IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 25, 2019, accepted January 27, 2020, date of publication February 5, 2020, date of current version February 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971780

Path Planning for UAV Ground Target Tracking via Deep Reinforcement Learning

BOHAO LI AND YUNJIE WU
State Key Laboratory of Virtual Reality Technology and System, Beihang University, Beijing 100191, China
School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
Science and Technology on Aircraft Control Laboratory, Beijing 100191, China

Corresponding author: Yunjie Wu (wyjmip@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 91216304.

ABSTRACT In this paper, we focus on the study of UAV ground target tracking under obstacle environments using deep reinforcement learning, *and* an improved deep deterministic policy gradient*(DDPG)* algorithm is *presented*. A reward function based on line of sight and artificial potential field is constructed to guide the behavior *of UAV to achieve target tracking*, and a penalty term of action *makes* the trajectory *smooth*. In order to improve the exploration ability, multiple UAVs, which controlled by the same policy network, are used to perform tasks in each episode. Taking into account that the history observations *have* a great degree of correlation with the policy, long short-term memory networks are used to approximate the state of environments, which improve the approximation accuracy and the efficiency of data utilization. The simulation results show that the *propose method* can make the UAV keep *target tracking and obstacle avoidance effectively*.

INDEX TERMS DDPG, deep reinforcement learning, obstacle avoidance, target tracking, UAV.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have the advantages of safety, low cost and high maneuverability. *They are* widely used in military or civil fields such as reconnaissance, strike, rescue *and* early warning, etc. *One of the typical application of UAV* [1] *is target tracking and obstacle avoidance.* High autonomy on-line trajectory planning of UAV for target tracking and obstacle avoidance in unknown working environment *arosed great attentions* [2]–[4].

With the rapid development of artificial intelligence technology in recent years, deep reinforcement learning(DRL) plays an important role in more and more fields *for* its excellent environmental awareness and decision control performance [5]. Reinforcement learning can directly map the environment state to the control signal, which provides a dynamic planning solution for UAV trajectory planning. Flight environment *is* usually local or completely unknown *during* on-line path planning. How to react to the dynamic environment using incomplete information is a key *issue* in UAV on-line path planning. Reinforcement learning has advantages of *strong robustness*, *independence on environment model and prior knowledge*, solves the on-line path planning problem by trial and error [6].

Q-learning [7] is *an effective tool in reinforcement learning*, which *is widely used and* followed by many improved algorithms such as SARSA [8], double Q-learning [9], and the first DRL algorithm DQN (Deep Q Network) [10]. *There for Q-learning has been applied into UAV path planning* [11]. Zhang *et al.* [12], uses geometric method to calculate value matrix containing threat information. *In this method*, it combines greedy strategy with Boltzmann strategy to improve exploration. Yijing *et al.* [13] proposes an adaptive random exploration Q-learning method which designs the learning, escape and action modules of UAV respectively. Yan *et al.* [14] *initializes* Q matrix with environment and target, and *adds* the function of avoiding repetitive actions in the initial stage of training. Most Q-learning path planning methods are restricted in grid like environments for Q-learning and its improves can be used only in discrete space. Policy gradient [15]–[18] and actor-critic [19]–[21] can be used to deal with path planing problems in continuous action space. Zhaowei *et al.* [22] proposes an actor-critic algorithm based on RBF neural network to achieve

The associate editor coordinating the review of this manuscript and approving it for publication was Victor S. Sheng.

UAV avoidance. Zhu *et al.* [23] designs an end-to-end DRL model to control UAV indoor target searching with images as input. Deep deterministic policy gradient (DDPG) [27] is a DRL algorithm which combines DQN with actor-critic and can be operated in continuous action space. Wang *et al.* [24] *transforms* the UAV path planning problem into a partially observable Markov decision process, builds *a* recurrent deterministic policy gradient framework to navigation in large-scale and complex environments. You *et al.* [25] proposes a DDPG algorithm based on a generative model and variational Bayesian estimation to search target in cognitive electronic warfare.

Complex, dynamic and partially observable environments are major challenges for UAV target tracking [26]. To overcome these *difficulties*, we improve DDPG in terms of reward function and data. A good reward function is *an excellent* description of the relationship between environment and mission objectives, which improves the ability and generalization of the algorithm. Considering the impact of dynamic environment, *a reward for DDPG is consisted of* line of sight(LOS) [28], [29] and artificial potential field [30]. The quality and utilization efficiency of data *are* the key to the success of model training [31]. The quality of data is closely related to the exploration which is limited by the scope of action. A large scope of actions is *beneficial* for exploration. *But UAV is hard to accomplish actions that if vary too much due to the dynamics limitation.* We add penalty term of action to the reward function. *The action range of UAV will decrease with the convergence of the algorithm correspondingly*. *Mean while* the exploration can be improved by exploring starts [32]. Multiple UAVs with different initial conditions, which controlled by the same model, are used to perform tasks in each episode. The utilization efficiency of data can be improved through the networks and input *states* [33]. LSTM [34]–[36] is a kind of recurrent neural networks with sequence data as input. A part of information of data is transmitted to the next moment in the form of memory and participates in the training of input-output data pairs. That is to say, the training results at the current moment are determined by both the current training data and the historical training data. In partially observable environments, the combination of LSTM and historical sequence of observations can approximate the value network and actor network better [37]. *The improved DDPG algorithm is trained in a virtual simulation environment, and the well-trained algorithm can be used for online target tracking and obstacle avoidance in new dynamic environments.*

The main *contributions* of this paper *are* as follows:

1) A DRL model for target tracking and obstacle avoidance is developed.

2) Critic *network* and actor *network* based on LSTM are designed and well trained.

3) A virtual simulation environment for target tracking and obstacle avoidance is constructed. The results verify that the algorithm has satisfactory performance and favorable generalization.

The structure of this paper is as follows: Section II, the background *is* introduced. Including the DDPG algorithm, the ground target tracking environment, the kinetic models of UAV, observation space and action space. *A new method s presented in Section* III *with the improvement in learning framework, reward function and networks.* Section IV gives the the network structure, the simulation parameters and results, and the efficiency of the improved method is proved. The summary and prospect *is given in* Section five.

II. BACKGROUND

In this section we give an introduction to the background knowledge of DRL ground target tracking and obstacle avoidance*, including* a DRL algorithm – DDPG and the environment used for tracking.

