Received February 22, 2019, accepted March 12, 2019, date of publication March 18, 2019, date of current version April 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905649

Deep Reinforcement Learning for Target Searching in Cognitive Electronic Warfare

SHIXUN YOU, MING DIAO, AND LIPENG GAO
College of Information and Communication, Harbin Engineering University, Harbin 150001, China

Corresponding author: Lipeng Gao (gaolipeng@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61701134, and in part by the Equipment Pre-Research Fund of China under Grant 61404150101.

ABSTRACT The recent appreciation of deep reinforcement learning (DRL) arises from its successes in many domains, but the applications of DRL in practical engineering are still unsatisfactory, including optimizing control strategies in cognitive electronic warfare (CEW). CEW is a massive and challenging project, and due to the sensitivity of the data sources, there are few open studies that have investigated CEW. Moreover, the spatial sparsity, continuous action, and partially observable environment that exist in CEW have greatly limited the abilities of DRL algorithms, which strongly depend on state-value and action-value functions. In this paper, we use Python to build a 3-D space game named Explorer to simulate various CEW environments in which the electronic attacker is an unmanned combat air vehicle (UCAV) and the defender is an observation station, both of which are equipped with radar as the observation sensor. In our game, the UCAV needs to accomplish the task of detecting the target as early as possible to perform follow-up tracking and guidance tasks. To allow an "infant" UCAV to understand what "target searching" is, we train the UCAV's maneuvering strategies by means of a well-designed reward shaping, a simplified constant accelerated motion control, and a deep deterministic policy gradient (DDPG) algorithm based on a generative model and variational Bayesian estimation. The experimental results show that when the operating cycle is 0.2 s, the search success rate of the trained UCAV in 10 000 episodes is improved by 33.36% compared with the benchmark, and the target destruction rate is similarly improved by 57.84%.

INDEX TERMS Deep reinforcement learning, cognitive electronic warfare, motion planning, deep deterministic policy gradient, variational Bayesian estimation, target searching.

I. INTRODUCTION

From the current achievements of deep reinforcement learning (DRL), visual observations and simple action instructions can motivate an intelligent agent to create value in many occasions and even gain the potential to surpass human beings. However, in general test environments, such as Atari 2600 games, state, action and reward require clear representations, which limits the universality of traditional DRL algorithms. DRL algorithms have different designs for different scenarios to achieve excellent performance. To produce a more promising application platform, we look in a novel direction – approximate end-to-end cognitive electronic warfare (CEW) simulation environments.

CEW, which is a very comprehensive concept, primarily includes advanced technologies such as automatic vehicle control, electronic countermeasures, multisource sensor fusion and so on. Members of CEW include one or more intelligent agents that are responsible for electronic attacks and electronic defenses. The operating strategies between each agent reflect the situation of the battlefield in real time. An unmanned combat air vehicle (UCAV) is the most ideal operational carrier in CEW due to its high maneuverability, multiweapon mounting capacity, excellent synergy, and low cost. A radar sensor system is primarily responsible for sensing the CEW environment; thus, partially observable tasks are the basic form of electronic warfare missions. CEW missions include four parts: target searching/detection, acquisition, tracking, and guidance. Since the battlefield situation is rapidly changing and all parts are continuous-time tasks, the shorter the operating cycle is, the better is the

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

decision-making effect. Dynamic programming is considered to be one approach to optimize the operating strategy at different stages, but a mission generally lasts tens to thousands or even tens of thousands of execution cycles. Clearly, the scale of CEW engineering or the model is the key factor that has restricted its development in recent years. Additionally, when performing tasks, the UCAV needs to perform motion planning for different scenarios. However, unlike path planning based on arrival point generation, motion planning is a more bottom-level task and needs to respect certain physical constraints of the body, including overload coefficients (radial overload and normal overload, which are related to the acceleration that an aircraft can sustain during acceleration and turning). In short, we do not need to design a single mathematical model that describes the entire CEW process because such a task would be impossible.

As the most basic task in CEW, target searching is also a necessary process to obtain prior knowledge for subsequent tasks. Unfortunately, in existing solutions for either CEW or traditional EW, there is no detailed consideration of how an electronic attacker or defender can make target searching decisions, or it is even assumed that the state information of the target is available a priori. Such a CEW model is meaningless.

According to the above factors, we built a simple but full-featured CEW test platform with Python, and we created a game named Explorer according to a real target searching task. Explorer is a flying-driving game. By controlling a UCAV, players can navigate in a limited perspective, as long as they find the target defense station within a given three-dimensional (3D) map and time limit. At the beginning of the experiments about CEW, our team found an interesting phenomenon: for a learning agent in their infancy, "target" itself is a very vague concept; in fact, the process of the UCAV constantly searching for a target in a 3D space can be the result of understanding the meaning of "target searching" and needs to be trained as knowledge.

An instance of Explorer can be parameterized by two key quantities. The first is the maneuvering characteristics of the UCAV, including maximum flight speed and overload. These characteristics determine the motion modes of the UCAV in all missions. The second quantity is the sampling/operating cycle. Additionally, these two quantities directly determine the density of motion planning, i.e., the sparsity of win/loss signals as rewards. The Explorer environment therefore allows us to evaluate and test the long-term behaviors of attackers and defenders with DRL algorithms and to develop insights about the robustness of solutions to changes in the environment. Analyses of CEW problems have generally been approached by much more abstract methods given the complexity in finding the step-by-step optimal representation.

II. RELATED WORK

Our work is related to both CEW and DRL; hence, briefly presenting the related works in these two areas is essential.

A. COGNITIVE ELECTRONIC WARFARE

The development of EW has been focused on the three aspects of electronic support measure, electronic countermeasure, and electronic counter-countermeasure systems [1]. Notably, in recent years, research on the cognitive system model of the EW environment has achieved great progress [2]–[5]. Noh and Jeong [6] introduced a threat model to realize an autonomous decision-making process for threat detection, classification, and the selection of alternative countermeasures against threats in EW settings, and they also presented a methodology that compiles the threat into a set of rules using soft computing methods. Based on artificial brain theory, the autonomous control system proposed by Meng *et al.* [7] is divided into three layers: perception fusion layer, cognition and decision layer, and behavior control layer. By controlling the speed and direction of the motion, the required track of the UCAV is realized, and the integrity of the system function is verified. Chen *et al.* [8] used Bayesian estimation to improve the ability of distinguishing underwater terrain and navigation ability of an autonomous underwater vehicle. Ogren *et al.* [9] aimed to solve the system conflict problem of a UCAV when performing different tasks. The main idea is to calculate the priority of each task according to the current level of satisfaction and then report it to the kernel controller. In addition, the electronic attack mode with a jammer as the main body has also received widespread attention [10], [11]. Osner and Plessis [12] designed a threat evaluation and jamming allocation system and implemented an exhaustive method to achieve the optimal cooperative jamming strategy of the UCAV to counter the target radar network, but it only lasted for 15 sampling cycles. Furthermore, considering that combat vehicles such as UCAVs are the agents that execute the decision results of all CEW systems, it is desirable that they have excellent maneuverability and situational awareness. Many methods have been proposed to overcome the various problems encountered by UCAVs during combat, such as target searching, collision avoidance, and motion planning [13]–[16], however, from the perspective of target searching, these works has failed to involve an end-to-end motion planning framework.

When a combat vehicle possesses a limited detection range in an unknown environment, the traditional target search policy is to first grid the mission map and then use carpet reconnaissance or Bayesian inference to calculate the possible position of the target [17], [18]. Of course, evolutionary computation methods can also be used to find an optimal search strategy by optimizing the fitness function to reduce the environmental uncertainty [19]. Unfortunately, both the continuous mission space and the approach to motion planning are simplified in the works mentioned above, which results in a large gap between the experimental results and the engineering requirements for end-to-end CEW (environment reconnaissance to action decision-making). Moreover, these methods are not sufficiently intelligent, resulting in unpromising application scenarios. Based on reasonable hypotheses and logical analysis approaches for proceeding from the collection and analysis of sensor data to the operation of combat vehicles, our achievement is an approximate end-to-end CEW test platform that maintains a perfect balance between the preservation of operating forms and the simplification of mathematical models.

B. DEEP REINFORCEMENT LEARNING

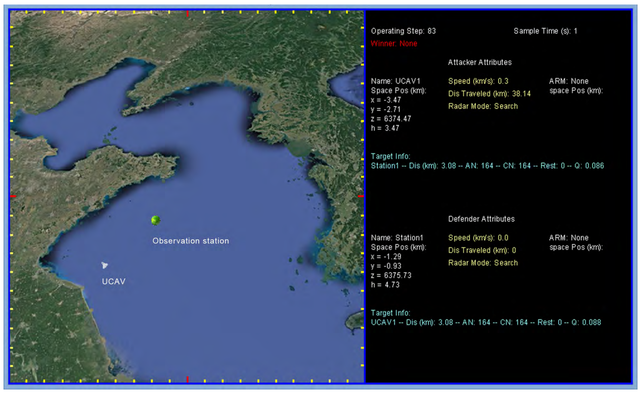
Deep learning is enabling reinforcement learning (RL) to scale to decision-making problems that were previously intractable. The intelligent agent with Deep Q-learning network (DQN) created by DeepMind's team displays a level that surpasses human beings in Atari 2600 games, which was undoubtedly a landmark step for artificial intelligence in industrial applications. Historically, DRL has enabled many amazing achievements [20]–[23] in game playing, simultaneous localization and mapping (SLAM), path planning, and so forth.

Many teams have also contributed to improving DRL algorithms for different application scenarios. Double DQN (DDQN) [24] solves the problem of overestimation caused by the greedy algorithm used in DQN. Deep deterministic policy gradient (DDPG) [25] based on an actor-critic framework enables the learning network to select a unique action in a continuous action space end-to-end. Sung *et al.* [26] proposed a framework of a DQN hybrid generative network and estimation network to guide robots with haptic sensors to complete click tasks in partially observable environments.

Because the agent of DRL is too difficult to train, researchers have performed substantial work [27]–[30] to further improve the performance of the DRL algorithm, including hierarchical parallel computing, modification of the loss function, memory and predictive modeling, and so on. Unfortunately, although these studies have made DRL powerful, there are still very few examples in conjunction with practical projects [4], [23], [31], [32], primarily because many engineering issues cannot be described by simple models or sensors and the DRL algorithms are model dependent. Thus, compared with improving the performance of DRL algorithms, our focus in this work is on how to make DRL effective in CEW environments.

