Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning

Ruilong Zhang, Qun Zong, *Member, IEEE*, Xiuyun Zhang, Liqian Dou, and Bailing Tian, *Member, IEEE*

***Abstract*— As one of the tiniest flying objects, unmanned aerial vehicles (UAVs) are often expanded as the "swarm" to execute missions. In this article, we investigate the multiquadcopter and target pursuit-evasion game in the obstacles environment. For high-quality simulation of the urban environment, we propose the pursuit-evasion scenario (PES) framework to create the environment with a physics engine, which enables quadcopter agents to take actions and interact with the environment. On this basis, we construct multiagent coronal bidirectionally coordinated with target prediction network (CBC-TP Net) with a vectorized extension of multiagent deep deterministic policy gradient (MADDPG) formulation to ensure the effectiveness of the damaged "swarm" system in pursuit-evasion mission. Unlike traditional reinforcement learning, we design a target prediction network (TP Net) innovatively in the common framework to imitate the way of human thinking: situation prediction is always before decision-making. The experiments of the pursuit-evasion game are conducted to verify the state-of-the-art performance of the proposed strategy, both in the normal and antidamaged situations.**

***Index Terms*— Multiagent reinforcement learning, multi-quadcopter motion planning, pursuit-evasion game, trajectory prediction.**

I. Introduction

COMPETITION and cooperation are the two essential elements for lots of complex ecosystems in nature, just like a pack of wolves hunting the sheep on the prairie. They are deemed as the driving forces of biological and social evolution. With the characteristics of lightweight, high maneuverability, and low cost, it is hard for a single quadcopter to accomplish complex missions alone. Therefore, the quadcopters have been derived "unmanned swarm system" application mode, which contains cooperation among partners and competition with enemies.

Manuscript received 28 January 2021; revised 14 July 2021 and 9 October 2021; accepted 24 January 2022. Date of publication 14 February 2022; date of current version 6 October 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102401; and in part by the National Natural Science Foundation of China under Grant 62073234, Grant 62003236, Grant 61873340, and Grant 62022060. *(Corresponding author: Xiuyun Zhang.)*

The authors are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: zhangrl1107@tju.edu.cn; zongqun@tju.edu.cn; zxy_11@tju.edu.cn; douliqian@tju.edu.cn; bailing_tian@tju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3146976.

Digital Object Identifier 10.1109/TNNLS.2022.3146976

One of the typical applications for multiquadcopter is target pursuit with obstacles avoidance [1]. The main idea of the traditional method is to continuously reduce the set of points that the evader can safely reach before being captured by the pursuer [2], [3] and adopt particle swarm optimization (PSO) [4], artificial potential fields [5], and other optimization methods to address the problem of collision and obstacle avoidance. However, most of the existing traditional methods ignored the physical size of the pursuer in addressing the obstacle avoidance and assumed that the pursuer's velocity is faster than the target's one.

With the rapid development of artificial intelligence (AI), the last decade has witnessed massive progress in deep reinforcement learning (DRL). From single AI unit (Aka-agent) defeated humans in various games including Atari video games [6] and DeepMind's Alphago beat a professional player in Go game [7] to Alphastar defeated a top professional player in the real-time strategy game StarCraft [8], DRL's excellent environmental awareness and decision control capacity have been clearly revealed. As an intelligent learning method, DRL is very suitable for online trajectory planning [9], [10]. A hierarchical approach that combines a model-free policy gradient method with a conventional feedback proportional–integral–derivative (PID) controller [11] is developed for autonomous target following. An actor-critic model-based [12] asynchronous advantage actor-critic (A3C) [13] for ground robots to target-driven visual navigation in indoor scenes was proposed.

In recent years, many reinforcement learning methods have emerged [14], such as counterfactual multi-agent (COMA) [15], CommNet [16], and multiple actor attention critic (MAAC) [17]. Multiagent deep deterministic policy gradient (MADDPG) [18] is one of the most classical multiagent DRL algorithms, which can train a centralized critic per agent to give all agents' policies during training to reduce the variance by removing the nonstationarity caused by the concurrently learning agents [19]. Various extension approaches have been proposed in recent years, for example, memory-driven MADDPG (MD-MADDPG) [20], using a shared memory as a means to multiagent communication; multiagent deep deterministic policy gradient with message dropout (MADDPG-MD) [21], extending dropout technique to robust communication in multiagent scenarios with direct communication; and minimax multiagent deep deterministic policy gradients (M3DDPGs) [22], extending MADDPG with

minimax objective to robustify the learned policy. However, a few approaches focus on designing scalable network structures applicable for the collaboration of the damaged "swarm" system (damaged quadcopter unit can cause the change in the number of "swarm" system during mission execution).

It can be seen that the combination of unmanned aerial vehicles (UAVs) and AI is the focus of future research. At present, many reinforcement learning methods have been applied for multi-UAV communication [23], [24], task allocation [25], and formation control [26]. However, lots of research work have been done for a group of dynamic agents to achieve objectives in the nonobstacle environment [27] or a single quadcopter agent [28], [29] to pursue the target in few obstacles environment. Type-2 Takagi–Sugeno–Kang fuzzy logic controllers (TSK-FLCs) [27] tuned by reinforcement learning implemented on quadcopters to pursue target in an open environment. An improved deep deterministic policy gradient (DDPG) [28] is employed to enable a single UAV to complete the path planning in an obstacles environment, and the author used DDPG and designed a reward function based on artificial potential field [29] to achieve target tracking and obstacle avoidance. However, the target pursuit for multiple collaborative quadcopters in a complex multiobstacle environment is still open.

In addition, prediction before the decision is an instinctive way of intelligent creatures thinking [30]. It is of immediate significance that quadcopter agents can possess the ability to predict the future position of the target, just like the police who pursue a thief will anticipate the thief's movements and surround the thief with a shortcut. A specific recurrent neural network (RNN) model [31] is presented for the trajectory prediction of UAVs. A constrained long short-term memory (LSTM) network [32] is proposed to achieve 4-D flight trajectory prediction. However, the target prediction method is rarely used in the field of reinforcement learning. In this article, we propose a target prediction network (TP Net) within a traditional DRL framework to assist agents' policy decision-making.

