IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 15, 2020, accepted September 3, 2020, date of publication September 8, 2020,
date of current version September 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022638

Recurrent MADDPG for Object Detection and Assignment in Combat Tasks

XIAOLONG WEI[1,2], LIFANG YANG[1,2], GANG CAO[1,2], (Member, IEEE), TAO LU[3], AND BING WANG[1,2]

[1]State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China
[2]School of Computer Science and Cybersecurity, Communication University of China, Beijing 100024, China
[3]Simulation Research Center, China Aerospace System Simulation Technology Co., Ltd, Beijing 100094, China

Corresponding author: Lifang Yang (yanglifang@cuc.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2019YFB1406201, and in part by the Fundamental Research Funds for the Central Universities under Grant CUC2019B021.

ABSTRACT With the development of artificial intelligence, multiagent algorithms have been applied to many real-time strategy games. Making plans on the human being is gradually passing away, especially in combat scenarios. Cognitive electronic warfare (CEW) is a complex and challenging work due to the sensitivity of the data sources. There are few studies on CEW. In the past, wargame simulations depended on differential equations and war theory, which resulted in high time and human resource costs. In the future, as other artificial intelligence theories are developed, artificial intelligence will play a more critical role in wargames. The capabilities of multiagent modeling to describe complex systems and predict actions in dynamic environments are superior to those of traditional methods. In this paper, we use a 3D wargame engine from China Aerospace System Simulation Technology Co., Ltd. (Beijing) named All Domain Simulation (ACS), which supports land, sea, and air combat scenarios, to simulate combat. In the simulations, there are several unmanned air vehicles (UAVs) as attackers and several radar stations as defenders, and both have the ability to detect the others. In the game, several UAVs need to learn to detect targets and track targets separately, and we train the UAV's behavior by well-designed reward shaping and multiagent reinforcement learning (MARL) with LSTM. We improved the RDPG algorithm and merged the MADDPG and RDPG algorithms. From the experimental results, we can see that the effectiveness and accuracy of the algorithm have been greatly improved.

INDEX TERMS UAV, RDPG, MADDPG, CWE.

I. INTRODUCTION

During the simulation and deduction of a wargame, an experienced player can judge and predict the missions of enemy aircraft according to their flight status, combat capabilities, rules of engagement and other information. With the continuous development and improvement of wargame systems, air missions are facing many new changes. Firstly, the number of combat units has increased dramatically, and the player needs to analyze and determine each target individually. Secondly, the continuous development of information technology allows wargame situations to evolve increasingly faster. Finally, massive amounts of wargame data are often incomplete, untimely, inaccurate, and even deceptive, making it difficult for players to analyze the hidden key situations. A series of profound changes have made the identification of air missions increasingly more difficult, and it is hard to adapt to the highly complicated and rapid transformation of a wargame situation using traditional artificial recognition methods. Therefore, the research can liberate the players from multi-source and complex battlefield data, and devote more energy to other crucial works, which is a major trend of intelligent wargame development in the future. With the continuous development of multiagent reinforcement learning (MARL), reinforcement learning (RL) has acquired the capabilities of autonomous learning and distributed computing. Agents generate their own behaviors, change their own state information, and finally achieve the goal efficiently through cooperation with others. The multiagent system can complete not only a single agent's goal, but also exceed the efficiency of the single agent, which means that many agents can increase its strength. MARL is robust, reliable, intelligent and

The associate editor coordinating the review of this manuscript and approving it for publication was Jianxiang Xi.

efficient at solving practical problems. At present, the related technology of a multiagent system has become a hot topic in artificial intelligence, communication technology, computer science and other fields. Multi-agent collaboration is a new and challenging topic in practical application. How to enable multiagents to learn efficiently in larger and more random environments is an ongoing challenge of reinforcement learning. In reinforcement learning, some algorithms use policy iteration to train agents, and then the training can be generalized to a larger environment; however, Lowe *et al.* [2] points out that traditional RL methods are poorly suited for multiagent environments.

Making multiagents team up like people to accomplish goals is a partially-observable Markov decision process (POMDP) task. Deep reinforcement learning uses an asynchronous framework to train multiagents, where each agent is independent of the others. Researchers have done a lot of work on single-agent environments. Lillicrap *et al.* [3] focuses on the problems with continuous action spaces, and designs a deep deterministic policy gradient algorithm (DDPG) within the actor-critic framework [4]. Heess *et al.* [1] proposes a recurrent deterministic policy gradient algorithm (RDPG). He demonstrates that the DDPG coupled with LSTM is able to solve a variety of physical control problems with an assortment of memory requirements. Wang *et al.* [5] proposes the R-MADDPG for partially-observable environments and limited communication, She investigates the recurrent effects of the use a team of agents on performance and communication. This centralized training of decentralized policies has recently attracted attention from the multiagent reinforcement learning community [6]. The Q-value path decomposition (QPD) is a value-based method that also aims to address the issues with a multiagent credit assignment. The QPD method proposes a network that integrates information from each agent into a multi-channel critic network and decomposes the global reward into individual rewards [7]. Subsequently, during its decentralized execution, the importance of each agent is correctly verified to approximate a global reward [8].

In some MARL algorithms, the interaction between agents is fully connected, which not only increases the complexity of the algorithm, but also makes it more difficult to apply the algorithm to reality. References [9]–[12] give studies of the interaction of multiagents with cooperative natures, and [13]–[15] give a series of studies on multiagents with competitive natures. How to manage the start and end of training is also a difficult problem of MARL. In multiagent training, no matter how good or bad an agent learns, each episode will end the current training at the same time. Therefore, it is also a challenge to keep an agent who performs well continuing with learning and start a new learning process if the agent performs badly.

