Multiagent Formation Control and Dynamic Obstacle Avoidance Based on Deep Reinforcement Learning

Zike Yuan, Chenhao Yao, Xiaoxu Liu, Zhiwei Gao, *Fellow, IEEE*, and Wenwei Zhang

***Abstract*—Multiagent formation obstacle avoidance is a crucial research topic in the field of multiagent cooperative control, and deep reinforcement learning has shown remarkable potential in this domain. However, most existing studies are not fully distributed and often involve relatively simple scenarios. In this article, we propose an advanced method based on multiagent deep reinforcement learning to address formation and obstacle avoidance in dynamic obstacles environments. For handling complex environments with an unknown number of obstacles, we use long short-term memory (LSTM) networks to encode dynamic obstacles, thereby improving the efficiency of obstacle avoidance. Our method achieves formation and obstacle avoidance in scenarios with both dynamic and static obstacles, where agents coordinate through fully independent and autonomous decision-making. We utilize the multiagent proximal policy optimization (MAPPO) algorithm for centralized training and distributed execution, enhancing the agents' formation and obstacle avoidance capabilities in complex settings. Through simulation and real-world experiments, and by comparing with benchmark methods, we demonstrate significant improvements in formation effectiveness and obstacle avoidance success rates, showcasing the superiority and practicality of our proposed approach.**

***Index Terms*—Collision avoidance, deep reinforcement learning, formation control, multiagent, multiagent proximal policy optimization (MAPPO).**

I. Introduction

MULTI-AGENT formation control have become hot topics recently, receiving extensive research in the context of vehicle swarm cooperation. It has great potential in various fields such as collaborative exploration [1], logistics [2], [3], and safety rescue [4]. One fundamental problem is how to maintain formation and avoid collisions with obstacles. For this problems, hierarchical control [5], [6], optimal control method [7], [8], adaptive control [9], and fuzzy control [10] have been employed to maintain formation while avoiding obstacles. This type of method mainly focuses on the interference and physical constraints of the agents themselves and the outside world in the formation system, and on this basis has achieved good results in formation and obstacle avoidance. However, they mainly focus on avoiding static obstacles, and their autonomous obstacle avoidance capabilities in dynamic obstacles environments, especially in dense scenarios, are weak. Furthermore, in many formation problems, a leader-follower structure is considered [11]. The effectiveness and quality of the formation are closely related to the leader. If the leader encounters problems,such as communication or external constraints, it can lead to the failure of the formation. Ref. [12] has conducted research on this issues and achieved some progress. In dynamic obstacle avoidance, Optimal Reciprocal Collision Avoidance (ORCA) [13] has gained significant attention for its real-time avoidance capabilities in dynamic obstacles environments. The ORCA algorithm predicts the trajectories of obstacles and optimizes the current and expected velocities to avoid collisions. It performs well in handling multiple moving obstacles, particularly in dense scenarios, where its advantages are more pronounced. However, several limitations significantly restrict its application. First, simulation-based studies [13], [14] assume that each agent has perfect perception of the surrounding environment, which is unrealistic in the real world. In addition, sensitivity to parameters and getting trapped in local optima can also lead to avoidance failures.

In recent years, learning-based methods have been extensively explored in the fields of agent planning and control. Among these, deep reinforcement learning (RL) not only provides the capability for automated decision-making, but also enables efficient policy learning in complex environments. By combining deep neural networks with the RL framework, and constructing Markov decision processes (MDP) [15], deep RL allows agents to systematically evaluate and optimize future decision. In the area of multiagent control, it has been widely studied in dynamic obstacle avoidance for agents [16], [17], [18], as well as in swarm formation control or planning [19], [20], [21], [22].

Received 5 November 2024; revised 1 January 2025; accepted 31 January 2025. This work was supported in part by National Natural Science Foundation of China under Grant 62003218, in part by the Stable Support Projects for Shenzhen Higher Education Institutions under Grant 20220717223051001, and in part by the Guangdong Provincial Engineering Technology Research Center for Materials for Advanced MEMS Sensor Chip under Grant 2022GCZX005. Paper no. TII-24-5862. *(Corresponding author: Xiaoxu Liu.)*

Zike Yuan, Chenhao Yao, Xiaoxu Liu, and Wenwei Zhang are with the Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, Shenzhen 518118, China (e-mail: yuanzike2022@email.szu.edu.cn; 202100103093@stumail.sztu.edu.cn; liuxiaoxu@sztu.edu.cn; zhangwenwei@sztu.edu.cn).

Zhiwei Gao is with the Department of Mathematics, Physics and Electrical Engineering, Northumbria University, NE1 8ST Newcastle upon Tyne, U.K. (e-mail: zhiwei.gao@northumbria.ac.uk).

Video: https://www.bilibili.com/video/BV1iy8terEG3.

Digital Object Identifier 10.1109/TII.2025.3545083

However, In formation problems, existing methods are either not fully distributed or adopt a leader-follower structure, and therefore, do not provide a truly distributed solution. In addition, in the realm of obstacle avoidance, most discussions focus on collision avoidance in static environments. Research on collision avoidance in dynamic obstacles environments and dense settings is limited, and the success rates are not ideal. Moreover, there are few studies that consider the combination of formation and obstacle avoidance, especially in scenarios involving dynamic obstacles. In cases where success rates are low, most of the research has not been validated through physical experiments.

Given the existing challenges in multiagent formation obstacle avoidance research, we are motivated to address and optimize the following issues.

1) Develop a method to achieve fully distributed multiagent formations using deep RL.
2) Design more efficient and flexible strategies for obstacle avoidance and formation maintenance in complex environments with both dynamic and static obstacles, adapting to varying numbers of obstacles and scenarios.

To address these issues, we have developed an algorithm for formation and dynamic obstacle avoidance based on multiagent RL, optimized using multiagent proximal policy optimization (MAPPO) [23]. We have noted the advantages of long short-term memory (LSTM) networks in fusing and extracting features of multiple obstacles during agent obstacle avoidance [17], and we incorporate this network into the policy network to enhance the system's obstacle avoidance capabilities in complex environments where multiple dynamic and static obstacles coexist.We adopted a multiagent RL approach to design policy learning that maintains formation characteristics, ensuring that agents strive to preserve formation consistency during the decision-making process. We compared our algorithm with other methods and validated it through physical experiments, demonstrating the superiority of our approach. The main contributions of this article are as follows.

1) We adopted a fully distributed multiagent RL method, MAPPO, enabling independent decision-making through centralized training and distributed execution. This method achieves better performance in complex environments.
2) To address complex environments with dynamic and static obstacles, we utilized LSTM to encode dynamic obstacle information, reducing network complexity and improving the adaptability of the algorithm. Compared to existing methods, this approach more effectively handles unknown and varying numbers of dynamic obstacles, demonstrating improved obstacle avoidance efficiency.
3) We achieved simultaneous formation maintenance and obstacle avoidance in challenging scenarios with both dynamic and static obstacles. Our approach enables agents to coordinate autonomously and independently. Compared to the two benchmark methods (ORCA-F [20] and APF-F [24]), our method significantly improved the success rate of formation obstacle avoidance and the efficiency of reaching the destination. This highlights the robustness and practicality of our approach in real-world applications.