Moreover, training and evaluating DRL algorithms in real environments are often complex and unpractical, especially for the scenario of multiquadcopters pursuing a target. One reason is that running systems in a physical space is time and cost consuming. Furthermore, acquiring large-scale action and state data in real environments is not trivial via common airborne sensors. Most of the existing simulation frameworks related to the pursuit-evasion game do not consider obstacles in the environment [27], [33], and they regard UAVs as points that cannot guarantee collision avoidance if the physical size of UAVs is considered [29]. Moreover, these work simply obstacles to black squares on blank backgrounds that cannot reflect the reality of the urban environment.

Motivated by the above observations, similar to AI2-THOR [12], multiagent particle environment [34], and PySC2 [35], we develop a simulation framework with high-quality 3-D scenes to simulate the pursuit-evasion game based on the Unity 3-D physics engine, called pursuit-evasion scenario (PES). Our simulation framework enables us to collect a large number of state values of quadcopter agents' actions and display the agents' motions intuitively. Furthermore, we investigate the multiagent coronal bidirectionally coordinated with target prediction network (CBC-TP Net) for a group of quadcopter agents to pursue a moving target in this article. Compared with the existing work, the main contributions are summarized as follows. First, we create a simulation framework PES with high-quality rendering, which enables real-time interactions for quadcopter agents with the multiobstacle environment, to simulate the process of quadcopters executing the mission in the urban environment more realistically. Second, we extend the prediction network within a traditional DRL framework to improve the policy decision capacity of quadcopter agents by predicting the target future trajectory. Third, we propose an improved DRL framework CBC-TP Net based on the centralized training with decentralized execution mechanism of MADDPG to generate cooperative action policy for both undamaged and damaged "swarm" systems.

The outline of this article is given as follows. Section II introduces the PES framework and rules of the game. In Section III, a novel reinforcement learning model CBC-TP Net for PES is developed, including TP Net, critic, and policy network architecture. The hierarchical multiagent gradient-descent algorithm for CBC-TP Net updating and a global-individual reward function are given in Section IV. The effectiveness of the proposed strategy is verified in Section V by experiments. Conclusion and future work are given in Section VI.

II. Pursuit-Evasion Scenario

In this section, we propose the PES in the urban environment. Then, we describe the pursuit-evasion problem constraint and objective.

A. Pursuit-Evasion Interaction Environment

To train and evaluate our multiagents DRL model, we propose PES in the urban setting, which is designed by integrating a physics engine (Unity 3-D) with a DRL framework based on Pytorch [36], communicating by Unity ML-Agents Toolkit [37], as shown in Fig. 1. The general idea is that quadcopter states operated by the physics engine embedded with the dynamic model are streamed to the DRL framework. Then, the DRL framework issues an action command accordingly and sends it back to the physics engine. The main advantages of PES are as follows.

1) The DRL framework communicates with the physics engine directly and the physics engine can show the global states of agents in real time.
2) The scenario of a real environment can be imitated as closely as possible. For example, quadcopters will be damaged if they crash into an obstacle.
3) We can change the block layout by replacing a few parameters in the physics engine, which can conveniently generate a new urban environment.

B. Problem Statement

The goal of this article is to imitate the scenario that multiquadcopters pursue the target in the urban environment

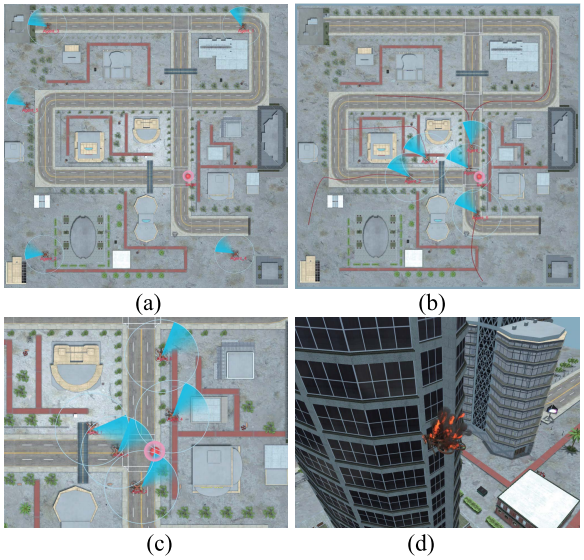


Fig. 1. PES framework is a rich interaction platform for AI agents. It enables physical interactions, such as moving or damaged by crashing into the obstacle. (a) Initial. (b) Process. (c) Mission success. (d) Crash into the building.

with multiobstacles. Based on this, the game rules of PES are given as follows.

Background: Multiquadcopter pursue single target in the urban environment.

Initial States: The quadcopter agents and target are randomly distributed in the edge and the center of the urban map, as shown in Fig. 1(a). The quadcopter agents are labeled as "Agent_1" to "Agent_N" and the moving target is labeled as "Target." The initial velocities of quadcopter agents and target are both zero.

Process: The forces generated by the DRL framework act on quadcopter agents to pursue the target while avoiding the obstacles (in this article, DDPG policy is used for the target to evade pursuit), as shown in Fig. 1(b) and (c).

Assumption: The multiquadcopter system is set to fly at a fixed altitude. To avoid the possible collisions between quadcopters, it is assumed that quadcopters fly at different altitudes.

Constraint: For each agent (including the target), real-time velocity and acceleration cannot exceed the set maximum values. To increase the difficulty of the game, we set that

$$\begin{aligned}\text{Max}(\text{velocity}_{\text{agent}}) &< \text{Max}(\text{velocity}_{\text{target}})\\ \text{Max}(\text{acceleration}_{\text{agent}}) &> \text{Max}(\text{acceleration}_{\text{target}}).\end{aligned}$$

Done: Three quadcopter agents pursue the target simultaneously, within a 2-m radius and last for five timesteps, as shown in Fig. 1(c).

