Journal of Intelligent & Robotic Systems
https://doi.org/10.1007/s10846-019-01106-x

Multi-robot Target Encirclement Control with Collision Avoidance via Deep Reinforcement Learning

Junchong Ma[1] · Huimin Lu[1] · Junhao Xiao[1] · Zhiwen Zeng[1] · Zhiqiang Zheng[1]

Received: 21 January 2019 / Accepted: 20 September 2019

Abstract

The target encirclement control of multi-robot systems via deep reinforcement learning has been investigated in this paper. Inspired by the encirclement behavior of dolphins to entrap the fishes, the encirclement control is mainly to enforce the robots to achieve a capturing formation pattern around a target, and can be widely applied in many areas such as coverage, patrolling, escorting, etc. Different from traditional methods, we propose a deep reinforcement learning framework for multi-robot target encirclement formation control, combining the advantages of the deep neural network and deterministic policy gradient algorithm, which is free from the complicated work of building the control model and designing the control law. Our method provides a distributed control architecture for each robot in continuous action space, relying only on local teammate information. Besides, the behavioral output at each time step is determined by its own independent network. In addition, both the robots and the moving target can be trained simultaneously. In that way, both cooperation and competition can be contained, and the results validate the effectiveness of the proposed algorithm.

Keywords Multi-robot · Deep reinforcement learning · Encirclement control · Collision avoidance

1 Introduction

Multi-robot system has received more and more attention among researchers owing to the broad applications, such as collaborative patrolling [3, 6], task allocation [7, 43], target tracking, and navigation [13, 21], autonomous search and rescue [24]. Among such research topics, the encirclement control of multi-robot systems has attracted the interest of many researchers for its practical value both in civil and military fields. The scope of the application includes reconnaissance and surveillance with multiple sensors or sensor-on-board robots [1, 28], capture of the enemy target by multiple UAVs [12], protection of the UGVs by a team of armed robotic vehicles (ARVs) [4]. The core issue of the above-mentioned applications is to control a group of intelligent unmanned platforms to form an expected formation to enclose a specific target and finally form a ring of encirclement.

Previous work on multi-robot target encirclement can be roughly divided into two categories based on the control architecture: centralized or distributed. The centralized method in a multi-robot system relies on a central robot, which plays a role of global control node for task assignment, trajectory planning, and even locomotion control for all robots. With centralized control, offline or online optimization algorithms can usually be used to obtain a global optimal control scheme. But the precondition is the full access to complete knowledge about all robots and perfect communication between robots, which is not always available in practice.

Usually, the centralized method is difficult to scale to a large number of robots as well as prone to disable the whole robot system with any failures of the central node.

In contrast, distributed methods have advantages over centralized methods and are widely used by researchers. In the existing literature about this issue, the method based on graph theory and classical control theory [17, 18, 29, 30]

Junchong Ma and Huimin Lu contributed equally to this work.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-019-01106-x) contains supplementary material, which is available to authorized users.

✉ Huimin Lu
lhmnew@nudt.edu.cn

Junchong Ma
junchong_nudt@outlook.com

[1] College of Intelligence Science and Technology, National University of Defense Technology, Changsha, 410073, People's Republic of China

Published online: 06 November 2019

has been favored by most researchers, since the topology of the connection of robots can be precisely described by a graph. However, there is an apparent limitation that it highly depends on the accurate physical modes, which are usually difficult to obtain in practice.

Due to the limitation of the traditional methods, we turn to the reinforcement learning to overcome the shortcomings of requiring precise models. In this work, we present a reinforcement learning algorithm based on deep deterministic policy gradient algorithm [32] and the neural network that allows multiple robots to track and encircle a target in continuous space. Additionally, both cooperation and competition scenario of the multi-robot system are considered in this work. The principle contribution contains four aspects: (1) a reinforcement learning framework combing the advantages of deep neural networks and deterministic policy gradient algorithms is provided; (2) this method achieves dynamic multi-robot formation control in continuous action and state space; (3) our algorithm provides a distributed control architecture for each robot that only relies on local teammates information as well as the output of behavior at each time step is only determined by its own independent network; (4) our method adapts to the mixed cooperation and competition situation.

The remainder of the paper is structured as follows: Section 2 briefly overviews the existing researches of multi-robot target encirclement and reinforcement learning in multi-robot system. Section 3 presents the problem formulation of target encirclement by multi-robot system. Section 4 introduces the framework of deep reinforcement learning algorithm used for target encirclement problem. Section 5 gives the validation of the algorithm through three simulations based on a multi-robot simulation platform developed by NuBot team[1]. The last section summarizes the paper and points out the research direction with potential significance. The video about the simulations is attached to this manuscript.

2 Related Works

In this section, the overview of multi-robot encirclement control and reinforcement learning in multi-agent systems will be given.

2.1 Multi-Robot Encirclement Control

For multi-robot encirclement control, all the robots are required to enclose specific stationary or moving target, and finally form a ring of encirclement to prevent the escape of the target.

The starting of encirclement control is initiated from the study of multi-agent cyclic pursuit, inspired by N-bugs problem [27]. Some previous studies including Kim et al. [17], Lan Ying and Yan Gangfeng [18], Kazuya Sato et al. [29], concentrate on the encircling of a stationary target. A few recent works have addressed the issue for moving target. Wang Chen and Xie Guangming et al. [38, 39] studied the distributed multi-robot target circumnavigation algorithm with any specified spacing. However, the algorithm required that all robots should be initially placed in a pre-desired position in circle. Nevertheless, Shi Yingjing [30] had studied the enclosing moving targets by multiple agents, but the algorithm they proposed implies that the locations of all agents must meet certain conditions. There are studies discussed more flexible distributed control algorithms for this problem. Franchi Antonio and Stegagno Paolo [11] designed a framework for achieving encirclement of a moving target by a multi-robot system in 3D space. Yao Weijia et al. [42] presented a distributed control law for encircling a moving target by a heterogeneous multi-robot system with dynamic formation spacings between robots.

However, previous works based on graph theory and control theory highly depend on the precise control model, which are not always available in practice.

Moreover, most of the literatures do not take into account the requirement of collision avoidance during encircling the target, which would likely to pose a danger to the robot team.

2.2 Reinforcement Learning in Multi-robot Systems

With the booming of machine learning, reinforcement learning has been increasingly studied. Through trial and error, animals or robots can adjust their behaviors in a renew or changing environment. This is the main intuition behind reinforcement learning [36]. The application of reinforcement learning has a rich history in many classical control issues, such as double inverted pendulum control [44], autonomous navigation [2], and motion control of humanoid robots [16]. Benefiting from powerful computing, deep neural networks, new algorithms, and mature software architecture, we have been witnessing the revival of reinforcement learning, especially, the deep reinforcement learning [25]. Deep reinforcement learning has recently achieved many eye-catching achievements, including defeating human players in game Atari with deep Q-learning [25], AlphaGo [31] and AlphaGo Zero [33]; dueling network architecture for model-free reinforcement learning [40], online learning for spoken dialogue system [34]; the dual-learning mechanism for machine translation [14].

The single agent reinforcement learning problem has been thoroughly studied and related algorithms have achieved impressive results. Unfortunately, when problems

[1] https://nubot.trustie.net

are extended from a single agent to multiple agents, the complexity will be greatly increased and most common methods are no longer effective. In Multi-agent reinforcement learning (MARL), both the transition between states and the reward assigning are functions of the joint actions of all agents in the environment. As the training progresses, each agent will eventually learn an optimal policy as the mapping of its observation to current optimal actions, but it cannot predict the actions of other agents.

Up to now, various problems in MARL have been comprehensively studied. The essence of interaction between agents includes cooperative, competitive, or both, thus many algorithms are designed for one specific interaction. Tampuu et al. [37] extended the Deep Q-Learning Network to the independent method in the classic two-agent computer game Pong. Foerster et al. [8] proposed the Reinforced Inter-Agent Learning (RIAL) algorithm and Differentiable Inter-Agent Learning (DIAL) algorithm, which realizes end-to-end learning in multi-agent collaborative communication. Similarly et al. [35] also explored a neural model named CommNet that is for learning to continuous communicate by backpropagation in complete cooperative tasks. Omidshafiei et al. [26] studied multi-agent cooperative problem with hysteretic Q function updates, which indicate that the goal of other agents is to improve collective reward. Leibo et al. [19] analyzed the sequential social dilemmas using deep Q-network in mixed-cooperation environments.

A few recent works have chosen popular video games as the research background and testbed for multi-agent reinforcement learning, and their research results have also received much attention. He et al. [15] proposed a neural network-based opponent behavior learning model with Mixture-of-Experts architecture to encode the opponent's observations into a deep Q network (DQN). The model has been validated in simple simulation football game and trivia games, both of which are the discrete problem. Several studies focused on independent Q-learning using a multi-agent variant of importance sampling [10], and using a centralized critic in policy gradient methods [9]. Both of them have been proved to be effective and verified in StarCraft unit micromanagement, which contains large state and action space in discrete domain.

Different from our contribution, these works mentioned above do not directly address multi-robot formation control in continuous space. In continuous domain, research [22] proposed an end-to-end algorithm for real-time obstacle avoidance of multi-robots. Lowe et al. [23] studied an Actor-Critic framework with centralized training and decentralized execution. But neither of them discussed multi-robot encirclement control issues, which includes both cooperative and competitive interactions.

3 Problem Formulation

In this section, we present the formal definitions and some notations of the multi-robot cooperation problem for target tracking and encirclement.

3.1 Encirclement Formation

Suppose that there are $n(n > 2)$ robots denoted by $r_1, r_2, \cdots, r_n$ in 2D space. The task of this multi-robot system is to form a circular formation that can encircle a stationary or moving target T on the same plane. In order to realize the target encirclement, the multi-robot formation should satisfy the following requirements as much as possible:

- The formation configuration should be a convex polygon which contains target inside composed of multiple robots;
- Whether the target is stationary or moving, the multi-robot formation should be able to adjust the position according to the speed of the target, and always keep the target inside the formation;
- On the formation configuration, any vertex of the convex polygon could be occupied by any robot in the group.