In view of the above, the efficiency of manual wargame simulations is low, and they do not achieve good solutions in a randomly changing environment. In this paper, we apply MARL to the field of wargame simulation using MADDPG [2], which makes each agent have not only the ability to be independent, but also the ability to cooperate with others. Over time, the multiagents will learn how to increase useful information and decrease useless information. In this way, each agent optimizes its strategy by increasing the interaction among multiple agents. Even if the environment changes, agents can accomplish their goals efficiently according to the strategies they have learned. However, the MADDPG based on DDPG can only act on the current state, and the previous state cannot affect the subsequent actions. To solve this problem that was mentioned above, the MADDPG will not retain the previous memory in each training iteration, and this paper adopts an improved recurrent deterministic policy gradient algorithm (Fast-RDPG) to accomplish the information inheritance of multiagents.

The main contributions of this paper are as follows:

1) We have constructed a 3D situation environment for electronic warfare. Multi-UAV navigation in a complex environment is regarded as a POMDP problem. A new MADRL framework has been developed for this problem.
2) We point out that MADDPG cannot play an effective role in an environment with obstacles; and as the number of agents increases, the convergence speed of MADDPG will also decrease. We integrate the concept of a recurrent neural network into the MARDL algorithm and conduct training using historical data in real-time.

The remainder of this manuscript is structured as follows. Section II introduces some background knowledge of electronic warfare technology and multiagent reinforcement learning. Section III formulates the navigation problem. The POMDP modeling process of the navigation problem is elaborated, and the proposed algorithm is demonstrated in Section IV. Simulation results and discussions are presented in Section V. Section VI concludes this paper and envisages some future work.

II. RELATED WORK

A. COGNITIVE ELECTRONIC WARFARE WITH TRADITIONAL METHODS

Cognitive electronic warfare combines cognitive science and electronic warfare technology and involves a wide range of subjects, including electronics, military science, computer science, philosophy, linguistics, anthropology, neurology, psychology, etc. If an entity has cognitive ability, it must be built on the basis of the certain intelligence of the entity itself. The cognition can be regarded as the process that assesses the target and surrounding environmental signals; and the process includes detection, analysis, processing, information formation, intelligent judgment and implementation [16]. The idea of cognition was first embodied in the field of radio communication. According to the characteristics of the environment, its core idea is to perceive the surrounding environment and optimize the parameters in real time. Research on the cognitive system model of the EW environment has achieved

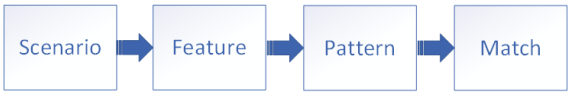


FIGURE 1. Processing task based on the traditional method.

great progress [16]–[19]. Therefore, a cognitive electronic warfare system could reduce the costs and decrease the risk of casualties.

The existing algorithms mainly adopt the traditional non-learning strategy to deal with the multiple UAV task. The strategy needs to obtain feature information from the situation of the complex scenario, then matches the processed feature information with the prior knowledge, and finally outputs the recognition result. The general process is shown in Figure 1. The traditional methods include fuzzy inference, Bayesian networks, neural networks, etc. These methods could solve the problem of uncertainty in small-scale tactical tasks on different levels. In [20], air task identification is regarded as the reverse process of the enemy's air task assignment that is inferred from the enemy's perspective and solved by the continuous Hopfield neural network optimization algorithm. In [21]–[23], the Bayesian network algorithm is improved to adapt to the dynamic changes of battlefield events and identify the combat intentions of enemy air targets. A Bayesian network has a strong probabilistic and causal inference ability, which can dynamically adapt to battlefield changes through the constant updating of the network parameters and solve the problem of uncertain reasoning regarding intentions. However, the adaptability of a Bayesian network to a complex battlefield is insufficient to deal with the deception from antagonistic intentions. A Bayesian network also difficulty determining the prior probability and conditional probability of each node event. Although the traditional methods have solved the problem of uncertain reasoning in some scenarios, there are problems such as subjective experience constraints, insufficient intelligence levels, complex systems and big data. Therefore, it is urgent to break through traditional technology to explore and study the intelligent methods oriented to big data.

B. MULTI-AGENT REINFORCEMENT LEARNING BASED ON POLICY GRADIENT

Since the introduction of deep reinforcement learning in 2015, algorithms such as Q-learning, Sarsa and TD-lambda have made many amazing achievements [24]–[28]. Compared with the Monte Carlo method, the temporal difference (TD) method will conduct a maximum likelihood estimation, which makes the estimation result more in line with the trend of future data. Deep reinforcement learning (DRL) combines reinforcement learning with decision-making and deep learning with perception. It has been used to solve increasingly more problems, such as intelligent robot arm motion in a factory, face recognition, and video analysis [29]. The DQN was proposed to improve Q-learning, which solved the ''dimension disaster'' of Q-learning; and its computing performance exceeded the performance of professional human players in several experimental games [30]. In order to eliminate the strong correlation among reinforcement learning data, the DQN uses a uniform sampling experience replay mechanism to train neural networks; however, the uniform sampling method ignores the importance of different experiences. Therefore, [31] put forward that for the prior experience from a replay, the DQN should use the TD error to measure and replay the importance of an agent's experience multiple times, thereby improving learning efficiency. Hasselt *et al.* [32] proposed a double DQN algorithm based on the DQN to separate action selection from value estimation to avoid overestimating the value. The actor-critic method [33] combined the policy gradient and TD algorithm and adopted a one-step update to solve the problem of low evaluation efficiency. Based on the AC algorithm, the Deterministic Policy Gradient (DPG) and Asynchronous Actor-Critic Agents (A3C) were proposed. A3C [34] is different from DQN's experience replay mechanism. It breaks the correlation between training data and makes parallel training possible. Different from AC, the DPG algorithm adopts the TD to reduce the variance, which allows the DPG to be able to be better applied to off-policing. The DDPG algorithm [35] improves the DPG algorithm since it can output continuous actions, which allows the deep reinforcement learning method to be applied to more complex and continuous action spaces. The PPO algorithm [35] is similar to TRPO algorithm, but PPO is more suitable for large-scale computations and achieves a new balance between the difficulty of the implementation and the complexity of the sampling.