A. DDPG

An agent-environment interaction process with time discreteness can be represented as a trajectory:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$$

where S_t, A_t*, and* R_t represented the state, action*, and* reward at step t *respectively*. If the trajectory satisfies the Markov property, that is, the values of R_{t+1} and S_{t+1} have well defined discrete probability distributions dependent only on the preceding state S_t and action R_t, we can call this interaction process as a Markov Decision Process (MDP) [38]. The agent tries to select actions so that the sum of the discounted rewards *is maximized over the future*. The sum of the discounted rewards after step t can be defined as the return G_t whose normal form is:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{\tau=0}^{+\infty} \gamma^\tau R_{t+\tau} \quad (1)$$

where $\gamma \in [0, 1]$ is a parameter, called the discount rate, *is used* to determines the *current* value of future rewards.

A policy is a map from state to action, which determines the behavior of agents and it is a probability distribution of states in generally. Deterministic policy gradient algorithm *outputs* specific behavior rather than probability. A deterministic policy π can be defined as a function:

$$a = \pi(s; \theta) \quad (2)$$

where θ refers to the parameters of the function. Our task is to find a set of parameters θ make π optimum or make $E_\pi[G_0]$ maximum, where $E_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π. In order to find the parameters, we *defines* two functions, the state-value function $v_\pi(s)$ and the action-value function $q_\pi(s, a)$. $v_\pi(s)$ represents the value of state s under the policy π, is the expected return when starting in s and following π thereafter, which can be expressed by formula:

$$v_\pi(s_t) = E_\pi[G_t | S_t = s_t] \quad (3)$$

$q_\pi(s, a)$ represents the value of taking action a in state s under a policy π, is the expected return starting from s, taking the action a, and thereafter following policy π, which can be expressed by formula:

$$q_\pi(s_t, a_t) = E_\pi[G_t|S_t = s_t, A_t = a_t]. \tag{4}$$

The relationship between state and action value function could be described by Bellman equation [39], and the Bellman equation for deterministic policy is defined as:

$$v_\pi(s) = q_\pi(s, a; \eta) \tag{5}$$

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, \pi(\theta))v_\pi(s') \tag{6}$$

where η is the parameter of value function. There are two steps for policy optimization, the first step is value update. Use temporal difference Learning to update the value functions, in each step:

$$y = R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}; \eta)) \tag{7}$$

and the parameter η can *be updated by*:

$$\eta \leftarrow \eta + \alpha[y - q_\pi(S_t, A_t; \eta)]\nabla_\eta q_\pi(S_t, A_t; \eta) \tag{8}$$

The second step is policy improvement. *Taking* the gradient of Bellman equation and *calculating* its expectations about S_t, we have the recurrence formula:

$$E[\nabla v_{\pi(\theta)}(S_t)] = E[\nabla\pi(S; \theta)[\nabla_a q(S, a)]_{a=\pi(S_t;\theta)}] + \gamma E[\nabla v_{\pi(\theta)}(S_{t+1})] \tag{9}$$

According to the formula (3) we have

$$\nabla E_\pi[G_0] = E[\nabla v_\pi(S_0)] \tag{10}$$

Using (9) and (10), we calculate the policy gradient:

$$\nabla E_\pi[G_0] = \sum_{t=0}^{+\infty} E[\gamma^t \nabla\pi(S; \theta)[\nabla_a q(S, a)]_{a=\pi(S_t;\theta)}] \tag{11}$$

In each time step, *we* update parameters θ according to:

$$\theta \leftarrow \theta + \beta\gamma^t \nabla_\theta\pi(S_t; \theta)[\nabla_a q(S_t, a; \eta)]_{a=\pi(S_t;\theta)}. \tag{12}$$

The principle of DDPG is the same as that of the deterministic policy gradient. The main difference is that the technology of experience replay and target network are used in DDPG whose detailes are given in algorithm 1.

Algorithm 1 Deep Deterministic Policy Graidient Algorithm

1: Initialize policy network $\pi(s)$ and value network $q(s, a)$ with parameters θ and η;
Initialize target policy network $\pi'(s)$ and target value network $q'(s, a)$ with parameters $\theta' \leftarrow \theta$ and $\eta' \leftarrow \eta$;
Initialize the learning rate of target network ε, batch size N, replay memory R;
2: Take action action according to the state $a_t = \pi(s_t)$;
3: Executes the action a_t, receives the reward r_{t+1}, acquire new state s_{t+1};
4: Save $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ to the memory;
5: Sample a batch size of N datas $\{(s_i, a_i, r_{i+1}, s_{i+1})\}_{i=1}^N$ from memory randomly;
6: Update the policy network and value network:

$$y_i = r_{i+1} + \gamma q(s_{i+1}, \pi(s_{i+1}; \theta'); \eta')$$
$$\eta \leftarrow \eta + \alpha\frac{1}{N}\sum_i [y_i - q(s_i, a_i; \eta)]\nabla_\eta q(s_i, a_i; \eta)$$
$$\theta \leftarrow \theta + \beta\frac{1}{N}\sum_i \nabla_\theta\pi(h_i; \theta)[\nabla_a q(h_i, a; \eta)]_{a=\pi(h_i;\theta)}$$

7: Update the target networks

$$\eta' \leftarrow \varepsilon\eta + (1 - \varepsilon)\eta'$$
$$\theta' \leftarrow \varepsilon\theta_c + (1 - \varepsilon)\theta'$$

8: Back to step 2 untill to the maximum number of training.

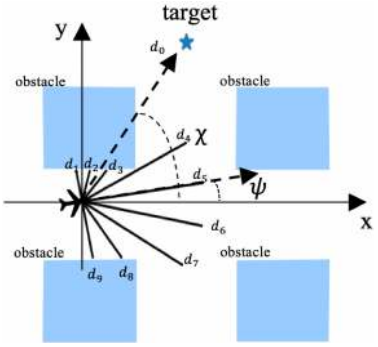


FIGURE 1. Target tracking and obstacle avoidance environment. d_0 denotes the distance between UAV and target; $d_1 \sim d_9$ represent the distances returned by the distance sensors; χ denotes the orientation of target; ψ denotes the orientation of the UAV velocity.

B. ENVIRONMENTS

The target tracking and obstacle avoidance environment can be illustrated in FIGURE 1. The location of the target changes in the environment with time steps. The distance and direction of target can be obtained by UAV. Sensors are equipped in the front of the UAV which can measure the distance of obstacles in the range of d_{max} ahead. Obstacles can be shaped as arbitrary polygons or circles. When the distance received by the sensor less than a certain threshold, it means that the UAV collision with obstacles and the task fails. When the distance between UAV and target is less than a certain threshold, *it* means the tracking task is being performed.