Varying from the number of the robots and the shape of the formation polygon, there are many formation configurations that can satisfy the above requirements. However, the main objective of this paper is to enclose the target and minimize the breach between two neighboring robots.

With this original intention, we explored a multi-robot formation configuration named *Encirclement Formation* as shown in Fig. 1, in which each robot performs an encirclement movement centering on the target. Moreover, depending on the dynamic performance of the target, consideration should be given to designing a variable number of multi-robot formation control frameworks.

For simplicity, we assume that the geometry of each robot and the target are modeled as a disc with an actual shape radius and a safe radius. Both the robots and the target are considered to be dynamic systems with holonomic

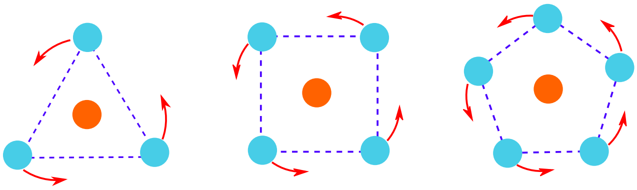


Fig. 1 Encirclement formation configurations

constraints, i.e., it can freely move to any direction on 2D plane.

We assume that each robot can obtain global target information, but only perceive local information from its neighboring robots, not all of the robots. Thus, instead of centralized architecture, a distributed formation control architecture is required.

To facilitate the representation of the relevant attributes of the formation, we describe the specific characteristics of the encirclement formation in polar coordinates of target body reference frame $\mathcal{B}$, centered at the target H. Robots are sorted and numbered in counter-clockwise in accordance with their phase in the polar coordinate, as shown in Fig. 2.

The position of robot r_i in the body reference frame is expressed as $\boldsymbol{q_i} = (\rho_i, \phi_i)$. $\Delta_i (\Delta_i > 0)$ denotes the angular spacing between two neighboring robots r_i and r_{i-1}. with the constraint of

$$\sum_{i=1}^{n} \Delta_i = 2\pi. \tag{1}$$

In particular,

$$\Delta_i = \begin{cases} \phi_i - \phi_n + 2\pi, & i = 1 \\ \phi_i - \phi_{i-1}, & i = 2, ..., n. \end{cases} \tag{2}$$

To formalize the problem, the definition of the multi-robot encirclement formation control problem is given as follows:

Definition 1 (Multi-robot encirclement control) For the multi-robot system containing $n(n \geq 2)$ mobile robots, the encirclement control aims to figure out a control law that satisfies the following conditions:

$$\lim_{t\to\infty} \rho_i(t) \quad = \rho^*, \rho^* > 0, \tag{3c}$$

$$\lim_{t\to\infty} \dot{\phi}_i(t) \quad = \omega^*, \omega^* > 0, \tag{3d}$$

$$\lim_{t\to\infty} \phi_i(t) = \bar{\phi}_i(t), \bar{\phi}_i(t) > 0. \tag{3e}$$

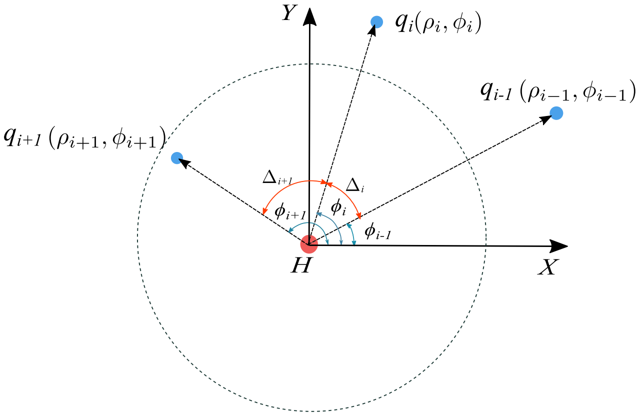


Fig. 2 Robots' positions in polar coordinate

where ρ^* and ω^* are the desired encirclement radius and angular velocity respectively; $\bar{\phi}_i(t)$ represents the desired angular position, which indirectly specifies the angular spacing Δ_i between neighboring robots.

3.2 Reinforcement Learning in Encirclement Formation

In the multi-robot encirclement problem, we consider a common reinforcement learning, which contains cooperative and competitive interactions between robots and the target. In this scenario, robots are in a cooperative relationship for a common purpose of enclosing the target; in contrast, robots are in a competitive relationship with the target to be enclosed. As an extension of the Markov decision process (MDP), this multi-agent reinforcement learning setup can be regarded as a stochastic game called the Markov game (MG).

A MG process of multi-robot encirclement can be described to by a tuple $G = \langle n, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \mathcal{O}, \gamma \rangle$, where $\mathcal{S}$ is the state space, describing the state of the environment as well as agents' states. The joint action for all robots is denoted as $\mathcal{A} = \{\boldsymbol{A_1}, \boldsymbol{A_2}, \cdots, \boldsymbol{A_n}\}$, where the real-valued $\boldsymbol{A_i} \in \mathbb{R}^N$ is the N-dimension continuous action space for each robot. In the interaction process, the state-action pairs of robots can be represented by the transition function $\mathcal{T}(s_t, a_t, s_{t+1}) : \mathcal{S} \times \boldsymbol{A}_1 \times \boldsymbol{A}_1 \times \cdots \times \boldsymbol{A}_n \times \mathcal{S}' \mapsto [0, 1]$, representing the probability distribution under conditional density $p(s_{t+1}|s_t, a_t)$, which satisfies the Markov property $p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_1, a_1, \cdots, s_t, a_t)$ in any sequential decisions $\{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$. Each robot interacts with the Markov game process to get return R, presented by the reward function $R_i(s_t, a_t^1, a_t^2, ..., a_t^n, s_{t+1}) : \mathcal{S} \times \boldsymbol{A}_1 \times \boldsymbol{A}_1 \times \cdots \times \boldsymbol{A}_n \times \mathcal{S} \mapsto \mathbb{R}$. The set of observations for all robots is described to be $\mathcal{O} = \{\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_n\}$ and $\boldsymbol{O}_i$ is the observation of each robot r_i.

During the dynamic encirclement, each robot computes its action in continuous 2D space through learned policy based on observation, striving to maintain the expected relative position with the teammates at each time step. The behavior of each robot r_i is defined by a policy $\boldsymbol{\pi}_{\theta_i}(a_t|s_t) : \boldsymbol{O}_i \times \boldsymbol{A_i} \mapsto \mathcal{P}(\mathcal{A})$, which transfers the state s_t to a new state s_{t+1} based on the transition function $\mathcal{T}$. $\mathcal{P}(\mathcal{A})$ is a set of probability computed on $\mathcal{A}$, and $\theta \in \mathbb{R}^m$ is a parameter vector with m elements.

The goal of each robot r_i is to learn an optimal policy $\boldsymbol{\pi}_{\theta_i}^*$ that maximizes its own cumulative discounted reward G_t from the starting state:

$$G_t = \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + ... = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}, \tag{4}$$

where $\gamma (0 < \gamma < 1)$ is the discount factor for each step.

In general, a standard reinforcement learning process uses the action-value function $Q^{\pi_\theta}(s,a)$ to describe the expected reward after taking action a_t at state s_t following the policy $\boldsymbol{\pi}_\theta$:

$$Q^{\pi_\theta}(s,a) = \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k} | s_t = s, a_t = a\right], \tag{5}$$

which is known as the Bellman equation and can be rewritten in recursive relationship:

$$Q^{\pi_\theta}(s,a) = \mathbb{E}\left[R_{t+1} + \gamma Q^{\pi_\theta}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a\right]. \tag{6}$$

As in many previous researches, Q-learning uses the greedy policy $\mu(s) = \arg\max_a Q(s,a)$. The function approximators are parameterized by θ^Q, optimized by minimizing the loss:

$$L(\theta^Q) = \mathbb{E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right], \tag{7}$$

where

$$y_t = R(s_t, a_t) + \gamma Q(s_{t+1}, \mu(t+1)|\theta^Q). \tag{8}$$

It is worth noting that in reinforcement learning of multi-robot system, whether the policies of robots are independently updated or unified updated, both the state transition and reward calculation are under the joint action of all the agents, which is shown in the Fig. 3.

In this work, we do not require that each robot has the perfect perception for the environment and all other robots, though such information is usually needed in some previous researches. We consider the observation vector $\boldsymbol{O}_i$ of each robot as a partially observable setting. More specifically, each robot can observe the position and speed of the target, but can only observe the position and speed of adjacent (or neighboring) robots in the formation, and cannot observe all the robots. Therefore the MG process in this formation control issue is exactly a partially observable Markov game problem.

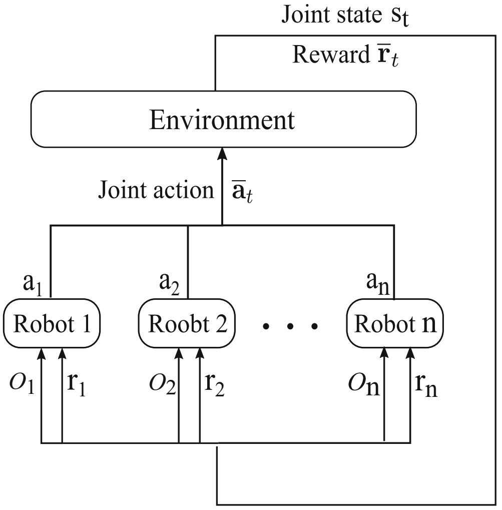


Fig. 3 Multiple robots interacting in reinforcement learning

4 The Proposed Approach

In this section, we introduce the main components of the reinforcement learning method, which is employed to deal with the mixed cooperative and competitive multi-robot problem.