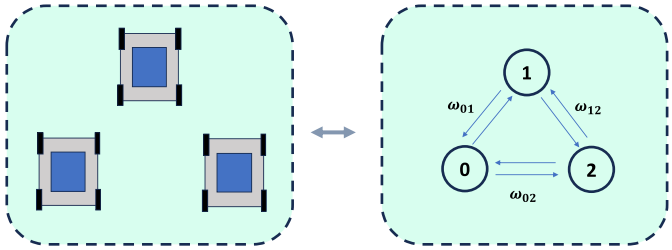

Fig. 1. Multiagent formation representation based on graph theory.

The rest of this article is organized as follows. Section II describes the problem formulation, including the formation and multiagent collaboration RL model. Section III introduces our methodology, covering state acquisition, reward design, network structure, and training methods. Section IV analyzes the results, demonstrating the superiority of our approach through comparison with two benchmark methods and deploying the method in practical experiments. We also compared our method with the method without LSTM to prove that feature extraction can improve obstacle avoidance performance. Finally, Section V concludes this article.

II. Problem Formulation

This article studies the formation control problem of multiple agents in a 2-D space, navigating from a starting point to a target point while being disturbed by dynamic obstacles. The agent model uses omnidirectional mobile robots with Mecanum wheels. The state of each agent i includes its current position $\boldsymbol{p_i}(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$ and current velocity $\boldsymbol{v_i}(t) = (v_{x_i}, v_{y_i})$. Agents select their velocity information as their control inputs.

A. Formation Description

A formation of N agents is modeled by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as shown in Fig. 1, where $\mathcal{V} := \{1, 2, \ldots, N\}$ is the set of vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. In graph $\mathcal{G}$, an edge $e_{ij} \in \mathcal{E}$ that connects vertex $i \in \mathcal{V}$ and vertex $j \in \mathcal{V}$ indicates that agents i and j can measure the geometric distance between each other. We assume that the communication between agents is mutual; hence, the formation graph $\mathcal{G}$ is a complete graph. The weight of edge e_{ij} is given by

$$w_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|^2, \quad (i, j) \in \mathcal{E}. \tag{1}$$

We use the adjacency matrix $\mathbf{A}$ to represent the mutual communication relationships between agents, and the degree matrix $\mathbf{D}$ to indicate the connection weights between agents. The Laplacian matrix $\mathbf{L}$ is then given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2}$$

B. Multiagent Reinforcement Learning

For the formation coordination problem of multiple agents, we treat it as a fully cooperative multiagent task [25]. The formation composed of N agents can be modeled as an N-agent MDP: $(\mathcal{S}, \mathcal{A}, P, r, \mathcal{O}, \mathcal{N}, \gamma)$. It includes the observation information set for each agent $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_N\}$ and the action set for

each agent $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$. $\mathcal{S}$ represents the global state of the environment comprising all agents' observation information. Based on the selected actions and the state transition function P, the environment changes and new states are obtained. All agents share the same reward function, $r(s_t, a_t)$. γ is the discount factor. Taking agent i as an example, the reward R_t^i obtained by the agent at time t depends on the state S_t of all agents and the actions $\boldsymbol{A_t}$ of all agents. Each agent's action decision a_i depends on the current policy $\pi_{\theta_i}(\boldsymbol{a_t}|\boldsymbol{o_t})$, where θ_i are the network parameters of the corresponding agent's policy network. The discounted return for an agent can be expressed as $R_t^i = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. In cooperation, we assume that all agents share the same return and value function, i.e., $Q_\pi(S, A) = Q_\pi^1(S, A) = Q_\pi^2(S, A) = \ldots = Q_\pi^i(S, A)$ and $V_\pi(S) = V_\pi^1(S) = V_\pi^2(S) = \ldots = V_\pi^i(S)$, where Q and V represent the action-value function and state-value function, respectively.

We aim to maximize the expected future state value $\mathbb{E}_S[V_\pi(S)]$ by optimizing the network parameters θ^i. The objective function J is defined as follows:

$$J(\theta^1, \ldots, \theta^N) = \mathbb{E}_S[V_\pi(S)]. \tag{3}$$

This implies finding the optimal parameters $\boldsymbol{\theta}^*$ that maximize J

$$\boldsymbol{\theta}^* = \arg \max_{\theta^1, \ldots, \theta^N} J(\theta^1, \ldots, \theta^N) \tag{4}$$

where $\boldsymbol{\theta}^*$ is a set of policy network parameters $(\theta^1, \ldots, \theta^N)$.

III. Reinforcement Learning Framework Design

In this section, we will provide a detailed explanation of our overall framework for formation obstacle avoidance using RL. First, we will describe the state space and action space we have chosen, as well as the reward function designed for formation and dynamic obstacle avoidance problems. Next, we will explain our specially designed policy network and value network, and how we employ a centralized training and distributed deployment approach to execute the action strategies for each agent.

A. State Space and Action Space

Considering that our mobile robots can move omnidirectionally using Mecanum wheels, we have adopted the same action space as described in [20]. The partitioning of this action space is illustrated in Fig. 2. Each agent i has an action input $\boldsymbol{u} = [v, \theta]$, where v denotes the speed selected by the agent, and θ indicates the chosen direction. We utilize a discrete action space, dividing the speed into five levels ranging from 0 to 1 m/s, and the direction into eight distinct directions. In the state space, the local state information obtained by the current agent consists of observations of obstacle information and the agent's own state observations. For the selection of the agent's own information, we choose $\boldsymbol{o}_{\text{agent}} = [g_x, g_y, v, \theta, v_{fx}, v_{fy}]$, where p_x and p_y are the current observed position information, g_x and g_y are the coordinates from the agent to the goal, v and θ are the current speed and direction of the agent, and $\boldsymbol{v_f} = [v_{fx}, v_{fy}]$ is the reference formation speed of the agent, which can be

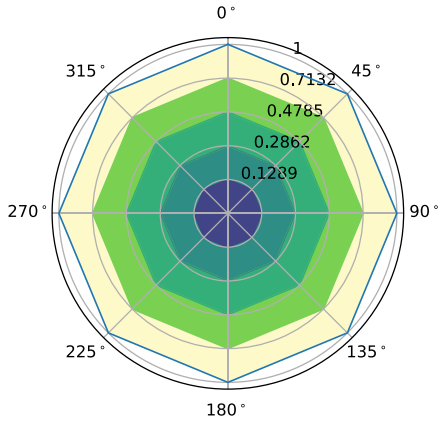

Fig. 2. Agent action space.

