the penalty is smaller, but the round is not interrupted for safety reasons.

$$r_c(d) = \begin{cases} r_{collision} & \text{if } d \le d_{collision} \\ r_{close} & \text{if } d_{collision} < d \le d_{close} \end{cases} \quad (8)$$

restrict the time

The r_t component is a time consumption penalty for each time step, which encourages the unmanned vehicle to find the optimal path to the target in a shorter time. The r_g component is the reward for reaching the target point safely, and it provides a large reward value when the unmanned vehicle makes contact with the target point.

Distance reward r_d is a key component of the reward design in this study. Specifically, the reward is computed as the difference between the Euclidean distance $d_{current}$ from the current agent to the target point and the Euclidean distance $d_{previous}$ from the previous agent to the target point. This difference is then multiplied by a bonus factor Δ to encourage the agent to make progress towards the target.

$$r_d(d_\Delta) = \left(d_{current} - d_{previous}\right) * \Delta \quad (9)$$

The total reward function is the accumulation of these five different components. This design allows the unmanned vehicle to obtain a safe, continuous, and smooth navigation trajectory while speeding up the convergence of the model.

D. Network Architecture

A novel approach that incorporates LSTM units into the Actor networks for dynamic obstacle navigation in unknown environments is proposed in this paper. The addition of LSTM units endows the neural network with the capacity to store memory, which enables the storage of obstacle-related information during navigation. Consequently, this enhances the algorithm's ability to navigate through dynamic scenes with a higher success rate.

The proposed structure for LTD3 networks is shown in Fig.1. The architecture of Actor network consists of two parallel channels with adjustable output channels in each layer to control model complexity. In channel 1, a two-layer LSTM network predicts the future state sequence, while in channel 2, FC layers extract features from the input state. The outputs from both channels are then combined and fed back into an LSTM memory module for final prediction. To ensure the angular and linear velocities fall within their respective ranges, we use hyperbolic tangent and sigmoid functions, respectively. Each middle layers in our model have 500 hidden neurons, and the network parameters are trained to minimize a suitable loss function.

To evaluate the quality of actions taken by the Actor network in reinforcement learning, the Critic network plays a crucial role by estimating the Q-value, which represents the expected cumulative reward of the current state and the output action of the Actor network. In our proposed approach, an evaluation module comprising fully connected neural network layers is used to process the input states and actions.

Specifically, the input states and actions are concatenated into a sequence and passed through the first two fully connected layers of the evaluation module. The resulting output is then fed into two separate Critic networks that are specific to the LTD3 algorithm. These Critic networks produce estimates of the Q-value, which are compared in magnitude to select the smaller Q-value output by the Critic_1 and Critic_2 networks as the final output of the Critic network.

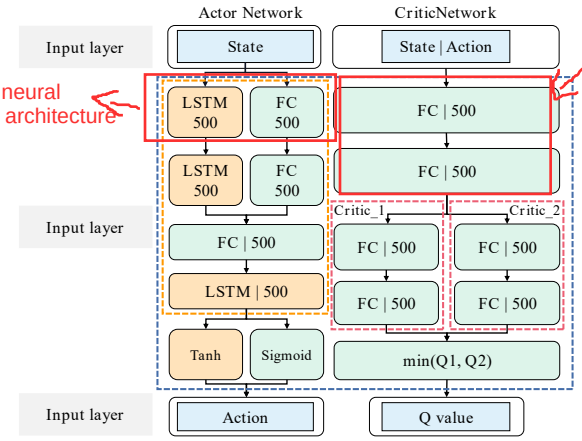


Figure 1. The structure of actor and critic networks.

III. EXPERIMENT

A. Training Setup

20 * 20 = 400 m2

The training environment used in the experiments is illustrated in Fig.2. The enclosed area is a $400m^2$ square with the center serving as the origin. The positive x-axis direction denotes the horizontal direction while the positive y-axis direction denotes the vertical direction. To ensure that training samples are dissimilar, the initial position of the robot is randomly set within the line segment from (10,10) to (10,-10). The target is represented as a red ball with a radius of 0.3m, which appears randomly at a point within the line segment (-10,10) to (-10,-10) each time. The robot is considered to have reached the target when its distance from the target is less than 0.2m. The static obstacle, which comprises squares, cylinders, and spheres (gray obstacles in the figure), are randomly distributed in the positive direction region. The squares have a

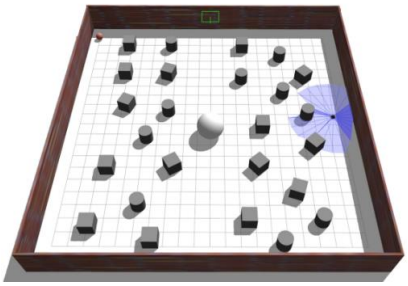
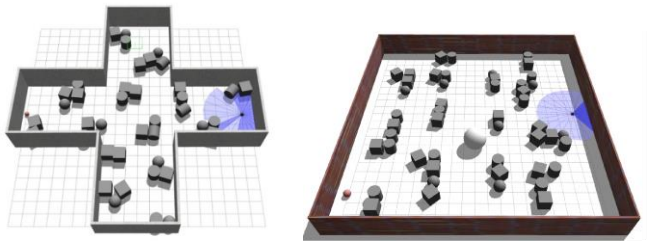


Figure 2. Training environment

side length of 1 m, the cylinders have a height of 1 m and a radius of 0.5 m. In the training scene, there is a dynamic obstacle (white sphere in the center of the figure with a radius of 1 m), which moves randomly between (0,10) and (0,-10) at variable speeds to prevent the robot from reaching the target.