4.1 Deep Deterministic Actor-Critic Algorithm

It is difficult to directly apply Q-learning to deal with continuous action spaces, as the policy optimization will become unacceptably slow [20]. In the process of multi-robot reinforcement learning for encirclement formation, each robot independently updates its own policy. Hence the environment seems to be non-stationary in the view of each robot, which does not meet the precondition of convergence in Q-learning.

The policy gradient algorithms have been favored by many researchers, which perform well in the continuous action spaces reinforcement learning.

According to whether to select actions randomly, policy gradient algorithms can be classified to stochastic policy gradient and deterministic policy gradient. The goal of the stochastic policy gradient algorithm is to maximize the following function [32],

$$\begin{aligned} J(\boldsymbol{\pi}_\theta) &= \int_{\mathcal{S}} \rho^{\pi_\theta}(s) \int_{\mathcal{A}} \boldsymbol{\pi}_\theta(a|s) R(s,a) \, \mathrm{d}a\mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\pi_\theta}, a\sim\boldsymbol{\pi}_\theta}[R(s,a)], \end{aligned} \tag{9}$$

where $\rho^{\boldsymbol{\pi}}(s)$ is the state distribution and $\mathbb{E}_{s\sim\rho^{\pi}, a\sim\boldsymbol{\pi}_\theta}[\cdot]$ is the expected value under the distribution $\rho^{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}_\theta$. The main idea of the stochastic policy gradient is to iterate the parameter θ toward the gradient direction of performance:

$$\begin{aligned} \nabla_\theta J(\boldsymbol{\pi}_\theta) &= \int_{\mathcal{S}} \rho^{\pi_\theta}(s) \int_{\mathcal{A}} \nabla_\theta \boldsymbol{\pi}_\theta(a|s) Q^{\pi_\theta}(s,a) \, \mathrm{d}a\mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\pi_\theta}, a\sim\boldsymbol{\pi}_\theta}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\pi_\theta}(s,a)]. \end{aligned} \tag{10}$$

For this purpose, it randomly selects actions, and explores the action and state space, at the cost of increasing the computational complexity. Furthermore, the algorithm has a high variation in the estimation of the performance gradient. For the multi-robot system case, if each robot does not consider other robots in the policy optimization process, the variance will be even larger, which counts against the convergence of the entire system.

The deterministic policy gradient algorithm is proposed to deal with the above-mentioned problem, which selects a certain action for each state, i.e., there is a mapping from state to action $\boldsymbol{\mu}_\theta : \mathcal{S} \mapsto \mathcal{A}, \theta \in \mathbb{R}^m$. By analogizing the stochastic policy gradient, we use $J(\boldsymbol{\mu}_\theta)$ as the performance objective and $\rho^{\boldsymbol{\mu}_\theta}(s)$ as the discounted state distribution. Then the performance objective in Eq. 9 needs to be rewritten as Eq. 11 [32].

$$\begin{aligned} J(\boldsymbol{\mu}_\theta) &= \int_{\mathcal{S}} \rho^{\boldsymbol{\mu}_\theta}(s) R(s, \boldsymbol{\mu}_\theta(s)) \ \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\boldsymbol{\mu}_\theta}}[R(s, \boldsymbol{\mu}_\theta(s))], \end{aligned} \tag{11}$$

where $\rho^{\boldsymbol{\mu}}(s)$ is the state distribution. In the continuous action space $\mathcal{A}$, the gradient of the deterministic policy is

$$\begin{aligned} \nabla_\theta J(\boldsymbol{\mu}_\theta) &= \int_{\mathcal{S}} \rho^{\boldsymbol{\mu}_\theta}(s) \nabla_\theta \boldsymbol{\mu}_\theta(s) \nabla_a Q^{\boldsymbol{\mu}_\theta}(s, a)|_{a=\boldsymbol{\mu}_\theta(s)} \ \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\boldsymbol{\mu}_\theta}}[\nabla_\theta \boldsymbol{\mu}_\theta(s) \nabla_a Q^{\boldsymbol{\mu}_\theta}(s, a)|_{a=\boldsymbol{\mu}_\theta(s)}]. \end{aligned} \tag{12}$$

In general, simply using the deterministic policy gradient algorithm limits the exploring space, which may guide the robot to a locally optimal solution. In order to deal with this problem, we implement the off-policy actor-critic architecture in the training process to help robots learn optimal strategies faster.

The off-policy actor-critic algorithm uses a different behavior policy $\boldsymbol{\beta}(a|s) \neq \boldsymbol{\pi}_\theta(a|s)$ to estimate the policy gradient. With the off-policy setting, the performance objective is changed to be the value function of the target policy that averaged with the state distribution in the behavior policy [5],

$$\begin{aligned} J_{\boldsymbol{\beta}}(\boldsymbol{\mu}_\theta) &= \int_{\mathcal{S}} \rho^{\boldsymbol{\beta}}(s) V^{\boldsymbol{\mu}_\theta}(s) \mathrm{d}s \\ &= \int_{\mathcal{S}} \rho^{\boldsymbol{\beta}}(s) Q^{\boldsymbol{\mu}_\theta}(s, \boldsymbol{\mu}_\theta(s)) \mathrm{d}s. \end{aligned} \tag{13}$$

Then the off-policy policy gradient can be obtained as an approximation of the differential of performance objective,

$$\begin{aligned} \nabla_\theta J_{\boldsymbol{\beta}}(\boldsymbol{\mu}_\theta) &\approx \int_{\mathcal{S}} \rho^{\boldsymbol{\beta}}(s) \nabla_\theta \boldsymbol{\mu}_\theta(a|s) Q^{\boldsymbol{\mu}}(s, a) \ \mathrm{d}a\mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^{\boldsymbol{\beta}}}[\nabla_\theta \boldsymbol{\mu}_\theta(s) \nabla_a Q^{\boldsymbol{\mu}}(s, a)|_{a=\boldsymbol{\mu}_\theta(s)}]. \end{aligned} \tag{14}$$

Within this framework, two main components play equally significant roles in policy optimization [32]: The *actor* explores and updates the parameter θ of deterministic policy $\boldsymbol{\mu}_\theta(s)$ through stochastic policy gradient ascent in Eq. 14, where the true action-value function $Q^{\boldsymbol{\mu}}(s, a)$ is substituted by a differentiable action-value function $Q^{\boldsymbol{w}}(s, a)$ with parameter vector w. The *critic* estimates the action-value function $Q^{\boldsymbol{w}}(s, a) \approx Q^{\boldsymbol{\mu}}(s, a)$ through an appropriate policy evaluation algorithm.

More specifically, the update of off-policy deterministic policy gradient is as follows:

$$\begin{aligned} \delta_t &= R_t + \gamma Q^{\boldsymbol{w}}(s_{t+1}, \boldsymbol{\mu}_\theta(s_{t+1})) - Q^{\boldsymbol{w}}(s_t, a_t), \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^{\boldsymbol{w}}(s_t, a_t), \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \boldsymbol{\mu}_\theta(s_t) Q^{\boldsymbol{w}}(s_t, a_t)|a = \boldsymbol{\mu}_\theta(s). \end{aligned} \tag{15}$$

where $w \in \mathbb{R}^m$ is a parameter vector with m elements, α_w and α_θ are the learning rates.

Motivated by the idea in [20], we take the advantage of neural network function estimator learning in large action and state space. However, if the function approximators are trained by the neural network directly with the data of reinforcement learning, there is a strong temporal correlation between the data (the Markov property in sequential interactions), which may result in the unstable convergence of training with the neural network. Hence, to satisfy the assumption that the sampled data are independent and identically distributed, we use a *replay buffer* $\mathcal{D}$ and *target networks* as used in the Deep Q-Network (DQN) [25].

Therefore, as in [20], we use the copy of actor and critic networks, denoted as $\mu'(s, a|\theta^{\mu'})$ and $Q'(s, a|\theta^{Q'})$ respectively, to calculate the target value. The weights of these target networks are updated slowly using ”soft” target updates: $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \tau \ll 1$, which enhances the stability of learning.

In the beginning, the stochastic exploration policy will sample the transitions from the environment, and the corresponding transition tuples (s_t, a_t, R_t, s_{t+1}) will be stored in a finite-sized replay buffer $\mathcal{D}$. Then the formal iterative training will start when the buffer $\mathcal{D}$ is full. At each step in training, a mini-batch data of tuples was sampled uniformly from the buffer to the update of the actor and critic.

We built the neural network based on the actor-critic framework, which implements policy search with learned value function. The parallel training on multiple processing threads can handle the large action and state space when asynchronous updating of policy and value function networks. The neural network in the proposed framework consists of two parts: the actor network and the critic network. The actor network takes the state of environment as input and returns the action of the robot at the current moment; the critic network returns the Q value based on the state of encirclement and the actions of all the robots.

Furthermore, the actor contains two networks with the same structure, the online policy network and the target policy network, which are used to calculate $\mu(s, a|\theta^{\mu})$ and $\mu'(s, a|\theta^{\mu'})$ respectively. Similarly, critic contains two networks with the same structure, the online Q network and the target Q network, which are used to calculate $Q(s, a|\theta^{Q})$ and $Q'(s, a|\theta^{Q'})$ respectively. The online network architecture of actor and critic used in the proposed algorithm are shown in Fig. 4. As shown in the figure, each neural network consists of 3 fully connected layers, and each layer containing 64 neurons.