The main contributions of this work are as follows:

- We develop a versatile CEW (vCEW) framework, in which the type of combat members is selectively created, the modules of detection sensors and countermeasure weapons can be expanded freely, and the interaction between each part and the environment has a clear mathematical model.
- We present a state-of-the-art approach for constructing a DRL framework to accomplish some partially observable tasks, such as Explorer (based on vCEW) in a continuous high-dimensional action space, and we provide some empirical tricks for designing algorithms.
- Persuasive simulation results are obtained by designing tasks with different difficulties. Based on the defined behavior angle, we analyze the potential behavior policies learned by the UCAV in optimizing navigation control, and we further explore how the learning unit of DRL understands the action generated during target searching.



```
Operating Step: 83                    Sample Time (s): 1
Winner: None

                    Attacker Attributes

Name: UCAV1         Speed (km/s): 0.3           ARM: None
Space Pos (km):     Dis Traveled (km): 38.14    space Pos (km):
x = -3.47           Radar Mode: Search
y = -2.71
z = 6374.47
h = 3.47

Target Info:
Station1 -- Dis (km): 3.08 -- AN: 164 -- CN: 164 -- Rest: 0 -- Q: 0.086

                    Defender Attributes

Name: Station1      Speed (km/s): 0.0           ARM: None
Space Pos (km):     Dis Traveled (km): 0        space Pos (km):
x = -1.29           Radar Mode: Search
y = -0.93
z = 6375.73
h = 4.73

Target Info:
UCAV1 -- Dis (km): 3.08 -- AN: 164 -- CN: 164 -- Rest: 0 -- Q: 0.088
```

FIGURE 1. The display interface of the Explorer game, with a total pixel size of 600 × 1050; the 600 × 600 panel on the left side displays the map from the global perspective, and the 600 × 450 panel on the right side displays the status information of the UCAV and the observation station.

III. EXPLORER: TARGET SEARCHING IN vCEW

Considering the model dependence of the DRL algorithm, we first introduce the vCEW-based game – Explorer. As shown in Figure 1, a player needs to control the maneuvering of a UCAV (such as tumbling and diving) until it finds the target in a given 3D map. During the manipulation process, the behavior and trends of exploration are considered as much as the outcome of the game. There are two key factors that restrict the UCAV's completion of the game (which are also the major problems to be addressed in CEW):

- The UCAV can only maneuver at high speed in space under the condition of satisfying the physical constraints of its airframe; thus, it is difficult to find the optimal or even feasible solution in real time for long-term (>100 sampling cycles) operation planning.
- The members in the game can only perceive the surrounding situation through radar (only a radar sensor is equipped in Explorer; more model details will be discussed in Section III-C). Since the received signal strength has a minimum threshold and is always accompanied by an error, only partial regions of observation can be provided for players. Consequently, the target searching task has to be modeled as a partially observable Markov decision process (POMDP).

A. MAP

In Explorer, the task region for the UCAV is a tiled space of 15 km in length (from -7.5 km to 7.5 km), 15 km in width (from -7.5 km to 7.5 km) and 7.5 km in height (from 0.5 km to 8 km). The geocentric coordinate system, $OXYZ$, is adopted as the base reference frame, and the coordinates of any point in the space are marked as $\boldsymbol{p} = [x, y, z]^T$ or $\hat{\boldsymbol{p}} = [x, y, h]^T$. The formula for transforming between the Z-axis coordinate

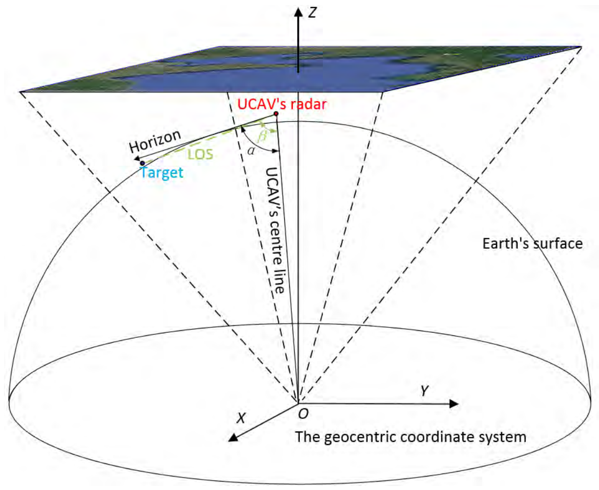


FIGURE 2. **Relationship between the Explorer game space and the projected map.**

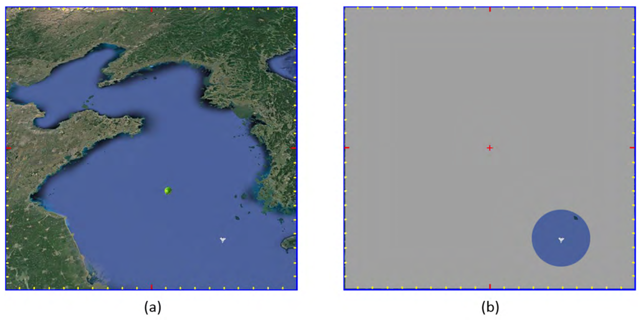


FIGURE 3. **(a) Player's global perspective and (b) maximum detection range of the UCAV (reconnaissance perspective).**

of any point in space and the height from the ground is

$$z = \sqrt{(R_e + h)^2 - x^2 - y^2} \Leftrightarrow \tag{1}$$
$$h = \sqrt{x^2 + y^2 + z^2} - R_e,$$

where R_e is the Earth's radius (6371 km) and h is the radial height of the object from the Earth.

The original map of Explorer can be obtained by projecting the game space in the opposite direction of the Z-axis, as shown in Figure 2. Figures 3 (a) and (b) demonstrate our observation perspective (i.e., the global perspective) and the UCAV's reconnaissance perspective, respectively. The center of the map corresponds to the northernmost point of the Earth. The map size in the vertical view is 600×600 pixels, and the ratio with respect to the real environment is $1 : 25$, i.e., 1 pixel corresponds to 25 m.

From Sections III-B to III-E, we present an overview of the necessary mathematical models in vCEW, which can be skipped by uninterested readers.

B. MANEUVERING CHARACTERISTICS OF THE UCAV

Referring to [16], the local coordinate system and airborne coordinate system of the UCAV can be defined as $O_lX_lY_lZ_l$

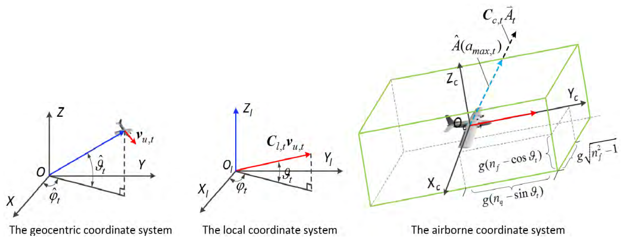

Z
$\boldsymbol{v}_{u,t}$
$\hat{\vartheta}_t$
Y
O
$\hat{\varphi}_t$
X
The geocentric coordinate system
Z_l
$\boldsymbol{C}_{l,t}\boldsymbol{v}_{u,t}$
Y_l
O_l
ϑ_t
φ_t
X_l
The local coordinate system
$\boldsymbol{C}_{c,t}\vec{A}_t$
$\hat{A}(a_{max,t})$
Z_c
Y_c
O_c
$g(n_f - \cos\vartheta_t)$
$g\sqrt{n_f^2-1}$
X_c
$g(n_q - \sin\vartheta_t)$
The airborne coordinate system

FIGURE 4. **Descriptions of the reference frames and constraint boundaries.**

and $O_cX_cY_cZ_c$, respectively. The local coordinate system of the UCAV is used to describe the relationship between the fuselage's posture and the inertial frame. Specifically, the direction of gravity is always towards the center of the Earth when the UCAV is flying in the mission space. By contrast, the airborne coordinate system is based on the position and velocity of the UCAV. Note that the axis O_cY_c points in the direction of the UCAV's velocity at sampling time t, the axis O_cX_c points towards the right wing of the aircraft (O_cX_c is perpendicular to O_cY_c), and O_cX_c, O_cY_c, and O_cZ_c satisfy the right-hand rule. Now, let the transformation matrix used to map vectors from $OXYZ$ to $O_lX_lY_lZ_l$ be

$$\boldsymbol{C}_{l,t} = \begin{bmatrix} \sin\hat{\varphi}_t & -\cos\hat{\varphi}_t & 0 \\ -\cos\hat{\varphi}_t \sin\hat{\vartheta}_t & -\sin\hat{\varphi}_t \sin\hat{\vartheta}_t & \cos\hat{\vartheta}_t \\ \cos\hat{\varphi}_t \cos\hat{\vartheta}_t & \sin\hat{\varphi}_t \cos\hat{\vartheta}_t & \sin\hat{\varphi}_t \end{bmatrix}, \tag{2}$$

where the angles $\hat{\varphi}_t$ and $\hat{\vartheta}_t$ refer to the azimuth and elevation, respectively, of the UCAV's projected location from the Earth. Similarly, the transformation matrix used to map vectors from $O_lX_lY_lZ_l$ to $O_cX_cY_cZ_c$ is given by

$$\boldsymbol{C}_{c,t} = \begin{bmatrix} \sin\varphi_t & -\cos\varphi_t & 0 \\ \cos\varphi_t \cos\vartheta_t & \sin\varphi_t \cos\vartheta_t & \sin\varphi_t \\ -\cos\varphi_t \sin\vartheta_t & -\sin\varphi_t \sin\vartheta_t & \cos\vartheta_t \end{bmatrix}, \tag{3}$$

where φ_t and ϑ_t represent the pitch and rotation angles, respectively, of the UCAV.

At any time, the maneuver performed by the UCAV in different directions is subject to normal overload and radial overload, as shown in Figure 4. According to the formula in [16] and [33], the boundary constraints received by the UCAV when performing motion planning are decoupled as follows:

$$a_{max,t} = (g\sqrt{n_f^2 - 1}, g(n_q - \sin\vartheta_t), g(n_f - \cos\vartheta_t)), \tag{4}$$

where n_f and n_q are the maximum normal overload and maximum radial overload of the UCAV, respectively. g is the gravitational acceleration. Based on $a_{max,t}$, any type of control strategy of planned acceleration is upper bounded. We concentrate on φ_t and ϑ_t as the UCAV's action input, i.e., $A_t = [\varphi_t, \vartheta_t]$, so the acceleration applied to the UCAV is given by

$$\hat{\boldsymbol{a}}_{u,t} = \boldsymbol{C}_{l,t}^{-1}\boldsymbol{C}_{c,t}^{-1}\hat{A}(a_{max,t}), \tag{5}$$

$\hat{A}$ is defined as a function of the maximum magnitude acceleration that the UCAV can generate for motion in the direction A_t.