C. Quadcopter Model

We embed the quadcopter model into the Unity 3-D physics engine to simulate the flight process of the quadcopter agent. Like all other aerial vehicles, the quadcopter has six-degree of freedom (DOF) related to translational and rotational motions. The dynamic equations for rotational motions are given as the simplified versions in [38]

$$\begin{aligned}\ddot{\phi} &= \frac{(I_{yy}-I_{zz})}{I_{xx}}\dot{\vartheta}\dot{\psi}+\frac{M_x}{I_{xx}}\\ \ddot{\vartheta} &= \frac{(I_{zz}-I_{xx})}{I_{yy}}\dot{\phi}\dot{\psi}+\frac{M_y}{I_{yy}}\\ \ddot{\psi} &= \frac{(I_{xx}-I_{yy})}{I_{zz}}\dot{\vartheta}\dot{\phi}+\frac{M_z}{I_{zz}}\end{aligned} \tag{1}$$

where ϕ, ϑ, and ψ are roll, pitch, and yaw angles, respectively; I_{xx}, I_{yy}, and I_{zz} denote the inertia moments of three axes; and M_x, M_y, and M_z are aerodynamic moments. The dynamic equations for translational motion coupling with rotational motion can be seen as in [38]

$$\begin{aligned}\ddot{x} &= (\sin(\psi)\sin(\phi)+\cos(\phi)\sin(\vartheta)\cos(\psi))\frac{F_z}{m}\\ \ddot{y} &= (-\cos(\psi)\sin(\phi)+\cos(\phi)\sin(\vartheta)\sin(\psi))\frac{F_z}{m}\\ \ddot{z} &= -g+(\cos(\phi)\cos(\vartheta))\frac{F_z}{m}\end{aligned} \tag{2}$$

where m is the quadcopter mass, x, y, and z are positions of the quadcopter, and F_z is the aerodynamic force in the vertical direction. In this article, considering the quadcopter flying at the fixed altitude, we define $\ddot{z} \equiv 0$. We use the DRL method to generate the desired accelerations $(\ddot{x}_{(\text{desired})}, \ddot{y}_{(\text{desired})})$ in the horizontal direction and use a PID controller to control the quadcopter to move.

III. Coronal Bidirectionally Coordinated With TP Net

A. Problem Formulation

In the PES, we need to control a group of quadcopter agents to pursue the target under multiobstacles conditions of the turban environment. The quadcopter agents are fully cooperative, and they compete with the target. Therefore, the pursuit-evasion process can be approximated as a stochastic Markov game, a multiagent extension of Markov decision processes (MDPs) [18], [35], [39]. A Markov game for N agents is defined by a set of states S describing the possible properties of all agents and a set of actions $A_1, \ldots, A_N$ and observations $O_1, \ldots, O_N$ for each agent as well. The actions chosen by each quadcopter agent i interact in the PES to produce the next state according to the state transition function, and $S \times A_1 \times A_2 \times \cdots \times A_N \mapsto S'$ denotes the transition from state S to the final state S'. Each quadcopter agent i can receive the individual observation correlated with the state O_i: $S \mapsto O_i$ and obtains rewards according to state and action r_i: $S \times A_i \mapsto R$. The goal of each quadcopter agent is to maximize its own total expected reward $R_i = \sum_{t=0}^{T=\infty} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ and T denote discount factor and the time horizon, respectively.

In order to solve the multiquadcopter collaborative control problem to compete with the target in the PES, we resort to DRL. DRL is an area of machine learning concerned with how the agent interacts with the environment through trial and error to take action to obtain the maximum cumulative reward [40]. The goal of the DRL model is to train an optimal

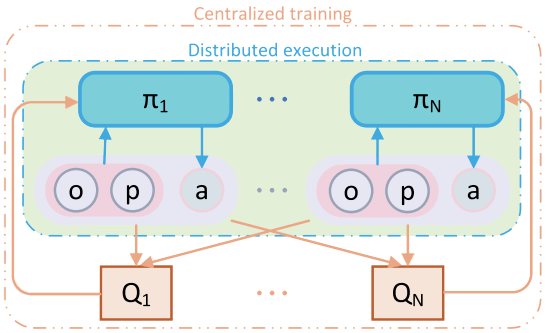


Fig. 2. Centralized training and distributed execution approach of CBC-TP Net.

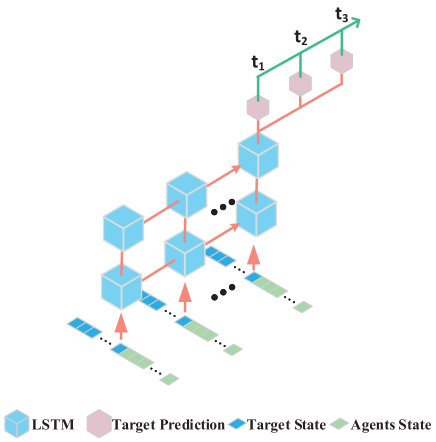


Fig. 3. Framework of TP Net. Target policy is influenced not only by its own state but also by quadcopter agents' state; therefore, we feed both target's and quadcopter agents' history states into TP Net to predict the future position of the target.

policy π_i, which defines the probability of choosing action a_i in observation o_i for each quadcopter agent, $\pi_i(a_i|o_i), a_i \in A_i, o_i \in O_i$, where a_i and o_i denote that the agent i chooses the action and obtains the observation at current timestep, respectively. Thus, the objective of N quadcopter agents, denoted by $J(\theta_i)$, is expressed as follows:

$$J(\theta_i) = \mathrm{E}\left[\sum_{t'=t}^{T=\infty} \gamma^{t'-t} r_{i,t'}(s_i, \pi_i(a_i|o_i))\right] \tag{3}$$

where θ denotes the parameter of the policy.

B. Multiagent Reinforcement Learning

Similar to [18], we achieve our goal by adopting the framework of centralized training with distributed execution, as shown in Fig. 2. Thus, we allow the policies to use coagent information to ease training, so long as this information is not used at test time. In addition, we expect that quadcopter agents possess the ability to predict the future position of the target. The private observation o_i of each quadcopter agent i can be updated by concatenating with target prediction p_{tar}: $(O_i + P_{\text{tar}}) \mapsto O_i$. Then, we can write the gradient of the expected return for agent i, $J(\theta_i)$, as

$$\nabla_{\theta_i} J(\theta_i) = \mathrm{E}\big[\nabla_{\theta_i} \pi_i(a_i|(o_i, p_{\text{tar}})) \cdot \nabla_{\pi_i} Q_i^{\pi}\left(o_{1,\ldots,N}, p_{\text{tar}}, a_{1,\ldots,N}\right)\big]. \tag{4}$$

The gradient of centralized action-value function Q_i^{π} can be obtained by using the sum of square loss

$$\nabla_{\varepsilon_i} L(\varepsilon_i) = \mathrm{E}\big[\left(Q_i^{\pi}\left(o_{1,\ldots,N}, p_{\text{tar}}, a_{1,\ldots,N}\right) - y\right) \cdot \nabla_{\varepsilon_i} Q_i^{\pi}\left(o_{1,\ldots,N}, p_{\text{tar}}, a_{1,\ldots,N}\right)\big] \tag{5}$$

where ε denotes the parameter of the critic network and y is approximated by temporal difference (TD) algorithms [41]

$$y = r_i + \gamma\, Q_i^{\pi'}\left(o'_{1,\ldots,N}, p'_{\text{tar}}, a'_{1,\ldots,N}\right) \tag{6}$$

where π' denotes the target policies with delayed parameters ε'_i.