The test environment for evaluating the generalization capability of the proposed method is shown in Fig.3. To investigate the method's ability to perform in diverse scenarios, two environments have been designed. Scenario 1, which is shown in Fig.3 (a), simulates an indoor corridor setting where the corridor is obstructed with miscellaneous objects, leaving only narrow gaps for the robot to navigate through. The robot and target are placed randomly at the end of the corridor, and the robot is never positioned at the same location as the target. This environment has been selected to assess the method's adaptability to complex and cluttered environments.



(a) Static test environment (b) Dynamic test environment

Figure 3. Test environments

Scenario 1, which is shown in Fig.3 (b), is a dynamic scenario designed to simulate an open room environment. In this scenario, obstacles are placed in an orderly manner with a passable gap, allowing the robot to navigate through the environment. The objective of this scenario is to evaluate the proposed method's ability to generalize in a dynamic and complex environment with diverse obstacle configurations.

The TurtleBot3 robot is selected as the training and testing object. A single-line LiDAR is used as the sensor to obtain information about surrounding obstacles. The LiDAR detection range is 0.2-3.5m, the detection angle is 360 degrees around the robot, and the data is divided into 24 dimensions of 15 degrees each. In order to obtain the dynamic obstacle features more accurately, we stitch the LiDAR data of three consecutive time steps and add the target point coordinate information and robot linear and angular velocities to form a 76-dimensional vector as the state space of this paper, which are shown in the TABLE I. In the reward function, $r_{unsmooth}$ is taken as -2, $r_{collision}$ as -200, $d_{collision}$ as 0.25m, d_{close} as 0.35m, r_{close} as -70 and the bonus factor Δ as 600.

TABLE I. STATE SPACE

State Space	Dim
LiDAR Detection in S_{t-2}, S_{t-1}, S_t	24+24+24
Speed	2
Goal Coordinate	2

3 steps: 3*24 =76

(Linear velocity, angular velocity)

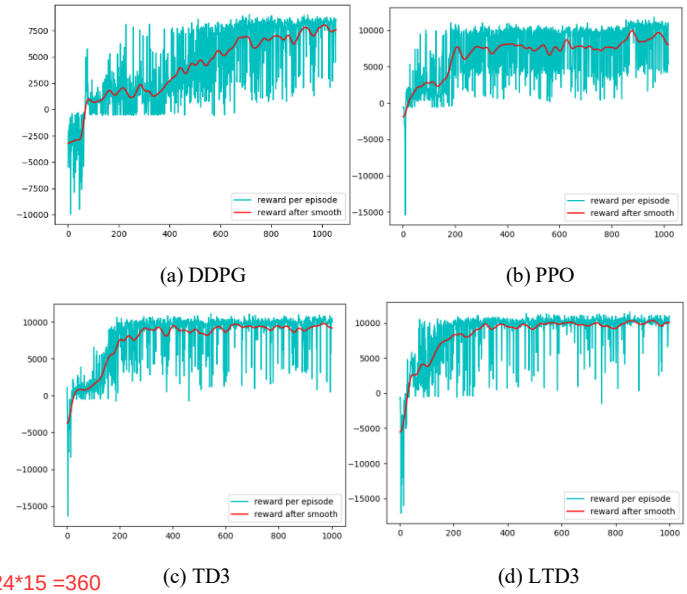
(x, y)

In this paper, all training and testing were performed on a computer with Ubuntu 20.04, the CPU is i9-12900k, and the GPU is RTX3080-12G. The training and testing environments were set up on the ROS noetic robot operating system, and the TurtleBot3-Waffle-Pi was utilized as the robot platform.

B. Comparison experiments

In this section, we present a comprehensive evaluation of the proposed LTD3 method in comparison to existing methods, including DDPG [15], PPO [16], and TD3. For this purpose, three quantitative metrics were selected: training efficiency, navigation success rate and navigation trajectory smoothness.

The performance evaluation results of DDPG, PPO, TD3, and LTD3 are presented in this section, with a focus on the training efficiency and stability of the models. The training results, in terms of the number of rounds and reward value curves, are shown in Fig.4, the X-axis is the number of trainings, and the Y-axis is the reward value, with the light blue curve representing the actual reward value curve obtained by the robot in each round, and the red curve representing the smoothed curve generated by Gaussian filtering of the reward value curve.



(a) DDPG (b) PPO

(c) TD3 (d) LTD3

24*15 =360

Figure 4. The reward curve for 1000 rounds from scratch.

