Multi-Robot Flocking Control Based on Deep Reinforcement Learning

PENGMING ZHU, WEI DAI, WEIJIA YAO, JUNCHONG MA, ZHIWEN ZENG, AND HUIMIN LU (Member, IEEE)
Robotics Research Center, College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
Corresponding author: Zhiwen Zeng (zengzhiwen@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1913202 and Grant U1813205, and in part by the Hunan Provincial Innovation Foundation for Postgraduate under Grant CX2018B010.

ABSTRACT In this paper, we apply deep reinforcement learning (DRL) to solve the flocking control problem of multi-robot systems in complex environments with dynamic obstacles. Starting from the traditional flocking model, we propose a DRL framework for implementing multi-robot flocking control, eliminating the tedious work of modeling and control designing. We adopt the multi-agent deep deterministic policy gradient (MADDPG) algorithm, which additionally uses the information of multiple robots in the learning process to better predict the actions that robots will take. To address the problems such as low learning efficiency and slow convergence speed of the MADDPG algorithm, this paper studies a prioritized experience replay (PER) mechanism and proposes the Prioritized Experience Replay-MADDPG (PER-MADDPG) algorithm. Based on the temporal difference (TD) error, a priority evaluation function is designed to determine which experiences are sampled preferentially from the replay buffer. In the end, the simulation results verify the effectiveness of the proposed algorithm. It has a faster convergence speed and enables the robot group to complete the flocking task in the environment with obstacles.

INDEX TERMS Multi-robot, deep reinforcement learning, flocking control, PER-MADDPG.

I. INTRODUCTION Multi-robot systems play an important role in a wide range of applications, such as target tracking and navigation, collaborative patrol, search, rescue, forest inspection, and agricultural spraying [3]–[7]. When multiple robots work together in a complex environment, it is crucial to ensure the safety of each robot. Inspired by the group behavior of biological colony, such as bird migration and fish gathering, where the entire system is in a coordinated and orderly state to respond to external threats without any organizer, many scholars conduct research on multi-robot flocking control. However, when the working environment of the robot group is relatively complex, the group behavior strategy is required to be real-time and able to avoid various obstacles. Besides, due to the limitation of actual communication capabilities, each robot’s communication range is limited. Therefore, the robot group must consider connectivity during the task to ensure that the robots can communicate with each other.

Regarding the flocking problem, Reynolds et al. [8] have come up with three basic rules, namely separation, aggregation, and consistency of velocity. The three rules are instructive to the establishment of flocking motion models, and most of the subsequent flocking models proposed are based on the three rules. Vicsek [9] studied the consistency of velocities in Reynolds’ rules, then he controlled agents perturbed by random noise such that their dictions of motions converge. Tian et al. [10] proposed an improved Vicsek model with limited field of view, and this model was further extended by Zhang et al. [11] with random line-of-sight directions. Among the extensive studies on flocking control problems, most of them used traditional methods such as those based on LQR [12], PCA [13] or a virtual leader [14], which are not effective in dealing with the external disturbance and the nonlinear time-varying nature of the flocking control problem. This paper uses the deep reinforcement learning (DRL) method to complete the flocking task without requiring accurate modeling and sophisticated control design that are required in traditional methods.

This paper adopts the DRL method, multi-agent deep deterministic policy gradient (MADDPG) [1], which combines neural network and deterministic policy gradient algorithm, to solve the multi-robot flocking control problem in 2D environments. Based on the MADDPG algorithm, we propose an improved version, called PER-MADDPG algorithm, by introducing the prioritized experience
replay (PER) [2] mechanism. Our new algorithm effectively improves the training efficiency and shortens the convergence time. The three main contributions of this paper are as follows: (1) To the best of our knowledge, this is the first work to use MADDPG method to solve the multi-robot flocking problem. (2) Combined with the features of centralized training and decentralized execution of the MADDPG algorithm, we use only one replay buffer to store information of all the robots. (3) We propose the PER-MADDPG algorithm that combines MADDPG and PER. Simulation results show that this new algorithm has significantly improved the training efficiency.

The rest of the paper is structured as follows: Section II overviews the existing studies on reinforcement learning in the field of cooperative flocking control. Section III formulates the multi-agent flocking problem and illustrates the multi-robot reinforcement learning process. Section IV introduces the algorithmic framework for the flocking task proposed in this paper. Section V verifies the algorithm through simulation experiments. Finally, Section VI concludes the paper.