Furthermore, with a constant acceleration (CA) control strategy, the path planning problem for the UCAV to solve in one sampling interval is

$$\boldsymbol{U}_{t+1} = \boldsymbol{\Phi}_{CA}\boldsymbol{U}_t \Leftrightarrow \begin{bmatrix} \boldsymbol{p}_{u,t+1} \\ \boldsymbol{v}_{u,t+1} \\ \boldsymbol{a}_{u,t+1} \end{bmatrix} = \boldsymbol{\Phi}_{CA} \begin{bmatrix} \boldsymbol{p}_{u,t} \\ \boldsymbol{v}_{u,t} \\ \hat{\boldsymbol{a}}_{u,t} \end{bmatrix}, \qquad (6)$$

where $\boldsymbol{U}_t$ represents the motion state of the UCAV at time t and $\boldsymbol{p}_{u,t}$ and $\boldsymbol{v}_{u,t}$ are the position and velocity, respectively, of the UCAV during the time interval. Additionally, $\boldsymbol{\Phi}_{CA}$ is the CA model:

$$\boldsymbol{\Phi}_{CA} = \begin{bmatrix} \boldsymbol{I} & \tau\boldsymbol{I} & \frac{\tau^2}{2}\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \tau\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \qquad (7)$$

where τ is the length of the sampling cycle, $\boldsymbol{I}$ is the third-order unit matrix, and $\boldsymbol{0}$ is the third-order zero matrix.

Finally, because the speed of the UCAV has an upper bound V_{max}, at time t, the following replanning process is essential [16]:

$$\hat{\boldsymbol{a}}_{ru,t} = \begin{cases} \frac{1}{\tau}((\boldsymbol{v}_{u,t} + \hat{\boldsymbol{a}}_{u,t}\tau)/\varepsilon - \boldsymbol{v}_{u,t}) & \text{if } \varepsilon > 1 \\ \hat{\boldsymbol{a}}_{u,t} & \text{else} \end{cases}, \qquad (8)$$

where $\varepsilon = \left\| \boldsymbol{v}_{u,t} + \hat{\boldsymbol{a}}_{u,t}\tau \right\| / V_{max}$. Equation 8 is used to adjust the impracticable speed generated by the control strategy back to the constraint boundaries.

C. RADAR DETECTION

In Explorer, the UCAV perceives the surrounding area through radar, i.e., the radar is the "eye" of the UCAV. In the vCEW platform, we assume that the UCAV is equipped with an active phase-controlled radar with an electric scanning cycle of an infinitesimal length (generally on the order of μs); thus, when the maximum detection range is ρ_{max} and the conditions given in Appendix A are satisfied, the visibility range for the UCAV can be considered a 360-degree sphere in 3D space. It is well known that radar determines whether a target exists according to the signal strength of the feedback echo. Under the assumption that the radar has been not jammed, the existence of an observable target is determined if the echo signal intensity exceeds its sensitivity. Note that there is no need to consider the terrain or other obstacles because the combat environment in Explorer is very sparse; because there are only two entities, namely, the UCAV and the observation station; and because the height specified by the UCAV's planned motion must be greater than 0.5 km.

After inputting the observation data of the radar into the end-to-end electronic warfare system, the action command of the UCAV will be directly output. However, for Explorer, since the subsystem for simulating the transmit/receive waveform is too complicated, to simplify the radar signal processing for locating the target and finding the direction, the target information with noise, e.g., the captured target's position and velocity with echo analysis, can be directly used as input.

We primarily focus on the relative distance between the radar and target, which is the core factor that affects radar detection. Let the radar's sensitivity and maximum detection range be denoted by κ and ρ_{max}, respectively. The relationship between κ and ρ_{max} satisfies $\rho_{max}^4 \propto \kappa$. For instance, if the detection range of a radar needs to be doubled, then its sensitivity must be increased by a factor of 15.

1) RADAR CROSS SECTION (RCS)

For an advanced UCAV, a layer of stealth coating is often applied to reduce its radar cross section (RCS) to improve its survival probability. Since the stealth effect is related to the observation angle, to simplify the analysis, the RCS of the UCAV, ξ, can be parameterized according to the following formula:

$$\xi = \begin{cases} 1 + \xi_{min} - 2\left\langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \right\rangle / \pi & \text{if} \left\langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \right\rangle \leq \pi/2 \\ 1 & \text{else} \end{cases}, \qquad (9)$$

where ξ_{min} is the minimum RCS of the UCAV, $\boldsymbol{n}_t$ is a unit vector along the line of sight (LOS) from the UCAV to the object, and $\left\langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \right\rangle$ is the angle between the direction of the UCAV's velocity and the LOS direction. Equation 9 models the UCAV as a cone-sphere chimera, as shown in Figure 5 (a). The RCS linearly decreases to ξ_{min} in the forward direction with respect to the UCAV and remains constant at 1 in the tail hemisphere. To simplify the calculation, we directly define $\kappa = \rho_{max}^4$ as an initial condition. Then, the visibility distance of the UCAV can be computed as $\rho = (\kappa\xi)^{0.25} = \rho_{max}\xi^{0.25}$.

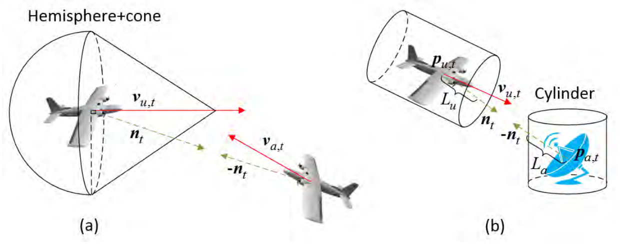


FIGURE 5. (a) Parametric model of the UCAV's RCS and (b) physical collision model of an object in Explorer.

2) LINE-OF-SIGHT (LOS) HIDING

Due to the curvature of the Earth's surface and the linear propagation of electromagnetic waves in space, the low altitude of the target will have a hiding effect on the high-altitude detector; the schematic diagram is shown in Figure 2. As shown in Figure 2, α is the angle between the UCAV's center line and the horizon, and β is the angle between the UCAV's center line and the LOS. Clearly, when α is larger than β, the target is invisible to radar, and vice versa.

Therefore, the following criteria are used as the basis for determining whether the target is detectable:

$$\alpha < \beta \ \& \ \left\| \boldsymbol{p}_{u,t} - \boldsymbol{p}_{a,t} \right\| < \rho_{max}\xi^{0.25}, \qquad (10)$$

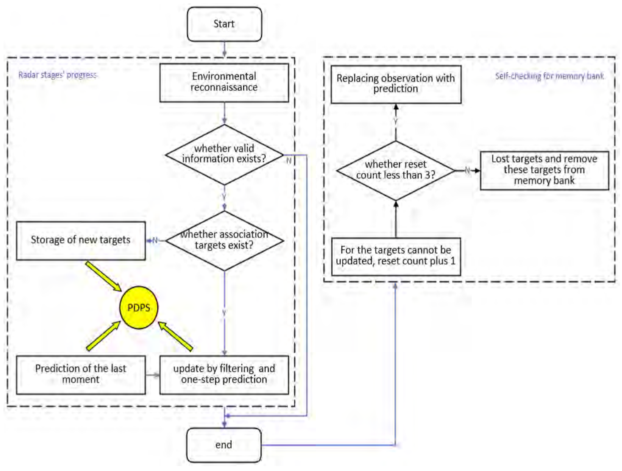

Start
Radar stages' progress
Environmental reconnaissance
whether valid information exists?
N
Y
whether association targets exist?
Storage of new targets
N
PDPS
Y
Prediction of the last moment
update by filtering and one-step prediction
end
Replacing observation with prediction
Self-checking for memory bank
Y
whether reset count less than 3?
N
Lost targets and remove these targets from memory bank
For the targets cannot be updated, reset count plus 1

FIGURE 6. Workflow of an event-triggered radar sensor.

TABLE 1. Explorer configuration.

Notation	Description	Value
n_f	Maximum normal overload	8
n_q	Maximum radial overload	8
g	Gravitational acceleration	9.8 m/s^2
V_{max}	Maximum flight speed	300 m/s
Vol_c	Volume of the UCAV, regarded as a cylinder	bottom radius=60 m height=120 m
Vol_s	Volume of the station, regarded as a cylinder	bottom radius=60 m height=120 m
ξ_{min}	Minimum RCS of the UCAV	0.004
τ_r	Electric scanning cycle of the radar	1 μs
ρ_{max}	Maximum detection range of the radar in search mode	3 km
ζ_{max}	Maximum quality factor of the PDPS	0.2
ζ_{min}	Minimum quality factor of the PDPS	0.05
F	Prediction scale of the PDPS	0.8

where $\boldsymbol{p}_{a,t}$ is the target's position in space at time t and $\|\boldsymbol{p}_{u,t} - \boldsymbol{p}_{a,t}\|$ represents the two-norm displacement value from the target to the UCAV.

D. COLLISION DETECTION

When simulating the behavior of objects in a real environment, a physics engine with high efficiency and granularity can offer increased immersion for the interaction between an object and the environment, and collision detection is definitely a crucial part of developing such a physics engine. In vCEW, we do not ignore the physical volume of objects in the environment, such as the UCAV and observation stations. Sacrifice self-destruction by colliding with the target is still considered to be an effective means of attack when the UCAV is not equipped with missiles; thus, collision detection is also necessary for shaping the reward of the attacker.

As shown in Figure 5 (b), the shape of a physical object is modeled as a cylinder. The condition for determining whether there is a collision relationship between two objects should be

$$\|\boldsymbol{p}_{u,t} - \boldsymbol{p}_{a,t}\| < L_u(\boldsymbol{n}_t) + L_a(-\boldsymbol{n}_t), \tag{11}$$

where the L_u function is used to calculate the length from the geometric center of the UCAV to the contour edge, pointing toward the direction $\boldsymbol{n}_t$. Similarly, L_a can be used to calculate the value of the target with the same meaning as L_u.

E. PROGRESSION OF RADAR STAGES

Although the events encountered by the radar will change in real time in the face of various combat situations, these events must be continuous time and interdependent. Therefore, when the UCAV performs target searching, its radar will work in the search stage/mode, and the system will not switch to another stage/mode until a new event is triggered. Moreover, even in the same stage, the radar performance will be affected by the subevents separated from the main event.

The UCAV stores the state information of the detected target and associates it with the data in its memory bank. Targets without successful associations may be discarded. To simulate the program of capturing observations, we design a stable data processing system. Kalman filtering (KF) is a classical filtering algorithm that has been proven to converge to the optimal solution, and it can be used to smooth, predict and estimate continuous observations. The simulation for KF needs to be combined with real-world data to generate process noise and measurement noise for objects. However, such data from field trials are confidential. Using the idea of the KF algorithm, we build a parametric data processing system (PDPS), which is as functional as KF. The convergence ability of the system can be adjusted manually, and the form of environmental noise can also be selected.

The procedure of vCEW processing continuous radar data is illustrated in Figure 6, and the function design and mathematical model derivation of PDPS are detailed in Appendix B.