In the case of dynamic obstacles in the environment, DDPG and PPO are unable to achieve satisfactory training results within 1000 rounds of formal training. The reward value obtained after the curve reaches the convergence state is significantly lower than that obtained after the convergence of TD3 and LTD3. The fluctuation of the reward curve is also larger and less stable for DDPG and PPO.

Comparing the smoothed reward value curves of the four algorithms, the slope of the LTD3 reward value curve within 0-200 rounds is the highest among the four algorithms. This indicates that LTD3 has a higher ability to obtain rewards than the other algorithms in the early stage of training, resulting in higher convergence efficiency in a more complex training environment. Moreover, the reward curves of LTD3 are smoother after convergence, and the ability to obtain reward values in the late training period is more stable when comparing the curves of four algorithms after convergence.

Scenarios1: Static test environment (cruciform corridor)
Scenarios2: Dynamic test environment (square-shaped randomised environment)

We tested the models trained by four algorithms in the training scenario for 200 rounds in each of the two test scenarios, and the navigation success rates are shown in the TABLE II.

In Scenario 1, the navigation success rates of the DDPG-trained and PPO-trained models are below 80%. In contrast, the models trained by TD3 and LTD3 achieve success rates of over 95%, demonstrating a substantial improvement over the other

TABLE II. COMPARISON OF NAVIGATION SUCCESS RATE WITH BASELINE ALGORITHMS

Algorithm	Navigation Success Rates (%)	
	Scenario1	*Scenario2*
DDPG	70.5	36.5
PPO	77.5	43
TD3	95	61.5
LTD3	96.5	83.5

two algorithms in static scenes. In Scenario 2, which includes dynamic obstacles, the success rates of the DDPG-trained and PPO-trained models drop to less than 50%. TD3 fares slightly better, achieving a success rate of 65.5%, but it is still significantly lower than the LTD3 algorithm's 83.5% navigation success rate. These results highlight the effectiveness of our proposed network structure in enhancing the agent's navigation performance in dynamic environments.

To evaluate the effectiveness of our proposed method in terms of trajectory smoothness and obstacle avoidance performance, we conducted a visual analysis of the navigation trajectories generated by the robot in the test scenario. The results are presented in Fig.5, where the blue trajectory represents the baseline trajectory generated by TD3, and the red trajectory represents the optimized trajectory generated by LTD3 after training with our proposed reward function. The trajectories generated by our method are smoother, more realistic, have better curvature control and fewer turning points, and are significantly superior to existing research

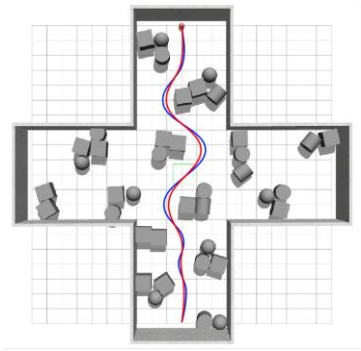


Figure 5. Trajectories generated in the static test environment.

IV. CONCLUSION

In this work, a deep reinforcement learning model called LTD3 is proposed, which aims to solve the problem of end-to-end navigation of robots in dynamic environments without maps. Simulation experiments show that the proposed LTD3 model is able to perform mapless navigation tasks in dynamic environments. By introducing an LSTM network structure and a novel reward function, the model is able to improve the success rate of navigation in dynamic environments and optimize the navigation paths to avoid path redundancy and unsmoothness.

REFERENCES

[1] M. Aizat, A. Azmin, and W. Rahiman, "A survey on navigation approaches for automated guided vehicle robots in dynamic surrounding," *IEEE Access*, vol. 11, pp. 33934–33955, 2023.

[2] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric a-star algorithm: An improved a-star algorithm for agv path planning in a port environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021.

[3] Z. Qadir, F. Ullah, H. S. Munawar, and F. Al-Turjman, "Addressing disasters in smart cities through uavs path planning and 5g communications: A systematic review," *Computer Communications*, vol. 168, pp. 114–135, 2021.

[4] D. Zou, P. Tan, and W. Yu, "Collaborative visual slam for multiple agents: A brief survey," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 461–482, 2019.

[5] W. Ali, P. Liu, R. Ying, and Z. Gong, "A feature based laser slam using rasterized images of 3d point cloud," *IEEE Sensors Journal*, vol. 21, no. 21, pp. 24422–24430, 2021.

[6] J. Chen, C. Du, Y. Zhang, P. Han, and W. Wei, "A clustering-based coverage path planning method for autonomous heterogeneous uavs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25546–25556, 2021.

[7] Q. Luo, H. Wang, Y. Zheng, and J. He, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Computing and Applications*, vol. 32, pp. 1555–1566, 2020.

[8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. P´erez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[9] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6569–6576, 2021.

[10] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 856–892, 2020.

[11] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[12] H. 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, 2019.

[13] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[14] F. Zhang, J. Li, and Z. Li, "A td3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment," *Neurocomputing*, vol. 411, pp. 206–215, 2020.

[15] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "Learn to navigate autonomously through deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342–5352, 2021.

[16] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.