II. RELATED WORK
A. MULTI-ROBOT REINFORCEMENT LEARNING
The problem of single robot reinforcement learning has been extensively studied, and a number of algorithms have been proposed in the literature [15]–[19]. However, only a few methods are available to solve the multi-agent reinforcement learning problem. Dai et al. [20] used DQN to solve the multi-agent task assignment problem and achieved some results. Sukhbaatar et al. [21] designed a neural network called CommNet to enable continuous communication in a collaborative environment. Also for multi-agent communication problems, Foerster et al. [22] used Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL) to enable the end-to-end collaborative communication within multiple robots. Palmer et al. proposed Lenient-DQN [23], which introduced Lenient loss function based on Double DQN [24] to adapt to the cooperation problem of multi-agent reinforcement learning. Foerster et al. [25] proposed COMA, which uses the centralized critic. The centralized critic can obtain global information to guide each agent to further improve each agent’s modeling capabilities for information. However, as there is only one centralized critic, agents are not allowed to have different reward functions. Recently, Rashid [26] proposed the QMIX algorithm, which uses a hybrid network structure and adds global state information to improve the algorithm’s performance during the training process. The MADDPG algorithm, used in this paper, adopts the centralized learning and decentralized execution mechanism and adds the action information of each agent into the training process. Empirically, obtaining the action information of each agent helps to understand the policy of other agents so that the MADDPG algorithm can be well adapted in the cooperative-competitive environment. The MADDPG algorithm has been applied in many aspects, such as multi-agent communication [27], multi-agent target assignment and path planning [28], and multi-agent target encirclement formation control [29].

B. THE APPLICATION OF REINFORCEMENT LEARNING IN FLOCKING
Several studies have been conducted to apply reinforcement learning methods to the flocking control problem. Having considered the model of flocking behavior, Morihiro et al. [30] proposed a multi-agent cooperative flocking control framework based on the Q-learning algorithm and implemented it in a simulation environment. On this basis, Tomimasu et al. [31] studied cooperative flocking control based on reinforcement learning. Adopting the Q-learning algorithm and introducing potential field methods, they further built a simulation model to make the robot learn flocking behavior. Hung et al. [32], [33] studied the flocking control problem of small fixed-wing UAVs under the background of model-free reinforcement learning. Xu et al. [34] proposed a flocking control framework based on the deep reinforcement learning multi-vehicle system (MVS) after considering the conditions with collision avoidance and communication maintenance. Although the multi-agent cooperative flocking control based on reinforcement learning methods has been initially verified on the simulation and physical platforms, most of the existing studies consider discrete action or state space, and do not deal with some problems such as slow convergence speed that may significantly deteriorate the control performance in a complex environment. Therefore, we propose the PER-MADDPG algorithm that combines features of both MADDPG and PER such that the training efficiency and convergence speed have been both noticeably improved. In addition, the introduction of the PER mechanism based on MADDPG enables robots to output actions in the continuous action space.

III. PROBLEM FORMULATION
A. SYSTEM MODELING
Suppose there are $M$ obstacles and $N$ homogeneous omnidirectional robots with mass $m$ in the two-dimensional space. We use $V = \{r_1, \ldots, r_N\}$ to represent all robots and $I_o = \{o_1, \ldots, o_M\}$ to represent all obstacles, where $N, M$ are finite numbers. In the flocking control problem, $p_t' = [p_{1t}', \ldots, p_{Nt}']$ and $v_t' = [v_{1t}', \ldots, v_{Nt}']$ are used to denote the positions and velocities of $N$ robots at time $t$, and $p_{i}' = [p_{i1}', \ldots, p_{iM}']$ is used to denote the positions of $M$ obstacles. The discretized dynamic model for each robot $r_i$ is described as

$$\begin{align*}
    v_{i,t+1} &= v_{i,t} + \frac{F_{i,t}}{m} \Delta t, \\
    p_{i,t+1}' &= p_{i,t}' + v_{i,t}' \Delta t,
\end{align*}$$

where $v_{i,t}$, $p_{i,t}'$ denote the velocity and the position of the $i$-th robot respectively at time instance $t$, and $\Delta t$ denotes the sampling period. We define the control input as $u_{i,t}' = F_{i,t}$, where $F_{i,t} \in \mathbb{R}^2$ is a force vector. In the following, if there is
In this paper, an undirected graph $G = (V, E)$ is used to describe the communication graph of $N$ robots, where $E$ is the edge set, defined as $E = \{(r_i, r_j)|d(r_i, r_j) = \|p_i - p_j\| \leq \rho_c\}$, where $\rho_c$ is the maximum communication distance. In other words, when the distance between two robots is less than the maximum communication distance, there is an undirected edge between them and they can communicate with each other. Therefore, we can define the neighbor set of $r_i$ as $N_i = \{r_j| (r_i, r_j) \in E, j \neq i\}$. The flocking task consists of three parts: reaching the target position, avoiding collision, and maintaining connectivity, as shown in Fig. 1.

1. Reaching the target position: Given a target position $g$, the control target is to minimize the sum of distance between the robot and the target in the flocking: $e = \sum_{i=1}^{N} ||p_i - g||_2$.

2. Avoiding collision: The safe distance between robots and that between the robot and the obstacle are given as $\rho_o$ and $\rho_p$. Then $d(r_i, r_j) \geq \rho_o$ and $d(r_i, o_j) = ||p_i - \tilde{p}_j|| \geq \rho_p$, $\forall r_i, r_j \in V, \forall o_j \in O$ should be satisfied, where $\tilde{p}_j$ represents the position of the $j$-th obstacle.