As each robot is relatively independent in the distributed multi-robot system, all of them have their own reinforcement learning networks. Finally, the entire framework and training process of the reinforcement learning for

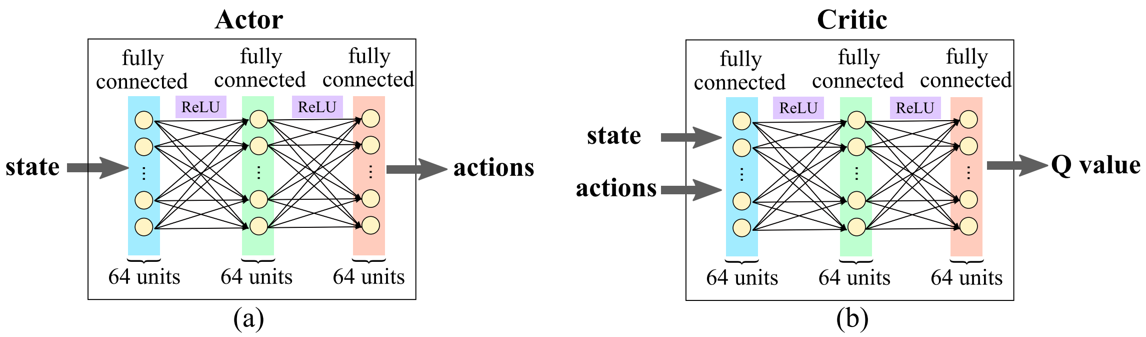


Fig. 4 Neural network design of Actor-Critic Framework. **a** and **b** represent the Actor network and the Critic network structure respectively

multi-robot encirclement formation control are illustrated in Fig. 5.

4.2 Priority in Reward Design

There are several specific constraints in the problem of multi-robot encirclement, which are exactly the control objectives of formation configuration, including encirclement radius ρ_i, angular velocity ω_i, phase difference Δ_i between neighboring robots, see Definition 1 for details.

If we directly use the algorithm framework described above, as shown in Section 5.1, the error of encirclement radius ρ_i can be stabilized in a small range after training. However, the other errors will be in a large oscillation range.

To solve the problem, we propose a priority-based method for reward function, which can make the control objectives converge one after another under the premise of ensuring the stability of the learning process.

Besides the constraints of formation configuration, collision avoidance is the ability with the highest priority that robots have to learn. We use a simple quadratic function r_i^c to define the collision avoidance reward for each robot r_i,

$$r_i^c = \begin{cases} -k_1 \left[(D_i^* + D_k^*) - \parallel \mathbf{p}_i - \mathbf{p}_k \parallel\right]^2, & \parallel \mathbf{p}_i - \mathbf{p}_k \parallel \leq D_i^* + D_k^* \\ 0, & \parallel \mathbf{p}_i - \mathbf{p}_k \parallel > D_i^* + D_k^*. \end{cases} \tag{16}$$

where k_1 is the weight coefficient; D_i^* and D_k^* are the safe radius of robot r_i and obstacle o_k respectively (all other entities in the environment are obstacles in the view of each robot); $\parallel \mathbf{p}_i - \mathbf{p}_k \parallel$ is the euclidean distance between robot r_i and obstacle o_k. In addition, the *collision_counter* grows one every time the distance between the robot and the obstacle is less than the safe distance.

Then, we use the following function to define the encirclement radius reward r_i^ρ for robot r_i, which has the second highest priority.

$$r_i^\rho = \begin{cases} -k_2 e_\rho + \max(r_i^\rho), & collision\_counter \leq 1 \\ 0, & otherwise. \end{cases} \tag{17}$$

where k_2 is the weight coefficient; $e_\rho = \rho_i - \rho^*$ is the error of encirclement radius of r_i; $\max(r_i^\rho)$ is the upper limit for the reward.

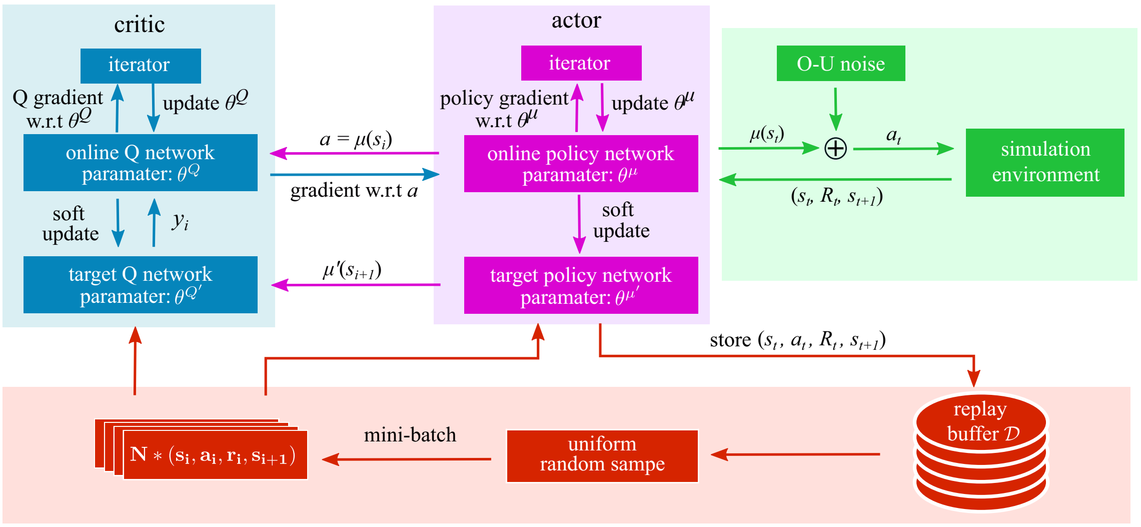

critic
iterator
Q gradient w.r.t θ^Q
update θ^Q
online Q network paramater: θ^Q
soft update
y_i
target Q network paramater: $\theta^{Q'}$
$a = \mu(s_i)$
gradient w.r.t a
$\mu'(s_{i+1})$
actor
iterator
policy gradient w.r.t θ^μ
update θ^μ
online policy network paramater: θ^μ
soft update
target policy network paramater: $\theta^{\mu'}$
O-U noise
$\mu(s_t)$
a_t
simulation environment
(s_t, R_t, s_{t+1})
store (s_t, a_t, R_t, s_{t+1})
replay buffer $\mathcal{D}$
uniform random sampe
mini-batch
$\mathrm{N} * (\mathrm{s_i}, \mathrm{a_i}, \mathrm{r_i}, \mathrm{s_{i+1}})$

Fig. 5 The Framework of Deep Deterministic Actor-Critic Algorithm

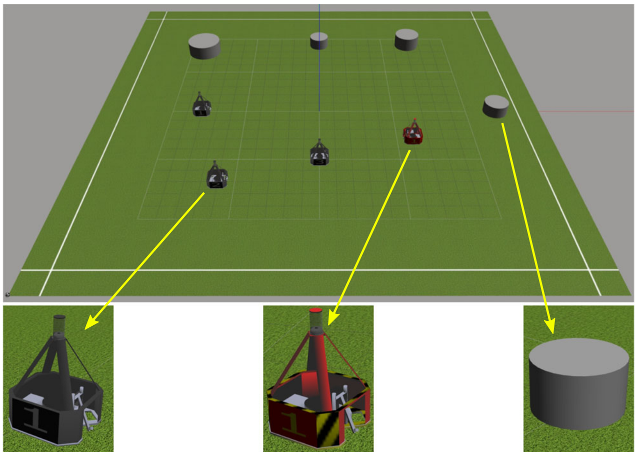


Fig. 6 The Environment of Gazebo Simulator

Next, we define the reward function r_i^{ϕ} of the phase that indicates the angular difference between neighboring robots, whose priority is lower than the encirclement radius.

$$r_i^{\phi} = \begin{cases} -k_3 e_{\phi} + \max(r_i^{\phi}), & e_{\rho} \leq 0.05 \max(e_{\rho}) \\ 0, & otherwise. \end{cases} \tag{18}$$

where k_3 is the weight coefficient; $e_{\phi} = \phi_i - \bar{\phi}_i$ is the error of phase of r_i; $e_{\rho} \leq 0.05 \max(e_{\rho})$ means that it can be rewarded only when e_{ρ} is less than a small error.

Finally, we use the same idea to define the angular velocity reward r_i^{ω}, which has the lowest priority for its less importance for realizing in the dynamic formation.

$$r_i^{\omega} = \begin{cases} -k_4 e_{\omega} + \max(r_i^{\omega}), & e_{\phi} \leq \pi/12 \\ 0, & otherwise. \end{cases} \tag{19}$$

where k_4 is the weight coefficient; $e_{\omega} = \omega_i - \omega^*$ is the error of angular velocity of r_i; $e_{\phi} \leq \pi/12$ means that only when e_{ϕ} is less than a small error can it be rewarded.

It should be noted that the output value of the three reward functions including the encirclement radius, the difference between neighboring robots, and the angular velocity should be set in the same range by adjusting weight coefficients. For example, they are set to [0, 20] in the simulations of Section 5. In contrast, the collision avoidance reward function mainly gives a punitive return, that is, a negative value whose range is set to $[-20, 0]$. The final reward function is the weighted sum of these four reward functions.

$$r_i = \alpha_1 r_i^c + \alpha_2 r_i^{\rho} + \alpha_3 r_i^{\phi} + \alpha_4 r_i^{\omega} \tag{20}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are weight coefficients.

How to prioritize each reward function depends on the importance of each control objective and the difficulty to learn. Moreover, the experimental results prove that using such a priority order, the best control effect can be obtained and the learning can be ensured to converge stably.

5 Experimental Results and Analysis

In order to verify the effectiveness of the proposed algorithm, we have implemented it under Linux and conducted three experiments based on our custom designed multi-robot simulation environment [41]: (1). encircle a stationary target; (2). encircle a moving target; (3). encircle an escaping target. The complexity of these three experiments is from low to high, and all the results prove that the algorithm works well in multi-robot encirclement formation control.

The simulation is set to be in a 2D square area, where a group of autonomous mobile robots can move freely to encircle a stationary/moving/escaping target while avoiding randomly placed obstacles. As shown in Fig. 6, the gray robots will encircle the red robot, while avoiding the gray cylinder obstacles and other robots.

In the simulation environment, each robot is an independent object, and for all objects, second-order dynamic models are considered, where control inputs are acceleration vectors. To make the simulation more realistic, the models also include physical properties such as friction and collision. The dynamic properties of the robots and target are listed in Table. 1, where all physical quantities are in standard international units.