The combination of a multiagent system and DRL will experience some problems, but the related research on multi-agents never stops. Schulman *et al.* [36] proposed a universal cooperative network (ACCNet), which combines AC and DL to learn the communication between agents from scratch in a partially-observed environment, to reduce the environmental instability. Reference [37] proposed Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL), which use the average interaction between all agents or neighboring agents to approximate the interaction between agents. This method greatly simplifies the problem of the action space increasing due to the number of agents. Lowe *et al.* [2] extended the DDPG method to MARL by observing the opponent's behavior. Meanwhile, the global critic function was built to evaluate the global state action, and a group of agents was trained to improve the robustness of the algorithm. Chu andYe [38] proposed an MADDPG algorithm (PS-MADDPG) based on parameter sharing that includes three actor-critic shared architectures to solve the problem of the poor expansibility of the MADDPG algorithm. Omidshafiei *et al.* [39] considers whether the agent should interact with another agent or only treat the other agent as part of the environment without interacting, thus reducing the complexity of the algorithm by determining the degree of interaction between agents. Foerster *et al.* [37] considers

the problem of the perception and action of multiple agents in an environment by making it possible for agents to learn communication protocols in environments and share the necessary information to solve tasks and maximize their shared utility. Since the AC algorithm will lose shared information and collaboration between the agents due to its independent training, Foerster *et al.* [40] gives preliminary solutions to this problem and proposes a method that centralizes critics with the COMA and transfers global information to each agent so as to improve the capability of sharing among each agent.

C. LSTM AND RDPG

The LSTM network is an improved Recurrent Neural Network (RNN). With memory units, LSTM stores feature information over long time intervals and solves the problem that the RNN is prone to the vanishing gradient problem. The RDPG [1] combines the DDPG and LSTM and it extends the framework of the DDPG to POMDPs by updating the policy parameters using the following policy gradient:

$$\nabla_{\eta}(\mu) = \mathbb{E}\left[\sum_{t} \gamma^{t-1} \nabla_{\theta}\mu(h_t)\nabla_{a}Q_u(h_t, a)|_{a=\mu(h_t)}\right]. \quad (1)$$

During training, the agent determines its actions based on the environment using the policy learned. After each episode, the RDPG will put all the empirical data into the memory cache. Although the RDPG has better results than the DDPG in many fields, the RDPG still has some disadvantages. Firstly, the RDPG is an offline algorithm that executes parameter optimization at the end of each episode. Secondly, the RDPG is updated based on historical trajectories instead of entire episodes, and this model makes it converge slowly. Wang *et al.* [41] optimized this problem by correcting the timing of using buffer data and proposed an online DRL method named the Fast-RDPG.

D. WARGAME PLATFORM

Due to the rapid development of military modeling and simulation technology, centralized simulation and distributed interactive simulation are widely used in air defense missile weapon system simulation. Based on the design method proposed above, this paper adopts the combat simulation platform developed by China Aerospace System Simulation Technology Co., Ltd., and shown in Figure 3, and presents a concrete example of an air defense combat simulation. The simulation test is planned using the simulator engine, and then we set the target's parameters and start the engine of the simulator. The red side sends aircrafts with passive radar to search and approach the blue side's target position. The blue side has the corresponding ground air defense radar.

III. PROBLEM DESCRIPTION

This paper mainly aims to accomplish the multiagent monitoring and search task in a wide range of large-scale, unknown, complex environments as quickly as possible. The search mission requires the UAVs to have the autonomous search, response and navigation capabilities according to different environmental characteristics; the capability to overcome information interference from the external environment; and the capability to track targets efficiently and autonomously in real-time.

FIGURE 2. 3D digital Earth of the experimental environment from All Domain Simulation (ACS). There are four radar stations and four targets on the blue side. The planes on the red side need to bypass the radar detector and find the corresponding positions.

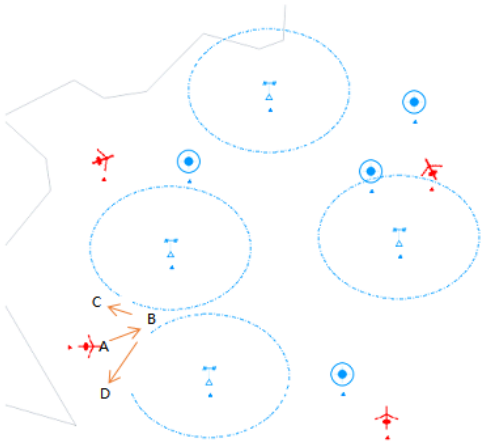


FIGURE 3. This figure illustrates the local observability of UAV navigation in a large range of complex environments. The solid blue circles represent the sensor ranges. The arrows indicate the likely trajectory of the drone.

The search field of a UAV is a real GIS map, and the corresponding longitude, latitude and ECEF 3-D coordinates are calculated according to the position of the operator. The position of the target and the position of the radar are generated randomly, as shown in Figure 2. In the simulation scenario, the mission area size is set to 10 km * 10 km, and four UAVs are searching for four static targets to bypass the radar coverage areas. Since the algorithm does not achieve effective results in the POMDP environment, we assume that the location of the targets and the radar locations are known. When a target is within the surveillance range of the UAV, it indicates that the UAV has found the target at that location. The flight process of the UAV is determined by the internal simulation engine of ACS, And the corresponding velocity and direction vector needed to be transmitted externally.