We use matplotlib - python to build a complete environment model. The whole environment is confined to a two-dimensional plane space of 500 meters in length and width with boundaries around it. Several stationary polygons are randomly placed in the boundaries as obstacles, and target at rest or in uniform motion. More informations about the environments are shown in Section IV.

C. OBSERVATION AND ACTION SPACE

The observations we can get from the environment include the position and velocity of UAVs, the distance and direction of target, obstacle distance in directions of the sensors. Considering the dynamic environment and generalization, we abandon the position of UAV, *retain* the speed v and direction ψ. *We* use the relative azimuth $[\chi, d_0]$ *to* indicate the relationship between UAV and target. *We use* the distance measured by sensors *indicates* the relationship between UAV and obstacles. The observation space: $o = [\psi, \chi, d_0, d_1, d_2, \ldots, d_n]$, where n indicates the number of sensors, $\psi, \chi \in [-\pi, \pi]$, $d_0 \in [0, +\infty]$, $d_1 \sim d_n \in [0, d_{max}]$.

UAV can be regarded as a point when planning the trajectory in large scale environments, its three-dimensional continuous-time motion model *is given* as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v} \\ \dot{\psi} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} v cos\psi cos\gamma \\ v sin\psi cos\gamma \\ v sin\gamma \\ u_{\dot{v}} \\ u_{\dot{\psi}} \\ u_{\dot{\gamma}} \end{bmatrix} \tag{13}$$

where $[x,y,z]$ is the coordinates of UAV in three-dimensional space. $[v,\psi, \gamma]$ indicate the speed, yaw angular, pitch angular, respectively.$[u_{\dot{\psi}}, u_{\dot{\gamma}}, u_{\dot{v}}]$ is the control *command*. For simplify, we limit the trajectory planning problem into a two-dimensional continuous state environment and set the velocity to a constant *value*. The simplified dynamic *discrete mode1 in time is presented* as follows:

$$\begin{bmatrix} x(t+1) \\ y(t+1) \\ \psi(t+1) \end{bmatrix} = \begin{bmatrix} x(t) + v cos\psi(t+1) \\ y(t) + v sin\psi(t+1) \\ \psi(t) + a(t) \end{bmatrix} \tag{14}$$

The task for target tracking is to drive the UAV reach the target position in the shortest time and keeps it within bounds without colliding with any obstacles.

The action a refers to the variation of ψ, while the the control quantity of true UAVs usually expressed as acceleration [40]. *In order to guarantee that the action can be followed, the mechanical system between acceleration and action is considered in subsequent part. In the horizontal direction, the force on a moving UAV satisifices the relationship:*

$$F = \frac{mV^2}{\rho} \tag{15}$$

and the normal acceleration a' is shown as:

$$a' = \frac{F}{m} = \frac{V^2}{\rho}. \tag{16}$$

where ρ is the curvature radius of the path, which can be shown in FIGURE 2. AB and BC have the same length which equals the distance traveled by UAV within half sampling period. According to the geometric relation, we have the curvature radius as:

$$\rho = 0.5V\Delta t cot\frac{a}{2} \tag{17}$$

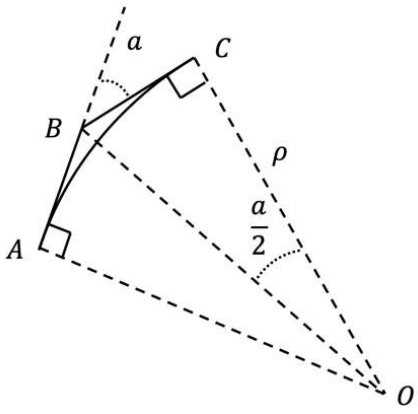


FIGURE 2. Action and curvature radius. a denotes the action; ρ represent the curvature radius; the broken line ABC denotes the generated trajectory in one sampling period; the arc AC denotes the real trajectory in one sampling period.

where Δt is the sampling time-interval. Thus we have the relationship between acceleration and action:

$$a' = \frac{2Vtan\frac{a}{2}}{\Delta t} \approx \frac{Va}{\Delta t}. \tag{18}$$

The relationship can be used to determine the range of action, or it can be used to check whether the selected action range is reasonable. If the acceleration range is covered by the mobility of real UAV, the generated trajectories can be followed by UAV.

III. PROPOSED METHOD

We improve the DDPG in framework, reward function and networks in this section.

A. LEARNING FRAMEWORK

DRL framework for UAV target tracking and obstacle avoidance *includes* three modules:

1) Environment description module. The function of this module is to sense target informations, describe the threats faced by UAVs and extract relevant features.

2) DRL Control module. DRL control is the corner of the framework, which receives the environment description signal, estimates the state value of environment and modifies the control policy continuously.

3) UAV module. UAV adjusts the attitude and position *by means of the control signal*, then affects the received environment description.

Compared with traditional DRL, our framework which *is* shown in FIGURE 3 is different in two aspects. The first is that feature extraction functions are placed in the DRL control module, the feature extraction network and the actor network can be trained simultaneously. The second is *that* multiple UAVs with the same policy interact with environments, which is helpful for UAVs to discover more unknown environments. The exploration ability and convergence speed are improved *simultaneously*. *Due to the improvement in exploration, the problem of local optimization has been also alleviated.*

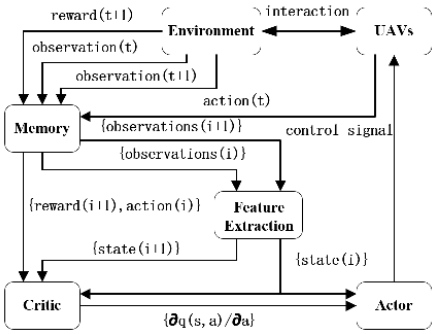


FIGURE 3. *The improved DRL framework for UAV target tracking. Feature extraction module connected with Actor-Critic modules directly, which is convenient for joint training. Multi-UAVs explore environments cooperatively, which speed up the convergence and alleviate the local optimization.*

B. REWARD FUNCTION

Reinforcement learning uses reward to estimate the expected return and gets the optimal strategy. The setting of reward function is closely related to the quality of training results. A simpler method is to set sparse rewards based on results, that is to say, each episode only gives positive or negative rewards according to whether the *mission* can be fulfilled *or not*. This method has the advantage of strong applicability and can be used in various environmental models, while the disadvantage is that the convergence speed is *slow* for the update of network only at the end of each episode, and the algorithm is easy to get stuck at locally optimal value for random exploration.