C. Target Prediction Network

Forecasting the future positions of the target can help quadcopter agents make decisions more efficiently. RNN has one or more neural networks with feedback loops, which can solve the problem of memory in time series signal effectively [42]. Therefore, we propose a TP Net based on deep LSTM to predict the future k timesteps trajectory of the target, as shown in Fig. 3.

Consider that next k timesteps $(t+k) \sim (t+k)$ positions of the target need to be predicted by using the n timesteps history target state data. For TP Net training, in the action exploration process of DRL, we store $(n+k)$ timesteps target state data into a separate target trajectory replay buffer, including n timesteps training data p^x (target position, quadcopter agents position, and velocity) and k timesteps label data p^y (target position). We train the TP Net with supervised learning. The TP Net P_ς can be updated by minimizing the loss

$$L(\varsigma) = \mathrm{E}\left[\left(p^y - P_\varsigma\left(p^x\right)\right)^2\right] \tag{7}$$

where ς denotes the parameter of the TP Net.

D. Policy and Critic Network

Although the work [18] proposed a principled mechanism to foster team-level collaboration, MADDPG is not applicable for the collaboration of variable numbers of agents, due to the limitation of fully connected network structure (i.e., the quadcopter agent may be damaged by a malfunction). In this article, we propose a novel CBC-TP Net, which can allow communications between variable numbers of agents.

Overall, CBC-TP Net consists of a multiagent policy network and a multiagent critic network for each individual agent, as shown in Fig. 4. Both the policy network and the critic network are based on the bidirectional RNN structure [43]. It is worth noting that the current quadcopter agent corresponds to the initial and the final cell of bidirectional LSTM simultaneously in order to make the network structure symmetric. The policy network, which takes in a current agent state and coagent relative state together, returns the action for the current quadcopter agent, and the input details of quadcopter agent i CBC-TP Net policy networks are shown in Table I. Depending on the advantages of the bidirectional LSTM structure, the

TABLE I
INPUT DETAILS OF QUADCOPTER AGENT i CBC-TP NET POLICY NETWORKS

Name	Input	Size
LSTM cell of agent i	[self position, self velocity, the nearest obstacle to agent i, target relative position to agent i]	1×8
LSTM cell of agent j ($j \neq i$)	[agent j relative position and velocity to agent i, the nearest obstacle to agent j, target relative position to agent j]	1×8
FC layer of Target Prediction	[target relative prediction position to agent i] in next 3 timesteps	1×6

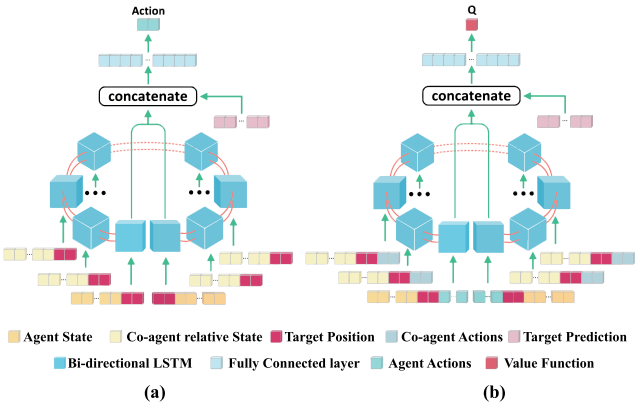


Fig. 4. Policy and critic network structures of CBC-TP Net. Quadcopter agent corresponds to the initial and the final bidirectional LSTM cell, and coagents correspond to other bidirectional LSTM cells. (a) CBC-TP Net policy networks. (b) CBC-TP Net Q networks.

policy and critic network can be expanded to fit variable numbers of agents.

IV. LEARNING METHOD FOR CBC-TP NET

In this article, we formulate the PES as the proposed CBC-TP Net training model. We propose a hierarchical multiagent gradient-descent method to train the model and design a global-individual reward function to promote the training process.

A. Multiagent Gradient-Descent Backpropagation

In this section, we propose a hierarchical multiagent DRL algorithm, which can extend the traditional training method of the critic-actor network to CBC-TP Net. In the macro view, we can think of computing the backward gradients of TP Net with supervised learning and actor-critic net in an unsupervised way. In the micro view, both TP Net and actor-critic net gradients apply backpropagation through time (BPTT) [44]. The gradients pass to both the Q function and the policy function that are aggregated from all the agents' state–action pairs. In other words, the gradient from the agent reward is influenced by each agent action and the resulting individual gradient is further backpropagate to updating the parameters of the network. The overall hierarchical multiagent DRL algorithm is presented in Algorithm 1.

We build two replay buffers for each quadcopter agent to update the TP, policy, and critic network efficiently. One is replay buffer $\mathcal{T}$, which collects the history state of quadcopter agents and history trajectory of the target to train the TP network. The other is replay buffer $\mathcal{R}$, which stores the elements of MDPs $(s^t, a, r, s^{t+1}, p^t_{\text{tar}})$ every timestep to update the policy and critic network. It is worth noting that the action a^t_i is decided by o^t_i, which is subset of s^t. To enhance the robustness and exploration ability of the proposed algorithm, we use Ornstein–Uhlenbeck noise (OUnoise) $\mathcal{N}$ for action exploration in training processing.