3. Maintaining connectivity: Given that maximum communication distance between two robots is $\rho_c$. To ensure the robots can communicate with each other, the distance between two robots should not exceed the perception range, i.e., $d(r_i, r_j) \leq \rho_o$, $\forall r_i, r_j \in N_i$.

To better conform to the actual situation of flocking, we assume that the target area is globally perceived, and the positions of the obstacles are locally perceived. We define a distance $\rho_p$ ($\rho_p > \rho_o$), then the robot can locally perceive the obstacle when $d(r_i, o_j) \leq \rho_p$. When a robot in the group senses the obstacle, the other robots within its communication range can also obtain their relative positions to the obstacle, as shown in Fig. 2.

**B. DESCRIPTION OF MULTI-ROBOT FLOCKING CONTROL PROBLEM**

Solving the multi-agent problem through reinforcement learning can avoid modeling the behavior of the agent in advance and control design. The agent only needs to interact with the environment to generate its strategy.

The reinforcement learning is usually described as a Markov decision process (MDP). The Markov process of multi-robot flocking can be represented as a tuple $G = (\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, O)$, where $\mathcal{S}$ is the state space to describe the state of the environment and the state of the robot. The joint actions of all robots can be expressed as $\mathcal{A} = A_1 \times \cdots \times A_N$, where $A_i \subseteq \mathbb{R}^2$ is the two-dimension continuous action space for each robot. In the iteration, the state-action pair of the robot can be expressed by the transition function $\mathcal{T} (s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$. The robot will get a reward $R$ during the iteration according to the reward function $R_t (s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$. The observations of all robots can be expressed as $O = \{O_1, O_2, \cdots, O_N\}$, where $O_i$ represents the observation of the robot $r_i$, and the observation includes its own velocity and the positions of each robot and target. The multi-robot learning process is shown in Fig. 3.

During the flocking task, each robot computes its action in continuous two-dimensional space through learned policy.
based on observation. The action taken by the robot \( r_i \) is defined by the policy \( \pi_{\theta_i}(a_i|s_t) : O_t \times A_i \rightarrow \mathcal{P}(A) \), where \( \mathcal{P}(A) \) is the probability calculated through \( A \), and \( \theta_i \in \mathbb{R}^d \) is a parameter with \( d \) elements. The actions convert the state \( s_t \) to a new state \( s_{t+1} \) according to the transition function \( T \). During the training process, the goal of the robots is to learn the best policy to maximize its own cumulative discounted reward \( G_t \):

\[
G_t = \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \ldots + \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k},
\]

where \( 0 < \gamma < 1 \) represents the discount factor in each step.

IV. THE FLOCKING CONTROL BASED ON PER-MADDPG

A. PRIORITIZED EXPERIENCE REPLAY MADDPG

1) MADDPG

The MADDPG algorithm is an extention of DDPG [19]. Similar to DDPG, MADDPG also uses the actor-critic [16] structure, and both of the actor and the critic have an online policy network and a target policy network. The actor online network calculates the action \( a_i = \pi_{\theta}(o_i) \) to be performed only based on the current state \( o_i \) observed by robot \( r_i \), and the critic online network evaluates the action to improve the performance of the actor online network. The target network regularly copies parameters from the online network.

In the stochastic policy gradient algorithm, if we use \( \pi = \{\pi_1, \ldots, \pi_N\} \) to represent the robots’ policy of \( N \) robots and use \( \theta = \{\theta_1, \ldots, \theta_N\} \) to represent the policy parameters, then we can write the gradient of the expected return for robot \( r_i \), \( J(\theta_i) = \mathbb{E}[R_t] \) as:

\[
\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^\pi, a \sim \pi_{\theta_i}} \left[ \nabla_{\theta_i} \log \pi_{\theta_i}(a_i|o_i) \nabla_{\theta_i} Q^\mu_i(\cdot) \right],
\]

where \( p^\pi \) is the state distribution, \( s = (o_1, \ldots, o_N) \) represents the joint state, \( a = (a_1, \ldots, a_N) \) represents the joint action, and \( Q^\mu_i(s, a) \) is a centralized action-value function, whose inputs are the joint action and joint state of all robots, and its output is the Q value of the robot \( r_i \).

However, MADDPG adopts the deterministic policy gradient. If we consider \( N \) continuous policies \( \mu_i \) with parameters \( \theta_i \) (abbreviated as \( \mu_i \)), the gradient can be written as:

\[
\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{s,a \sim \mathcal{D}} \left[ \nabla_{\theta_i} \mu_i(a_i|o_i) \nabla_{\theta_i} Q^\mu_i(s,a) \bigg|_{a_i = \mu_i(o_i)} \right],
\]

where \( \mu = \{\mu_1, \ldots, \mu_N\} \), \( \mathcal{D} \) represents the experience replay buffer which contains a series of tuples \( (s, s', a, r) \) recording the experiences of all robots, \( s' \) is the new state of the robots after executing the actions and the \( r = (r_1, \ldots, r_N) \) is the reward of all robots.