Autonomous navigation is a special mobile robot moving path planning problem. According to its cognition of the environment, the precondition for the robot to perform other tasks is that the agent can plan a path to avoid obstacles. This requires a more comprehensive understanding of the environment and mobile route planning algorithms through research. There are two kinds of path planning problems: global path planning with a known environment and local path planning with an unknown environment. Q-learning methods based on different greedy strategies, such as annealing Q-learning, can be used. As is known to all, the output space of Q-learning is finite and discrete. The information of a discrete environment and the action output can meet the needs of autonomous navigation in a simple laboratory environment. However, it cannot meet the requirements for the control accuracy of a continuous value such as the direction and speed of autonomous navigation. Thanks to the rapid development of reinforcement learning algorithms based on previous research in recent years, the DDPG neural network performs well at solving problems in a continuous state. Therefore, the DDPG neural network is used as the algorithm in this chapter to explore the autonomous navigation problem of the UAVs. In practice, the pilot controls its movement based on visual or azimuth instructions. The terrain is rough and the obstacles have different sizes. During the movement time, the drone not only needs to dodge obstacles, but it also needs to find its target. In order to simplify the experimental variables, only fixed obstacles appear in this chapter.

In this paper, the UAVs in a cluster are defined as $1, 2 \ldots N$; and X_i is the status sequence of UAV_i, including its position, sensing distance and orientation. N is the number of UAVs, t is the simulation moment, T is the target position set in the environment, and $(x, y)_r$ is the target position coordinate. B is the location set of the obstacles in the environment, $(x, y)_b$ is the location coordinates of the obstacles, $(x, y)_{X_i}$ is the dynamic position coordinates of UAV_x at time t, and $(v_x, v_y)_{X_i}$ is the dynamic velocity coordinates of UAV_x at time t.

To facilitate the research, the following assumptions are made:

1) Goals do not overlap.
2) The target and radar range do not overlap.
3) The UAV cluster is fixed at a stable flying altitude.

This paper aims to use a group of unmanned aerial vehicles (UAVs) to detect dynamic and static targets in a large-scale unstructured environment with obstacles.

IV. METHODS

A. MULTI UAV NAVIGATION AS A POMDP

As shown in Figure 3, the left UAV located at point A needs to approach the target and find a route that is not found by the opponent under the condition that its perception ability is limited. If there is no path of previous memory, the UAV will go to point B and proceed in an infinite loop. On the contrary, if the UAV has memory, the UAV can fly from point A to point B and find that there is nowhere to go, and then the UAV can exit the infinite loop from B to C or B to D and continue to find the nearest target point.

B. OBSERVATION SPACE AND ACTION SPACE SPECIFICATION

In order to support the navigation task of multiagents, the UAV group needs to obtain three information sources: the first is the speed $[x, y]$ and position $[x, y]$ of the agent itself, the second is the relative coordinate vector $tau = [d_0 \ldots d_N]$ of other agents and the current agent, and the third is the relative coordinate vector $psi = [d_0 \ldots d_N]$ of the current agent and the obstacles.

C. REWARD DESIGN

In reinforcement learning, the reward value is the convergence direction of the neural network. When multiagents navigate in a complex environment, the reward of each agent depends on the distance to the nearest target and whether it collides with other agents. The closer the distance is, the greater the reward.

$$C = \begin{cases} -1, & collision \\ 0, & not\ collision \end{cases} \tag{2}$$

Suppose that the distance between agent i and target j is $D(i, j)$. Then, the reward of agent i is:

$$R_i - \min_{j\ le\ N}(D(i, j)) + C. \tag{3}$$

D. MADDPG

In the multiagent environment, N combat units can be expressed as Markov decision-making processes (MDPs). We define state S as all possible conditions of all agents and action A as a series of actions $a_1, a_2, \ldots a_n$. The policy for each agent is defined by π_i, which is used to calculate the action based on the current status of s_i. After its execution, $agent_i$ gets the reward r_i. The goal of each agent in each iteration is to maximize its expected total revenue. All agents share the same reward function and discount factor γ, which is the hyperparameter of the reward function that determines the degree to which the policy favors immediate reward over long-term benefits [42].

The MADDPG uses the method of centralized training distributed execution. The actor uses the policy gradient to learn and select the agent's actions under the current given environment while the critic uses the policy evaluation to evaluate the value function and generate signals to evaluate the actions of the actor. In the path planning process, the environmental data are input into the actor network and the actions needed by the agent are output. The environmental state and actions of agents are input into the critic network, and the corresponding Q is output as the evaluation standard.

In the MADDPG algorithm, both actors and critics are represented by a deep neural network (DNN). The actor network and critic network respectively approximate the function of

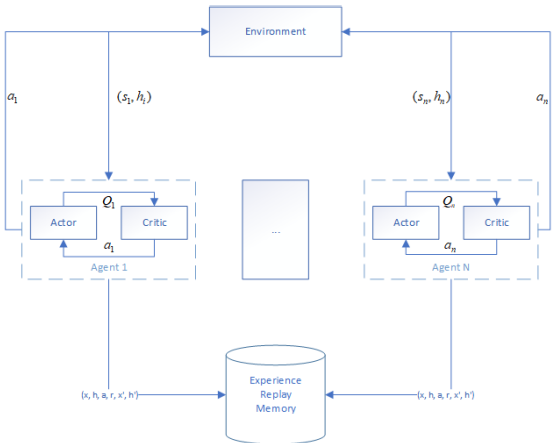


FIGURE 4. The training framework.

the deterministic policy and value-based function. When the algorithm is iteratively updated, firstly, the sample data of the experience pool are accumulated until the specified minimum batch size is reached. Then, the sample data are used to update the critic network and update the parameter through the loss function, and the gradient of the relative action is obtained. Then, the actor will be updated by the Adam optimizer.