In the simulation, the range of the movement for the robots and target is limited to a square area with the dimension of 2m*2m. The radius of the robots and target are 0.05 meters, and the radius of obstacles are random values within the range of 0.05 ∼ 0.07 meters.

Table 1 The parameters for simulation environment and models

Different parameters		Simulation I	Simulation II	Simulation III
Target	Max Velocity	0	0.6	0.8
	Max Acceleration	0	0.2	0.2
Robot	Max Velocity	0.2	0.8	1
	Max Acceleration	0.2	0.3	0.4
Obstacles number		4	0	4
Common parameters		Mass = 1, Velocity Damping = 0.1, Randomly Initial Position		

During training, the size of the replay buffer $\mathcal{D}$ is set to 100,000, the learning rate is set to 1e-3, the discount factor γ is 0.9, and the parameter τ used for "soft" target update is set to 1e-2. In the simulation, episodes are used for learning. In the process of training, when one of the following two conditions is satisfied, the episode will be terminated. The first condition is that the number of steps has reached the maximum in one episode. The other condition is that all errors of the three control objectives in the formation are less than the expected values.

Considering the characteristics of distributed formation control, each robot has its own reinforcement learning training network, which is used to determine the action at each time step. We assume that each robot can observe the global position information of obstacles and the target, but it can only observe the position and velocity information of adjacent robots.

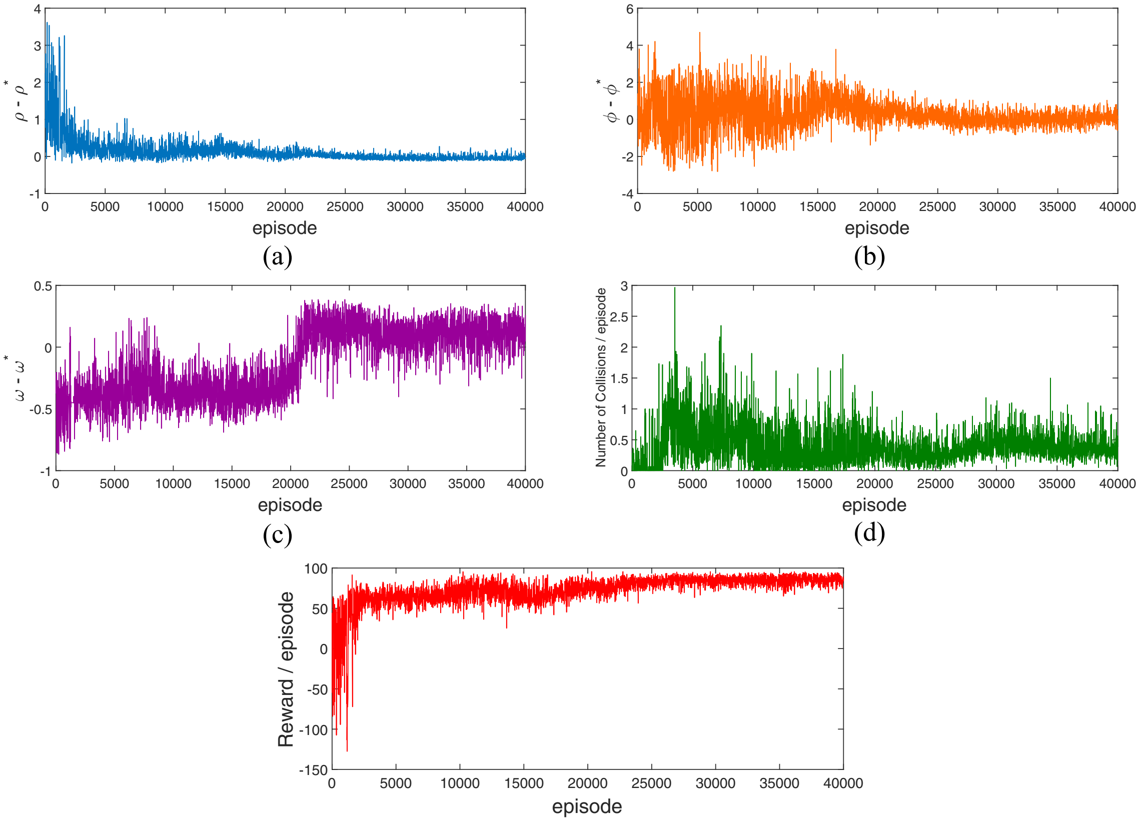
5.1 Simulation I: Stationary Target Encirclement

In this section, we conducted two experiments, where three robots will learn to cooperate with each other to implement the encirclement task while avoiding collisions. Note that in the first experiment, the priority scheme given in Section 4.2 was not used, and in the second experiment, we added the priority scheme. The comparison of these two simulation results can validate the effectiveness of the proposed priority scheme.

The expected control objectives of the multi-robot encirclement formation are set as follows: the circumstance radius is 0.25 m, the phase difference between two neighboring robots is $2\pi/n$ rad, which means that the robots are equally distributed on a circle, and the angular velocity of the encirclement movement is 1.5 rad/s.

In the first experiment, 40,000 episodes of training were performed on three robots, and each episode consisted of a 60-step iteration. Finally, the robots learned to cooperate with each other to form an encirclement formation and enclose the target. During the entire training process, the error data of control objectives for all robots are recorded in each episode, and the error curve of robot 1 is shown in Fig. 7.

It can be seen from the results that, as the training proceeds, the errors of the objectives gradually converge



(a)

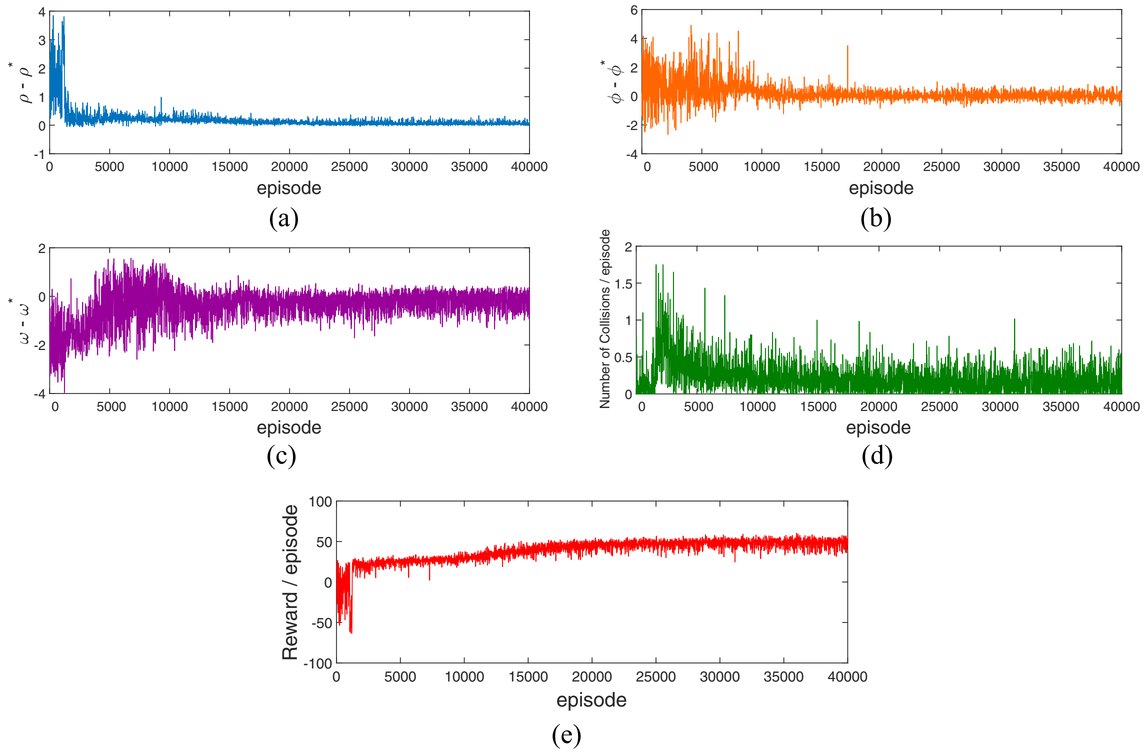
(b)

(c)

(d)

(e)

Fig. 7 Error curves of objectives in the training of the first experiment of Simulation I. a-c are the error curves for $\rho - \rho^*$, $\phi - \phi^*$, $\omega - \omega^*$ respectively, d plots the number of the collision with other robots or obstacles in each episode, and e represents the cumulative reward in each episode



(a)

(b)

(c)

(d)

(e)

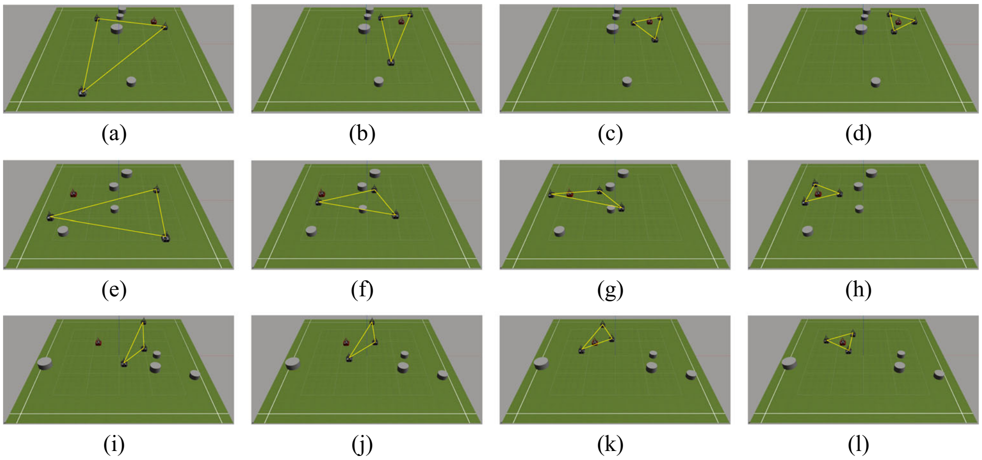
Fig. 8 Error curves of objectives in the training of the second experiment of Simulation I. a-c are the error curves for $\rho - \rho^*$, $\phi - \phi^*$, $\omega - \omega^*$ respectively, d plots the number of the collision with other robots or obstacles in each episode, and e represents the cumulative reward in each episode

to a small range. At the same time, the cumulative reward increases gradually from negative to a stable positive reward, which means that the robot has gradually learned a high reward policy under the reward and punishment mechanism.