In order to improve the efficiency and *practicability*, a non-sparse reward is designed to guide UAV tracking and obstacle avoidance in the special application environment, which consists of LOS reward, distance reward, terminal reward and action penalty. LOS is the line between UAV and target, and the LOS reward is designed as:

$$r_{LOS} = \lambda(\frac{\pi}{2} - |\chi - \psi|) \tag{19}$$

where λ is a positive constant, $|\chi - \psi|$ is the LOS angle in velocity coordinate system. The smaller the LOS angle is, the faster UAV approach to the target. Once the LOS larger than $\frac{\pi}{2}$, UAV will fly away from the target. *The physical significance of r_{LOS} is that no matter where the UAV is, it can get a higher reward as long as flies towards the target. r_{LOS} is the most important reward to guide the UAV fly to the target.*

The obstacle reward is designed as:

$$r_{obstacle} = \sigma\{[\sum_{i=1}^{n}(\frac{1}{d_i} - \frac{1}{d_{max}})cos\frac{i\pi}{n-1}]^2 + [\sum_{i=1}^{n}(\frac{1}{d_i} - \frac{1}{d_{max}})sin\frac{i\pi}{n-1}]^2\} \tag{20}$$

where σ is a negative constant. The reward is transformed from artificial potential field, which *represents* the overlap value of obstacle repulsive fields in sensor directions.

The terminal reward relate to the success of the mission, which *is* designed as:

$$r_{terminal} = \begin{cases} -k & if\ d_i \le d_{min1}, \quad i \in [1,n] \\ k & if\ d_0 \le d_{min2} \\ 0 & else \end{cases} \tag{21}$$

where k is a positive constant and d_{min1} is the threshold of obstacle avoidance, d_{min2} is the threshold of target tracking. *$r_{terminal}$ is a sparse reward whose physical significance is that no matter what the direction of the UAV is, it can get a higher reward as long as its distance from the target is less than a threshold, and it will get a higher penalty as long as its distance from the obstacles is less than another threshold. If any of these two conditions are triggered, the UAV will be re-initialized. The main function of $r_{terminal}$ is to guide UAV avoid obstacles and hover around the target.*

In order to make *trajectories* smoother, an action penalty is given as:

$$r_{action} = \alpha|\Delta a| \tag{22}$$

where α is a negtive constant, Δa indicates the change of *actions* in adjacent time.

To summarize, we *give* the final reward function:

$$r = r_{LOS} + r_{obstacle} + r_{terminal} + r_{action} \tag{23}$$

In previous studies, the sparse reward $r_{terminal}$ is a common reward function. However, we found that the model trained by $r_{terminal}$ has a high probability to fall into local optimum. The UAV controlled by the local optimum model only consider the obstacle avoidance while ignoring the target. The local optimization problem has been alleviated when using multi-UAVs to explore environment cooperatively, while the training results are still unstable. In our study, we design the reward r_{LOS}, UAVs will head to the target to get more rewards. The application of r_{LOS} reduces the meaningless patrols and circling, improves the training successful ratio greatly, and alleviate the local optimization problem further.

C. ACTOR RECURRENT CRITIC

The tasks for ground target tracking are performed in dynamic and partially observable environments where the observations *are* quite different from the states. We receive the observation and corresponding action at a certain time, but the reward may comes later, which can be illustrated *by* the function:

$$r(o_t, a_t) = \sum_{i=1}^{T} \kappa_t^{t-T+i} r'(o_{t-T+i}, a_{t-T+i}) \tag{24}$$

where $r(o_t, a_t)$ indicates the reward received in time step t, and $r'(o_{t-T+i}, a_{t-T+i})$ *indicates* the reward produced in time step $t - T + i$. Define $\kappa_t^{t-T+i} r'(o_{t-T+i}, a_{t-T+i})$ as a part of reward which is produced in time step $t - T + i$ while received in time step t, where $\kappa_t^{t-T+i} > 0$ is a discount factor and $\sum_i^T \kappa_t^{t-T+i} = 1$. For the function (20), we have the conclusion that the reward we detected in each time step is a

function about history observations which can be represented as a trajectory:

$$O_0, A_0, O_1, A_1, O_2, \ldots, S_t$$

In order to speed up the computation, we set a maximum history length T, use the historical observations from $t-T$ to t to train the network in each time t. For the actions could get from the *adjacent* observations, *it* can be *omitted*. We have the history as:

$$h_t = o_{t-T}, o_{t-t+1}, \ldots, o_t \tag{25}$$

DDPG updates the value network with the detected reward in each step, so the value *network can be seen as* a function about h_t. We have already *known* in *equation* (3) that value *function* is a function about state, so that we can simulate the state by history. Recurrent networks could synthesize the historical observations, have a better representation of states. LSTM is a kind of excellent recurrent neural networks, which consists of keep gate, write gate and read gate, it has a great ability *of* controlling historical information *to* participates in training. Using the LSTM network to simulate the state from observation history, We have the state as:

$$s_t = f(h_t; \omega) \tag{26}$$

where $f(\cdot)$ is determined by the LSTM network, ω *represents* the parameters of LSTM.

There are two kinds of networks in DDPG. Actor *network* is used to adjust the parameters of policy, determine the best action in a specific state. Critic *network* is used to evaluate the value of current action. We improve the framework of DDPG and name the new structure as ARC(Actor - Recurrent - Critic Network). The main structure of the ARC network *is* shown in FIGURE 4. The actor network and the critic network are *consisted* of Dense network, which computes the continuous actions and value functions respectively. LSTM and actor combine together *to* make up the policy network, *the value network is consisted of LSTM and critic.* Policy and value networks share the same structure and parameters of LSTM. The policy *is* defined as:

$$a = \pi(h_t; \omega, \theta) \tag{27}$$

where θ *represents* the parameters of actor network. The value *is* defined as:

$$q = Q(h_t, a_t; \omega, \eta) \tag{28}$$

where η *represents* the parameters of critic network. In each time step, *we* update parameters according to (29)-(32).

$$y = r_{t+1} + \gamma Q(h_{t+1}, a_{t+1}) \tag{29}$$

$$\eta \leftarrow \eta + \alpha[y - Q(H_t, A_t)]\nabla_\eta Q(h_t, a_t; \omega, \eta) \tag{30}$$

$$\omega \leftarrow \omega + \alpha[y - Q(h_t, a_t)]\nabla_\omega Q(h_t, a_t; \omega, \eta) \tag{31}$$

$$\theta \leftarrow \theta + \beta\nabla_\theta \pi(h_t; \omega, \theta)[\nabla_a q(h_t, a; \omega, \eta)]_{a=\pi(h_t;\omega,\theta)} \tag{32}$$

Algorithm 2 provides the overall steps of our ARC algorithm.