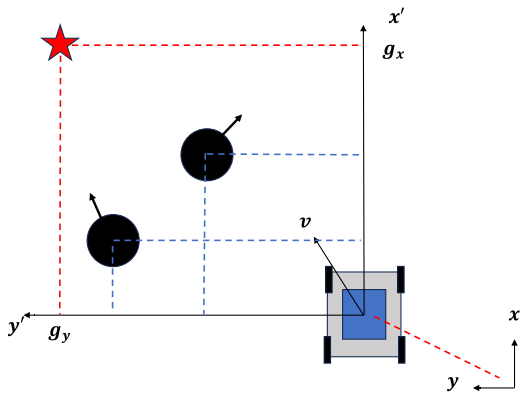

Fig. 3. Schematic diagram of coordinate transformation and status acquisition under current agent observation.

written as

$$\boldsymbol{v_f} = \sum_{j \in \mathcal{N}} \boldsymbol{k_j}(\boldsymbol{x} - \boldsymbol{x_j} - \boldsymbol{x}_{\text{des}}) \tag{5}$$

where $\boldsymbol{k_j}$ represents the gain between the current communicating agents and $\boldsymbol{x}_{\text{des}}$ denotes the desired formation distance between them.We avoided incorporating communication states with other agents into the network by representing formation features as reference formation velocities through a consensus protocol [26]. Compared to [21], we reduced the state inputs to the policy network, thereby decreasing the computational resource consumption. And instead of directly feeding the state information of other agents to the policy network, we transform the formation features into a form that is easier for the policy to learn.

We consider the obstacle avoidance problem in scenarios with dynamic obstacles. For dynamic obstacles, we assume that the agent can obtain relatively accurate position information $\boldsymbol{p_o^i}$ and estimate the velocity $\boldsymbol{v_o^i}$ of the obstacle through sensors. This forms part of the obstacle observation information $\boldsymbol{o}_{\text{obstacle}}^{\boldsymbol{i}} = [p_{ox}^i, p_{oy}^i, v_{ox}^i, v_{oy}^i]$. Fig. 3 illustrates the coordinate transformation relationship in the state space, where g_x, g_y, and the current observation $\boldsymbol{o}_{\text{obstacle}}^{\boldsymbol{i}}$ of each obstacle are all

information transformed from the current coordinates $\boldsymbol{p_i}$ of agent i.

B. Reward Function Design

In the process of policy learning, the appropriate reward function has a great impact on the learning effect, so we focus on the design of the reward function, aiming of maintaining the formation as much as possible while avoiding dynamic obstacles in the environment to successfully reach the target point. The reward function is expressed as follows:

$$r = \alpha_1 r_{\text{avoid}} + \alpha_2 r_{\text{nav}} + \alpha_3 r_{\text{formation}} \tag{6}$$

where r is the total reward for one time step, and r_{avoid}, r_{nav}, and $r_{\text{formation}}$ represent the obstacle avoidance reward, navigation reward, and formation maintenance reward, respectively. The terms α_i are the corresponding discount coefficients.

The reward for avoiding collisions between agents and dynamic obstacles is shown as follows:

$$r_{\text{avoid}} = \begin{cases} -\sum_{i\in\mathcal{N}, j\in\mathcal{N}\cup\mathcal{M}} r_{\text{danger}}, & \text{if } \delta_{ij} \le \delta \\ -\sum_{i\in\mathcal{N}, j\in\mathcal{N}\cup\mathcal{M}} r_{\text{collision}}, & \text{if collision.} \end{cases} \tag{7}$$

This includes the danger distance penalty r_{danger} and the collision penalty $r_{\text{collision}}$ between agents and obstacles. δ_{ij} represents the actual distance between agent i and obstacle j, while δ is the predefined danger distance. The r_{danger} refers to the idea in [24] and can be expressed as

$$r_{\text{danger}} = \frac{b}{e^{\frac{\delta_{ij}}{c}} - e^{\frac{\delta}{c}}} \tag{8}$$

where b and c are constants, both of which are set to 0.5, δ needs to consider the radius of the robot and the obstacle, and set the minimum distance between the two boundaries, we set $\delta = 0.5$ during training. Considering that this value will become very large when the obstacle is about to collide with the agent, we limit its upper limit to half of $r_{\text{collision}}$.

The navigation reward is given by (9), which includes the process reward r_{action} for the agent's actions and a one-time reward r_{goal} when all agents reach the goal

$$r_{\text{nav}} = \begin{cases} r_{\text{action}}, & \text{if } \|p - p_{\text{goal}}\| > d \\ r_{\text{goal}}, & \text{if } \|p - p_{\text{goal}}\| \le d \end{cases} \tag{9}$$

where p and p_{goal} indicates the current position and the target position, d is the allowable target difference, which is set to half of the robot radius. This can reduce the shaking of the robot near the end point during training. r_{action} is a reward designed to encourage the agent to move closer to the goal, thereby preventing sparse rewards for reaching the goal during training, as given as follows:

$$r_{\text{action}} = \sum_{i\in\mathcal{N}} (D_i^t - D_i^{t-1}) \tag{10}$$

where D_i^t represents the distance between the current agent and the goal at time t.

For the reward based on formation characteristics, we refer to the ideas in [27]. The goal is to enable agents to learn partial eigenvalue invariance and scale invariance through rewards. First, we normalize the Laplacian matrix $\mathbf{L}$ as follows:

$$\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{11}$$

Next, we take the Frobenius norm of the normalized Laplacian matrix to obtain the formation's characteristic value

$$f = \|\hat{\mathbf{L}} - \hat{\mathbf{L}}_{\text{des}}\|_F^2 = \text{tr}\{(\hat{\mathbf{L}} - \hat{\mathbf{L}}_{\text{des}})^T(\hat{\mathbf{L}} - \hat{\mathbf{L}}_{\text{des}})\} \tag{12}$$

where $\hat{\mathbf{L}}_{\text{des}}$ represents the desired Laplacian matrix for the ideal formation. Consequently, we derive the formation reward $r_{\text{formation}}$ as

$$r_{\text{formation}} = -f. \tag{13}$$

In the procedure, we set $r_{\text{collision}} = 5$ and $r_{\text{goal}} = 5$. The discount coefficients were set to $\alpha_1 = 50$, $\alpha_2 = 80$, and $\alpha_3 = 30$, respectively.

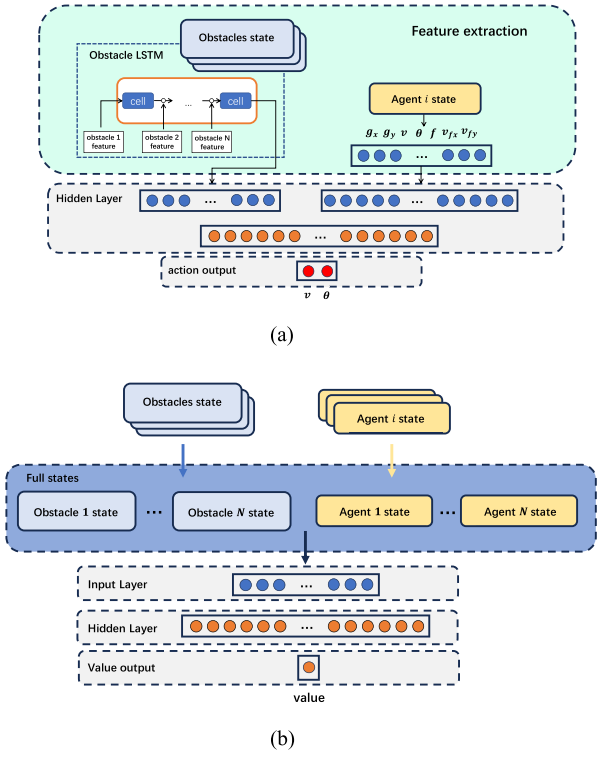

Fig. 4. Two network structures. (a) Policy network framework. (b) Value network framework.