F. EXTENSION OF EXPLORER CONTENT

In addition to the functional modules already described above, the Explorer supports the extension of the motion model of any object, such as adding a missile model to the properties of the UCAV. However, since this work solves a pure target searching problem, Other attachments can be ignored.

All model-related notations and the corresponding settings used in our configuration are summarized in Table 1. Moreover, there are some additional parameter settings related to the radar's PDPS; however, the description of this system is quite cumbersome, and its configuration remains unchanged during environmental testing (at least in Explorer). Therefore, in this paper, no space is devoted to such a description, and little introduction to these parameters is provided. Interested researchers can refer directly to our open source code to obtain functional information on the PDPS.

IV. DEEP REINFORCEMENT LEARNING IN THE EXPLORER GAME

This section aims to establish an RL framework that allows the UCAV to express and reason for the maneuvering strategies generated by its interactions with the environment. RL frameworks can typically be divided into three categories: action-value-based frameworks [34], [35], policy gradient frameworks [28], [36], [37], and actor-critic (AC) frameworks [25], [38]–[40].

A. ALGORITHM SELECTION

Because the UCAV in Explorer needs to realize continuous long-term action planning, the selected DRL algorithms must be based on continuous control models, such as normalized advantage functions (NAF) [34], trust region policy optimization (TRPO) [37], proximal policy optimization (PPO) [28], the asynchronous advantage AC (A3C) algorithm [40], and the deep deterministic policy gradient (DPPG) algorithm [25]. NAF is based on a single action-value function network, which is exquisitely designed but complex to implement; TRPO, PPO, A3C, and DPPG are all based on AC frameworks (although policy gradient optimization lies at the core of the TRPO and PPO algorithms). Considering the real-time requirements of CEW, we adopt the currently popular DDPG algorithm. DDPG is considered to be the fastest converging algorithm for continuous control based on an AC framework [41]. More importantly, DDPG adopts a deterministic policy when updating the actor network, which is a very suitable approach for a system with a high-dimensional action space such as a UCAV.

In an AC framework, an actor is used to select actions, and a critic is used to estimate the quality of the current actions. At time step t, the agent obtains the observed state S_t. The action set is denoted by $\mathcal{A}$ of size $K = |\mathcal{A}|$. The DDPG algorithm continuously improves the policy as it explores the environment. The agent first chooses the optimal action $A_t \in \mathcal{A}$ from the deterministic policy function $A_t = \mu(S_t) + \mathcal{N}_t$, where $\mathcal{N}_t$ is the exploration noise introduced by humans. Then, the environment will feed back a timely reward signal R_t in accordance with a reward function $R_t = f(S_t)$, which will be discussed in Section V-B. The new state S_{t+1} is updated in an unknown but certain state transition mode at the next moment $t+1$ it clearly receives the impact of the environment. For a continuous task, the observed trace of the agent can be represented as a combination of multiple state transition pairs, i.e., $\phi = \{\ldots(S_t, A_t, R_t, S_{t+1}), (S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2})\ldots\}$. Meanwhile, we define the action-value function $Q(S_t, A_t)$ for each state-action pair in terms of the expected accumulated reward in the future when action A_t is taken in the given state S_t.

In particular, when the dimensions of the agent's state/action space are discrete or even continuous, the recognition ability in the state/action space is enhanced by the powerful generalization ability of a neural network (NN), such as a deep convolutional neural network (ConvNet). Hence, an NN model is embedded into the policy function and the action-value function, and then they are rewritten as $\mu(S_t|\theta^\mu)$ and $Q(S_t, A_t|\theta^Q)$, respectively. We apply gradient descent to train θ^μ and θ^Q over the minibatch data randomly

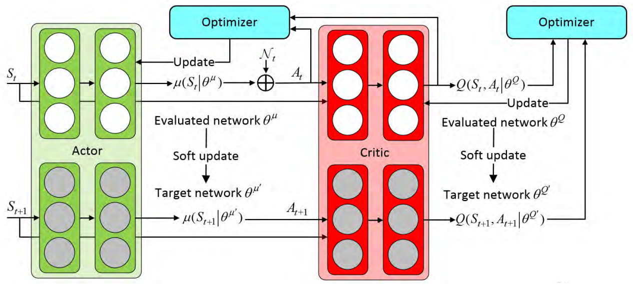

Optimizer
Optimizer
Update
$\mathcal{N}_t$
S_t
$\mu(S_t|\theta^\mu)$
A_t
$Q(S_t, A_t|\theta^Q)$
Update
Evaluated network θ^μ
Evaluated network θ^Q
Actor
Critic
Soft update
Soft update
Target network $\theta^{\mu'}$
Target network $\theta^{Q'}$
S_{t+1}
$\mu(S_{t+1}|\theta^{\mu'})$
A_{t+1}
$Q(S_{t+1}, A_{t+1}|\theta^{Q'})$

FIGURE 7. The DDPG network framework.

sampled from the trace ϕ, and this operation is called memory replay.

Q-learning is a common learning method for the critic, which is subject to the action-value function. However, since the $Q(S_t, A_t|\theta^Q)$ update is prone to divergence, the direct implementation of Q-learning with NNs proved to be unstable in many environments. To solve this problem, Lillicrap *et al.* [25] use a dual network structure similar to that proposed in [35], i.e., a target network combined with an evaluation network. First, the target networks $\mu'(S|\theta^{\mu'})$ and $Q'(S, A|\theta^{Q'})$ are copied for the actor and critic, respectively, and then the weights in these networks are updated via a soft update as follows:

$$\begin{cases} \theta^{\mu'} \leftarrow \sigma\theta^{\mu} + (1-\sigma)\theta^{\mu'} \\ \theta^{Q'} \leftarrow \sigma\theta^{Q} + (1-\sigma)\theta^{Q'} \end{cases}, \quad \sigma \ll 1. \tag{12}$$

As shown in Figure 7, in DDPG, the closer to supervised learning design allows the actor and critic to learn according to their respective objective functions, resulting in a more convergent learning process.

- For the actor network, the gradient is computed as follows:

$$\nabla_{\theta^\mu} \mu|_{S_t} \approx \frac{1}{B}\sum_t \nabla_A Q(S, A|\theta^Q)\Big|_{S=S_t, A=\mu(S_t)} \nabla_{\theta^\mu} \mu(S|\theta^\mu)\big|_{S_t}, \tag{13}$$

  where B is the batch size.
- For the critic network, the loss function that should be backpropagated through the deep action-value network is

$$\frac{1}{B}\sum_t (R_t + \gamma Q'(S_{t+1}, \mu'(S_{t+1}|\theta^{\mu'})||\theta^{Q'}) - Q(S_{t+1}A_{t+1}|\theta^Q))^2 \tag{14}$$

  where γ is the discount factor, which takes values in the range [0, 1].

When DDPG is applied in actual engineering, there are other tricks to make the performance of the algorithm more stable, such as state/action clipping. Details will be introduced in Section V.

B. PARTIALLY OBSERVABLE MARKOV DECISION PROCESS (POMDP)

The UCAV in Explorer takes sensor feedback from a partially observable environment and stores this feedback within a memory bank. For environmental observation, the feedback signal received by the UCAV at time t can be resolved into a nine-dimensional vector:

$$\begin{aligned} O_t &= CON(\hat{\boldsymbol{p}}_{u,t}, \hat{\boldsymbol{p}}^o_{a,t} - \hat{\boldsymbol{p}}_{u,t}, \boldsymbol{v}^o_{a,t} - \boldsymbol{v}_{u,t}) \\ &= CON([x_{u,t}, y_{u,t}, h_{u,t}], [dx^o_t, dy^o_t, dh^o_t], \\ &\qquad [dv^o_{x,t}, dv^o_{y,t}, dv^o_{z,t}]), \end{aligned} \tag{15}$$

where CON represents a function used to concatenate vectors, $[dx^o_t, dy^o_t, dh^o_t] = [x^o_{a,t} - x_{u,t}, y^o_{a,t} - y_{u,t}, h^o_{a,t} - h_{u,t}]$, and $[dv^o_{x,t}, dv^o_{y,t}, dv^o_{z,t}] = [v^o_{ax,t} - v_{ux,t}, v^o_{ay,t} - v_{uy,t}, v^o_{az,t} - v_{uz,t}]$. Equation 15 indicates that O_t is composed of three 3D vectors: the spatial position of the UCAV, $[x_{u,t}, y_{u,t}, h_{u,t}]$; the relative displacement between the target and the UCAV, $[dx_t, dy_t, dh_t]$; and the relative velocity between the target and the UCAV, $[dv^o_{x,t}, dv^o_{y,t}, dv^o_{z,t}]$.

Three important points to note are as follows:

- The target's status information, including the position and velocity accompanied by noise (at least in Explorer), is only valid if the UCAV's observability conditions are satisfied, as expressed in Equation 10.
- In practice, for DRL algorithms, we find that it is easier to converge when high-order difference variables are used to describe the environment state than when pure absolute variables are used. For example, the generalization ability of an NN for the relative displacement between two objects (expressed as a 3D vector) is stronger than that for the concatenated vector of two objects' positions (expressed as a six-dimensional vector).
- We can effectively reduce the sparse representation of state by rewriting the space coordinates of the object, i.e., $[x, y, z]$ to $[x, y, h]$, where the conversion between z and h refers to Equation 1. State sparsity can be well explained by an example: in Explorer (the specific environment design of Explorer is presented in Section III-A), if the UCAV wants to learn a pull-up motion of 0.5 km, the change that the network needs to learn may have a magnitude of only 0.5/6371 (6371 km is the Earth's radius) when expressed in terms of the coordinate value on the Z-axis but at least 0.5/8 when expressed in terms of the off-ground height. Clearly, the latter is more advantageous for the training of the NN. Interestingly, compared with using batch normalization at the output of the NN, manual operations such as this can often make DRL converge faster.

1) GENERATIVE MODEL

Although we can set up the feedback mechanism manually, our goal is to build a framework that allows the UCAV (player) to infer global information from interactions with its environment. For example, when the UCAV flies to a location and does not find a target, it will review the path that it has traveled before, and then it will use this information to guess the possible location of the target and the confidence level in the action that the UCAV will take. However, simply extracting information from the observation itself is not sufficient to guide the UCAV's actions, particularly in the early stages of the mission when the UCAV has explored less of the environment. If other factors such as the noise influences of the environment are added, then it will become more difficult for the UCAV to understand the probability distribution of the global/original state from the existing state information.