B. Reward Function

The reward function provides useful motivation for DRL agents training, which has a significant influence on the learning results. The goal of PES is to control the quadcopter agents to pursue the target while avoiding the obstacle. A potential drawback of only using the global reward or individual reward is that it ignores the fact that a team collaboration typically consists of individual collaborations [39]. Therefore, the reward function should include a team collaboration reward and individual reward. Moreover, in practice, each agent tends to have a common objective and its own objective (conforming to the common objective), which drives the collaboration. To model this, we propose a global-individual reward function for agent i as

$$r_i = r_{\text{co},i} + r_{\text{self},i} \tag{8}$$

where $r_{\text{co},i}$ and $r_{\text{self},i}$ denote global and individual reward, respectively, which are defined as

$$\begin{cases} r_{\text{co},i} = -\sum_{j=1}^{\text{num}} 0.1 \times \rho \times \text{dis}(j, \text{tar}) \\ r_{\text{self},i} = -(0.5 \times \rho \times \text{dis}(j, \text{tar}) + \log(\exp(0) \\ \qquad + \exp((\text{dis}_{\text{safe}} - \text{dis}(j, \text{obs}))/k)) \times k), \quad j = i \end{cases} \tag{9}$$

where $\text{dis}(j, \text{tar})$ is the distance from quadcopter agent j to the target and $\text{dis}(j, \text{obs})$ denotes the distance from quadcopter agent j to the nearest obstacle. We define dis_{safe} to be a safe buffer distance between quadcopter agents and the nearest obstacle, and ρ is a normalized factor to balance the reward into $(0, 1)$, and k is a constant parameter, which is set to 10^{-3} in our experiment. Generally speaking, the normalized factor ρ is necessary in the PES game. Without proper normalization, the policy network will have difficulty in converging, even trapped in local minima of action boundary. Apart from the basic pursuit-evasion process reward, some extra rewards are considered in our strategy, as the intrinsic motivation and external restrictions to speed up the training process. When a

Algorithm 1 Hierarchical Multiagent Gradient Descent

for episode = 1 to Max episodes (M) **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state s^1
  for $t = 1$ to Terminal timesteps (T) **do**
    $p^t{}_{tar} = (pos^{t-n}_{tar}, \ldots, pos^t_{tar})$ (n timesteps history positions of the target. If None, padding 0)
    For each agent i, choose action $a^t_i = \pi_i(o^t, P_\varsigma(p^t{}_{tar})) + \mathcal{N}_t$ for the current policy and exploration
    receive reward r_i, obtain new state s^{t+1} and target position pos^{t+1}_{tar} form o^{t+1}
    Store $(s^t, a, r, s^{t+1}, p^t_{tar})$ in replay buffer $\mathcal{R}$
    Store $p^x = [(pos^{t-(n+k)}_{tar}, s^{t-(n+k)}), \ldots, (pos^{t-k}_{tar}, s^{t-k})]$ and $p^y = (pos^{t-k+1}_{tar}, \ldots, pos^t_{tar})$ into target trajectory replay buffer $\mathcal{T}$
    $s^t \leftarrow s^{t+1}$
    for agent $i = 1$ to Number of agents (N) **do**
      Sample a random minibatch of J samples (p^x_j, p^y_j) from $\mathcal{T}$
      Update TP Net by minimizing the loss Eq. 7 $L(\varsigma) = \mathrm{E}[(p^y_j - P_\varsigma(p^x_j))^2]$
      Sample a random minibatch of M samples $(s^t_m, a_m, r_m, s^{t+1}_m, p^t_m)$ from $\mathcal{R}$
      Set $y_{m,i} = r_{m,i} + \gamma\, Q_{m,i}{}^{\pi'}(o'_{1,\ldots,n}, p'{}_{tar}, a'_{1,\ldots,n})$
      compute critic gradient estimation according to Eq. 5:
      $\nabla_{\varepsilon_i} L(\varepsilon_i) = \frac{1}{M}\sum_{m=1}^{M} \mathrm{E}[(Q^\pi_i(o_{1,\ldots,N}, p_{tar}, a_{1,\ldots,N}) - y_{m,i}) \cdot \nabla_{\varepsilon_i} Q^\pi_i(o_{1,\ldots,N}, p_{tar}, a_{1,\ldots,N})]$
      compute actor gradient estimation according to Eq. 4:
      $\nabla_{\theta_i} J(\theta_i) = \frac{1}{M}\sum_{m=1}^{M} \mathrm{E}[\nabla_{\theta_i}\pi_i(a_i|(o_i, p_{tar})) \cdot \nabla_{\pi_i} Q^\pi_i(o_{1,\ldots,N}, p_{tar}, a_{1,\ldots,N})]$
    end for
    Update target network parameters for each agent i:
    $\theta_i{}' \leftarrow \tau\theta_i + (1-\tau)\theta_i{}'$, $\varepsilon_i{}' \leftarrow \tau\varepsilon_i + (1-\tau)\varepsilon_i{}'$
  end for
end for

quadcopter agent crash into the obstacle, we introduce an extra negative reward and set it to -2 in our experiment. However, we would like to let the damaged quadcopter agent continue the mission in the training process to obtain more replay buffer data. Besides, in order to encourage our quadcopter agents to accomplish the mission more efficiently, we give all quadcopter agents a big positive reward, which is set to 5.

V. Results and Discussion

In this section, we compare the results in different DRL methods and discuss our CBC-TP Net performance. It is mentioned in Section II that the mission successful condition is at least three quadcopter agents to pursue the target simultaneously while considering the configuration of the simulation computer; therefore, without loss of generality, we choose five quadcopter agents for the simulation experiment, i.e., $N = 5$. First, we introduce the experience setting of the simulation. Then, we analyze the results in PES and discuss the CBC-TP Net model's performance by comparing other classical DRL models. Moreover, we demonstrate the effect of the TP Net by comparing CBC Net. Finally, we introduce the antidamage performance of CBC-TP Net in the PES where part of the quadcopter is damaged. For better comprehension, we analyze the mission successful rates, mean episode steps and rewards during training, and the learned strategies.

A. Experiment Settings

In the training process, we set the discount factor γ to 0.95, the learning rate of prediction, policy and critic network to 1×10^{-3}, and $\tau = 1 \times 10^{-2}$ for soft updating the target networks, the timesteps of history state $n = 10$. The size of the replay buffer is 1×10^6 and we update the network parameters after every 50 added to the replay buffer. The batch size of updating is 512. Moreover, the maximum steps of each episode are set to 500. The experiment is applied on a computer with an Intel Xeon Silver 4114 CPU, 128 GB of memory, and Nvidia GTX 2080Ti GPU. The parameters of CBC-TP Net are shown in Table II. The technical specifications of the quadcopters are chosen as in Table III.