C. Network Structure

We designed a specialized network architecture tailored to our problem. In complex environments, the number and movement of dynamic obstacles are typically uncertain. A critical challenge during network training is managing the network state input and feature extraction when the number of obstacles varies. To address this issue, we adopted the prediction method for multiple moving obstacles as described in [17]. Our RL agent consists of two primary components: 1) the policy network; and 2) the value network, as shown in Fig. 4. The policy network selects actions based on the current state. To achieve this, we extract

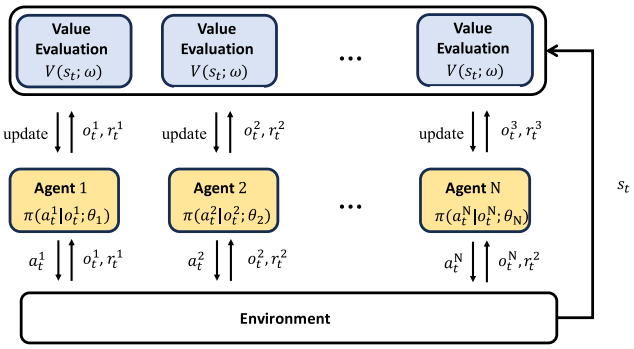

Fig. 5. Centralized training distributed execution framework.

features of obstacle clusters and integrate them into the policy network. Obstacles are sorted by distance, from far to near, to establish spatial relationships, ensuring closer obstacles have a greater influence on the final hidden state. Obstacle information is transformed into the robot's current coordinate system, sorted by distance, and sequentially input into an LSTM for feature extraction. These features are then combined with the agent's own features, processed through a fully connected layer. The final action is determined using two fully connected layers in the hidden layer.

The value network estimates the value of the current state. The observed full state of each agent is concatenated and passed through an input layer and a hidden layer to compute the state value. To accommodate varying numbers of obstacles, we set an upper limit for the obstacle count and replace nonexistent obstacles with zero states.

D. Training

We employ the policy gradient method [28] for optimization, directly obtaining the control policy for each agent. Specifically, we utilize the MAPPO algorithm [23], an extension of the proximal policy optimization (PPO) algorithm [29] for multiagent systems, sharing a similar strategy update concept.

Our training approach follows the centralized training and distributed execution paradigm [30], as depicted in Fig. 5. During training, all agents share global information, allowing for more stable learning. Each agent selects actions based on its current observation and policy parameters θ_i. After executing actions at time t, agents receive rewards r_t, their own observations O_t^i, and the global state observation s_t.

The RL agent interacts with the environment in discrete time steps. At each step, the agent observes the current state, which includes the positions and velocities of dynamic obstacles transformed into the robot's coordinate system. These observations guide the agent in making decisions to select appropriate operators that navigate the environment effectively while avoiding collisions. The observations utilized within the RL framework encompass both local and global information, the policy network use the local information and the value network use the global information.The information contained in each is shown in the network structure diagram.

For updating the value network, we use the Temporal Difference (TD) Error method with the loss function defined as

$$\mathbf{loss}(\boldsymbol{\omega}) = \hat{\mathbb{E}}(\mathbf{MSE}(V(s_{t-1};\boldsymbol{\omega}), \hat{y}_t)) \tag{14}$$

where MSE denotes the function for calculating the mean squared error, $\hat{y}_t = r_t + \gamma V(s_t;\boldsymbol{\omega})$, and the TD error δ_t can be expressed as

$$\delta_t = r_t + \gamma V(s_{t-1};\boldsymbol{\omega}) - V(s_t;\boldsymbol{\omega}). \tag{15}$$

For the policy network parameters $\boldsymbol{\theta^i}$ of each agent, our objective is to maximize the return by updating the policy network $\boldsymbol{\theta^i}$. According to the policy gradient theorem, we obtain

$$\begin{aligned}&\nabla_{\boldsymbol{\theta_i}} J(\boldsymbol{\theta^1}, \ldots, \boldsymbol{\theta^N}) \\ &= \mathbb{E}_{S,A}\left[A_t \cdot \nabla_{\boldsymbol{\theta_i}} \ln \pi(A \mid O; \boldsymbol{\theta^1}, \ldots, \boldsymbol{\theta^N})\right].\end{aligned} \tag{16}$$

A_t is the advantage function estimated by Monte Carlo approximation, which can be written as (17). The advantage function represents the relative value benefit of taking a certain action compared to the current policy action

$$\begin{aligned}A_t &= Q^\pi(s,a) - V^\pi(s) \\ &\approx r + \gamma V(s_t;\boldsymbol{\omega}) - V(s_{t-1};\boldsymbol{\omega}) \\ &= \delta_t.\end{aligned} \tag{17}$$

To further stabilize the estimation of the policy gradient and enhance sample efficiency, we replace the advantage function in (17) with Generalized Advantage Estimation (GAE) [31] as follows:

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \cdots\right) \\ &= (1-\lambda)\left(\delta_t + \lambda(\delta_t + \gamma\delta_{t+1}) + \cdots\right) \\ &= \sum_{l=0}^{N} (\gamma\lambda)^l \delta_{t+l}\end{aligned} \tag{18}$$

where $\hat{A}_t^{(i)}$ for $i = 1, 2, 3, \cdots$ are the advantage functions obtained at each time step according to Equation (17), λ is a newly introduced discount factor used to integrate the advantage estimates of multiple time steps to balance low bias (typically corresponding to estimates from shorter time steps) and low variance (typically corresponding to estimates from longer time steps).

We refer to the improved objective function of the PPO algorithm for task optimization. The loss function can be defined as (19) instead of (3)

$$\begin{aligned}\mathbf{loss}(\boldsymbol{\theta}^i) = \hat{\mathbb{E}}\Big[&\min\Big(r_t(\boldsymbol{\theta}^i)\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}, \\ &\mathrm{clip}\left(r_t(\boldsymbol{\theta}^i), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}\Big)\Big]\end{aligned} \tag{19}$$

wherein, $r_t(\boldsymbol{\theta^i}) = \frac{\pi(a_t, o_t; \boldsymbol{\theta^i})}{\pi(a_t, o_t; \boldsymbol{\theta^i_{\mathrm{old}}})}$ denotes the probability ratio between the new policy and the old policy.The term ϵ is a clipping factor used to limit large discrepancies between the new and old policies, thereby preventing excessively large strategy updates. The essence of PPO lies in constraining the extent of policy updates to avoid substantial policy shifts, which in turn enhances

Algorithm 1: MAPPO Algorithm.