The definition of the input and output generated by the critic network is as follows:

$$y^i = r_i^j + \gamma Q_i^{\mu'}(x^{j'}, a_1', \ldots, a_N')|_{a_k'=\mu_k'(s_k^j)} \tag{4}$$

Let S be denoted by the environmental state before the execution of the action, and y^i is the Q generated by the intelligent agent choosing the path planning behavior according to the policy and μ under state s. Algorithm training is carried out with the goal of maximizing the reward value while minimizing the $L(\theta_i)$ function.

$$L(\theta_i) = \frac{i}{S}\left(\sum_j y^i - Q_i^{\mu}(x^j, a_1^j, \ldots, a_N^j)\right)^2 \tag{5}$$

The policy network training seeks to find the optimal solution of the network parameters. The MADDPG algorithm uses stochastic gradient ascent (SGA) to update the policy network parameters. The formula of the actor network update gradient is:

$$\nabla_{\theta_i} J \approx \frac{i}{S}\sum_j \nabla_{\theta_i}\mu_i(s_i^j)\nabla_{a_i} Q_i^{\mu}(x^j, a_1^j, \ldots, a_N^j)|_{a_k=\mu_k(s_k^j)} \tag{6}$$

E. RECURRENT MADDPG

Aiming at the current simulation environment, this paper proposes a multiagent actor-criticism model based on long short-term memory (LSTM) network recursion. The model only accepts data from one time frame per simulation step. The LSTM model extends the actor-critic framework of the MADDPG by storing the information before the step through the hidden layers, which makes learning using time series possible in a multiagent warfare environment. The flowchart

Algorithm 1 Recurrent Multi-Agent Deep Deterministic Policy Gradient for N Agents

for episode = 1 to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state x initialize empty history h
  for t = 1 to max-episode-length **do**
    for each agent i select action:
    $a_i = \mu_{\theta_i}(o_i, h_i) + \mathcal{N}_i$
    Execute actions $a = (a_i, \ldots, a_N)$
    Get reward r
    Get new state x'
    Get new history h'
    store (x, h, a, r, x', h') in replay buffer $\mathcal{D}$
    $x \leftarrow x'$
    $h \leftarrow h'$
    for agent $i = 1$ to N **do**
      Sample a random mini-batch of $\mathcal{S}$ samples
      $(x^j, a^j, r^j, x'^j, h'^j)$ from $\mathcal{D}$
      $y^i =$
        $r_i^j +$
        $\gamma Q_i^{\mu'}(x^{j'}, h^{j'}, a_1', \ldots, a_N')|_{a_k'=\mu_k'(s_k^j, h_k^j)}$
      Update critic by minimizing the loss:
      $\mathcal{L}(\theta_i) = \frac{i}{S}\left(\sum_j y^i - Q_i^{\mu}(x^j, h^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor:
      $\nabla_{\theta_i} J \approx$
        $\frac{i}{S}\sum_j \nabla_{\theta_i}\mu_i(s_i^j, h_i^j)\nabla_{a_i} Q_i^{\mu}$
        $(x^j, h^j, a_1^j, \ldots, a_N^j)|_{a_k=\mu_k(s_k^j, h_k^j)}$
    end for
    Update target network parameters
    for each agent i
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
  end for
end for

of the algorithm of this paper is shown in Figure 4, And the pseudo-code is shown in Algorithm 1.

The real battlefield environment is modeled using MARL in which the agents in the environment have two networks, namely, the actor network and critic network. The input value of the actor network is the speed and position of the current agent and the relative position of the other agents and the current agent. In addition, there is the relative position s_i of obstacles and the current agent. The actor network outputs the action value a_i to the simulation environment. The input value of the critic network is the combination of s_i and a_i, and the output value of the critic network is Q_i. Q_i is used to evaluate the actions generated by the actor network. Through continuous learning, the actor network and the critic network use the method of centralized training distributed execution to optimize the parameters. During training, the experience buffer stores the current state and action, the next state and action, and the LSTM hidden elements $(s_i, a_i, s_i', a_i', h_i, h_i')$, respectively. The critic network is the same as the MADDPG,

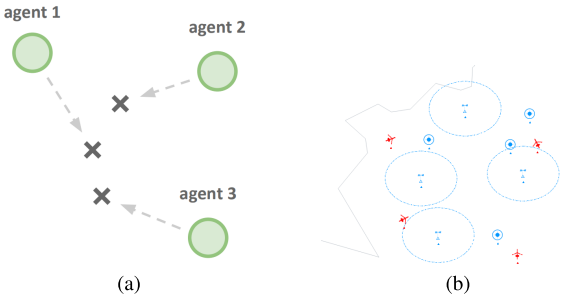


FIGURE 5. Multi-agent cooperative navigation environment developed by OpenAI and the simulator named ACS developed by China Aerospace System Simulation Technology Co., Ltd.

and the $L(\theta_i)$ function is same as equation (5). The formula of the actor network update gradient is $\nabla_{\theta_i} J$. Each actor network has a corresponding critic network corresponding with the critic network improvement policy. Even if the agent has only part of the state information, it can still make the right decision.

The recently-proposed recurrent MADDPG (R-MADDPG) uses recurrent neural networks based on the MADDPG to handle partially-observable environments. Similar to our work, although Wang *et al.* [5] used recurrent actor-critic networks, their recurrent layers had the purpose of remembering the messages from previous time steps to solve certain domains that are not allowed to communicate every time step. However, we use recurrent layers for remembering the infomation of previous time steps to solve the problem of consistent navigation.

V. EXPERIMENTS

We construct the combat scenario described in Section 2 by combining the multiagent particle environment developed by OpenAI [2] and the simulator named ACS developed by China Aerospace System Simulation Technology Co., Ltd. There are several agents in this combat scenario. We combine the two training techniques with the MADDPG and fast-MARDPG, respectively, and conduct comprehensive experiments to illustrate the performances of the two techniques.