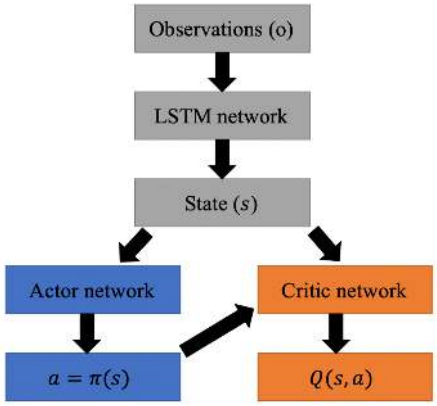
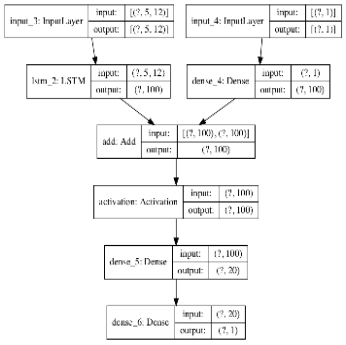
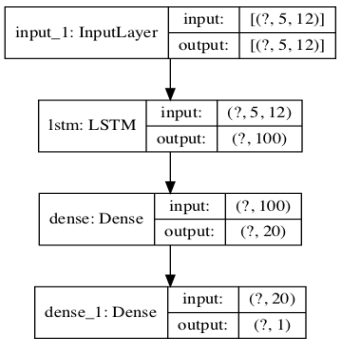


FIGURE 4. Actor - Recurrent - Critic(ARC) networks architecture. The LSTM network simulate the state; The actor network selects actions; The critic network applies to estimate state-action values.



(a) value network



(b) policy network

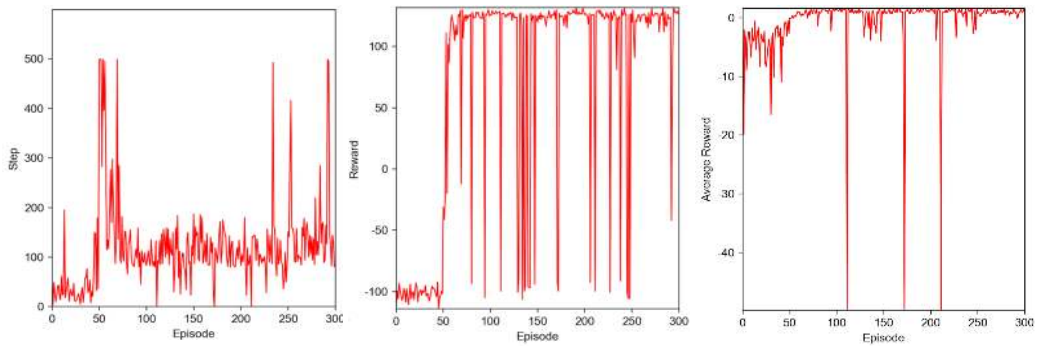
FIGURE 5. Policy and value networks of ARC. The number of trainable params of policy network is 47,241, and the number of trainable params of value network is 47,441. These two networks share the same LSTM structure and parameters.

IV. EXPERIMENTS

In this section, simulation experiment is carried out based on TensorFlow2.0 - python, the network and *hyper parameters* required by the experiment are given, and the experimental results are analyzed.

A. EXPERIMENTAL SETTINGS

There are two kinds of network structures in DDPG – the value and the policy, which are shown in FIGURE 5. The



(a) Steps in each episode (b) Reward in each episode (c) Average reward in each episode

FIGURE 6. Simulation results of ARC. The x-axis represent the number of episodes, and the y-axis represent the steps, reward and average reward in each episode respectively. Approach the target or colliding with an obstacle indicates the end of an episode, and each episode have the most time steps 500.

Algorithm 2 Actor - Recurrent - Critic Algorithm

1: Initialize policy network $\pi(h)$ and value network $q(h, a)$ with parameters ω, θ and η;
Initialize target policy network $\pi'(h)$ and target value network $q'(h, a)$ with parameters $\omega' \leftarrow \omega$, $\theta' \leftarrow \theta$ and $\eta' \leftarrow \eta$;
Initialize the learning rate of target network ε, batch size N, maximum history length T, replay memory R;
2: Take action action according to the historical observation $a_t = \pi(h_t)$;
3: Executes the action a_t, receives the reward r_{t+1}, acquire new observation o_{t+1};
4: Update the history $h_{t+1} = o_{t-T+1}, o_{t-t+2}, \ldots, o_{t+1}$ and save $\{h_t, a_t, r_{t+1}, h_{t+1}\}$ to the memory;
5: Sample a batch size of N datas $\{(h_i, a_i, r_{i+1}, h_{i+1})\}_{i=1}^{N}$ from memory randomly;
6: Update the policy network and value network:

$$y_i = r_{i+1} + \gamma q(h_{i+1}, \pi(h_{i+1}; \omega', \theta'); \omega', \eta')$$
$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_i [y_i - q(h_i, a_i; \omega, \eta)] \nabla_\eta q(h_i, a_i; \omega, \eta)$$
$$\omega \leftarrow \omega + \alpha \frac{1}{N} \sum_i [y_i - q(h_i, a_i; \omega, \eta)] \nabla_\omega q(h_i, a_i; \omega, \eta)$$
$$\theta \leftarrow \theta + \beta \frac{1}{N} \sum_i \nabla_\theta \pi(h_i; \omega, \theta) [\nabla_a q(h_i, a; \omega, \eta)]_{a=\pi(h_i;\omega,\theta)}$$

7: Update the target networks

$$\eta' \leftarrow \varepsilon\eta + (1-\varepsilon)\eta'$$
$$\omega \leftarrow \varepsilon\omega + (1-\varepsilon)\omega'$$
$$\theta' \leftarrow \varepsilon\theta_c + (1-\varepsilon)\theta'$$