Initialize policy parameters $\boldsymbol{\theta^i}$ for each agent i, old policy parameters $\boldsymbol{\theta^i_{\text{old}}}$ for each agent i, centralized value function parameters $\boldsymbol{\omega}$
Initialize environment initial state s_0
for iteration $= 1, 2, \ldots, N$ **do**
  for $t = 0, 1, \ldots, T-1$ **do**
    for each agent i **do**
      Sample action $a_t^i \sim \pi_i(a, o_t^i; \boldsymbol{\theta^i_{\text{old}}})$
    end for
    Execute joint action $\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^n)$ in the environment
    Observe joint independent observations of each agent $\mathbf{o}_t = (o_t^1, o_t^2, \ldots, o_t^n)$
    Observe shared reward r_t and new state s_t
    Store transition $(s_t, \mathbf{o}_t, \mathbf{a}_t, r_t, s_t)$
  end for
  Compute advantage estimates $_t$ for each agent using the value function $V(s_t; \boldsymbol{\omega})$
  for each agent i **do**
    Update $\boldsymbol{\theta^i}$ by $\mathbf{loss}(\boldsymbol{\theta^i})$
    Update $\boldsymbol{\theta^i_{\text{old}}} \leftarrow \boldsymbol{\theta^i}$
  end for
  Update $\boldsymbol{\omega}$ by $\mathbf{loss}(\boldsymbol{\omega})$
end for
return $\boldsymbol{\theta}, \boldsymbol{\omega}$

the stability of training.The whole pseudocode of MAPPO is presented in Algorithm 1

IV. Experimental Results

We trained our method in a custom simulation environment. To demonstrate its effectiveness, we compared it with several baseline methods, such as APF-F [24] and ORCA-F [20] in the simulation. These comparisons validate the effectiveness of our proposed method. In addition, we also conducted hardware experiments to verify the algorithm on a physical platform.

A. Training and Simulation

The relevant hyperparameters used during the training process are provided in Table I. l_r^p and l_r^v denote the learning rates for the policy network and the value network, respectively. The maximum length of each episode is 3000 steps, with each step lasting 0.1 s. To improve sampling efficiency, we ran 300 parallel threads of the same environment to gather data.

All fully connected layers in our proposed network structure contain 128 neurons. The policy network incorporates an LSTM with a hidden state size of 128. In the simulation, we set the radius of dynamic obstacles and agents to 0.5 m, and the training field size 20 m $\times$ 20 m. The random speed of dynamic obstacles is between 0–1 m/s.

We adopt a curriculum learning approach [32] to gradually increase the complexity of the environment. This allows the strategy to better learn formation and obstacle avoidance tactics,

TABLE I
Reference Parameters

Parameter	Value
Learning Rate l_r^p	0.0005
Learning Rate l_r^v	0.0005
Max Steps per Episode	600
PPO Epochs	15
Threads number	300
Clip Range ϵ	0.2
Discount Factor δ	0.99
Discount Factor λ	0.95

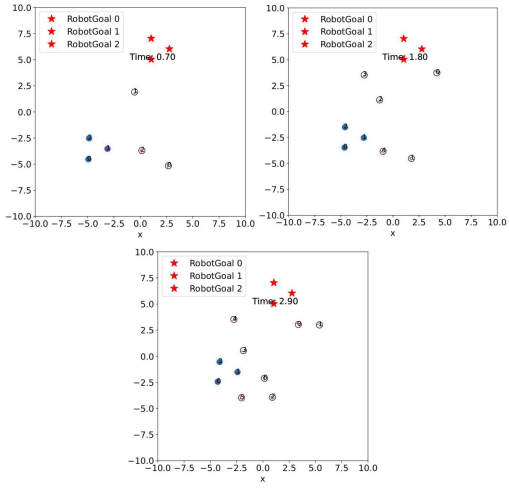

Fig. 6. Increasing the environmental complexity through curriculum learning.

preventing it from getting stuck in local optima due to an overly complex initial environment.During training, we gradually increased the number of dynamic obstacles in the simulation environment. The obstacle avoidance difficulty increases from sparse to dense, as shown in Fig 6. We used a formation of 3 agents. The reward variation over episodes is shown in Fig. 8. We conducted the training on a workstation equipped with an Intel i9-11900 K processor and an NVIDIA A4000 GPU. The training process took approximately 3.5 h to complete around 5000 iterations.

We compared our proposed RL-based formation and obstacle avoidance method with two baseline methods: 1) ORCA-F; and 2) APF-F. The difficulty of dynamic obstacles was categorized into three levels: sparse, medium, and dense. Table II presents the comparisons of success rates, total time consumed, and formation error for each method over 100 experiments. The results indicate that our method consistently outperforms the other methods in terms of success rate and total time consumed.

In sparse obstacle environments, our method achieved a 100% success rate, significantly higher than ORCA-F's 92% and APF-F's 77%. In addition, our method required the least amount of time, taking only 12.1 s compared to ORCA-F's 18.4 s and APF-F's 13.05 s. In medium-density environments, our method

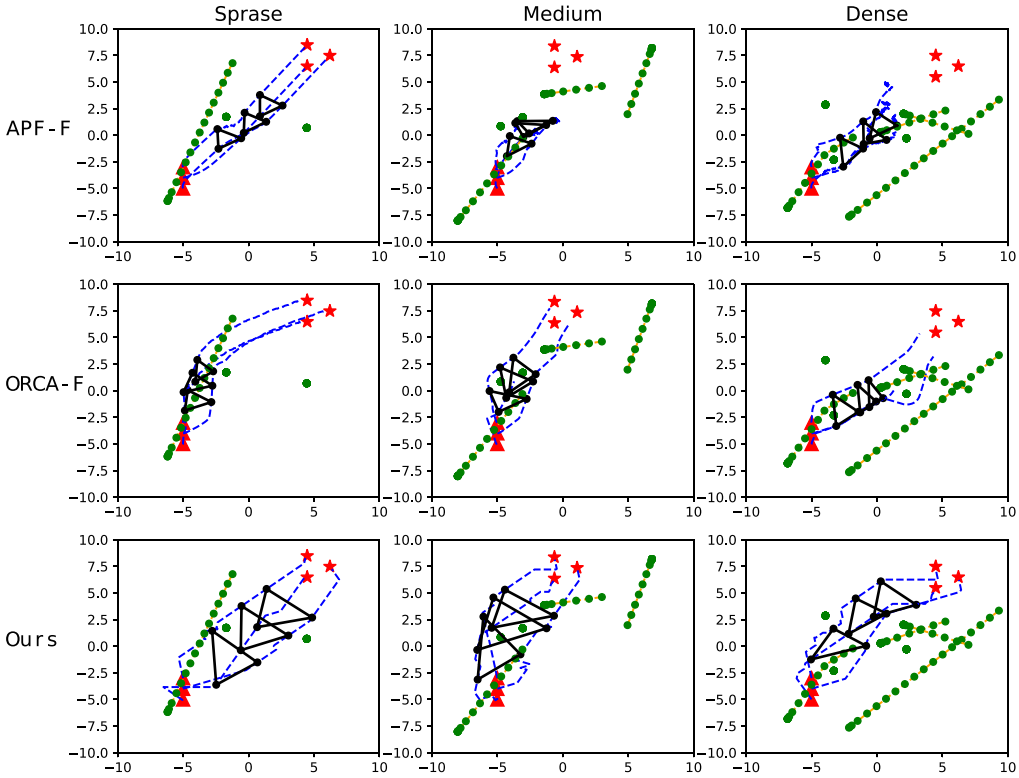

Fig. 7. Formation trajectories using different methods in various scenarios, including formation maintenance at key frames and the movement trajectories of dynamic obstacles.