Every once in a while, experiences will be randomly sampled from \( \mathcal{D} \) to update network parameters. The critic network \( Q^\mu_i \) is updated by the loss function as follows:

\[
\mathcal{L}(\theta_i) = \mathbb{E}_{s,a,r,s'} \left[ (Q^\mu_i(s,a) - y)^2 \right],
\]

where \( y = r_i + \gamma Q^\mu_i(s', a') \bigg|_{a' = \mu_i(o')} \).

The actor network is updated by minimizing the policy gradient of robot \( r_i \) which can be written as:

\[
\nabla_{\theta_i} J(\mu_i) \approx \frac{1}{K} \sum_k \nabla_{\theta_i} \mu_i \left( a_i^k \right) \nabla_{a_i} Q^\mu_i \left( s^k, a' \right) \bigg|_{a' = \mu_i(o_i^k)},
\]

where \( K \) is the minibatch size of samples and \( k \) is the index of samples.

2) PRIORITIZED EXPERIENCE REPLAY MECHANISM

The Experience Replay method overcomes the correlated data problem and non-stationary distribution problem of experience through storage-sampling. Due to the uneven quality of randomly extracted experience led by random sampling, the MADDPG algorithm faces difficulties of low learning efficiency and slow convergence speed.

To solve the problem mentioned above, this paper introduces the PER mechanism. The PER method has been widely used in DQN, DDPG, and other algorithms, and it performs well in the single-agent reinforcement learning problem. However, in multi-agent tasks, as each agent has a separate replay buffer to store its own experience, storing and replaying according to their respective evaluation would disrupt the relevance of the centralized experience training, thereby failing to complete the training.

In view of the characteristics of the centralized training of the MADDPG algorithm, this paper uses a centralized experience buffer, which stores the joint information of the agents \( (s, a, r, s') \). And then the agents preferentially sample experience according to the importance, thereby improving the algorithm’s efficiency.

The main idea of PER is to replay more frequently those experiences that are more important to network updates, so how to define the importance of these experiences is the
critical issue. TD-error is often used in most reinforcement learning algorithms to update the estimate of the critic network $Q^\mu_i(s, a)$. Since the TD-error performs the maximum likelihood estimation, its value can be used as a correction for estimation. It can implicitly reflect the degree that the agent can learn from the experience, thereby making the estimated result more in line with the trend of future data. The bigger the value of the TD-error, the more positive the correlation of the expected action-value. On the contrary, the smaller the TD-error value is, the worse the action taken by the agent in this state would be. Replaying these experiences more frequently helps the agents gradually get the correct result of the wrong behaviors and avoids the wrong behaviors occurring again, thereby improving the overall performance of the algorithm. In this paper, we use the absolute value of the TD-error $|\delta|$ as the basis for ranking. The TD-error of experience $k$ is calculated through the formula below:

$$\delta_k = r + \gamma Q^\mu_i(s', a') - Q^\mu_i(s, a).$$  \hspace{1cm} (8)

The larger TD-error shows that the difference between the evaluation value of the target network and the actual value for this experience is significant. Hence, the sampling frequency needs to be increased to update the value of the target network as well as the evaluation network as soon as possible to achieve the optimal training effect. We define the probability that experience $k$ is sampled as:

$$P(k) = \frac{D_k^\alpha}{\sum_j D_j^\alpha}.$$  \hspace{1cm} (9)

In the formula, $D_k = \frac{1}{\text{rank}(k)} > 0$ and $\text{rank}(k)$ represents the rank of experience $k$ in the experience replay buffer based on the absolute value of the TD-error. The parameter $\alpha$ determines the degree of priority, and when $\alpha = 0$, it becomes the uniform sampling. The relationship between the probability of experience being sampled and rank is shown in Fig. 4. From the definition of sampling probability, it can be seen that even the experience with lower TD-error value is also probable to be sampled, thereby ensuring the diversity of the sampled experience and preventing the neural network from overfitting. Nevertheless, those experiences with higher TD-error value will replay more frequently, thus changing the sampling frequency of each state and further resulting in oscillation or even divergence during training. To deal with this issue, we adopt importance sampling to adjust and update the model by reducing the weight of the top-ranking experience.

$$\omega_k = \frac{1}{S^\beta \cdot P(k)^\beta},$$ \hspace{1cm} (10)

where $S$ is the size of the experience replay buffer, $P(k)$ is the probability of the sampled experience $k$, and the parameter $\beta$ is used to control the impact of importance sampling weights on learning. The parameter $\beta$ will gradually increase to 1 during the training process. As $\beta$ increases, the weight of the high-priority samples is almost unchanged in (10), while the weight of the low-priority samples is greatly increased. When training starts to converge eventually, the unbiased update is crucial for error convergence. In order to improve the stability of the algorithm model training, we always normalize weights by $1/\max_k \omega_k$ so that they only scale the update downwards. Therefore, the definition of the loss function in (5) is changed to:

$$\mathcal{L} (\theta_i) = \frac{1}{K} \sum_k \omega_k \delta_k^2.$$ \hspace{1cm} (11)

Based on the above introduced prioritized experience replay method, an integrated algorithm of MADDPG with prioritized experience replay is shown as in Algorithm 1. The framework of PER-MADDPG is shown in Fig. 5.