TABLE II
BCB-TP Net Configurations

Name	Parameter
TP Net hidden units (LSTM)	256
TP Net hidden layers (LSTM)	2
Policy network hidden units (LSTM)	256
Policy network hidden layers (LSTM)	1
Policy network hidden units (MLP)	64
Critic network hidden units (LSTM)	256
Critic network hidden layers (LSTM)	1
Critic network hidden units (MLP)	64
CBC-TP Net activation function (MLP)	ReLU

B. Comparison to Other Reinforcement Learning Methods

In five quadcopters pursuing one target scenario, we implement our CBC-TP Net and evaluate it on the environments presented in Section II. The simulation rendering is shown

TABLE III
QUADCOPTER SPECIFICATIONS

Feature	Description	Pursuer	Evader
m	Mass (kg)	0.8	1
V	Max velocities (m/s)	10	12
A	Max acceleration (m/s^2)	5	4
I_{xx}		0.006	0.006
I_{yy}	Inertia moments (kgm^2)	0.006	0.006
I_{zz}		0.001	0.001

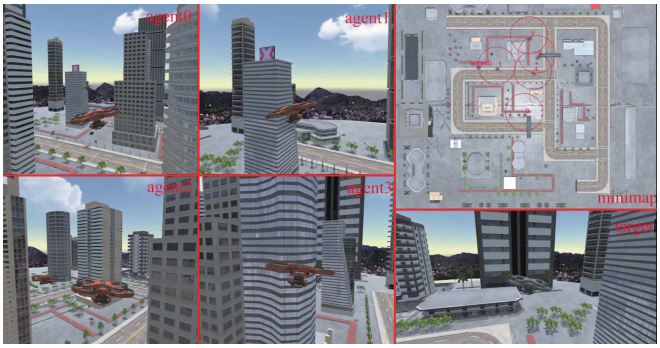


Fig. 5. Simulation rendering of PES.

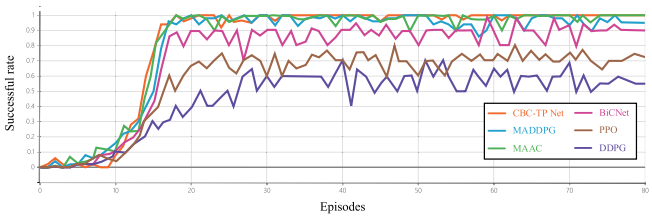


Fig. 6. Mission successful rate of CBC-TP Net and other algorithms in PES from every 100 episodes' training.

in Fig. 5. To evaluate the performance of policies learned in competitive settings, we pitch CBC-TP quadcopter agents against the DDPG target agent. We train our models with 4000 episodes and over 1 million steps and then evaluate them by averaging various metrics for 1000 further iterations. Meanwhile, we will also use PPO, DDPG, MADDPG, BiCNet, and MAAC [17], which is trained in the same environment, as a reference.

1) Mission Successful Rates: Mission successful rates play an important role in measuring the learning performance of the DRL model. To evaluate the mission successful rates, we test our model and MADDPG every 100 episodes and depict the results in Fig. 6. We can see that our quadcopter agents can hardly complete the mission before 500 episodes. With the progress of training, quadcopter agents start to pursue the target successfully, and the curve of win rates has an impressive increase after 800 episodes. After 2000 episodes' training, our quadcopter agents can almost complete pursuit mission 100%. Compared to the curve of MADDPG, our CBC-TP Net framework has outstanding and stable performances in PES.

2) Average Rewards: Generally speaking, a powerful intelligent quadcopter agent in PES should pursue the target while avoiding collision with obstacles. Here, we introduce the

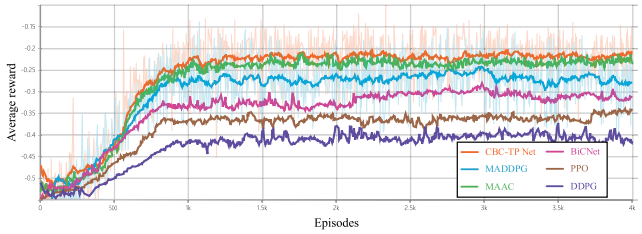


Fig. 7. Average reward of CBC-TP Net and other algorithms quadcopter agents in PES during training.

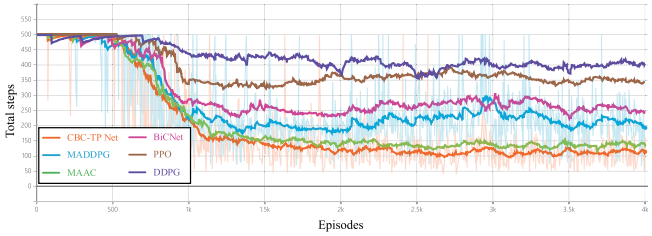


Fig. 8. Episode steps of CBC-TP Net and other algorithms quadcopter agents in PES during training.

average rewards, dividing the total rewards by episode steps in every episode. The curves of CBC-TP Net and MADDPG quadcopter agents' average rewards are shown in Fig. 7. The average rewards of both two DRL methods have an obvious increase in the beginning, grow steadily during training, and keep smooth after about 1000 episodes. However, the curve of CBC-TP Net quadcopter agents' average rewards has a sooner uptrend than MADDPG after about 500 episodes.

3) Episode Steps: We depict the average episode steps of CBC-TP Net and MADDPG quadcopter agents during training in Fig. 8. It is obvious that the curve of the average episode steps has four stages. In the beginning, episode steps are almost always 500 steps because quadcopter agents have learned nothing and explore an unknown environment. Then, quadcopter agents start to learn how to avoid the collision first in this process, and this is why the curve in Fig. 7 is growing but in Fig. 8 is keeping 500 steps. After that, quadcopter agents start to realize that they are far from the target causes a negative reward. They learn to close to the target, meanwhile avoiding the collision. In this phase, CBC-TP Net quadcopter agents pursue the target with fewer steps because the quadcopter agent can predict the trajectory of the target. In the end, quadcopter agents have learned an appropriate action policy to pursue the target without collision, and they are able to catch the target cooperatively in almost 100 to 150 steps.