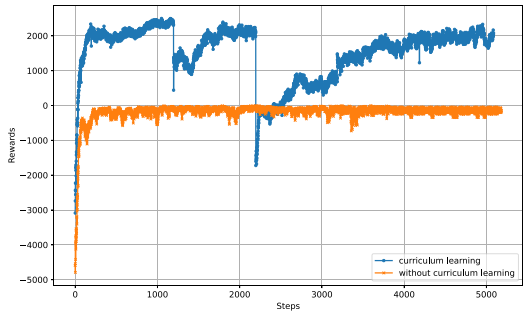

Fig. 8. Curve of the reward function.

TABLE II
COMPARISON OF METHODS ACROSS DIFFERENT SCENARIOS

Scenario	Method	Success rate (%)	Total time (s)
Sparse	ORCA-F [20]	92	18.4
	APF-F[24]	77	13.1
	Ours	**100**	**12.1**
Medium	ORCA-F [20]	84	19.1
	APF-F[24]	61	18.8
	Ours	**98**	**13.5**
Dense	ORCA-F [20]	79	19.2
	APF-F[24]	53	–
	Ours	**97**	**14.1**

The bold values highlight that the corresponding method achieves the highest success rate in obstacle avoidance or the shortest runtime in the comparisons.

maintained a high success rate of 98%, while ORCA-F and APF-F had success rates of 84% and 61%, respectively. The time efficiency of our method was also superior, requiring only 13.5 s. Even in dense environments, where the challenge is greatest, our method achieved a 97% success rate, significantly higher than ORCA-F's 79% and APF-F's 53%. The total time consumed by our method was also the lowest at 14.1 s.

To further analyze the performance, we sampled the trajectories of agents and obstacles under sparse, medium, and dense scenarios using the three different methods. These experiments were conducted using the same random seed and formation keyframes to ensure consistency. The detailed trajectory results are shown in Fig. 7. In sparse environments, all three methods maintained good formation, but our method exhibited higher efficiency and shorter completion times. In more complex environments, the performance of the other methods deteriorated significantly. The APF-F method exhibited substantial collisions and struggled to reach the destination. The ORCA-F method, constrained by the dual objectives of obstacle avoidance and formation maintenance, often became trapped in local minima, failing to reach the destination while maintaining formation. In contrast, our proposed method demonstrated a high success rate for obstacle avoidance and maintained effective formation in both medium and dense scenarios.

In addition, we conducted comparative experiments with standard MAPPO, using 100 random obstacle scenarios (Table III). The key difference lies in using the nearest obstacle to the agent as the state input, instead of the feature-extracted state. In sparse and medium environments, both methods achieve similar success rates and times, so comparisons are omitted. However, in dense obstacle environments, the LSTM-based model performs better. Fig. 10 shows a "pincer movement" scenario, where the non-LSTM model fails due to a lack of anticipation, leading to

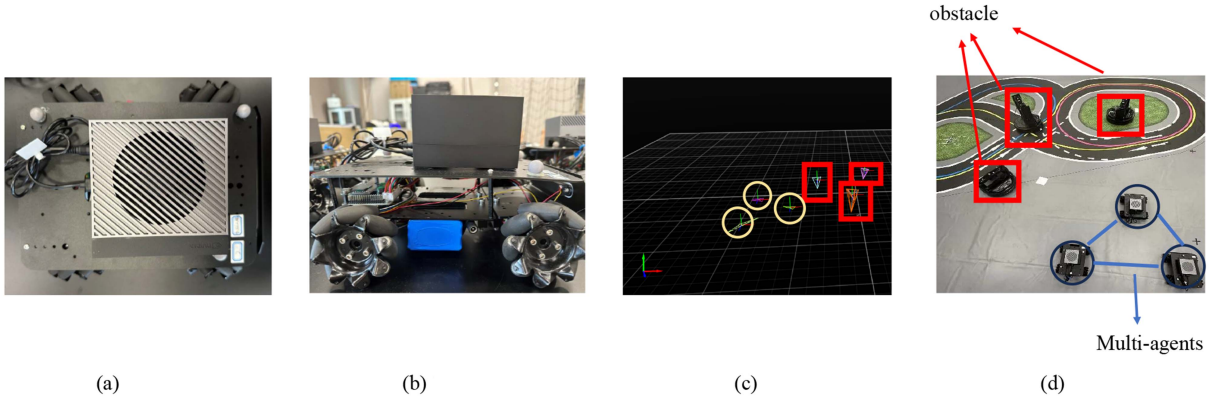

Fig. 9. Hardware platform. (a) Top view of the agent. (b) Side view of the agent. (c) Optitrack view of the formation. (d) Formation scenario of 3 agent.

TABLE III
COMPARISON BETWEEN THE ORIGINAL MAPPO AND THE METHOD WITH LSTM ADDED

Scenario	Method	Success rate (%)	Total time (s)
Dense	MAPPO	93	14.5
	Ours	**97**	**14.1**

The bold values highlight that the corresponding method achieves the highest success rate in obstacle avoidance or the shortest runtime in the comparisons.

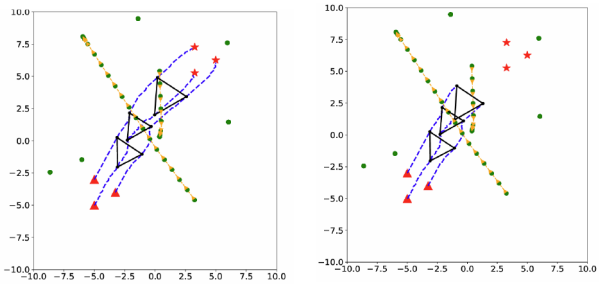

Fig. 10. Compare with MAPPO with LSTM and MAPPO method. (a) Our method. (b) MAPPO without LSTM.

collisions (also unsolvable by ORCA [13]). The LSTM model, however, predicts and avoids collisions. In dense environments, the success rate of the non-LSTM model is 93%, lower than the LSTM model.

In summary, our RL approach, enhanced by LSTM, significantly boosts the robot's planning and obstacle avoidance abilities in dynamic obstacles environments. Extensive simulations and real-world tests demonstrate that our method achieves the highest success rate in a short time, outperforming traditional tracking methods and RL approaches without feature extraction.

B. Real World Deployment

We conducted hardware deployment on the omnidirectional wheel robot, as shown in Fig. 9. We first introduce the low-level controller of the hardware platform used in this work. The robot adopts a Mecanum wheel-based chassis, as shown in the Fig. 11. Here, V_x represents the linear velocity of the robot along the x-axis in its own coordinate system, V_y represents the linear velocity along the y-axis, and ω is the angular velocity around the robot's center in its own coordinate system. V_A, V_B, V_C, V_D denote the velocities of the four wheels, respectively.

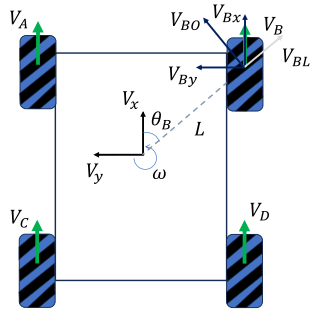

Fig. 11. Schematic diagram of Mecanum wheel car.