**B. REWARD FUNCTION SETTING**

Based on this algorithm framework, we design the reward function of the robot, according to the multi-robot flocking behavior. As shown below, the reward function is mainly used to reward the expected behaviors and punish the undesirable actions:

$$R_i = \omega_g R_g^i + \omega_s R_s^i + \omega_c R_c^i + \omega_p R_p^i + \omega_o R_o^i,$$ \hspace{1cm} (12)

where $R_g^i$ is the reward for reaching the target point in the flocking task, $R_s^i$ is the reward for ensuring that robots in the flock can avoid collision among robots, $R_c^i$ is the reward for maintaining the flocking aggregation to enable robots‘ communication, $R_p^i$ is the reward for ensuring that robots in the group can avoid obstacles, and $R_o^i$ is the reward for making the velocity of each robot relatively smooth. In the expression (12), $\omega_g, \omega_s, \omega_c, \omega_o, \omega_p$ are positive weighting factors. Details about the reward function are given below.

1) **REACHING THE TARGET**

This reward function is to ensure that the robot group can reach the target. Each robot will receive a reward $r_{goal}$ when it reaches the target point $g$, and will get a punishment when it moves away from the target. The punishment is proportional to the distance from the robot to the target point.

$$R_g^i = \begin{cases} r_{goal} & \text{if } ||p_i - g||_2 \leq \rho_g \\ -||p_i - g||_2 & \text{otherwise}, \end{cases}$$ \hspace{1cm} (13)

where $p_i$ represents the position of the robot $r_i$, $\rho_g > 0$ represents the radius of the target area, and $r_{goal}$ is a positive constant.
Algorithm 1 MADDPG With Prioritized Experience Replay

1: Initialize priority parameters $\alpha$, $\beta$ and minibatch $K$, replay buffer $D$
2: for episode = 1 to total-episode do
3:   Initialize a random process $N$ for action exploration
4:   Receive initial state $s = (o_1, \ldots, o_N)$
5:   for $t = 1$ to max-episode-length do
6:     for each agent $r_i$, select action $a_i = \mu_i(o_i) + N_i$ w.r.t the current policy and exploration
7:     Execute actions $a = (a_1, \ldots, a_N)$ and receive reward $r$ and new state $s'$
8:     Store $(s, a, r, s')$ in replay buffer $D$
9:     $s' \leftarrow s$
10:    for agent $r_{i-1}$ to $r_{i+N}$ do
11:       for $j = 1$ to $K$ do
12:          Sample experience $k$ with probability $P(k)$ from $D$
13:          Compute corresponding importance-sampling weight $\omega_k$ and TD-error $\delta_k$
14:          Update the priority of experience $k$ according to absolute TD-error $|\delta_k|$
15:       end for
16:       Update critic by minimizing the loss $L(\theta_i) = \frac{1}{K} \sum_k \omega_k \delta_k^2$
17:       Update actor using the sampled policy gradient:
18:       $\nabla \theta_i J \approx \frac{1}{K} \sum_k \nabla \theta_i \mu_i(o_k) \nabla_a Q_i(s_k, a_k) \bigg|_{a_k = \mu_i(o_k)}$
19:    end for
20:    Update target network parameters for each agent $r_i$: $\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i$
21: end for
22: end for

FIGURE 5. The framework of PER-MADDPG.

2) AVOIDING COLLISION

This reward function is used to avoid collisions among robots. When the distance between two robots is less than the minimum safety distance $\rho_n > 0$, they will be punished. Conversely, if the distance is greater than the minimum safety distance, they will be rewarded.

$$ R^i_t = \begin{cases} r_{\text{avoid}} & \text{if } d(r_i, r_j) \geq \rho_n \\ - (\rho_n - d(r_i, r_j)) & \text{otherwise} \end{cases} \quad (14) $$

where $r_{\text{avoid}}$ is a positive constant.

3) MAINTAINING COMMUNICATION

This reward is used to promote connectivity within the group. When the distance between a robot and any other robot in the group exceeds the maximum communication distance $\rho_c > 0$, the robot will be punished. The greater the distance is, the more severe the penalty will be. When the distance between the two robots in the group is less than the maximum communication distance, they will get a reward. Such a setting is to maintain connectivity within the group.

$$ R^c_i = \begin{cases} r_{\text{comm}} & \text{if } d(r_i, r_j) \leq \rho_c \\ -(d(r_i, r_j) - \rho_c) & \text{otherwise} \end{cases} \quad (15) $$

where $r_{\text{comm}}$ is a positive constant.