For comparison, in the second experiment, we added the priority scheme to the reward function and performed the training with 40,000 episodes, where each episode consisted of a 60-step iteration. During the training process, the error data of control objectives for all robots are recorded in each episode, and the error curve of robot1 is shown in Fig. 8.

After training, the robots have learned to form an encirclement formation and move around the target with the desired angular speed. Figure 9 shows the dynamic process of the trained robots in the form of encirclement formation.

Since the priority scheme was added to the reward design in the second experiment, the convergence of the control



(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Fig. 9 The second experiment of Simulation I: Dynamic process of the robots trapping the target. The four pictures in each row represent an enclosing process, which shows the process of the robot's encirclement when the target is in three different positions

objectives error in the training process is faster than that in the first experiment. Moreover, the final errors of the control objectives converge to a smaller range than that in the first experiment.

5.2 Simulation II: Moving Target Encirclement

On the basis of Simulation I, the modification in this scenario of Simulation II is to give the target a velocity to make the encirclement task more difficult. Like the the second experiment of simulation I, we added prioritized scheme to the reward function as proposed in Section 4.2.

Specifically, the target can move freely within the square area. However, the robots can not predict the target's speed. During training, the velocity vector $\left[v_x, v_y\right]$ of the target is randomly generated, but the acceleration of the target is limited as shown in Table 1.

In this experiment, 50,000 episodes of training were performed on four robots, and each episode consisted of a 60-step iteration. Finally, the robots learned to cooperate with each other to form an encirclement formation and enclose the moving target. During the entire training process, the error data of control objectives for all robots are recorded in each episode, and the error curve of robot 1 is shown in Fig. 10.

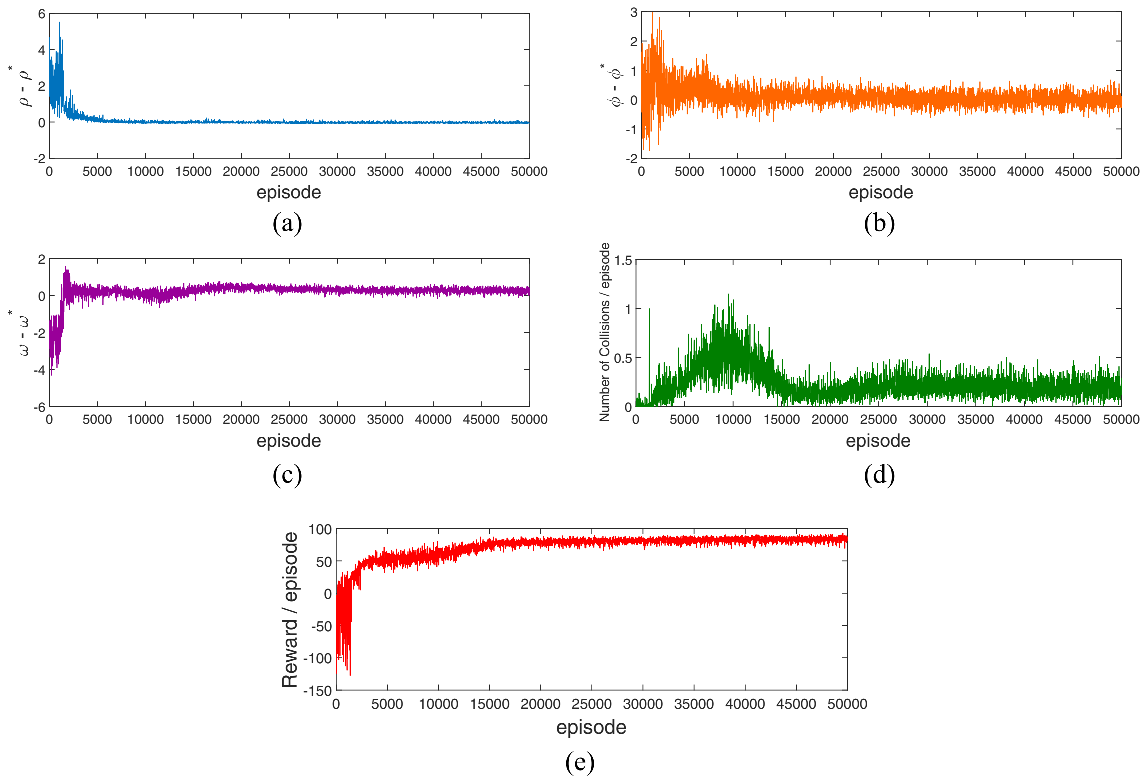
It can be seen from the results that, as the training proceeds, the errors of the objectives gradually converge to a small range. At the same time, the cumulative reward of the robot increases gradually from negative to a stable positive reward, which means that the robot has gradually learned a high reward policy under the reward and punishment mechanism.

After training, the robots have learned to form an encirclement formation and move around the target with the desired angular speed. Figure 11 shows the dynamic process of the trained robots in the form of encirclement formation in simulation environment.

In the test, we designed two typical movement trajectories for the target, which are circle shape trajectory and "∞" shape trajectory, and the initial positions of the robots are randomly generated. The trajectories of the robots and the target are shown in Fig. 12

5.3 Simulation III: Escaping Target Encirclement

Simulation I and Simulation II verify the effectiveness of the proposed algorithm in the case of the stationary target and the moving target. However, these two simulations only involve the cooperative game between the robots and do not consider the competitive relationship between the



(a)

(b)

(c)

(d)

(e)

Fig. 10 Error Curves of objectives in the Training of Simulation II. a-c are the error curves for $\rho - \rho^*$, $\phi - \phi^*$, $\omega - \omega^*$ respectively, d plots the number of the collision with other robots or obstacles in each episode, and e represents the cumulative reward in each episode

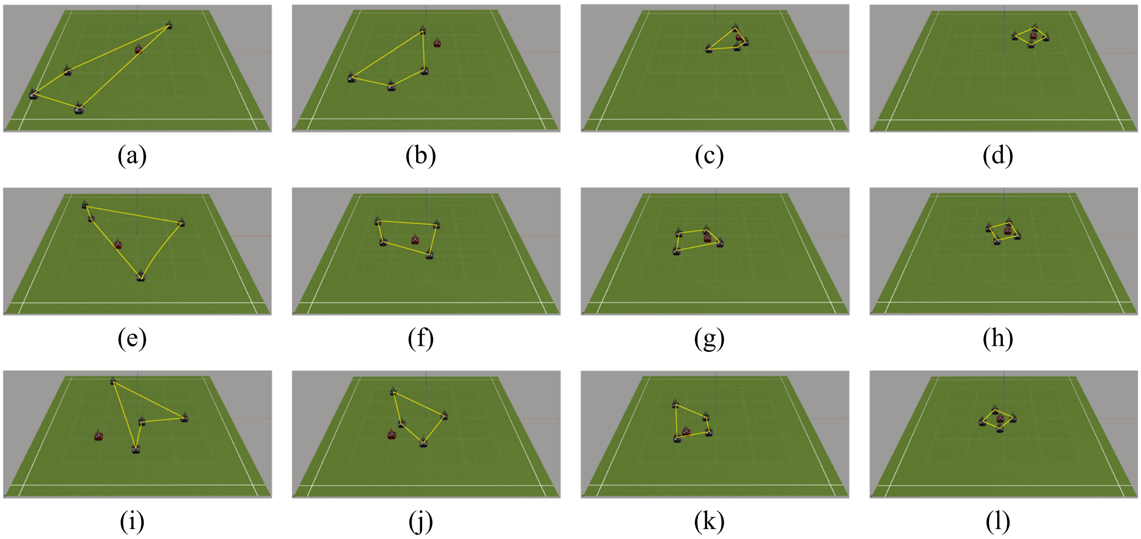


Fig. 11 Simulation II: Dynamic process of the robots trapping the target. The four pictures in each row represent an enclosing process, which shows the process of the robot's encirclement when the target moves randomly in the environment

robots and the target, which is more common in the real applications.

In the practical problem of multi-robot target encirclement, the robots and the target are holding the two sides of the zero-sum game, and their objectives are opposite. Robots need to jointly enclose the target, and they have a collaborative relationship with each other. On the other hand, when the robot is collaborating to enclose the target, the target will try to escape from being captured, showing the competitive relationship between the robots and the target. In order to verify the effectiveness of the algorithm for the practical target encirclement, the escape capability is added to the target in Simulation III.