As shown in Figure 5, two situations were created based on the OpenAI environment: one contains three agents and three targets, and the other includes four agents and four targets. In addition, two situations were created based on ACS environment. Compared with the OpenAI environment, there are more radar workstations in ACS. Similarly, there are three agents and three targets and four agents and four targets, respectively. The difference is that there are four obstacles in ACS environment. The four obstacles are four radar detectors. The agent needs to bypass radar detection, and then navigate to the target location automatically. ACS is a simulator with a 3D digital map of Earth as a display and it has simulation and deduction abilities. The engine of the former is provided by OpenAI, and the latter is provided by ACS.

We trained agents using the fast-MARDPG and MADDPG, respectively. For the MADDPG, the neural network for each red agent includes 2 hidden layers. Each hidden layer contains 64 hidden units with ReLU activations. Similarly, regarding the neural network of the critic for each red agent, the input of the critical network is the state and actions output by the actor network, and the architectures of the critical network and the actor network are basically the same. For the fast-MARDPG, Regarding the actor network each red agent, the input layer is followed by an LSTM layer, And after each LSTM layer is 2 hidden layers. Each hidden layer contains 64 hidden units with ReLU activations. The critic network is same as that of the MADDPG. To train the neural network, we use the Adam optimizer. The learning rate is set to 0.01, the gamma value is set to 0.95 and the batch-size is set to 1024. During training, the maximum number of steps for each episode was 45 for the "Cooperative Navigation" of OpenAI, and the maximum number of steps for each episode was 1000 for ACS. Those parameters will be used in the following experiments if they are not explicitly specified.

1) SIMPLE SPREAD

The result of this simple spread experiment is shown in Figure 6. An episode is a combat round that is terminated if the number of steps exceeds the maximum number of steps. This figure shows the respective reward rates of the two methods for the training of red agents. As we can see, both fast-MARDPG and MADDPG converge in this simple task. The first subfigure shows the average reward of each agent, and the second subfigure shows the average reward for all agents.

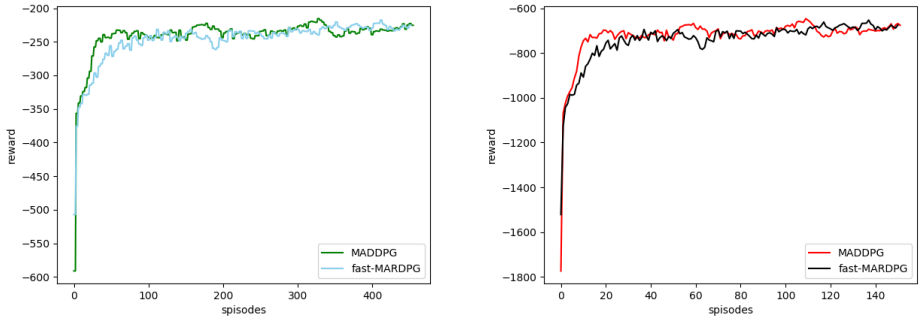
Correspondingly, we increase the number of agents and the number of targets. Both the number of agents and targets are four. It can be seen in these figures that both algorithms have achieved good convergence effects; but compared to the previous MADDPG, the fast-MARDPG has not produced a particularly obvious improvement. Since the neural network with memory does not play its role in this situation, it has delayed the convergence rate to a certain extent.

2) WARGAME

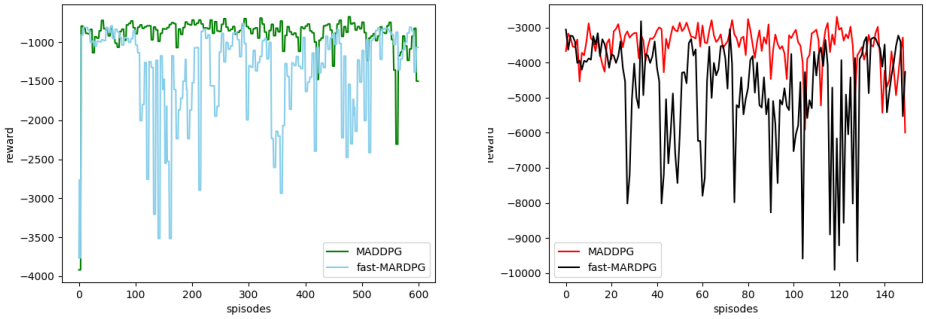
In a complex multiagent environment, we use the same neural network and the same data as the previous experiment. However, we do not run the same simulation. We add obstacles in the environment, that is, the number of the opponent's ground radar stations is four. Our plane needs to bypass the radar detector and find the corresponding position. The results can be seen in Figure 6. Adding the LSTM layer to the MADDPG increases the convergence speed, and it also increases the average reward of all agents and the average reward value of a single agent. In Table 1, we compare the winning percentages of the two algorithms of the MADDPG and the fast-MARDPG for 1000 episodes and the average number of steps of all episodes.

3) BEHAVIORAL ANALYSIS

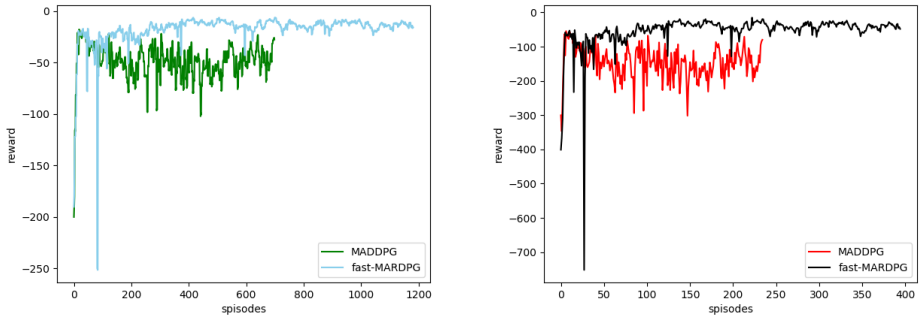
Aiming at the multiagent navigation problem, this paper uses the same neural network structure to perform the above two



(1) the maximum steps for each episode is 45 in the environment of OpenAI.