4) AVOIDING OBSTACLE

Besides, in order to effectively avoid the obstacles in the process of completing the flocking task, a robot in the
flocking will be punished when its distance to any obstacle is less than the safe distance $\rho_o > 0$.

$$R_i^\rho = \begin{cases} r_{\text{obstacle}} & \text{if } d(r_i, o_j) \geq \rho_o \\ -(\rho_o - d(r_i, o_j)) & \text{otherwise} \end{cases} \quad (16)$$

where $r_{\text{obstacle}}$ is a positive constant.

5) SMOOTHING VELOCITY

Furthermore, we appropriately restrict the velocity direction change to make sure that the robot group can move relatively smoothly. Suppose the angle difference between velocities at two consecutive time instances is denoted by $\varphi$ (see Fig. 6), then we do not want $\varphi$ to be too large. Therefore, we define the following reward:

$$R_i^\varphi = - \arccos \left( \frac{v_i^t, v_i^{t+1}}{\|v_i^t\| \|v_i^{t+1}\|} \right) = -\varphi, \quad (17)$$

where $<a, b>$ denotes inner product of two vectors, and the arccos function’s range is $[0, \pi]$.

V. SIMULATION AND EXPERIMENT

A. EXPERIMENT SETTING

1) ENVIRONMENT SETTING

We have designed a simulation training environment of multiple robots flocking based on the OPENAI platform, including robots, obstacles, and target locations. The experimental area is a square centered on the origin with a side of 2, and the

FIGURE 7. The results of the obstacle-free experiment.

FIGURE 8. The results of the experiment with static obstacles.
radius and mass of each robot are 0.05 and 1, respectively. The maximum speed of the robot is limited to 2, and the maximum acceleration is limited to 2. The radius of each obstacle is 0.05, and the radius of the target area is 0.1. At the beginning of the training, the robots in the group are generated with coordinates \((-0.7, 0.7)\), \((-0.7, 0.9)\), \((-0.9, 0.7)\), and the obstacles are generated with random coordinates \(\{(O_x, O_y) | -1 \leq O_x, O_y \leq 1\}\), and the target position is selected randomly within the area of \(\{(T_x, T_y) | 0.8 \leq T_x, T_y \leq 1\}\).

2) PARAMETER SETTING AND THE TRAINING PROCESS

The four networks of Actor and Critic have the same structure, i.e., each network has three fully connected layers, and each layer has 64 units. The learning rate \(l_r\) is 0.01, and the discount factor \(\gamma\) is 0.95. The mini-batch is 1024, and the network parameters are updated every 100 steps. The robot can move 60 steps per episode, and the total episode of training is set to 60,000. The action that the \(i\)-th robot takes is the force \(F_i^t \in \mathbb{R}^2\) at time instance \(t\), so the velocity and position are updated by (1). The maximum communication range \(\rho_c\) is 0.2, the collision avoidance distance between robots \(\rho_n\) is 0.1, the target area radius is 0.125, and the collision avoidance distance between robots and obstacles \(\rho_o\) is 0.1.

B. THE EXPERIMENTAL RESULTS

In order to verify the effectiveness of our PER-MADDPG algorithm to complete the multi-robot flocking task in different scenarios, we carried out experiments in the presence of no obstacles, static obstacles and dynamic obstacles respectively. We evaluate the performance of our algorithm and the MADDPG algorithm in terms of four indices: the collision counts between robot and robot( CCBRR), collision count between robot and obstacle( CCBRO), mean distance between robot and target(MDBRT), mean distance between robot and robot(MDBRR). If the distance between the two robots is less than \(\rho_n\), the CCBRR will be increased by 1. If the distance between the robot and obstacle is less than \(\rho_o\), the CCBRO will be increased by 1.
1) OBSTACLE-FREE
We first conducted an experiment without obstacles. We selected the results of the first 10,000 episodes as shown in Fig. 7. It is obvious that PER-MADDPG has a faster convergence speed. In addition, the PER-MADDPG algorithm performs better in avoiding collisions among robots. The paths of the trained robots are shown in Fig. 10a.

2) STATIC OBSTACLE
In this experiment, five randomly distributed obstacles are added. The experiment results are shown in Fig. 8. It can be seen that, in the stable stage, the difference between the two reward curves is small, but in the process of convergence, the convergence speed of PER-MADDPG is faster than MADDPG. Fig. 10b is the trajectory diagram of multiple robots when there are static obstacles in the scene.

3) DYNAMIC OBSTACLE
The five randomly generated obstacles in this experiment all move at random velocities within the range $v = \{(v_x, v_y) : 0.5 < v_x, v_y < 0.5\}$, making it more difficult for the robots to complete the flocking task. The training results are shown in Fig. 9. In the dynamic obstacle scene, compared to MADDPG, the convergence speed of PER-MADDPG has been significantly improved, and the number of collisions have been obviously reduced.