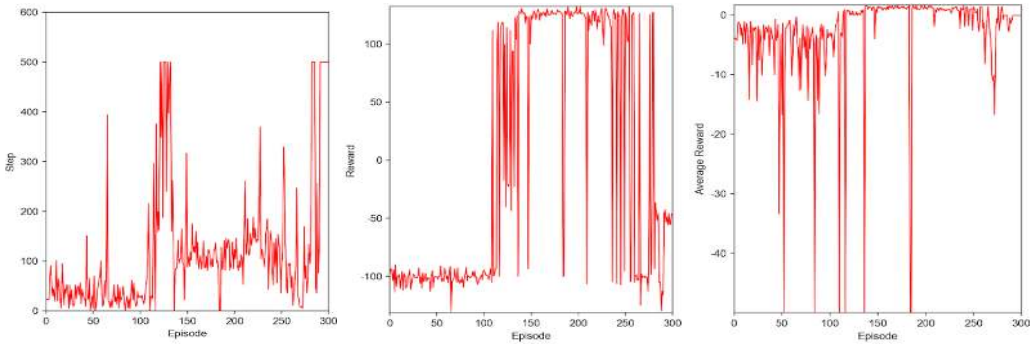
8: Back to step 2 untill to the maximum number of training.

layer of LSTM *is* used to extract feature informations from history observations, those two networks share the same LSTM and update it synchronously. In our environments, the speed is set to $V = 3m/s$, the action range of UAVs set as $[-\pi/20, \pi/20]$ *and the sampling time-interval* $\Delta t = 1s$. *According to equation(18), we have the normal acceleration* a' *with small range* $[-0.47, 0.47]m/s^2$ *which makes the generated trajectories easier to follow.* The observations of UAV are normalized to [0,1] and the action signal is normalized to [−1,1]. The reward is instantiated as: $\lambda = 1, \sigma = 0.1, k = 100, \alpha = 0.01$. The maximum history length is set to 5. The capacity of memory is set to 4000. The batch size is set to 32. RMSprop optimizer [41] is employed to learn the network parameters with a learning rate of 10^{-3}. The discount factor is $\gamma = 0.9$ and the soft target update rate is $\varepsilon = 0.01$. The exploration noise is set to $Var(-0.2, 0.2)$. The number of train episodes is n = 300, the maximum steps in each episode is m = 500.

B. SIMULATION AND PERFORMANCE ANALYSIS

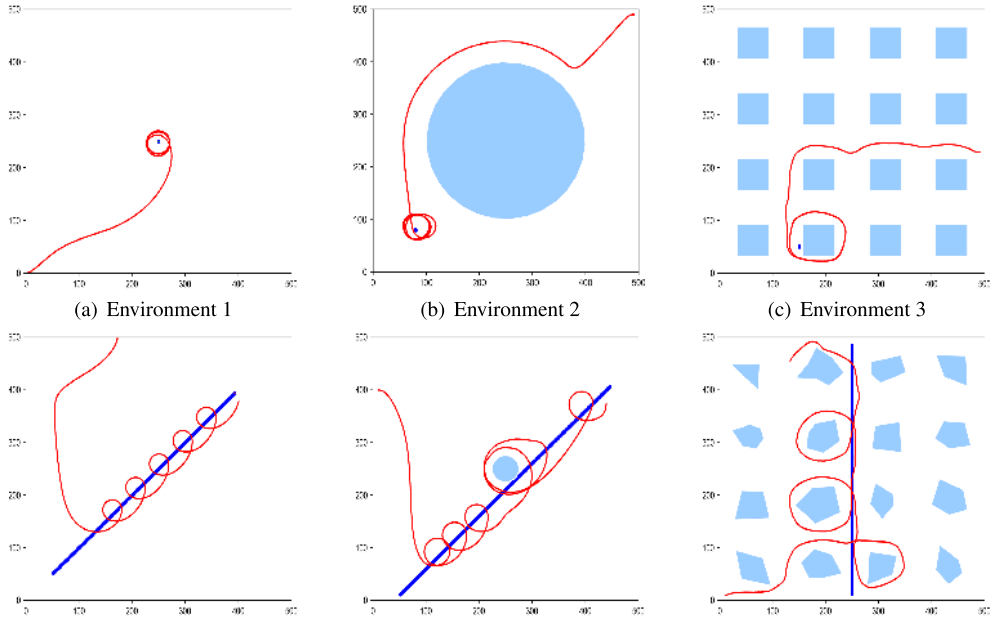
In this subsection, the training process *is* given. We observe the number of steps and rewards in each episode, *statistic and analysis their changes* from the initial to the convergence. FIGURE 6 shows the simulation results, where (a) represents the number of training steps in each episode. In the first 40 episodes, the training steps of UAV are less than 100, which indicates the termination condition is triggered, i.e. the UAV will collision with obstacles. Then the number of training steps began to increase, *creep up to 100*, or even arrive at 500, which indicates that the UAV learned how to avoid obstacles. After about 70th episode, the number of training steps *began to decrease and stable between 100 and 200 finally*, which indicates that the UAV learned better policy and *could* approach the target successfully.

The purpose of DRL is to improve the cumulative reward through continuous learning, so as to obtain the maximum cumulative reward. Therefore, the higher the reward is, the better the *traning effect is*. The cumulative reward *of* each episode is shown in FIGURE 6 (b). By comparison, we find that the trend of cumulative reward is consistent with the *obtained analysis results.* The average reward for each step in each episode reflects the effect of the *training* process. As shown in FIGURE 6 (c), the average reward *increase*



(a) Steps in each episode (b) Reward in each episode (c) Average reward in each episode

FIGURE 7. Simulation results of DDPG. The x-axis represent the number of episodes, and the y-axis represent the steps, reward and average reward in each episode respectively. Approach the target or colliding with an obstacle indicates the end of an episode, and each episode have the most time steps 500.



(a) Environment 1 (b) Environment 2 (c) Environment 3

(d) Environment 4 (e) Environment 5 (f) Environment 6

FIGURE 8. Some examples of the simulated complete environments. Obstacles in different types of environments are distinguished from shapes, sizes and numbers.

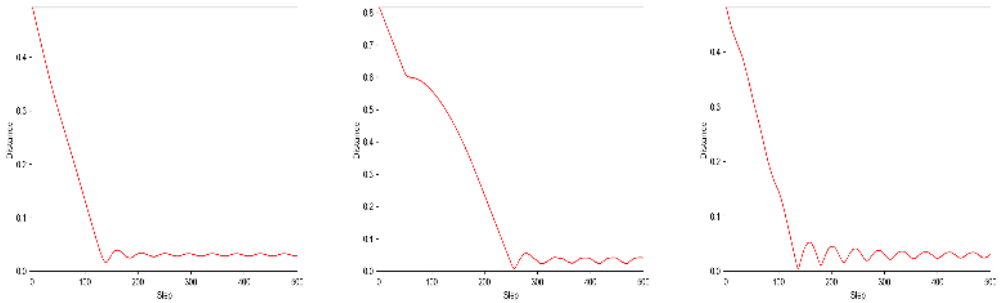
gradually. After about *100th* episode, the average reward *reaches* the top and *stays* stable. *Due to the exploration noise and random initial state,* UAV *collides* with obstacles in a small number of episodes *results* in low average reward.