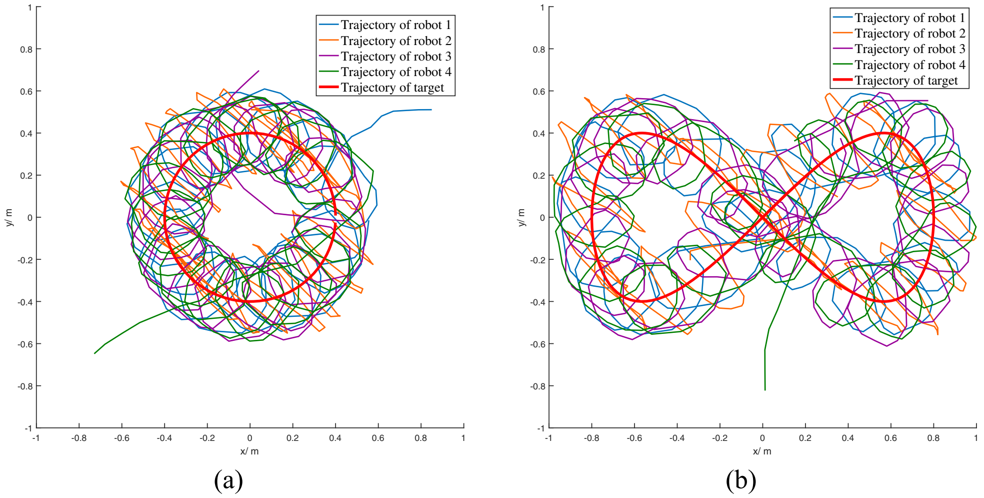
For the target, it is unfavorable to be trapped by robots, so a reasonable target will try to escape when it is being enclosed by other robots. Such an intellectual target is designed in this simulation according to the multi-agent Markov Game theory introduced in Section 3.2. During the training process, as the robots are gradually optimizing their policies of encirclement, the target is also trained and simultaneously optimizing its escape policy.

In order to evolve the target's policy, we designed two simple reward functions for the target training: (1) a collision avoidance reward (punishment) as described in Eq. 16; (2) a simple escape reward function r^e based on the distances with the robots:

$$r^e = k_5 \sum_{i}^{n} \parallel \mathbf{p}_t - \mathbf{p}_i \parallel + \max(r^e) \tag{21}$$

where k_5 is the weight coefficient, and $\parallel \mathbf{p}_t - \mathbf{p}_i \parallel$ represents the distance between the target and robot r_i. Reward function (21) shows that when the robots are closer

Fig. 12 The trajectories of the robots and the target in Simulation II



to the target, the target will be punished if it stays within the "dangerous" distance instead of choosing action to escape; conversely, if the target escapes and stays away from any robot with a distance larger than the "dangerous" distance, it will be rewarded.

Correspondingly, we defined a reward function on the target state based on the velocity and position of the target, working with the four reward functions described in Section 4.2. For each robot r_i, the reward function r_i^t of the target state is defined as:

$$r_i^t = -k_6 \parallel \mathbf{p}_i - \mathbf{p}_t \parallel - k_7 \parallel \mathbf{v}_t \parallel \tag{22}$$

where k_6 and k_7 are the weight coefficients, and $\mathbf{v}_t$ is the speed vector of the target.

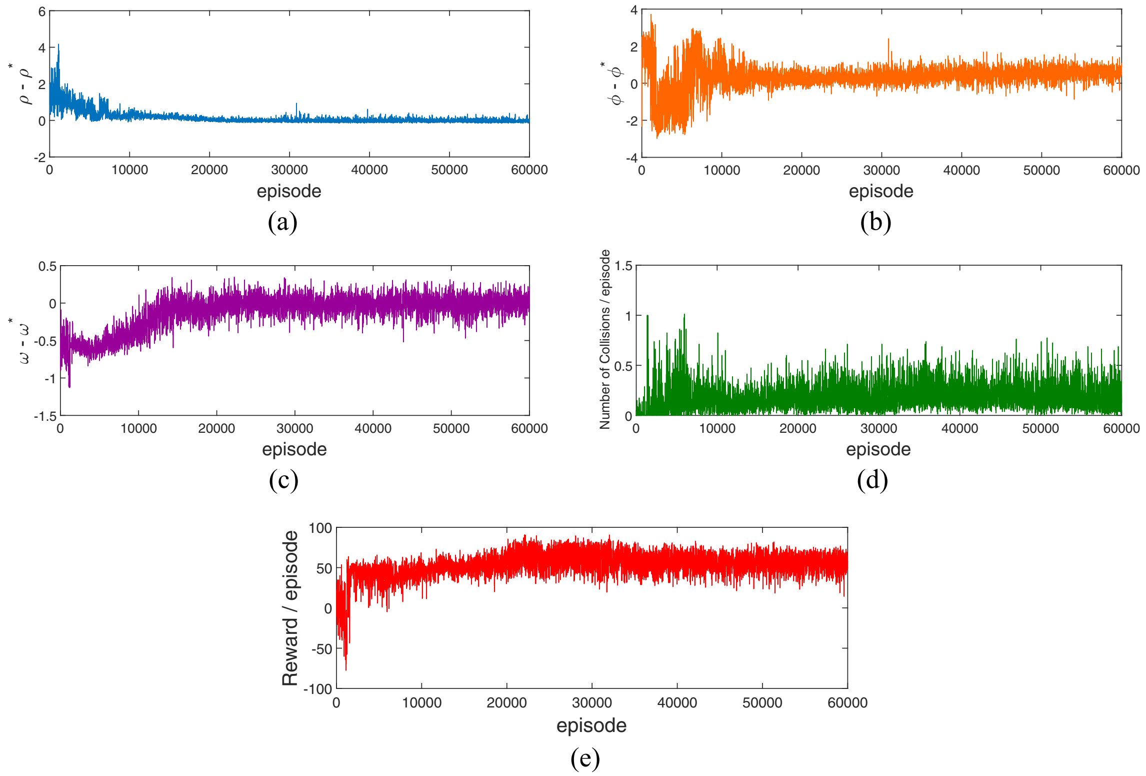
In this simulation, 60,000 episodes of training were performed on four robots and one target, and each episode consisted of a 60-step iterative process. Finally, the robots learned to cooperate with each other to form an encirclement formation and enclose the escaping target. During the entire training process, we recorded the error data of objectives for all robots in each episode, and the error data of robot 1 is shown in Fig. 13.

From the training data, when the target has the ability to escape, the encirclement of the robots becomes extremely difficult. Among the control objectives, only the errors of the encirclement radius $\rho - \rho^*$ and the error of phase difference between neighboring robots $\phi - \phi^*$ are reduced to a small range as the training proceeds. The error of angular velocity and the number of the collisions in each episode keeps oscillating along the training process.

Since the definition of angular velocity refers to the target as the center of the circle, the angular velocity of each robot changes drastically when the target is trying to escape. For the same reason, the final positions of the robots are not uniformly distributed around the target, but are affected by the velocity direction of the target, that is, in the positive direction of the velocity vector of the target, the phase difference between two neighboring robots is relatively larger; conversely, in the negative direction of the velocity vector of the target, the phase difference between two neighboring robots is smaller.

Even in this case, the cumulative reward of the robots still increases gradually from negative value to stable positive reward, which means that the robots have gradually learned a high reward policy under the reward and punishment mechanism.

After training, the robots have learned to form an encirclement formation and move around the target with the desired angular speed. Figure 14 shows the dynamic encircling process of the trained robots in simulation environment.



(a)

(b)

(c)

(d)

(e)

Fig. 13 Error curves of objectives in the training of Simulation III. a-c are the error curves for $\rho - \rho^*$, $\phi - \phi^*$, $\omega - \omega^*$ respectively, d plots the number of the collision with other robots or obstacles in each episode, and e represents the cumulative reward in each episode

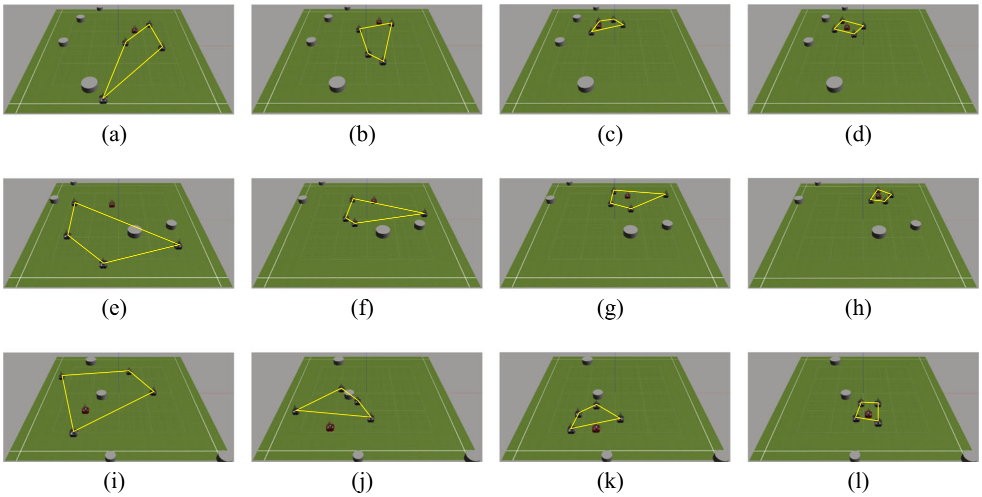


Fig. 14 Simulation III: Dynamic process of the robots trapping the target. The four pictures in each row represent an enclosing process, which shows the process of the robot's encirclement when the target escapes at different positions

As described in the figure, during the encirclement when the target is trying to escape, the number of collisions of the robot is increasing. It can be reasonably inferred, based on the dynamic process, that the main reason for the increase in the number of collisions is twofold. Firstly, when the target is moving towards outside the circle to escape, the probability of the collision with the robots increases. Secondly, with the advancement of the training process, the target's escape strategy is evolving as well. An interesting phenomenon is that these obstacles in the environment become "shelters" for the target to avoid being enclosed. The target learned a wise strategy that moving toward obstacles make it more difficult to be enclosed by the robots, thus the probability of the collision between the robots and the obstacle increased. When the target moves toward a place with obstacles, if the robots are still in the encirclement state and do not change the speeds in time, they may collide with the obstacle.

The result of Simulation III shows that the robots have basically learned the policy of enclosing the target, and the target has also learned an appropriate escape policy when facing being enclosed.

In contrast, the results of Simulation II and Simulation III demonstrate the benefits of the prioritization in multiple reward functions, such that these errors in the control objectives converge to a small range. Finally, the results of Simulation III prove that our algorithm is also effective in the occasion where there are both cooperative and competitive game.