VI. CONCLUSION
This paper uses MADDPG to solve the multi-robot flocking control problem without requiring complex control design as most traditional analysis methods do. Besides, to solve the algorithmic problem of low learning efficiency and slow convergence speed, this paper proposes a novel deep reinforcement learning algorithm, namely PER-MADDPG, by introducing the prioritized experience replay mechanism to enable the training results to converge faster. In addition, the experimental results also show that PER-MADDPG is better than MADDPG in completing multi-robot flocking tasks, having fewer collisions. Considering the cooperative features of the flocking task, we will improve the algorithm’s training efficiency through parameter sharing in the future.

REFERENCES
[1] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 6379–6390.
[2] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” 2015, arXiv:1511.05952. [Online]. Available: http://arxiv.org/abs/1511.05952
[3] K. Hausman, J. Müller, A. Haritharan, N. Ayarian, and G. S. Sukhatme, “Cooperative multi-robot control for target tracking with onboard sensing,” Int. J. Robot. Res., vol. 34, no. 13, pp. 1660–1677, Nov. 2015.
[4] L. Liu, C. Luo, and F. Shen, “Multi-agent formation control with target tracking and navigation,” in Proc. IEEE Int. Conf. Inf. Autom. (ICIA), Jul. 2017, pp. 98–103.
[5] A. Luis Bustamante, J. M. Molina, and M. A. Patricio, “A practical approach for active camera coordination based on a fusion-driven multi-agent system,” Int. J. Syst. Sci., vol. 45, no. 4, pp. 741–755, Apr. 2014.
[6] A. Farinelli, L. Iocchi, and D. Nardi, “Distributed on-line dynamic task assignment for multi-robot patrolling,” Auto. Robots, vol. 41, no. 6, pp. 1324–1345, Aug. 2017.
[7] A. Macwan, J. Vilela, G. Nejat, and B. Benhabib, “A multirobot path-planning strategy for autonomous wilderness search and rescue,” IEEE Trans. Cybern., vol. 45, no. 9, pp. 1784–1797, Sep. 2015.
[8] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in Proc. 14th Annu. Conf. Comput. Graph. Interact. Techn., 1987, pp. 25–34.
[9] T. Wicseck and A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shoched, “Novel type of phase transition in a system of self-driven particles,” Phys. Rev. Lett., vol. 75, no. 6, p. 1226, 1995.
[10] B.-M. Tian, H.-X. Yang, W. Li, W.-X. Wang, B.-H. Wang, and T. Zhou, “Optimal view angle in collective dynamics of self-propelled agents,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 79, no. 5, May 2009, Art. no. 052102.
[11] X. Zhang, S. Jia, and X. Li, “Improving the synchronization speed of self-propelled particles with restricted vision via randomly changing the line of sight,” Nonlinear Dyn., vol. 90, no. 1, pp. 43–51, Oct. 2017.
[12] O. Saif, I. Fantoni, and A. Zavala-Río, “Flocking of multiple unmanned aerial vehicles by LQR control,” in Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS), May 2014, pp. 222–228.
[13] S. Al-Abri, S. Maxon, and F. Zhang, “Integrating a PCA learning algorithm with the SUSD strategy for a collective source seeking behavior,” in Proc. Annu. Amer. Control Conf. (ACC), Jun. 2018, pp. 2479–2484.
[14] H. Su, X. Wang, and Z. Lin, “Flocking of multi-agents with a virtual leader,” IEEE Trans. Autom. Control, vol. 54, no. 2, pp. 293–307, Feb. 2009.
[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[16] T. Degriss, M. White, and R. S. Sutton, “Off-policy actor-critic,” 2012, arXiv:1205.4839. [Online]. Available: http://arxiv.org/abs/1205.4839
[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1928–1937.
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347
[19] 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,” 2015, arXiv:1509.02971. [Online]. Available: http://arxiv.org/abs/1509.02971
[20] W. Dai, H. Lu, J. Xiao, Z. Zeng, and Z. Zheng, “Multi-robot dynamic task allocation for exploration and destruction,” J. Intell. Robot. Syst., vol. 98, no. 2, pp. 455–479, May 2020.
[21] S. Sukhbaatar and R. Fergus, “Learning multiagent communication with backpropagation,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 2244–2252.
[22] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 2317–2325.
[23] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, “Lernient multi-agent deep reinforcement learning,” 2009, arXiv:0705.1402. [Online]. Available: http://arxiv.org/abs/1705.04402
[24] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” 2015, arXiv:1509.06461. [Online]. Available: https://arxiv.org/abs/1509.06461
[25] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” 2017, arXiv:1705.08926. [Online]. Available: https://arxiv.org/abs/1705.08926
[26] T. Rashid, M. Sennveyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multi-agent reinforcement learning,” 2020, arXiv:2003.08837. [Online]. Available: http://arxiv.org/abs/2003.08837
[27] Y. Zhang, Z. Zhuang, F. Gao, J. Wang, and Z. Han, “Multi-agent deep reinforcement learning for secure UAV communications,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCN), May 2020, pp. 1–5.
[28] H. Qie, D. Shi, T. Shen, X. Yu, Y. Li, and L. Wang, “Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning,” IEEE Access, vol. 7, pp. 146264–146272, 2019.
[29] J. Ma, H. Lu, J. Xiao, Z. Zeng, and Z. Zheng, “Multi-robot target encirclement control with collision avoidance via deep reinforcement learning,” J. Intell. Robot. Syst., vol. 99, no. 2, pp. 371–386, Aug. 2020.