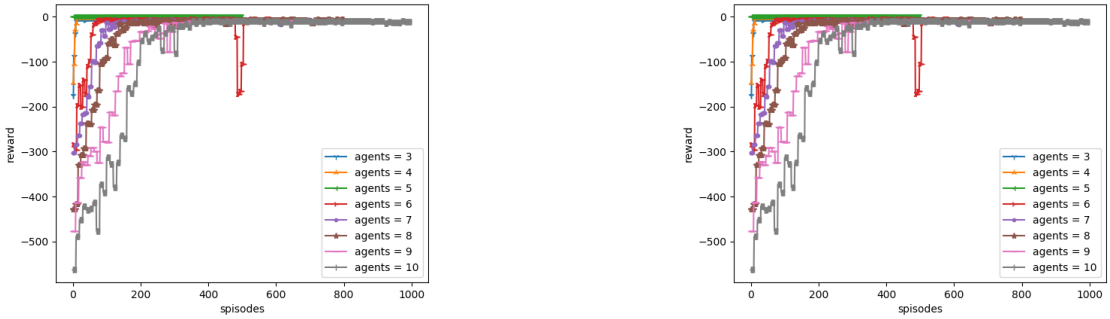


(2) the maximum steps for each episode is 1000 in the environment of OpenAI.



(3) the maximum steps for each episode is 1000 in the environment of ACS.

FIGURE 6. The comparison of the fast-MARDPG and MADDPG and the average reward of all agents for a simple navigate task with several agents and targets without obstacles in the OpenAI and ACS environments, each set of images describes the average reward value of three agents and four agents, where the maximum number of steps for each episode is 45 and 1000, respectively.



(a) (b)

FIGURE 7. Comparison of the average reward of all agents and the average reward of a single agent in a simple combat task with different numbers of agents.

experiments in the environments without and with obstacles. The detailed results are shown in Table 1. In the former, the maximum training step length is 45. Since 45 steps in the test environment cannot have a good effect, the number of test steps is increased to 1000. Compared with the fast-MARDPG, the MADDPG achieves results that are basically

TABLE 1. Performance of the proposed DRL framework at different game levels.

Level		Number of agents	Steps per episode		Arrivals per episode	
			MADDPG	fast -MARDPG	MADDPG	fast -MARDPG
Simple spread	45 stpes	3	494	480	2.96	2.95
		4	997	999	0.01	0.01
	1000 steps	3	998	999	0.02	0
		4	999	999	0	0
Wargame	1000 steps	3	534	429	1.89	2.23
		4	828	612	0.93	2.02

consistent with those of the fast-MARDPG. From the experimental results of the former, it can be seen that as the number of agents increases, a small number of steps cannot meet the requirements of the experiment; therefore, the latter uses 1000 steps for training and testing, and the effect is significantly improved after adding LSTM. In the case of three agents and three targets, the average number of steps decreased by 19.6%, and the average reach of the target was increased by 17.9%. Similarly, in the case of four agents and four targets, the average number of steps has dropped by 26%, and the average goal has been increased by 117%. However, it can be seen in Figure 7 that as the number of agents increases, the hit rate also continues to decline. Based on the above considerations, we focused on three and four agents.

VI. CONCLUSION

In this paper, we focus on the two methods of MARL and the LSTM network, and we combine and improve these two methods. Without map navigation and path planning, our method enables multiple UAVs to fly from arbitrary departure points to destinations in large-scale, complex environments. The results demonstrate that the fast-MARDPG is more efficient than the MADDPG. However, there are still great defects in the experiment. Regarding their time performance, there is still some work to do in the future. For example, it is necessary to address how to make the accuracy rate no longer decrease under the situation of increasing agents. Furthermore, how to design and fine-tune the reward function is necessary; otherwise, it may yield unsatisfactory performance.