Thus, in this work, we model the task using a partially observable Markov decision process (POMDP), and we use Equation 15 as the continuous state of POMDP. Given a sequence of radar signals $\vec{O} = (O_1, O_2, \ldots, O_t)$ up to the current time step t along with a sequence of actions taken $\vec{A} = (A_1, A_2, \ldots, A_t)$, our goal is to obtain an approximate estimate of the original state sequence $\vec{S} = (S_1, S_2, \ldots, S_t)$ such that we can generate a new state transition through the execution of action A_{t+1}, i.e., S_{t+1} [26]. Simply rewriting the transition pair (S_t, A_t, R_t, S_{t+1}) in the observed trace ϕ to $(S_t, A_t, R_t, T_t, O_t, S_{t+1})$ can yield a correct definition of the POMDP model. $T_t \in \mathcal{T}$ is a state transition function, and $O_t \in \mathcal{O}$ represents a probability function about the observation. Figure 8 (a) depicts a graphical model of the state transition when the UCAV interacts with its environment.

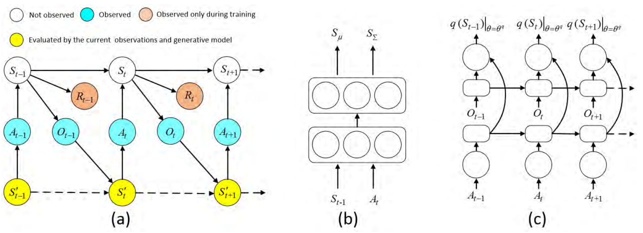


FIGURE 8. The target searching task can be considered as a POMDP (a) generated by the interaction between the UCAV and the environment. We parameterize the transfer function and emission function with an NN (b). To find the representation of the real states in the POMDP, we approximate the posterior distribution by means of a recurrent recognition network, which consists of two LSTM layers (c).

It is important that the state S_t is unknown such that the functions of R_t, T_t, and O_t determined by S_t cannot be output as a priori information.

Referring to the approach in [26], we employ a generative model to estimate the state transition probability. Although the probability distribution can be any form of distribution, we consider using a Gauss distribution to simplify the analysis of the model. The parameters of the generative network are defined as $\theta^g = \{S_\mu, S_\sum, O_\mu, O_\sum, R_\mu, R_\sum\}$, which correspond to the means and variances of the states and observations:

$$\begin{aligned} S_1 &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \\ S_t &\sim \mathcal{N}(f_{S_\mu}(S_{t-1}, A_t), f^2_{S_\sum}(S_{t-1}, A_t)\boldsymbol{I}) \\ O_t &\sim \mathcal{N}(f_{O_\mu}(S_t), f^2_{O_\sum}(S_t)\boldsymbol{I}) \\ R_t &\sim \mathcal{N}(f_{R_\mu}(S_t), f^2_{R_\sum}(S_t)\boldsymbol{I}). \end{aligned} \tag{16}$$

Figure 8 (b) shows a network with two fully connected (FC) layers for parameterized transfer functions; emission networks possess a similar structure.

2) VARIATIONAL BAYESIAN ESTIMATION

Directly calculating the posterior distribution $p(\vec{S}|\vec{O}, \vec{R}, \vec{A})$ by relying on the NN is difficult. We therefore use the variational Bayesian method combined with a recurrent neural network (RNN) to maximize the evidence lower bound (ELB) of the training data and obtain the approximation solution for the posterior distribution of p with an encoder network $q_{\theta^q}(\vec{S}|\vec{O}, \vec{R}, \vec{A})$, where θ^q represents the network parameters of the RNN. The network with parameters θ^g can be regarded as serving the function of a decoder.

Concurrently, because the environmental model is globally achievable, timely rewards $\vec{R} = (R_1, R_2, \ldots, R_t)$ are clearly obtained during training. There are many forms of encoder structures that can be selected. Since $q_{\theta^q}(\vec{S}|\vec{O}, \vec{R}, \vec{A})$ based on θ^q needs to process long-term trajectory data, and because the ability of long short-term memory (LSTM) to generate vector-navigation cells has been confirmed in [21], we first employ two LSTM layers as the basic framework for the deep RNN, as shown in Figure 8 (c). Additionally, to jointly learn the parameters of the generative network (θ^g) and the recognition network (θ^q), we need to maximize the objective function, a lower bound, which is derived through variational Bayes inference on a conditional log-likelihood [26], [42]:

$$\log p_{\theta^g}(\vec{O}, \vec{R}|\vec{A}) = KL(q_{\theta^q}(\vec{S}|\vec{O}, \vec{R}, \vec{A})||p_{\theta^g}(\vec{S}|\vec{O})) + \mathcal{L}(\theta^q, \theta^g) \geq \mathcal{L}(\theta^q, \theta^g), \quad (17)$$

where KL denotes Kullback-Leibler divergence. Since the value of the Kullback-Leibler divergence is greater than or equal to 0, when $\log p_{\theta^g}(\vec{O}, \vec{R}|\vec{A})$ is to be maximized, the ELB $\mathcal{L}(\theta^q, \theta^g)$ can be maximized instead. $\mathcal{L}(\theta^q, \theta^g)$ can be further simplified as follows:

$$\mathcal{L}(\theta^q, \theta^g) = -KL(q_{\theta^q}(\vec{S}|\vec{O}, \vec{R}, \vec{A})||p_{\theta^g}(\vec{S}|\vec{A})) + \mathbb{E}_q[logp_{\theta^g}(\vec{O}, \vec{R}|\vec{S}, \vec{A})], \quad (18)$$

where $\mathbb{E}$ denotes the expectation function. The purpose of maximizing $\mathcal{L}$ is to accurately estimate the posterior probability.

According to the reparameterization trick of [42], Equation 18 can be further rewritten as

$$\begin{aligned}&\mathcal{L}(\theta^q, \theta^g)\\ &\approx -KL(q_{\theta^q}(S_1|\,\vec{O}, \vec{R}, \vec{A})||p(S_1))\\ &\quad -\frac{1}{N}\sum_{t=2}^{T}\sum_{n=1}^{N}[KL(q_{\theta^q}(S_t|\,S_{t-1}, \vec{O}, \vec{R}, \vec{A})||p(S_t|\,S_{t-1}^{(n)}, A_{t-1}))]\\ &\quad +\frac{1}{N}\sum_{n=1}^{N}[logp_{\theta^g}(\vec{O}|\vec{S}^{(n)}) + logp_{\theta^g}(\vec{R}|\vec{S}^{(n)})], \quad (19)\end{aligned}$$

where $\vec{S}^{(n)} = q_{\theta^q,\mu} + \varpi^{(n)}q_{\theta^q,\sum}$, $\vec{S}_{t-1}^{(n)} = q_{\theta^q,t-1,\mu} + \varpi^{(n)}q_{\theta^q,t-1,\sum}$, and $\varpi^{(n)} \sim p(\varpi)$. Here, $\varpi^{(n)}$ depends only on Monte Carlo samples drawn from $p(\varpi)$. Given multiple data points from the dataset, we finally used AdaDelta to jointly backpropagate the two sets of encoder and decoder parameters on the NNs θ^q and θ^g by using a minibatch to maximize the ELB. Based on the experimental results of [42], if the batch size is sufficiently large ($\geq$ 100), the number N of samples per data point can be set to 1.

More details about the derivations of Equations 18 and 19 can be found in [26], [42], and [43]. Except for the change of definition symbols, the other contents are the same.

V. SYSTEM DETAILS

To encapsulate a complete target searching system, it is necessary to adapt the learning of the UCAV to the Explorer environment to create an optimal control policy for the UCAV. Meanwhile, the manual work of reward shaping and state representation will play a significant role in the convergence of the DRL algorithms.

A. STATE DEFINITION

One of the greatest benefits of vCEW is that the training samples can be tailored to meet certain demands, which was unimaginable in the past because the costs of field experiments were too high and because unpredictable states could cause irreversible damage to the vehicle's body.

Because of the complex transformations in 3D space, the forward direction of the UCAV can change within a very short time, and the UCAV's motion planning is sparse compared with the spatial radar observations ($\rho_{max}/\left\|\boldsymbol{p}_{u,t+1} - \boldsymbol{p}_{u,t}\right\| > 100$). This situation suggests that the UCAV cannot use visual observations as the input states for RL; thus, it is necessary to handle the effective state of the UCAV manually. Based on the conclusions from Section IV-B, Equation 15 can be used to represent an effective state of the UCAV.

As a supplement, the following two tricks that are commonly used in practice are introduced.

1) BOUNDARY CLIPPING

The actions of the agent are generally constrained, and the results fitted by the NN are likely to exceed the actions' boundary; thus, it is necessary to clip the portion beyond the boundary. This boundary clipping is called action clipping. However, state clipping is often overlooked, and this phenomenon is described below in conjunction with Figure 9. In this figure, when the UCAV takes action A_{t-1} under the state S_{t-1}, it will clearly leave the boundary of the map. At this time, two approaches are generally adopted: 1) environment reset and 2) state clipping.

In the actual project, we found that in the early phase of training, the actions output by the NN cannot significantly change the path of the UCAV; it will only be the simplest constant velocity maneuver, so it is easy to cause the UCAV to fly out of the mission area. Frequent restarting of the game environment will lead to a serious problem, i.e., the characteristics of the state transition tuples in the memory pool are

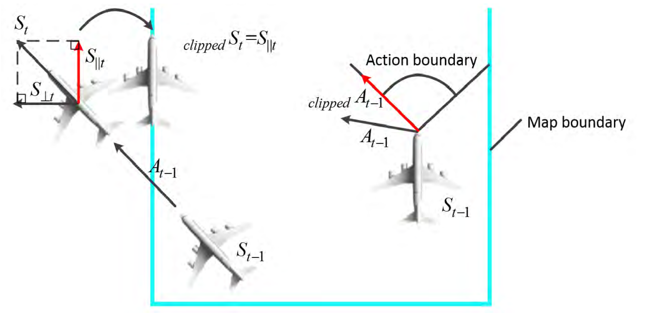


FIGURE 9. **Schematic diagram of the boundary clipping method.**

too few or the content that can be learned is not rich enough, the result of which is, of course, that the algorithm does not converge. In high-dimensional space, each state component needs to be clipped. As shown in Figure 9, the state clipping method is similar to action clipping – let the state component that is beyond the boundary correct to the original boundary.

2) STATE NORMALIZATION

State normalization is an effective means to accelerate the convergence of the DRL algorithm. In the environment of a sparse state space with different scales of state values, the importance of state normalization is self-evident. In this work, based on Equation 15, we present the following state normalization approach:

$$\begin{aligned}\oslash O_t &= CON(\oslash \hat{\boldsymbol{p}}_{u,t}, \oslash(\hat{\boldsymbol{p}}^o_{a,t} - \hat{\boldsymbol{p}}_{u,t}), \oslash(\boldsymbol{v}^o_{a,t} - \boldsymbol{v}_{u,t}))\\ &= CON([(x_{u,t}+7.5)/15, (y_{u,t}+7.5)/15, (h_{u,t}-0.5)/7.5],\\ &\quad [dx^o_t/15, dy^o_t/15, dh^o_t/7.5],\\ &\quad [dv^o_{x,t}, dv^o_{y,t}, dv^o_{z,t}]/V_{max}), \qquad (20)\end{aligned}$$

where $\oslash$ indicates the normalization operator.