Focusing on the motion of a single wheel, we take wheel B as an example. The linear velocity of wheel B, denoted as V_B, is generated by the motor's rotation and transmitted to the hub. Through the roller's contact with the ground, a relative sliding motion generates a component velocity V_{BL}. The resulting velocity (V_{Bx},V_{By},V_{BO}) is the vector sum of V_B and V_{BL}. We can derive the following relations:

$$V_{Bx} = V_B + V_{BL} \cdot \sin\beta \tag{20}$$

$$V_{By} = -V_{BL} \cdot \cos\beta \tag{21}$$

where β is the angle between the roller axis and the wheel's main rotational axis. For the experimental robot used, $\beta = 45°$.

Assuming the robot chassis and the four Mecanum wheels are part of a rigid body, the relationship between the wheel velocity and the overall robot velocity can be derived. By decomposing the velocity, we have

$$V_{Bx} = V_x + \omega \cdot L \cdot \sin\theta_B \tag{22}$$

$$V_{By} = V_y + \omega \cdot L \cdot \cos\theta_B \tag{23}$$

where L denotes the distance from the robot's center of mass to the center of wheel B, and θ_B is the angle of wheel B relative to the robot's coordinate frame.

By combining and simplifying (20), (21) and (22), (23) we obtain

$$\begin{aligned} V_B &= V_x + V_y + \omega \cdot L \cdot (\sin\theta_B + \cos\theta_B) \\ &= V_x + V_y + \omega \cdot (H + W) \end{aligned} \tag{24}$$

where H and W represent half the length and half the width of the robot's chassis, respectively.

Similarly, using the reference velocity of the robot, we can derive the desired wheel velocities for all four wheels

$$V_A = V_x - V_y - \omega \cdot (H + W) \tag{25}$$

$$V_B = V_x + V_y + \omega \cdot (H + W) \tag{26}$$

$$V_C = V_x + V_y - \omega \cdot (H + W) \tag{27}$$

$$V_D = V_x - V_y + \omega \cdot (H + W)\,. \tag{28}$$

To achieve precise velocity control, the desired wheel velocities are compared with the current velocities measured using motor encoders. Given the reference velocity signals of the vehicle, V_x, V_y, and ω, the rotational speed of each wheel can be obtained through (25)–(28). During a single sampling period, the relationship between the wheel velocity V and the encoder increment N is expressed as $N = K_m V$, where K_m is determined by factors such as the wheel radius, motor reduction ratio, and encoder resolution. We design the following PID controller to regulate the motor speed:

$$u_i = K_p \left[e_i(t) + \frac{1}{T_I} \int e_i(t)\, dt + T_D \frac{de_i(t)}{dt} \right] \tag{29}$$

where i denotes the motor index (A, B, C, D), $e_i = N_{id} - N_{ic}$ represents the error between the desired encoder increment N_{id} and the current encoder increment N_{ic}.

Remark 1: During reference signal tracking, external disturbances such as dead-zone effects, wheel slippage, and other dynamic factors may affect motor control [33]. However, this work focuses on high-level velocity decision and does not consider dynamic disturbances in motor signal tracking.

The RL strategy was deployed on the upper computer, which communicated with the robot's chassis via ROS to send velocity commands. Three additional mobile robots were used as moving obstacles for demonstration purposes. The experimental setup consisted of a 6 m × 6 m area, where each robot utilized the Optitrack motion capture system to obtain both its own state information and the state information of the environment. The robots communicated their state information with other robots in the cluster via ROS. As shown in Fig. 12, each robot acquires information about obstacles and cluster formation through communication, makes decisions using the policy network trained in simulation, and sends velocity commands (V_x, V_y, ω) to the chassis control node for execution. The radius of both the moving obstacles and the robots was set to 0.15 m. The velocity range of the moving obstacles was consistent with the simulation, between 0 and 1 m/s, and the decision frequency of the robots was 10 Hz. We use the Jetson Orin Xavier as the host computer for policy deployment. Testing results show that the average action output time of the policy is 5 ms, which is far below the required decision-making time and fully meets the real-time

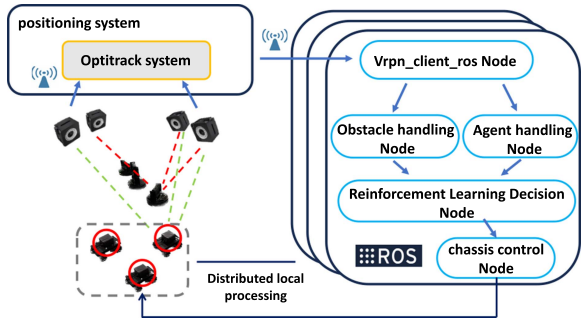

Fig. 12. Overall solution framework deployed through ROS.

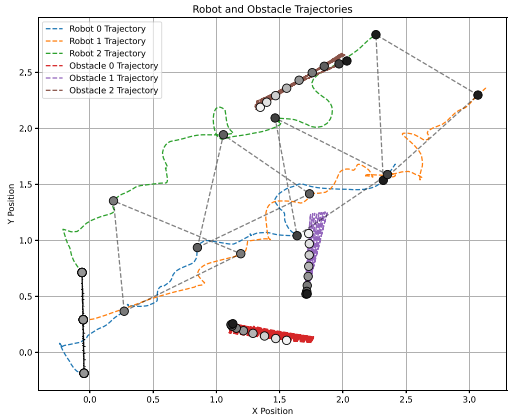

Fig. 13. Formation obstacle avoidance trajectory in real environment.

output requirements of the policy. By replacing the positioning and obstacle detection solutions with a more robust vision-based method, the cluster formation scenario can be easily migrated to an outdoor environment.

The trajectories tested in the real environment are shown in Fig. 13. The agents successfully avoided obstacles, maintained formation, and reached the destination. The results demonstrate that our proposed method exhibits excellent performance in real-world validation.

V. Conclusion

This article introduced a multiagent RL-based method for formation control and dynamic obstacle avoidance. By using deep RL strategies, each agent's actions were determined to achieve formation control and dynamic obstacle avoidance in complex navigation environments. We optimized our policy using the MAPPO algorithm and compared our method with other conventional formation and dynamic obstacle avoidance techniques. We also conducted formation experiments in a real-world environment using three Mecanum-wheeled robots, which confirmed the practical feasibility of our algorithm.The success rate of our method in achieving formation and obstacle avoidance to the endpoint has significantly improved compared to other methods. The experimental results clearly demonstrate the superior performance of our approach.

In the future, we will further study the obstacle avoidance method for multiagent formations. We found that our method

collides or fails to reach the end point at a certain moment because of the difference in dynamic and physical characteristics between simulation and real objects, as well as external interference [33], and uncertainties [34], and we will further improve the method, refer to [8] and [9], to enhance the team's mission completion capability in the presence of external interference, failures, and uncertain situations.

REFERENCES

[1] B. Zhou, H. Xu, and S. Shen, "RACER: Rapid collaborative exploration with a decentralized multi-UAV system," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1816–1835, Jun. 2023.