REFERENCES

[1] N. Heess, J. Hunt, T. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015, *arXiv:1512.04455*. [Online]. Available: https://arxiv.org/abs/1512.04455
[2] R. Lowe, Y. 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.*, 2017, pp. 6379–6390.
[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," Sep. 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971
[4] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Neural Inf. Process. Syst. Conf.* Cambridge, MA, USA: MIT Press, 2000, pp. 1008–1014.
[5] R. E. Wang, M. Everett, and J. How, "R-MADDPG for partially observable environments and limited communication," 2020, *arXiv:2002.06684*. [Online]. Available: https://arxiv.org/abs/2002.06684
[6] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn. (PMLR)*, 2019, pp. 2961–2970.
[7] Y. Yang, J. Hao, G. Chen, H. Tang, Y. Chen, Y. Hu, C. Fan, and Z. Wei, "Q-value path decomposition for deep multiagent reinforcement learning," 2020, *arXiv:2002.03950*. [Online]. Available: https://arxiv.org/abs/2002.03950
[8] Y. Yang, J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang, "Qatten: A general framework for cooperative multiagent reinforcement learning," 2020, *arXiv:2002.03939*. [Online]. Available: https://arxiv.org/abs/2002.03939
[9] J. Han, C.-h. Wang, and G.-x. Yi, "Cooperative control of UAV based on multi-agent system," in *Proc. IEEE 8th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2013, pp. 96–101.
[10] P. Frasca, R. Carli, and F. Bullo, "Multiagent coverage algorithms with gossip communication: Control systems on the space of partitions," in *Proc. Amer. Control Conf.*, 2009, pp. 2228–2235.
[11] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Hysteretic Q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 64–69.
[12] E. Semsar-Kazerooni and K. Khorasani, "Multi-agent team cooperation: A game theory approach," *Automatica*, vol. 45, no. 10, pp. 2205–2213, Oct. 2009.
[13] 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.
[14] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, Apr. 2017, Art. no. e0172395.
[15] N. Rashedi, M. A. Tajeddini, and H. Kebriaei, "Markov game approach for multi-agent competitive bidding strategies in electricity market," *IET Gener., Transmiss. Distrib.*, vol. 10, no. 15, pp. 3756–3763, Nov. 2016.
[16] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 201–220, Feb. 2005.
[17] T. E. D. Greef, H. F. R. Arciszewski, and M. A. Neerincx, "Adaptive automation based on an object-oriented task model: Implementation and evaluation in a realistic c2 environment," *J. Cognit. Eng. Decis. Making*, vol. 4, no. 2, pp. 152–173, Jun. 2010.
[18] J. E. Summers, J. M. Trader, C. F. Gaumond, and J. L. Chen, "Deep reinforcement learning for cognitive sonar," *J. Acoust. Soc. Amer.*, vol. 143, no. 3, p. 1716, Apr. 2018.
[19] P. Braca, R. Goldhahn, K. Lepage, S. Marano, V. Matta, and P. Willett, "Cognitive multistatic AUV networks," in *Proc. 17th Int. Conf. Inf. Fusion (FUSION)*, Jul. 2014, pp. 1–7.
[20] L. J. C. Yuan, "Tactical task of operation platform recognition in situation assessment for antiaircraft defending," *Command Control Simul.*, Jul. 2015.
[21] G.-L. Meng and G.-H. Gong, "Threat assessment of aerial targets based on hybrid Bayesian network," *Xi Tong Gong Cheng Yu Dian Zi Ji Shu/Syst. Eng. Electron.*, vol. 32, pp. 2398–2401, 2010, doi: 10.3969/j.issn.1001-506X.2010.11.31.
[22] X. X. G. Shun, "DSBN used for recognition of tactical intenrion," *Syst. Eng. Electron.*, Jul. 2014.
[23] H.-J. Deng, Q.-J. Yin, J.-W. Hu, and Y.-B. Zha, "Tactical intention recognition based on multi-entity Bayesian network," *Syst. Eng. Electron.*, vol. 32, pp. 2374–2379, 2010, doi: 10.3969/j.issn.1001-506X.2010.11.26.
[24] X. L. Wei, X. L. Huang, T. Lu, and G. G. Song, "An improved method based on deep reinforcement learning for target searching," in *Proc. 4th Int. Conf. Robot. Autom. Eng. (ICRAE)*, Nov. 2019, pp. 130–134.
[25] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *IEEE Signal Process. Mag.*, vol. 34, Jul. 2017.
[26] A. Banino *et al.*, "Vector-based navigation using grid-like representations in artificial agents," *Nature*, no. 7705, pp. 429–433, 2018.
[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[28] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr, "Playing doom with SLAM-augmented deep reinforcement learning," 2016, *arXiv:1612.00380*. [Online]. Available: https://arxiv.org/abs/1612.00380
[29] K. G. Kim, "Book review: Deep learning," *Healthcare Informat. Res.*, vol. 22, no. 4, p. 351, 2016.
[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602
[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: https://arxiv.org/abs/1511.05952
[32] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015, *arXiv:1509.06461*. [Online]. Available: https://arxiv.org/abs/1509.06461
[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, vol. 1, Jun. 2014, pp. 1–9.
[34] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[35] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, Sep. 2015.
[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jul. 2017, *arXiv:1707.06347*. [Online]. Available: https://arxiv.org/abs/1707.06347
[37] J. Foerster, Y. Assael, N. Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.
[38] X. Chu and H. Ye, "Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning," 2017, *arXiv:1710.00336*. [Online]. Available: https://arxiv.org/abs/1710.00336
[39] S. Omidshafiei, J. Pazis, C. Amato, J. How, and J. Vian, "Deep decentralized multi-task multi-agent RL under partial observability," Tech. Rep., Mar. 2017.
[40] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," May 2017, *arXiv:1705.08926*. [Online]. Available: https://arxiv.org/abs/1705.08926
[41] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.
[42] C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How, "Modeling and planning with macro-actions in decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 64, pp. 817–859, Mar. 2019.

XIAOLONG WEI received the B.S. degree from the Guilin University of Electronic Technology, Guangxi, China, in 2012, and the M.S. degree from the University of Science and Technology, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Computer Science, Communication University of China, Beijing. His current research interests include artificial intelligence algorithms and multiagent reinforcement learning algorithms and simulation.

LIFANG YANG received the B.S. degree in electronic information from Qingdao University, Qingdao, Shandong, China, in 2005, and the M.S. and Ph.D. degrees in signal and information processing from the Communication University of China, Beijing, China. She is currently an Associate Professor with the Communication University of China. Her research interests include intelligent retrieval and high-dimensional index structures.

GANG CAO (Member, IEEE) received the B.S. degree from the Wuhan University of Technology, Hubei, China, in 2005, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University, Beijing, China, in 2013. He was with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2010. He is currently a Faculty Member with the School of Computer Science, Communication University of China, Beijing. His current research interests include digital forensics, data hiding, and multimedia signal processing.

TAO LU received the Ph.D. degree from Nankai University, in 2011. He is currently an Associate Professor with the Simulation Research Center, China Aerospace System Simulation Technology Company Ltd., Beijing. His research interests include artificial intelligence algorithms, simulation, and parallel computing.

BING WANG received the B.S. degree from Xuchang University, Henan, China, in 2015, and the M.S. degree from the Zhengzhou University of Light Industry, Henan, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science, Communication University of China, Beijing, China. His current research interests include multimedia content analysis and deep learning.