Equation 20 shows that by scaling the original state component by the upper bound of the set interval, an ideal normalized state with a value of less than 1 can be generated. In fact, a recent technology called batch normalization [44] in deep learning can also achieve similar results. This technique normalizes each dimension across the samples in a minibatch to have unit mean and variance. We separately used these two methods to test the Explorer game. The results show that the effect of manually ensuring that the state units are within the set range is less time consuming in training than the batch normalization, i.e., the former's convergence rate is faster. This phenomenon may be related to the simplification of the learning process.

B. REWARD SHAPING

Reward shaping can be used to modify the behavioral purpose of the agent in RL and to highlight the signal most desired by the designer. In Explorer, we are able to access the internal states of interest and develop the desired reward function in an intuitive way:

- If the UCAV is reluctant to execute its mission or simply wishes to desert, then it should be punished when it comes close to the map boundary or when it shows a tendency to escape the map boundary. For such a case, we define the following penalty function:

$$\begin{aligned}{}_{es}R_t = &-clip((x_{u,t} - 7.5)/l + 1, b)\\ &-clip((y_{u,t} - 7.5)/l + 1, b)\\ &-clip((h_{u,t} - 8)/l + 1, b)\\ &-clip(1 - (x_{u,t} + 7.5)/l + 1, b)\\ &-clip(1 - (y_{u,t} + 7.5)/l + 1, b)\\ &-clip(1 - (h_{u,t} - 0.5)/l + 1, b), \qquad (21)\end{aligned}$$

where $clip(a, b)$ represents a clipping function. This function outputs the original value when a is within the interval b; otherwise, it outputs the closest boundary value of b. In Equation 21, b is set to $[0, 1]$. l is the maximum travel distance of the UCAV in one cycle, $l = V_{max}\tau$.
- After estimating the target's position, the operator wants the UCAV to tentatively approach this position to verify the accuracy of the estimate produced by the NN. In practice, we want the combat vehicle to observe the potential target for a long time, so target tracking should be encouraged; thus, we shape the rewards for positions close to the target as follows:

$${}_{tr}R_t = \begin{cases} 1 - \| \oslash (\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}) \| \text{if} \|\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}\| < \rho_{max} \\ -\| \oslash (\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}) \| \text{else.} \end{cases} \qquad (22)$$

- Since the UCAV is not equipped with missiles in the Explorer game, the only way for the UCAV to destroy an enemy observation station is to sacrifice itself through a suicide attack. This should be recognized as a special offensive behavior, and the reward created by this behavior can be defined as follows:

$${}_{sa}R_t = \begin{cases} 100 & \text{if } \textit{Equation 11 is satisfied} \\ 0 & \text{else.} \end{cases} \qquad (23)$$

The state reward of the UCAV at any time is equal to the sum of the results of the three reward functions defined above, i.e., $R_t = {}_{es}R_t + {}_{tr}R_t + {}_{sa}R_t$.

The overall system details are shown in Figure 10. We have shared the source code of the streamlined version of Explorer on our team's GitHub [1] to facilitate communication and learning among a wider community of scholars.

[1]https://github.com/youshixun/vCEW

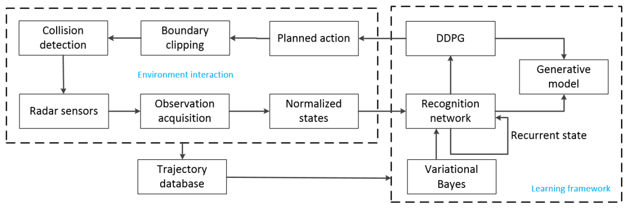


FIGURE 10. **Our proposed system for learning and simulation experiments involving target search missions in Explorer.**

VI. EXPERIMENTAL RESULTS

This section first configures the primary parameters of RL system, and then it introduces our simulation environment and hardware/software configuration in the experiment. Finally, according to the required metrics, we use the algorithm to test games of different difficulty and present the corresponding results and analysis.

A. NETWORK ARCHITECTURE

We train a network as our searcher. The basic network structures are shown in Figures 6 and 7. The detailed hyperparameters inside the network are listed in Table 2.

TABLE 2. Detailed network configurations.

Network	Layers	Hyperparameters
DDPG (actor-critic)	Actor: $2 \times$ 2FC-(9, 300, 2) Critic: $2 \times$ 2FC-(11, 300, 1)	Learning rate for actor: 3e-5 Learning rate for critic: 3e-5 Discount factor γ: 0.9 Soft replacement factor σ: 5e-3 Memory capacity: 3e5 Batch size: 128 Optimizer: Adam
Generative network	2FC-(11, 300, 18/18/2)	Learning rate: 5e-5 Batch size: 128 Optimizer: AdaDelta
Recognition network	2LSTM-(256, 256, 256)	Learning rate: 5e-5 Batch size: 128 Optimizer: AdaDelta

In Table 2, $2 \times$ 2FC-(9, 300, 2) means that there are two networks, and each network consists of two FC layers, which have 9 input units, 300 connection units, and 2 output units. Note that the generative network is related to S_t, O_t and R_t, which respectively represent three independent output networks. Moreover, the output action only includes rotation angle and pitching angle; thus, its dimension is 2.

B. SIMULATION PLATFORM OF SOFTWARE AND HARDWARE

For DRL, the computational burden of the algorithm is a very important issue; thus, when comparing the performance of various algorithms horizontally, the consistency of the software and hardware versions in the testing environment must be guaranteed. Table 3 presents the testing environment that we utilized.

C. SIMULATION DETAILS AND METRICS

We divide the game difficulty into four levels, denoted by $\mathcal{D} = 0$ to 3, to evaluate our DRL system on the task of learning autonomous navigation in Explorer based on operating cycles of different lengths: leftmargin=*

- $\mathcal{D} = 0$: for each episode, the maximum number of operational steps of the UCAV is 400, and the length of an operating cycle is $\tau = 1\ s$.
- $\mathcal{D} = 1$: for each episode, the maximum number of operational steps of the UCAV is 800, and the length of an operating cycle $\tau = 0.5\ s$.
- $\mathcal{D} = 2$: for each episode, the maximum number of operational steps of the UCAV is 2000, and the length of an operating cycle is $\tau = 0.2\ s$.

TABLE 3. Testing environment.

Item	Description	Model/Version
CPU	Central processing unit	Xeon E5-1620 v4 3.50 GHz
GPU	Graphics processing unit	NVIDIA Quadro P2000
Memory	Storage	16 GB (DDR4 RDIMM)
Python	Programming language	v3.5.4
TensorFlow	A symbolic mathematical system based on data flowprogramming, which is widely used in the programming of various machine learning algorithms	v1.7.0
Pyglet	A multimedia framework under Python	v1.3.2

- $\mathcal{D} = 3$: for each episode, the maximum number of operational steps of the UCAV is 4000, and the length of an operating cycle is $\tau = 0.1\ s$.

The training data for the tasks at each of the four difficulty levels are randomly generated from a specific random seed. At each level, 10000 episodes of the game are run; for each episode, the total navigation time is 400 s, and the condition $\tau > \sqrt[3]{\frac{4\rho_{max}^2}{V_{max}^2}}\tau_r = 0.07\ s$ is satisfied. Apparently, the difficulty of the game increases as the $\mathcal{D}$ value increases. For each episode, at the beginning, the state information of the UCAV and the observation station is reinitialized, such as position, velocity, and so forth. Additionally, if the UCAV tracking of the target lasts for 50 cycles or there is a collision with the target, the episode's task will be completed ahead of time.

Since there are not enough experimenters involved in the game and Explorer appears to be not very friendly to laymen, we use a completely random strategy as the baseline in the comparative experiments. In this baseline, random values are generated within the possible numerical range of actions at each moment.

Three metrics are employed for the experiments: the mean accumulated reward (MAR), game win rate (WR), and percentage of mission completion (PMC) of the UCAV. For the four levels of experiments, the WR and PMC values after all episodes have been run are reported, where WR refers to the percentage of episodes in which the UCAV finishes by destroying the target and PMC refers to the percentage of episodes in which the target is successfully located. Note that there may be multiple local optimal solutions that exist in the continuous state space; even if the algorithm has converged, the agent of DRL may still possesses a large jitter in the MAR value of each episode, i.e., an unstable variance's company in training duration. Therefore, we adopt the following calculation for data smoothing according to the completeness of different tasks:

$$MAR = \frac{1}{\wedge(|\phi|, \upsilon_s)} \sum_{t=-1}^{\vee(-|\phi|, -\upsilon_s)} R_t$$

$$\overline{MAR} = \frac{1}{\upsilon_e} \sum_{e=1}^{\upsilon_e} MAR_e, \tag{24}$$

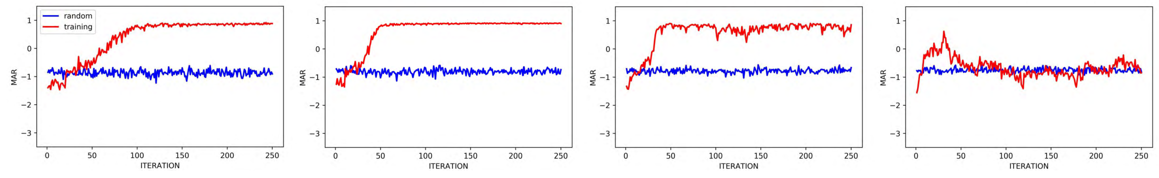


FIGURE 11. Evaluating the performance of the DDPG algorithm in Explorer. The four subfigures from left to right correspond to the results for $\mathcal{D} = 0$ to 3, respectively.

where $\vee$ and $\wedge$ are the maximum and minimum operators, respectively. $|\phi|$ represents the total step size of a single task trajectory. υ_s is the cumulative step size of MAP after completing a single task, and υ_e denotes the segmentation size of averaging all episodes' MARs. $t = -1$ means the reciprocal first step of the trajectory. The reason for this design used for data smoothing is that we are more concerned with what the UCAV has learned in one episode than the UCAV behavior at the beginning of the task. If the UCAV remains in a close-to-goal trend before each episode ends, then it is considered to have truly understood what the "target" is and what "target searching" behavior means. In our work, to adaptively observe the performance of the algorithm, υ_e is set to 1/250 of the maximum episodes, and υ_s is set to 1/40 of the maximum operational steps (from $\mathcal{D} = 0$ to 3).