4) Results Comparing: In addition, we use the PID controller with potential energy function as a baseline method and compare above the DRL algorithms in the evaluating progress. We introduce a 5% horizontal disturbance at stochastic timesteps to verify the robustness of each algorithm. In Table IV, we present the mission successful rate and mean episode steps of CBC-TP Net and baseline methods. We measure these models' performance in 1000 test episodes. We can see that the PID controller with potential energy function cannot complete the mission well because the target is faster

TABLE IV

PERFORMANCE COMPARISON OF CBC-TP NET WITH BASELINE METHOD

	PID(PF)	PPO	DDPG	MADDPG	BiCNet	MAAC	CBC-TP Net
Successful rate	27.3%	70.6%	56.8%	96.6%	90.6%	98%	98.8%
Episode steps	482	362	412	208	265	140	132
Average reward	−−	-0.35	-0.42	-0.28	-0.33	-0.23	-0.21

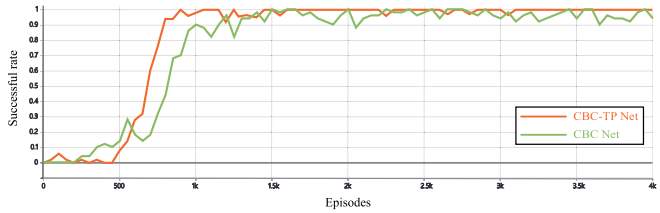


Fig. 9. Mission successful rate of CBC-TP Net and CBC-TP Net in PES from every 100 episodes' training.

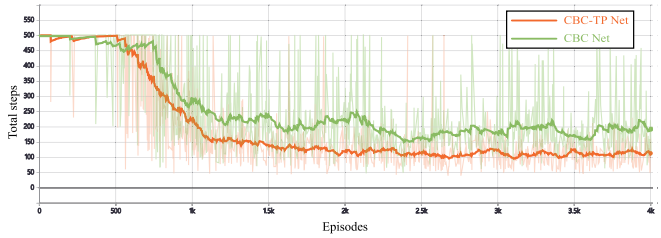


Fig. 10. Episode steps of CBC-TP Net and CBC-TP Net quadcopter agents in PES during training.

than the quadcopter agents and the quadcopter agents can only pursue the target but not round up. Through the comparison, our DRL framework can achieve a mission successful rate of 98.8%, which is much higher than other DRL method and need fewer steps to pursue the target.

C. Effect of TP Net

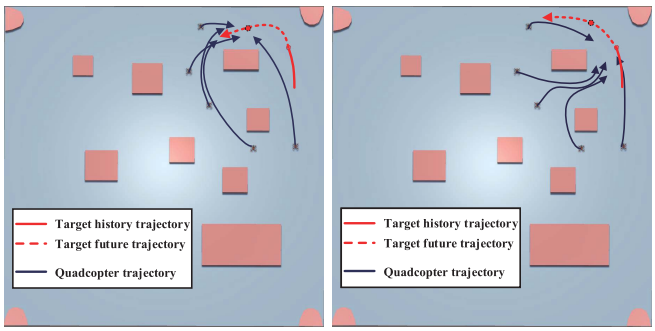
In this section, we compare CBC-TP Net and CBC Net in PES. In the comparison of prediction with none, we have a better understanding of TP Net. First, we compare final displacement error (FDE) and average displacement error (ADE) [45] of different prediction timesteps, as shown in Table V. We can find that we can obtain a target trajectory prediction result with acceptable accuracy when $k = 3$. We draw the mission successful rates and mean episode steps in Figs. 9 and 10. In Fig. 9, without target trajectory prediction, the mission successful rates are below about 40% in start 500–1000 episodes. In the latter part of the curve, CBC-TP Net quadcopter agents can keep a steady and high successful percentage, but the successful rate curve of CBC Net agents oscillates between 0.9 and 1. We can see that intuitively, CBC-TP Net quadcopter agents need less about 80 steps every episode after 3000 episodes than CBC Net in Fig. 10.

Here, we design a contrast experiment and make a brief analysis on TP Net, which makes our quadcopter agents more effective. We set that two PESs with the same initial condition and policy of target evasion are both DDPG with the same parameter, as shown in Fig. 11. Without TP Net,

TABLE V

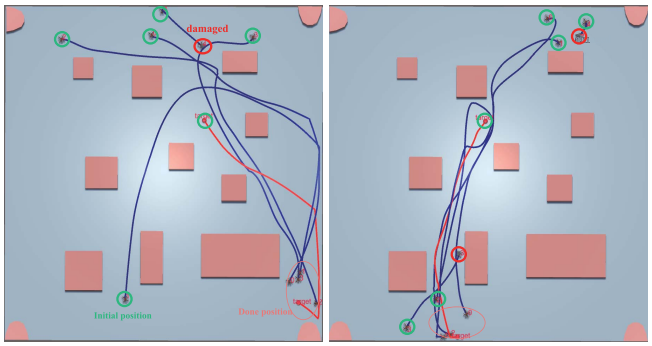
FDE AND ADE OF DIFFERENT PREDICTION TIMESTEPS (NORMALIZATION ERROR)

k	1	2	3	4	5
FDE	0.026	0.038	0.062	0.118	0.196
ADE	0.026	0.032	0.042	0.061	0.088



(a) (b)

Fig. 11. Comparison of CBC-TP Net quadcopter agents' trajectory and CBC Net quadcopter agents'. Just to make it clear, we depict the trajectory in the no rendering simulation environment. (a) CBC-TP Net. (b) CBC Net.



(a) (b)

Fig. 12. Pursuit trajectory of the rest while one or two quadcopter agents are damaged. (a) One quadcopter agent damaged. (b) Two quadcopter agents damaged.

CBC Net quadcopter agents generate action policy only based on the current state of the target. This action policy causes quadcopter agents to chase the target all the time but cannot complete the mission (the target is faster than the quadcopter agent). However, CBC-TP Net quadcopter agents can generate a current action policy according to the current and future state of the target. Therefore, CBC-TP Net quadcopter agents can learn how to besiege the target to complete the PES mission speedily.

D. Performance of Antidamage of Multiagent System

Antidamage of the multiagent system means that during the execution of the mission by multiagent, if part of agents is accidentally damaged and unable to continue the mission, the rest agents can continue to complete the mission without being affected. Benefitting from LSTM unique network characteristics (weight sharing and it can handle variable-length sequences), a well-trained CBC-TP Net agent can generate

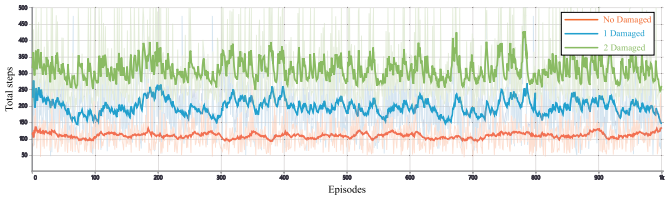


Fig. 13. Episode steps of the CBC-TP Net quadcopter agents with none, one, and two quadcopter agents damaged.