[2] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.

[3] Y. Lian, Q. Yang, W. Xie, and L. Zhang, "Cyber-physical system-based heuristic planning and scheduling method for multiple automatic guided vehicles in logistics systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7882–7893, Nov. 2021.

[4] L. Marconi et al., "The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *Proc. IEEE Int. Symp. Safety, Security, Rescue Robot.*, 2012, pp. 1–4.

[5] Y. Xia, X. Na, Z. Sun, and J. Chen, "Formation control and collision avoidance for multi-agent systems based on position estimation," *ISA Trans.*, vol. 61, pp. 287–296, 2016.

[6] R. S. Sharma, A. Mondal, and L. Behera, "Tracking control of mobile robots in formation in the presence of disturbances," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 110–123, Jan. 2021.

[7] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *J. Franklin Inst.*, vol. 354, no. 4, pp. 2068–2085, 2017.

[8] N. Le-Dung, P. Huynh-Lam, N. Hoang-Giap, and N. Tan-Luy, "Event-triggered distributed robust optimal control of nonholonomic mobile agents with obstacle avoidance formation, input constraints and external disturbances," *J. Franklin Inst.*, vol. 360, no. 8, pp. 5564–5587, 2023.

[9] J. Liu and J. Dong, "Distributed fault-tolerant control for nonlinear heterogeneous multiagent system with fault indication under safety constraint," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 54, no. 6, pp. 3843–3853, Jun. 2024.

[10] B. Ranjbar-Sahraei, F. Shabaninia, A. Nemati, and S.-D. Stan, "A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3124–3134, Aug. 2012.

[11] S. He, M. Wang, S.-L. Dai, and F. Luo, "Leader–follower formation control of USVs with prescribed performance and collision avoidance," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 572–581, Jan. 2019.

[12] X. Min, S. Baldi, and W. Yu, "Funnel-based asymptotic control of leader–follower nonholonomic robots subject to formation constraints," *IEEE Trans. Control Netw. Syst.*, vol. 10, no. 3, pp. 1313–1325, Sep. 2023.

[13] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3475–3482.

[14] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1928–1935.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.

[16] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.

[17] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10357–10377, 2021.

[18] Y. Han et al., "Deep reinforcement learning for robot collision avoidance with self-state-attention and sensor fusion," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6886–6893, Jul. 2022.

[19] Y. Zhao, Y. Ma, and S. Hu, "USV formation and path-following control via deep reinforcement learning with random braking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5468–5478, Dec. 2021.

[20] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2358–2372, Jun. 2021.

[21] Y. Yan et al., "Relative distributed formation and obstacle avoidance with multi-agent reinforcement learning," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 1661–1667.

[22] L. Chen et al., "Transformer-based imitative reinforcement learning for multirobot path planning," *IEEE Trans. Ind. Informat.*, vol. 19, no. 10, pp. 10233–10243, Oct. 2023.

[23] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. A. Mohamed, D. Agarwal, K. Belgrave Cho, and A. Oh, Eds., vol. 35, Red Hook, NY, USA: Curran Associates, Inc., 2022, pp. 24611–24624.

[24] X. Zhu, Y. Liang, and M. Yan, "A flexible collision avoidance strategy for the formation of multiple unmanned aerial vehicles," *IEEE Access*, vol. 7, pp. 140743–140754, 2019.

[25] D. Guo, L. Tang, X. Zhang, and Y.-C. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13124–13138, Nov. 2020.

[26] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[27] L. Quan et al., "Robust and efficient trajectory planning for formation flight in dense environments," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4785–4804, Dec. 2023.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

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

[30] G. Chen, "A new framework for multi-agent reinforcement learning–centralized training and exploration with decentralized execution via policy distillation," 2019, *arXiv:1910.09152*.

[31] S. L. M. J. P. A. John Schulman and P. Moritz, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

[32] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, no. 181, pp. 1–50, 2020.

[33] L. N. Tan, "Omnidirectional-vision-based distributed optimal tracking control for mobile multirobot systems with kinematic and dynamic disturbance rejection," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5693–5703, Jul. 2018.

[34] S. Gu, C. Qian, and N. Zhang, "Finite-time integral control for a class of nonlinear planar systems with non-vanishing uncertainties," *Automatica*, vol. 136, 2022, Art. no. 110016.

Zike Yuan received the B.S. degree in automation from the Department of Automation, Zhejiang University of Technology, Zhejiang, China, in 2022. He is currently working toward the M.S. degree in mechanical engineering with Shenzhen University, Shenzhen, China.

His research interests include the application of reinforcement learning, motion planning, and data-driven control in robotics.

Chenhao Yao is currently working toward the undergraduate degree in automation with the School of Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, Shenzhen, China.

His research interests include reinforcement learning in robotics applications, multiagent system, and data-driven control.

Xiaoxu Liu received the B.S. degree in information and computing sciences and the M.S. degree in operations research and cybernetics from Northeastern University, Shenyang, China, in 2012 and 2014, respectively, and the Ph.D. degree in electrical engineering from the University of Northumbria at Newcastle, Newcastle upon Tyne, U.K., in 2018.

She is currently an Associate Professor with the Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, Shenzhen, China. Her research interests include robust fault diagnosis, fault-tolerant control, nonlinear systems, stochastic systems, fuzzy modeling, distributed systems, and data-driven methods.

Dr. Liu is an Associate Editor for IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Zhiwei Gao (Fellow, IEEE) received the B.Eng. degree in industrial automation and the M.Eng. and Ph.D. degrees in systems engineering from Tianjin University, Tianjin, China, in 1987, 1993, and 1996, respectively.

His research interests include diagnosis and control, digital twins, artificial intelligence, machine learning, data-driven approaches, biometrics, wind energy systems, wave converter, electric vehicles, and offshore energy.

Dr. Gao was the recipient of the Alexander von Humboldt Fellowship in 2004. He is the Co-Editor-in-Chief for IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, Senior Editor for *IEEE Access*, Area Editor for *Journal of Ambient Intelligence and Humanized Computing*, and Associate Editor for IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS. He was the Associate Editor for IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, and IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. For contributions to real-time diagnosis and control for wind turbines, he was elevated to IEEE Fellow in 2023.

Wenwei Zhang received the B.S. and M.S. degrees in precision instruments and the Ph.D. degree in measurement technology and instrumentation from Tianjin University, Tianjin, China, in 1989, 1994, and 1997, respectively, and the MBA degree from the University of Glasgow, Glasgow, U.K., in 2000.

He is currently the Director of the Advanced MEMS Sensor Chip Engineering Technology Research Center, Guangdong Province Universities, and a Distinguished Professor and the Dean of the Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, Shenzhen, China. Previously, he served as R&D Director with Honeywell U.K., Tyco U.K., and Micronas Germany, accumulating over 15 years of experience in the R&D and management of magnetic sensor SOC chips. His expertise lies in core technologies for automotive magnetic sensor chips, with extensive practical experience in device physics, device materials, and the design and optimization of mixed-signal integrated circuits.

Dr. Zhang is the Chair of the Professor Committee and Vice-Chair of the Academic Committee at Shenzhen Technology University.