Note that because the agent tends to converge before all episodes have been run during training, we use the same random seed to generate the testing data and focus on observing the convergence of the MAR in the last 50% of the total episodes. If the UCAV's MAR value does not converge within the appropriate number of iterations, this indicates that the DRL algorithm has failed on this task.

The performance results of our system are shown in Figure 11. In the tasks with difficulties of 0 to 2, we observed that the UCAV very quickly learns a policy that can achieve close to the maximum episode return of 1, which proved that the DDPG algorithm possesses great convergence for various Explorer environments. Meanwhile, DDPG shows noticeable decreases in robustness with harder difficulty settings. Unfortunately, there is no evidence to support that the UCAV has learned enough knowledge across the most difficult task ($\mathcal{D} = 3$), so mission failure appears to be inevitable. Conceivably, increasing the number of episodes will allow the fourth task to achieve better results, but this will increase the training time, which is detrimental to our performance requirements for real-time CEW (i.e., it will lead to inconsistencies between the time of design tasks and the time of planning tasks, or increase the gap).

Detailed performance values for the UCAV on four tasks, such as the WR and PMC values, are reported in Table 4. Through comparisons, we find that our NN configurations allow the UCAV to perform at its best on the task with $\mathcal{D} = 1$. For the task with $\mathcal{D} = 0$, the performance may be lower because the motion planning cycle is too long, resulting in a rough maneuvering policy for the UCAV learned from insufficient exploration samples; by contrast, for the task with $\mathcal{D} = 2$, the motion planning cycle may be too dense, causing the policy that the UCAV needs to learn to be too rich for the DDPG algorithm to steadily converge within a given episode. Videos of Explorer experiments are available at the following website: `https://github.com/youshixun/vCEW`.

TABLE 4. Performance of the proposed DRL framework at different game levels.

Level	WR (%)		PMC (%)		Computation time (h)	
	Random	Training	Random	Training	Random	Training
$\mathcal{D} = 0$	2.22	71.45	56.54	90.29	0.6	2.0
$\mathcal{D} = 1$	2.13	83.49	56.96	93.62	1.5	4.2
$\mathcal{D} = 2$	1.94	59.78	56.27	89.63	3.9	11.5
$\mathcal{D} = 3$	1.76	6.86	51.48	58.48	9.5	22.3

D. BEHAVIOR ANALYSIS

Although the DRL-based approach enables the UCAV to demonstrate a state-of-the-art target searching capability in Explorer, we still wish to shed further light on the essence of the policies learned by the UCAV; therefore, conducting a behavior analysis of the UCAV's actions as generated via the NN is crucial. Equation 5 shows that the control of the UCAV is a virtual force, while the direction of the virtual force needs to be optimized. Using this idea in combination with vector-based navigation design, we perform a behavior analysis based on a new metric, i.e., the behavior angle (BHA), $\varrho = \left\langle \vec{A}_t, \boldsymbol{n}_t \right\rangle$. The BHA can be calculated to observe the movement trend of the UCAV. For convenience of analysis, ϱ is further normalized to $\oslash\varrho = \varrho/\pi$. The value of $\oslash\varrho$ is within the range [0, 1]; when it is less than 0.5, the UCAV is considered to be approaching the target, and vice versa.

Similarly, for tasks of consecutive difficulty, we reveal the transformation process of the UCAV's BHA through data smoothing. The simulation results are shown in Figure 12. As shown in Figure 12, the trained UCAV has undergone significant changes in navigation behavior. Because the benchmark strategy is completely random, its mean BHA is maintained at 0.5, while the trained BHA converges to 0.4 or even less (even in the most difficult game that has failed, changes in BHA reflect the same trend slightly). Note that the difference between the behaviors of the UCAV produced for BHA values of 0.5 and 0.4 is much larger than the intuitive difference between these values (0.1).

Based on the following two points, we will further discuss the reasons why the BHA does not continue to converge to

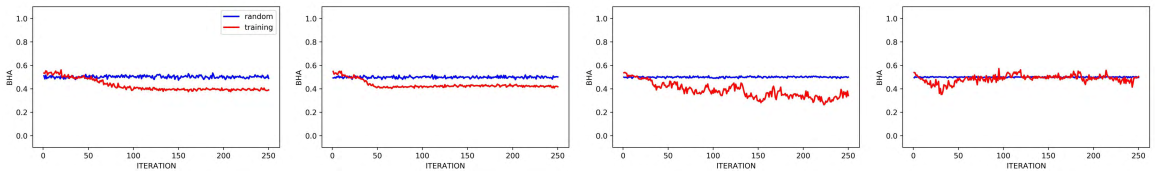


FIGURE 12. Investigating the potential behavior policies learned by the UCAV in Explorer. The four subfigures from left to right correspond to the results for $\mathcal{D} = 0$ to 3, respectively.

smaller values, thus producing a more ideal navigation state: 1) The target position inferred from the historical trajectory of the UCAV and the Bayesian network is inevitably subject to certain errors. Therefore, the UCAV learns a kernel strategy in which the arc curve is used to quickly search for a sufficiently large area, a guess is made regarding the location with the maximum probability of being the target location, and the approach posture of the UCAV is then constantly adjusted. 2) The state of the UCAV as designed in vCEW is high-dimensional and continuous. If the position of the target cannot be accurately estimated, the value of the output BHA is prone to large oscillations. However, because the control matrix of the UCAV is linear and continuous, the short-term oscillations of the BHA between values greater than and less than 0.4 do not affect the overall trend (the trajectory must be differentiable everywhere). In short, the performance of the DRL algorithm and the prediction performance of the generative and recognition networks together lead to a large variance in the BHA of the UCAV; i.e., the effect of training is such that it is difficult to keep the BHA at a stable low value. Note that the well-performing policy adopted by the UCAV for the target searching task is to approach the estimated position of the target as closely as possible, which is surprisingly consistent with the theory of vector-based navigation [21], [45].

VII. CONCLUSIONS

This paper has developed an innovative framework for testing various CEW tasks, in which the DRL algorithm has been applied to the target searching task for the first time. During task execution, the lack of prior information, the partial observability of the environment, and the physical constraints faced by the agent in maneuvering undoubtedly increase the difficulty of continuous action planning, but we successfully overcome these problems with a novel learning framework. Specifically, the variational Bayesian method allows us to approximate the posterior model with a deep recurrent recognition network that consists of two LSTM layers. Using hand-designed state representation and exquisite reward shaping, we also introduce a DDPG method that is able to learn optimal control in learned potential state space by utilizing experiences sampled in the memory pool and learned generative model for transition. Additionally, as the training time progresses, we find that the agent's behavior is constantly evolving in the direction expected in the case of vector-based navigation, which may provide evidence for DRL being anthropomorphic as a popular artificial intelligence technology. Although the work in this paper focuses on target searching and achieving excellent results, we look forward to solving more intricate tasks in CEW with better performing DRL algorithms or frameworks.

The trained artificial intelligence possesses superior control and transient-response capability beyond that of humans, thus proving that the development of DRL can be fully expected to advance the capabilities of military equipment. In the future, we will work with a more advanced model engine to extend the contents of the vCEW framework, such as the full view of the mission environment and the specific combat vehicles designed in Unity3D. Additionally, we hope to further enhance the agent's ability to learn complex policies by improving the existing DRL algorithms or even proposing some new methods. In short, enriching the mission types that can be addressed with vCEW and developing advanced intelligent vehicles that can efficiently handle these missions will be our main research directions in the future.

APPENDIX A
SPHERICAL SPACE DETECTION ASSUMPTION

Although an active phased array radar can achieve full-coverage reconnaissance of the surrounding space, when the length of the radar's electric scanning cycle τ_r is fixed, the length of the operating cycle τ ($\tau > \tau_r$) will determine the maximum number of units that can be scanned per unit time, i.e., $M_u = \tau/\tau_r$. Moreover, the total area of each unit is constrained by the vehicle's speed and the detection range of the radar. Therefore, the assumption that the UCAV has a 360-degree observational perspective during an operating cycle is based on the following premise:

$$M_u > \frac{4\pi\rho_{max}^2}{\pi(V_{max}\tau)^2} \Rightarrow \tau > \sqrt[3]{\frac{4\rho_{max}^2}{V_{max}^2}}\tau_r. \tag{25}$$

The meaning of Equation 25 is that the entire spherical surface of the detectable space is divided into many small circles, where the number of divisions is equal to the number of units that the radar needs to scan in one unit of time. The area of each small circle can be defined as the operation resolution of the UCAV. Since the movement of the UCAV can cause only limited changes in the detection space during an operating cycle, the minimum resolution must be greater than $\pi(V_{max}\tau)^2$ to allow observations from different regional units to be distinguished.

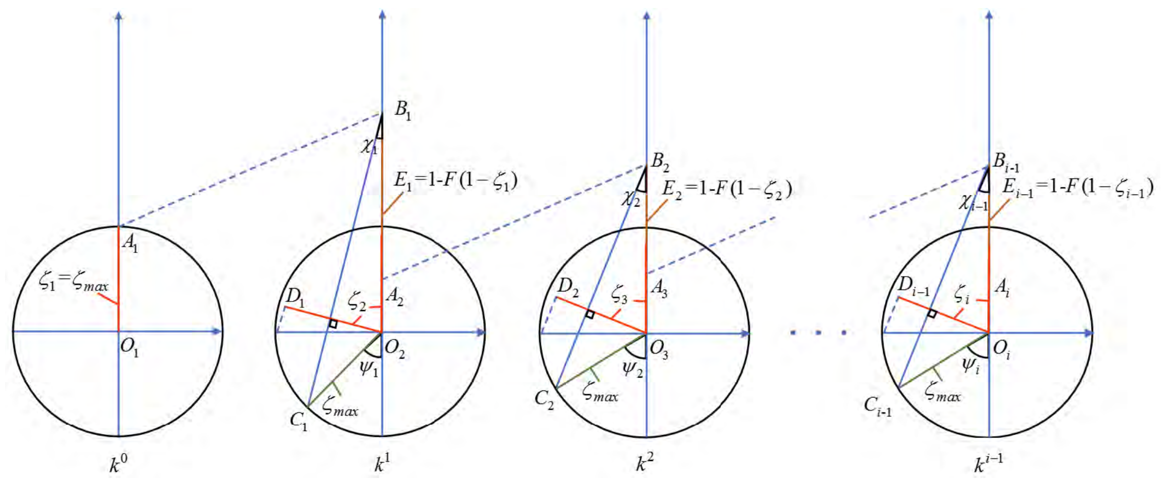


FIGURE 13. Our proposed parametric data processing system for learning and simulation experiments involving target search missions in Explorer. The extent of the entire parametric space is defined such that the values of all quality factors can be mapped into it; the value at the center of the circle is 0, while the value at infinity is 1.