TABLE VI

AVERAGE EPISODE STEPS AND MISSION SUCCESSFUL RATE OF CBC-TP NET QUADCOPTER AGENTS WITH NONE, ONE, OR TWO DAMAGED AGENT

	No damaged	1 damaged	2 damaged
Successful rate	98.8%	95.6%	86.7%
Episode steps	132	212	308

an action policy by communicating with the variable number agents that are no damage. Fig. 12 shows the pursuit trajectory of the other multiquadcopter agents when one or two agents are damaged. The rest of the multiquadcopter agents can generate action decisions that are favorable to the completion of the mission and are almost not affected by damaged agents. In Fig. 13 and Table VI, we depict the average episode steps and mission successful rate of CBC-TP Net quadcopter agents with none, one, or two damaged agents. It is apparent to see that although the performance of the pursuit decreases with the increase of the number of damaged quadcopter agents, the rest is still able to carry out the mission efficiently.

VI. CONCLUSION AND FUTURE WORK

This article focuses on the pursuit-evasion game of multiquadcopter in the obstacles environment. In this article, we have presented several contributions, including: 1) we propose a DRL interaction environment for PES based on Unity 3-D and python; 2) we introduce a vectorized actor-critic framework CBC-TP Net, where each dimension corresponds to the existed quadcopter agent and extends a TP Net in the traditional reinforcement learning framework for target trajectory prediction; and 3) we design a hierarchical multiagent DRL algorithm for CBC-TP Net updating and an effective global-individual reward function for training. The effectiveness and the superior performance over other DRL methods have been verified by the simulation experiments. It is remarkable that the proposed method enables the rest of the quadcopter agents to continue the PES mission even though parts of the quadcopters are damaged.

In addition, there are still some areas for future work. In the next step, we plan to carry on experiments of sharing the policy network to extend to big-scale unmanned swarm combat missions. At present, we can train quadcopter agents in 2-D space and obtain the desired results, while training six-DOF multiquadcopter agents to attack more flexible targets using the DRL methods are still an open problem. We will improve our methods for 3-D space missions and more complex scenarios in the future. Finally, applying our DRL algorithm to the real UAV trajectory planning is also one of the focuses of future work.

ACKNOWLEDGMENT

The authors would like to thank Kangrui Zhu for establishing the urban scene in Unity 3-D and providing exquisite virtual quadcopter model.

REFERENCES

[1] S. A. P. Quintero, F. Papi, D. J. Klein, L. Chisci, and J. P. Hespanha, "Optimal UAV coordination for target tracking using dynamic programming," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 4541–4546.

[2] H. Huang, W. Zhang, J. Ding, D. M. Stipanović, and C. J. Tomlin, "Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers," in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.*, Dec. 2011, pp. 4835–4840.

[3] A. Pierson, Z. Wang, and M. Schwager, "Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 530–537, Apr. 2017.

[4] R. Palm and A. Bouguerra, "Particle swarm optimization of potential fields for obstacle avoidance," in *Proc. Recent Adv. Robot. Mech.*, 2013, pp. 117–123.

[5] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, 1992.

[6] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[8] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multiagent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[9] S. Yin, S. Zhao, Y. Zhao, and F. R. Yu, "Intelligent trajectory design in UAV-aided communications with reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8227–8231, Aug. 2019.

[10] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 4896–4901.

[11] S. Li, T. Liu, C. Zhang, D.-Y. Yeung, and S. Shen, "Learning unmanned aerial vehicle control for autonomous target following," 2017, *arXiv:1709.08233*.

[12] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.

[13] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[14] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[15] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2017, *arXiv:1705.08926*.

[16] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[17] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[18] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6379–6390.

[19] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Oct. 2019.

[20] E. Pesce and G. Montana, "Improving coordination in small-scale multiagent deep reinforcement learning through memory-driven communication," 2019, *arXiv:1901.03887*.

[21] W. Kim, M. Cho, and Y. Sung, "Message-dropout: An efficient training method for multi-agent deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 6079–6086.

[22] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4213–4220.
[23] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.
[24] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.
[25] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerosp. Sci. Technol.*, vol. 92, pp. 588–594, Sep. 2019.
[26] L. Dou, X. Su, X. Zhao, Q. Zong, and L. He, "Robust tracking control of quadrotor via on-policy adaptive dynamic programming," *Int. J. Robust Nonlinear Control*, vol. 31, no. 7, pp. 2509–2525, May 2021.
[27] E. Camci and E. Kayacan, "Game of drones: UAV pursuit-evasion game with type-2 fuzzy logic controllers tuned by reinforcement learning," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2016, pp. 618–625.
[28] Y. Li, S. Zhang, F. Ye, T. Jiang, and Y. Li, "A UAV path planning method based on deep reinforcement learning," in *Proc. IEEE USNC-CNC-URSI North Amer. Radio Sci. Meeting, Joint AP-S Symp.*, Jul. 2020, pp. 93–94.
[29] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.
[30] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Netw.*, vol. 22, no. 3, pp. 200–212, 2009.
[31] K. Xiao, J. Zhao, Y. He, and S. Yu, "Trajectory prediction of UAV in smart city using recurrent neural networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[32] Z. Shi, M. Xu, and Q. Pan, "4-D flight trajectory prediction with constrained LSTM network," *IEEE Trans. Intell. Transp.*, vol. 22, no. 11, pp. 7242–7255, Nov. 2021.
[33] X. Zhao, Q. Zong, B. Tian, and M. You, "Finite-time dynamic allocation and control in multiagent coordination for target tracking," *IEEE Trans. Cybern.*, early access, Jun. 30, 2020, doi: 10.1109/TCYB.2020.2998152.
[34] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," 2017, *arXiv:1703.04908*.
[35] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.* vol. 3, no. 1, pp. 73–84, Feb. 2019.
[36] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 32, 2019, pp. 8026–8037.
[37] A. Juliani *et al.*, "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.
[38] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 5, Apr./May 2004, pp. 4393–4398.
[39] P. Peng *et al.*, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games," 2017, *arXiv:1703.10069*.
[40] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *Proc. 9th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, 2006, pp. 1–6.
[41] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.
[42] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.
[43] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[44] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[45] Y. Zhu, D. Qian, D. Ren, and H. Xia, "StarNet: Pedestrian trajectory prediction using deep neural network in star topology," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 8075–8080.