In order to prove the reliability of the improved method further, traditional DDPG with the same reward function and *hyper* parameters is used for comparison, and the results *are* shown in FIGURE 7. The simulation shows that the number of training steps began to convergence after about 130th episode, which is slower than the improved method. Due to inadequate exploration, there are large fluctuations in the last 20 episodes and the reward *drops* considerably. It can be found that both *the* stability and *the* convergence speed are *enhanced significantly* in the proposed *method*.

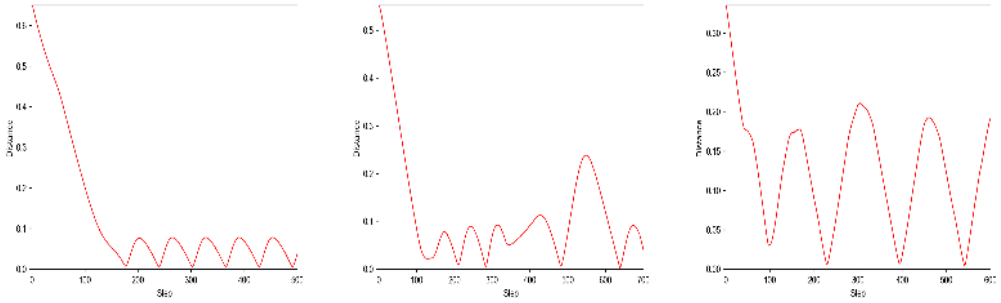
C. EXPERIMENT RESULT

This subsection shows the effect of target tracking by trajectories and the normalized distance between UAV and target. FIGURE 8 shows the trajectories in different environments, and FIGURE 9 shows *their* normalized tracking distance *correspondingly*.

Environment 1 shows the tracking *results* for stationary target in a barrier-free environment. At first the UAV approaches the target quickly, and then *circles* around the target to keep continuously *tracking* and *observing* within the range of maneuverability. Environment 2 shows the tracking *results* for stationary target in the case of simple obstacle interference. UAV can avoid *the obstacle successfully and approach the target quickly*. In the final phase, UAV also



(a) Environment 1 (b) Environment 2 (c) Environment 3



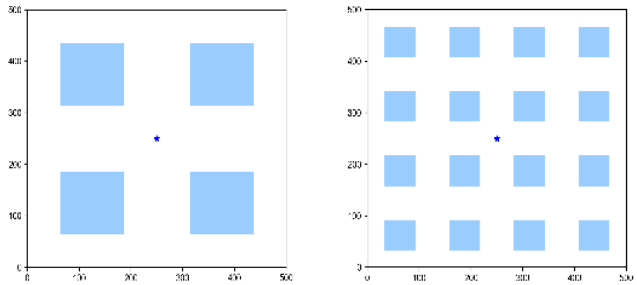
(d) Environment 4 (e) Environment 5 (f) Environment 6

FIGURE 9. Some examples of the simulated complete environments. Obstacles in different types of environments are distinguished from shapes, sizes and numbers.

flights around the target. Environment 3 shows the tracking results for stationary target *in a environment with complex obstacles.* Due to the *dynamic constraints and interference of obstacles*, the UAV cannot fly around the target. However, we can find that, the UAV can still complete the *task of* continuous observation and tracking by keeping a certain distance from the target on the premise of guaranteeing its own safety.

Environment 4-6 shows the tracking results for moving target. The moving speed is set to be 0.8m/s which is less than the speed of UAV. Therefore, the UAV *has the ability to fuifil the tracking task* under the correct guidance. Environment 4 shows the tracking results in a barrier-free environment. The results show that after approaches the target, the UAV always hovers within a certain range of the target and can observe and track the target stably. Environment 5 shows the tracking results for moving targets *in a environment with a simple obstacle*. The UAV approaches the target quickly and *keeps tracking* in the initial stage. When the obstacle is encountered, after a short adjustment to ensure flight safety, UAV flies to the target again and maintain tracking. Environment 6 shows the tracking results *for* the moving target in the environment *with complex obstacles*. Due to the density of obstacles and the *constraint of flying ability*, the UAV cannot maintain a stable observation distance to the target, but it can still fly to the target on the premise of avoiding obstacles.

In order to prove the effectiveness and practicability further. *Two environments shown in FIGURE* 10 *are selected to repeat the experiment.* The target is set to stationary in the center, and the initial state of UAV is random in each



(a) Environment 7 (b) Environment 8

FIGURE 10. Environments used for repeat the experiment. Environments with stationary target in the center. Obstacles in different types of environments are distinguished from shapes, sizes and numbers.

episode. For target tracking problem in the environment 7 *which* has sparse obstacles in it, compared with the traditional DDPG algorithm, the success rate of the improved algorithm is increased from 70.0% to 91.8%. For target tracking problem in the environment 8 *which* has dense obstacles in it, the success rate is increased from 13.6% to 67.5%.

V. CONCLUSION AND PROSPECT

In this paper, we improve the DDPG algorithm make it more suitable for UAV target tracking. The simulation *results show* that the training process is more stable and the convergence is faster. *In the verification process, we observe that the UAV could generate the collision free trajectory for target tracking. Besides, the failure rate decreases significantly compared with traditional DDPG.*

Despite the better results, there are still areas for improvement. The following scheme is proposed for future work.

1) State space. The visual based DRL method can extract the obstacle information directly from the depth images collected by the camera, but it can't detect the target position information. One solution is that the relative position information of the target is calculated and *fused* with the images. Then the expanded high-level image information *can be* used as the input of DRL.

2) Reward function. The design of reward function is *crucial* to the *training* effect of DRL. *In order to obtain satisfactory results, we have compared more than 10 kinds of reward functions. More effective method for reward function definition will be the focus of the follow-up research.*

3) *Combination with rule-based methods. Although the target tracking method based on DRL can ensure convergence, it lacks security and practicability. The rule-based path planning algorithms are usually more stable and effective. Therefore, the combination of DRL with rule-based methods will be a promising research direction, which can not only deal with complex and changeable environment, but also enhance the stability and efficiency.*

REFERENCES

[1] C. Hu, Z. Zhang, Y. Tao, and N. Wang, "Decentralized real-time estimation and tracking for unknown ground moving target using UAVs," *IEEE Access*, vol. 7, pp. 1808–1817, 2019.