[30] K. Morihiro, T. Isokawa, H. Nishimura, and N. Matsui, “Characteristics of flocking behavior model by reinforcement learning scheme,” in Proc. SICE-ICASE Int. Joint Conf., 2006, pp. 4551–4556.

[31] M. Tomimasu, K. Morihiro, H. Nishimura, T. Isokawa, and N. Matsui, “A reinforcement learning scheme of adaptive flocking behavior,” in Proc. 10th Int. Symp. Artif. Life Robot. (AROB), Oita, Japan, 2005.

[32] S.-M. Hung, S. N. Givigi, and A. Noureldin, “A dyna-Q (Lambda) approach to flocking with fixed-wing UAVs in a stochastic environment,” in Proc. IEEE Int. Conf. Syst., Man, Cybern., Oct. 2015, pp. 1918–1923.

[33] S.-M. Hung and S. N. Givigi, “A Q-learning approach to flocking with UAVs in a stochastic environment,” IEEE Trans. Cybern., vol. 47, no. 1, pp. 186–197, Jan. 2017.

[34] Z. Xu, Y. Lyu, Q. Pan, J. Hu, C. Zhao, and S. Liu, “Multi-vehicle flocking control with deep deterministic policy gradient method,” in Proc. IEEE 14th Int. Conf. Control Autom. (ICCA), Jun. 2018, pp. 306–311.

PENGMING ZHU received the bachelor of engineering degree from the Beijing Institute of Technology (BIT), China, in 2018. He is currently pursuing the master’s degree with the National University of Defense Technology (NUDT). His research interests include multi-agent reinforcement learning algorithms, multi-robot cooperation, distributed systems, and robot soccer.

WEIJIA YAO received the bachelor’s and master’s degrees in control engineering from the National University of Defense Technology, China, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Engineering and Technology Institute (ENTEG), University of Groningen, The Netherlands. His research interests include nonlinear systems and control, mobile robotics, and multi-agent systems.

WEI DAI received the bachelor of engineering and master of engineering degrees from the National University of Defense Technology (NUDT), China, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree. From 2018 to 2019, he was a Visiting Ph.D. Student with the Faculty of Science and Engineering, University of Groningen, The Netherlands. His research interests include shared control, multi-robot cooperation, distributed systems, and robot soccer.

JUNCHONG MA received the bachelor of engineering and master of engineering degrees from the National University of Defense Technology (NUDT), China, in 2016 and 2018, respectively. His research interests include multi-robot coordination and robot soccer.

WEIJIA YAO received the bachelor’s and master’s degrees in control engineering from the National University of Defense Technology, China, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Engineering and Technology Institute (ENTEG), University of Groningen, The Netherlands. His research interests include nonlinear systems and control, mobile robotics, and multi-agent systems.

PENGMING ZHU received the bachelor of engineering degree from the Beijing Institute of Technology (BIT), China, in 2018. He is currently pursuing the master’s degree with the National University of Defense Technology (NUDT). His research interests include multi-agent reinforcement learning algorithms, multi-robot cooperation, distributed systems, and robot soccer.

WEI DAI received the bachelor of engineering and master of engineering degrees from the National University of Defense Technology (NUDT), China, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree. From 2018 to 2019, he was a Visiting Ph.D. Student with the Faculty of Science and Engineering, University of Groningen, The Netherlands. His research interests include shared control, multi-robot cooperation, distributed systems, and robot soccer.

ZHIWEN ZENG received the bachelor of engineering degree from the University of Electronic Science and Technology of China, in 2009, and the master of engineering and Ph.D. degrees from the National University of Defense Technology (NUDT), China, in 2011 and 2016, respectively. Later, he joined the Department of Automation, NUDT, in 2016, where he is currently an Associate Professor. His research interests include mobile robotics, especially on multi-robot coordination, and robot control.

HUIMIN LU (Member, IEEE) received the bachelor of engineering, master of engineering, and Ph.D. degrees from the National University of Defense Technology (NUDT), China, in 2003, 2005, and 2010, respectively. Later, he joined the Department of Automation, NUDT, in 2010, where he is currently a Professor. His research interests include mobile robotics, especially on robot vision, multi-robot coordination, robot soccer, and robot rescue.