In a practical situation, when τ is too large, the sampling of the environmental data can easily be insufficient, preventing the UCAV from observing the evolution of the combat situation in real time. By contrast, if τ is too small, the radar detection model will degenerate from a sphere to a cone, making the analysis of the search process more difficult. Consequently, τ must be set such that a compromise is achieved between the real-time nature of the system and the difficulty of model analysis.

APPENDIX B
PARAMETRIC DATA PROCESSING SYSTEM

Let us define a quality factor $\zeta \in [0, 1]$ to describe the radar performance. Let the maximum quality factor be ζ_{max} and the minimum be ζ_{min}; note that the radar performance increases as the value of the quality factor decreases. Assuming that the transfer model of the tasks in vCEW is linear and the inheritance factor is $F \in [0, 1]$, the one-step prediction quality factor E_i can be computed as $E_i = 1-F(1-\zeta_i)$, where i refers to the i-th step of processing the target data in the same radar stage. Next, we further illustrate how to generate the filtering result ζ_{i+1} through E_i in conjunction with Figure 13.

As shown in Figure 13, the radii of all circles (green lines) in the figure are equal to ζ_{max}; the red lines represent the current iteration of ζ_i, such as O_1A_1, O_2A_2, O_3A_3, and the brown lines represent the E_i obtained at the previous moment, such as O_2B_1, O_3B_2. Note that C_i only appears in the bottom semicircle because ψ_i is a random angle generated according to the Gauss distribution $\psi_i \sim \mathcal{N}(0, \pi/6)$; this method enables the probability that ψ_i belongs to $[-\pi/2, \pi/2]$ to be 99.7%; if the generation is still outside this range, then it can be clipped to the boundary. The recursion method of producing ζ_i from ζ_{i-1} is as follows: First, determine the intercept point B_i on the vertical axis based on the predicted value E_{i-1}. Then, randomly generate point C_{i-1} and connect C_{i-1} to the predicted point B_{i-1}. Finally, take the circle's center O_i as the starting point to draw a line perpendicular to the line $B_{i-1}C_{i-1}$, and draw a line parallel to $B_{i-1}C_{i-1}$ from the left end of the circle that intersects this vertical line at point D_{i-1}, such that the length of O_iD_{i-1} is ζ_i. Consequently, the iterative formula for ζ_i is

$$\zeta_i = \begin{cases} \zeta_{max} & \text{if } i = 1 \\ \zeta_{min} + k^{i-1}(\zeta_{max}\cos\chi_{i-1} - \zeta_{min}) & \text{else if } i > 1, \end{cases} \tag{26}$$

where $\chi_{i-1} = arc\tan(\sin\psi_i/(\frac{E_{i-1}}{\zeta_{max}} + \cos\psi_{i-1}))$, and $k \in (0, 1]$ denotes the convergence control factor. The larger value of k is, the worse the convergence of the quality factor, and the whole PDPS will oscillate when $k = 1$. In Explorer, k is set to 0.99.

Regarding the quality factor as the current filtering value, it is observable that in PDPS, the old predictive value and the new filtering value appear alternately. Furthermore, the quality factor can guarantee two points in each iteration: 1) predictive value and filtering value are interdependent, and a poor prediction quality will lead to a poor filtering quality; 2) all parameters are mapped to the range of [0, 1], and the errors randomly generated by angle ψ are irrelevant. We simulate the acquisition of observation data and the processing of observation data with the quality factor. During target searching, the quality factor can be regarded as a scale factor to calculate the relative distance error between the UCAV and target. For example, when the radar's quality factor is 0.2, the ranging error of the radar is 20%.

REFERENCES

[1] G. N. Rao, C. V. S. Sastry, and N. Divakar, "Trends in electronic warfare," *IETE Tech. Rev.*, vol. 20, no. 2, pp. 139–150, Feb. 2003.

[2] S. Xue, G. Jiang, and Z. Tian, "Using fuzzy cognitive maps to analyze the information processing model of situation awareness," in *Proc. 6th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Aug. 2014, pp. 245–248.

[3] 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.

[4] 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.

[5] P. Braca, R. Goldhahn, K. D. Lepage, S. Marano, V. Matta, and P. Willett, "Cognitive multistatic auv networks," in *Proc. Int. Conf. Inf. Fusion*, Jul. 2014, pp. 1–7.

[6] S. Noh and U. Jeong, "Intelligent command and control agent in electronic warfare settings," *Int. J. Intell. Syst.*, vol. 25, no. 6, pp. 514–528, Jun. 2010.

[7] X. Meng, J. Zhu, G. Li, and X. Tu, "Research on autonomous control system structure of UCAV based on cognitive science," in *Proc. IEEE Int. Conf. Netw. Infrastruct. Digital Content*, Nov. 2009, pp. 861–865.

[8] X. Chen, Y. Li, Y. Pang, and P. Chen, "Bayesian terrain-based underwater navigation using an improved state-space model," in *Proc. 3rd Int. Conf. Digital Manuf. Automation*, Apr. 2012, pp. 1002–1005.

[9] P. Ogren, A. Backlund, T. Harryson, L. Kristensson, and P. Stensson, "Autonomous ucav strike missions using behavior control lyapunov functions," in *Proc. AIAA Guid., Navigat., Control Conf. Exhibit*, 2006, pp. 21–24.

[10] P. Wei, Z. Song, and M. Lin, "Research on force assignment for ground-to-air radar jamming system based on chaos genetic algorithms," in *Proc. Chin. Control Decision Conf.*, May 2016, pp. 1215–1220.

[11] X. F. Zhai and Y. Zhuang, "IIGA based algorithm for cooperative jamming resource allocation," in *Proc. Asia Pacific Conf. Postgraduate Res. Microelectron. Electron. (PrimeAsia)*, Jan. 2010, pp. 368–371.

[12] N. Osner and W. P. D. Plessis, "Threat evaluation and jamming allocation," *IET Radar Sonar Navigat.*, vol. 11, no. 3, pp. 459–465, Mar. 2017.

[13] Q. Fan, F. Wang, X. Shen, and D. Luo, "Path planning for a reconnaissance UAV in uncertain environment," in *Proc. IEEE Int. Conf. Control Autom.*, Jun. 2016, pp. 248–252.

[14] H. Ergezer and M. K. Leblebicioğlu, "3D path planning for UAVs for maximum information collection," in *Int. Conf. Unmanned Aircraft Syst.*, May 2013, pp. 45–55.

[15] F. Vanegas and F. Gonzalez, "Enabling UAV navigation with sensor and environmental uncertainty in cluttered and GPS-denied environments," *Sensors*, vol. 16, no. 5, p. 666, May 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/5/666

[16] S. You, L. Gao, and M. Diao, "Real-time path planning based on the situation space of UCAVS in a dynamic environment," *Microgr. Sci. Technol.*, vol. 30, no. 6, pp. 899–910, Dec. 2018.

[17] B. Lavis, T. Furukawa, and H. F. Durrant Whyte, "Dynamic space reconfiguration for Bayesian search and tracking with moving targets," *Auton. Robot.*, vol. 24, pp. 387–399, Jan. 2008.

[18] T. H. Chung and J. W. Burdick, "Analysis of search decision making using probabilistic search strategies," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 132–144, Feb. 2012.

[19] M. Dadgar, S. Jafari, and A. Hamzeh, "A PSO-based multi-robot cooperation method for target searching in unknown environments," *Neuro Comput.*, vol. 177, pp. 62–74, Feb. 2016.

[20] A. Kai, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[21] A. Banino *et al.*, "Vector-based navigation using grid-like representations in artificial agents," *Nature*, vol. 557, no. 7705, p. 429, May 2018.

[22] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[23] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr. (Dec. 2016). "Playing doom with slam-augmented deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1612.00380v1

[24] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Comput. Sci.*, to be published.

[25] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *Comput. Sci.*, vol. 8, no. 6, p. A187, Apr. 2015.

[26] J. Sung, J. K. Salisbury, and A. Saxena, "Learning to represent haptic feedback for partially-observable tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jun. 2017, pp. 2802–2809.

[27] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a gpu," in *Proc. ICLR*, 2017, 1–12.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. (2017). "Proximal policy optimization algorithms." [Online]. Available: https://arxiv.org/abs/1707.06347

[29] T. Weber *et al.* (2017). "Imagination-augmented agents for deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1707.06203

[30] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in minecraft," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, May 2016, pp. 2790–2799.

[31] X. B. Peng, G. Berseth, K. Yin, and M. V. D. Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, p. 41, Jun. 2017.

[32] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. (2016). "One-shot learning with memory-augmented neural networks." [Online]. Available: https://arxiv.org/abs/1605.06065

[33] L. Zhu, X. Cheng, and F. G. Yuan, "A 3D collision avoidance strategy for UAV with physical constraints," *Measurement*, vol. 77, pp. 40–49, Jan. 2016.

[34] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 2829–2838.

[35] M. Volodymyr *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[36] O. Buffet and D. Aberdeen, "The factored policy-gradient planner," *Artif. Intell.*, vol. 173, no. 5, pp. 722–747, 2009.

[37] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *Comput. Sci.*, vol. 37, pp. 1889–1897, Jul. 2015.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, Mar. 2014, pp. 387–395.

[39] L. Espeholt *et al.* (2018). "Impala: Scalable distributed deep-RL with importance weighted actor-learner architectures." [Online]. Available: https://arxiv.org/abs/1802.01561

[40] V. Mnih *et al.* (2016). "Asynchronous methods for deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1602.01783

[41] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, May 2016, pp. 1329–1338.

[42] D. P. Kingma and M. Welling. (2013). "Auto-encoding variational bayes." [Online]. Available: https://arxiv.org/abs/1312.6114

[43] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," *Comput. Sci.*, to be published.

[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, Feb. 2015, pp. 448–456.

[45] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Auto. Robots*, vol. 13, no. 3, pp. 207–222, 2002.

SHIXUN YOU received the bachelor's degree from the Taiyuan University of Technology, China, in 2015. He is currently pursuing the master's degree and the Ph.D. degree in information and communication engineering with Harbin Engineering University, China. His research interests include machine learning, evolutionary computation, cognitive jamming, and cooperative jamming.

MING DIAO received the bachelor's, master's, and Ph.D. degrees from Harbin Engineering University, China, where he is currently a Professor with the College of Information and Communication. His research interests include wideband signal processing, pattern recognition, machine learning, and telecommunication. He is a member of the China Society of Image and Graphics, China, and he is also a Fellow of the China Institute of Communications, China. He was a recipient of four ministerial and provincial-level science and technology awards.

LIPENG GAO received the bachelor's, master's, and Ph.D. degrees from Harbin Engineering University, China, where he is currently a Professor with the College of Information and Communication. His research interests include wideband signal processing, information fusion, machine learning, and cooperative jamming. He was a recipient of three ministerial and provincial-level science and technology awards.