In summary, the deep reinforcement learning framework enables multiple robots to learn to coordinate and enclose the target. By combining the advantages of neural networks and deterministic policy gradient algorithms, we propose this framework for multi-robot formation control problems. The contribution of our work is to use the machine learning framework to achieve the multi-robot cooperation control without accurate modeling and complex derivation.

6 Conclusion and Future Work

In this work, we propose a reinforcement learning framework that combines the advantages of the deterministic policy gradient algorithm and neural networks to allow multiple robots to collaborate in completing the target encirclement in continuous space. Our algorithm is not limited by the complicated control model deduction which is usually necessary for most traditional solutions.

With this algorithm, each robot has its own separate training network and the behavioral output at each time step is determined by independent policy. In the training or testing process, there is no global central control network in the whole system, so our algorithm can be considered to be distributed.

There are still several points worth improving in our work:

- The convergence performance of the algorithm is flawed. It can be seen from the error curves in the training process that these errors still slightly oscillated in a small range till the end of the training, which indicates that the robots' policies still have small randomness. Therefore, in further research, it is necessary to reduce the randomness of exploration to make the policy converge stably at the later stage of the training.
- We use the simulation model in the training, and ideally simulate a distributed control scheme based on local information. However, the practical situation is much more complicated. So there is lots of work to do to apply the algorithm to physical platforms in real environments.

- Our algorithm solves the formation control problem of multi-robots in 2D finite continuous space, but further research on highly nonlinear and high-dimensional motion space platforms such as UAV and UGV is needed.

References

1. Aguilar-Ponce, R., Kumar, A., Tecpanecatl-Xihuitl, J.L., Bayoumi, M.: A network of sensor-based framework for automated visual surveillance. J. Netw. Comput. Appl. **30**(3), 1244–1271 (2007)
2. Arleo, A., Smeraldi, F., Gerstner, W.: Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning. IEEE Trans. Neural Netw. **15**(3), 639–652 (2004)
3. Bustamante, A.L., Molina, J.M., Patricio, M.A.: A practical approach for active camera coordination based on a fusion-driven multi-agent system. Int. J. Syst. Sci. **45**(4), 741–755 (2014)
4. Chen, F., Ren, W., Cao, Y.: Surrounding control in cooperative agent networks. Syst. Control Lett. **59**(11), 704–712 (2010)
5. Degris, T., White, M., Sutton, R.S.: Off-policy actor-critic. In 29th International Conference on Machine Learning (2012)
6. Farinelli, A., Iocchi, L., Nardi, D.: Distributed on-line dynamic task assignment for multi-robot patrolling. Auton. Robot. **41**(6), 1–25 (2016)
7. Finke, J., Passino, K.M., Sparks, A.G.: Stable task load balancing strategies for cooperative control of networked autonomous air vehicles. IEEE Trans. Control Syst. Technol. **14**(5), 789–803 (2006)
8. Foerster, J., Assael, I.A., de Freitas, N., Whiteson, S.: Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In: Advances in Neural Information Processing Systems, pp. 2137–2145 (2016)
9. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. arXiv:1705.08926 (2017)
10. Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H., Kohli, P., Whiteson, S.: Stabilising experience replay for deep multi-agent reinforcement learning. arXiv:1702.08887 (2017)
11. Franchi, A., Stegagno, P., Oriolo, G.: Decentralized multi-robot encirclement of a 3d target with guaranteed collision avoidance. Auton. Robot. **40**(2), 1–21 (2016)
12. Hafez, A.T., Iskandarani, M., Givigi, S.N., Yousefi, S., Beaulieu, A.: Uavs in formation and dynamic encirclement via model predictive control. IFAC Proc. **47**(3), 1241–1246 (2014)
13. Hausman, K., Mueller, J., Hariharan, A.: Cooperative multi-robot control for target tracking with onboard sensing. Int. J. Robot. Res. **34**, (2015)
14. He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., Ma, W.Y.: Dual Learning for Machine Translation. In: Advances in Neural Information Processing Systems, pp. 820–828 (2016)
15. He, H., Boyd-Graber, J., Kwok, K., Daumé, H., III: Opponent Modeling in Deep Reinforcement Learning. In: International Conference on Machine Learning, pp. 1804–1813 (2016)
16. Iida, S., Kanoh, M., Kato, S., Itoh, H.: Reinforcement Learning for Motion Control of Humanoid Robots. In: Ieee/Rsj International Conference on Intelligent Robots and Systems, vol.4, pp. 3153–3157 (2004)
17. Kim, T., Hara, S., Hori, Y.: Cooperative control of multi-agent dynamical systems in target-enclosing operations using cyclic pursuit strategy. Int. J. Control. **83**(10), 2040–2052 (2010)
18. Lan, Y., Yan, G., Lin, Z.: Distributed control of cooperative target enclosing based on reachability and invariance analysis. Syst. Control Lett. **59**(7), 381–389 (2010)
19. Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 464–473. International Foundation for Autonomous Agents and Multiagent Systems (2017)
20. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015)
21. Liu, L., Luo, C., Shen, F.: Multi-Agent Formation Control with Target Tracking and Navigation. In: IEEE International Conference on Information and Automation (2017)
22. Long, P., Fanl, T., Liao, X., Liu, W., Zhang, H., Pan, J.: Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6252–6259. IEEE (2018)
23. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
24. Macwan, A., Vilela, J., Nejat, G., Benhabib, B.: A multirobot path-planning strategy for autonomous wilderness search and rescue. IEEE Trans. Cybern. **45**(9), 1784–1797 (2015)
25. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
26. Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. arXiv:1703.06182 (2017)
27. Parrish, J.K., Viscido, S.V., Grunbaum, D.: Self-organized fish schools: an examination of emergent properties. Biol. Bullet. **202**(3), 296–305 (2002)
28. Sarwal, A., Agrawal, D., Chaudhary, S.: Surveillance in an Open Environment by Co-Operative Tracking Amongst Sensor Enabled Robots. In: 2007. ICIA'07. International Conference On Information Acquisition, pp. 345–349. IEEE (2007)
29. Sato, K., Maeda, N.: Target-Enclosing Strategies for Multi-Agent Using Adaptive Control Strategy. In: IEEE International Conference on Control Applications, pp. 1761–1766 (2010)
30. Shi, Y.J., Li, R., Teo, K.L.: Rotary enclosing control of second-order multi-agent systems for a group of targets. Int. J. Syst. Sci. **48**, 13–21 (2017)
31. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484 (2016)
32. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic Policy Gradient Algorithms. Proceedings of the 31st International Conference on Machine Learning (ICML-14), pp. 387–395 (2014)
33. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
34. Su, P.H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.H., Young, S.: On-line active reward learning for policy optimisation in spoken dialogue systems. arXiv:1605.07669 (2016)
35. Sukhbaatar, S., Fergus, R., et al.: Learning Multiagent Communication with Backpropagation. In: Advances in Neural Information Processing Systems, pp. 2244–2252 (2016)

36. Sutton, R., Barto, A.: Reinforcement Learning:An introduction. MIT Press, Cambridge (1998)
37. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning, vol. 12 (2017)
38. Wang, C., Xie, G., Cao, M.: Forming circle formations of anonymous mobile agents with order preservation. IEEE Trans. Autom. Control **58**(12), 3248–3254 (2013)
39. Wang, C., Xie, G., Cao, M.: Controlling anonymous mobile agents with unidirectional locomotion to form formations on a circle. Automatica **50**(4), 1100–1108 (2014)
40. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv:1511.06581 (2015)
41. Xiao, J., Xiong, D., Yao, W., Yu, Q., Lu, H., Zheng, Z.: Building Software System and Simulation Environment for RoboCup MSL Soccer Robots Based on ROS and Gazebo. Springer International Publishing (2017)
42. Yao, W., Lu, H., Zeng, Z., Xiao, J., Zheng, Z.: Distributed static and dynamic circumnavigation control with arbitrary spacings for a heterogeneous multi-robot system. Journal of Intelligent & Robotic Systems **94**, 883–905 (2019)
43. Zhang, Y., Parker, L.E.: Multi-Robot Task Scheduling. In: IEEE International Conference on Robotics and Automation, pp. 2992–2998 (2016)
44. Zheng, Y., Luo, S., Lv, Z.: Control Double Inverted Pendulum by Reinforcement Learning with Double Cmac Network. In: International Conference on Pattern Recognition, pp. 639–642 (2006)

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

Junchong Ma received his Bachelors of Engineering (2016) and Masters of Engineering (2018) from the National University of Defense Technology (NUDT), China. The focus of his research lies on multi-robot coordination and robot soccer.

Huimin Lu received his Bachelors of Engineering (2003), Masters of Engineering (2005) and Ph.D. (2010) from the National University of Defense Technology (NUDT), China. Later, he joined the Department of Automation, NUDT (2010) where he is an associate professor. The focus of his research is on mobile robotics, especially on robot vision, multi-robot coordination, robot soccer and robot rescue.

Junhao Xiao received his Bachelors of Engineering (2007) from the National University of Defense Technology (NUDT), China, and Ph.D. (2013) from the Institute of Technical Aspects of Multimodal Systems (TAMS), Department Informatics, University of Hamburg, Germany. Later, he joined the Department of Automation, NUDT (2013) where he is an associate professor. The focus of his research is on mobile robotics, especially on localization, mapping, robot soccer and robot rescue.

Zhiwen Zeng received his Bachelors of Engineering (2009) from the University of Electronic Science and Technology of China, Masters of Engineering (2011) and Ph.D. (2016) from the National University of Defense Technology (NUDT), China. Later, he joined the Department of Automation, NUDT (2016) where he is an assistant professor. The focus of his research is on mobile robotics, especially on multi-robot coordination, and robot control.

Zhiqiang Zheng received his Ph.D. (1994) from University of Liege, Belgium. Later, he joined the Department of Automation, National University of Defense Technology (NUDT), China, where he is a professor. The focus of his research is on mobile robotics and flight control, especially on multi-robot coordination and robot vision.