[2] J. Kim and Y. Kim, "Moving ground target tracking in dense obstacle areas using UAVs," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 8552–8557, 2008.

[3] 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.

[4] J. Wu, H. Wang, Y. Huang, Z. Su, and M. Zhang, "Energy management strategy for solar-powered UAV long-endurance target tracking," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 4, pp. 1878–1891, Aug. 2019.

[5] K. Balakrishnan, P. Narayanan, and M. Lakehal-Ayat, "Automatic navigation using deep reinforcement learning," U.S. Patent 15 944 563, Oct. 3, 2019.

[6] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[7] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[8] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, "An improved Sarsa(λ) reinforcement learning algorithm for wireless communication systems," *IEEE Access*, vol. 7, pp. 115418–115427, 2019.

[9] H. V. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2613–2621.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[11] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid path planning with deep reinforcement learning: preliminary results," *Procedia Comput. Sci.*, vol. 123, pp. 347–353, 2018.

[12] B. Zhang, Z. Mao, W. Liu, and J. Liu, "Geometric reinforcement learning for path planning of UAVs," *J. Intell. Robot. Syst.*, vol. 77, no. 2, pp. 391–409, Feb. 2015.

[13] Z. Yijing, Z. Zheng, Z. Xiaoyi, and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," in *Proc. 36nd Chin. Control Conf. (CCC)*, 2017, pp. 3397–3402.

[14] C. Yan and X. Xiang, "A path planning algorithm for UAV based on improved Q-learning," in *Proc. 2nd Int. Conf. Robot. Automat. Sci. (ICRAS)*, Jun. 2018, pp. 1–5.

[15] Y. Zhang, Q. H. Vuong, K. Song, X.-Y. Gong, and K. W. Ross, "Efficient entropy for policy gradient with multidimensional action space," 2018, *arXiv:1806.00589*. [Online]. Available: https://arxiv.org/abs/1806.00589

[16] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[17] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, "Q-Prop: Sample-efficient policy gradient with an off-policy critic," 2016, *arXiv:1611.02247*. [Online]. Available: https://arxiv.org/abs/1611.02247

[18] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for linearized control problems," in *Proc. ICLR*, 2018, pp. 1–28.

[19] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*. [Online]. Available: https://arxiv.org/abs/1802.09477

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

[21] P. Khuntia and R. Hazra, "An actor-critic reinforcement learning for device-to-device communication underlaying cellular network," in *Proc. IEEE Region Conf. (TENCON)*, Oct. 2018, pp. 50–55.

[22] M. Zhaowei, N. Yifeng, and S. Lincheng, "Vision-based behavior for UAV reactive avoidance by using a reinforcement learning method," in *Proc. 12th World Congr. Intell. Control Automat. (WCICA)*, Jun. 2016, pp. 3301–3306.

[23] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 3357–3364.

[24] 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.

[25] S. You, M. Diao, and L. Gao, "Deep reinforcement learning for target searching in cognitive electronic warfare," *IEEE Access*, vol. 7, pp. 37432–37447, 2019.

[26] T. Wang, R. Qin, Y. Chen, H. Snoussi, and C. Choi, "A reinforcement learning approach for UAV target searching and tracking," *Multimedia Tools Appl.*, vol. 78, no. 4, pp. 4347–4364, Feb. 2019.

[27] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.

[28] S. Hanna, H. Yan, and D. Cabric, "Distributed UAV placement optimization for cooperative line-of-sight MIMO communications," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 4619–4623.

[29] G. Li, Y. Wu, and P. Xu, "Fixed-time cooperative guidance law with input delay for simultaneous arrival," *Int. J. Control*, pp. 1–10, Sep. 2019.

[30] Y.-B. Chen, G.-C. Luo, Y.-S. Mei, J.-Q. Yu, and X.-L. Su, "UAV path planning using artificial potential field method updated by optimal control theory," *Int. J. Syst. Sci.*, vol. 47, no. 6, pp. 1407–1420, Apr. 2016.

[31] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, G. Xu, H. Li, and X. Liang, "UAV autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, vol. 7, pp. 117227–117245, 2019.

[32] A. Gruslys, W. Dabney, M. Gheshlaghi A. B. Piot, M. Bellemare, and R. Munos, "The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning," 2017, *arXiv:1704.04651*. [Online]. Available: https://arxiv.org/abs/1704.04651

[33] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare, "Dopamine: A research framework for deep reinforcement learning," 2018, *arXiv:1812.06110*. [Online]. Available: https://arxiv.org/abs/1812.06110

[34] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[35] G. G. Henry, T. Parks, and K. T. O'Brien "Neural network unit that performs concurrent LSTM cell calculations," U.S. Patent 10 380 481, Aug. 13, 2017.

[36] Y. Lakretz, G. Kruszewski, T. Desbordes, D. Hupkes, S. Dehaene, and M. Baroni, "The emergence of number and syntax units in LSTM language models," 2019, *arXiv:1903.07435*. [Online]. Available: https://arxiv.org/abs/1903.07435
[37] D. R. Song, C. Yang, C. McGreavy, and Z. Li, "Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge," 2017, *arXiv:1710.02896*. [Online]. Available: https://arxiv.org/abs/1710.02896
[38] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. Lau, "Dynamic virtual machine management via approximate Markov decision process," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[39] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The uncertainty bellman equation and exploration," 2017, *arXiv:1709.05380*. [Online]. Available: https://arxiv.org/abs/1709.05380
[40] G. Li, Y. Wu, and X. Liu, "Adaptive fixed-time consensus tracking control method for second-order multi-agent systems with disturbances," *J. Franklin Inst.*, to be published.
[41] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of Adam and RMSProp," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11127–11135.

BOHAO LI was born in Hebei, China, in 1990. He received the B.E. degree from Lanzhou University, Lanzhou, China, in 2012, and the M.S. degree from the Lanzhou University of Technology, Lanzhou, in 2017. He is currently pursuing the Ph.D. degree in navigation, guidance and control with Beihang University, Beijing, China. His research interests include deep learning, deep reinforcement learning, and guidance.

YUNJIE WU was born in 1969. She received the Ph.D. degree in navigation guidance and control from Beihang University, Beijing, China, in 2006. She is currently a Professor with the School of Automation Science and Electrical Engineering, Beihang University. Her research interests include system simulation, intelligent control, servo control, aircraft guidance, and control technology.