electronics

MDPI

Article

Completing Explorer Games with a Deep Reinforcement Learning Framework Based on Behavior Angle Navigation

Shixun You, Ming Diao and Lipeng Gao *

College of Information and Communication, Harbin Engineering University, Harbin 150001, China; youshixun@hrbeu.edu.cn (S.Y.); diaoming@hrbeu.edu.cn (M.D.)

* Correspondence: gaolipeng@hrbeu.edu.cn; Tel.: +86-139-4614-8397

Received: 7 May 2019; Accepted: 20 May 2019; Published: 25 May 2019

Abstract: In cognitive electronic warfare, when a typical combat vehicle, such as an unmanned combat air vehicle (UCAV), uses radar sensors to explore an unknown space, the target-searching fails due to an inefficient servoing/tracking system. Thus, to solve this problem, we developed an autonomous reasoning search method that can generate efficient decision-making actions and guide the UCAV as early as possible to the target area. For high-dimensional continuous action space, the UCAV's maneuvering strategies are subject to certain physical constraints. We first record the path histories of the UCAV as a sample set of supervised experiments and then construct a grid cell network using long short-term memory (LSTM) to generate a new displacement prediction to replace the target location estimation. Finally, we enable a variety of continuous-control-based deep reinforcement learning algorithms to output optimal/sub-optimal decision-making actions. All these tasks are performed in a three-dimensional target-searching simulator, i.e., the Explorer game. Please note that we use the behavior angle (BHA) for the first time as the main factor of the reward-shaping of the deep reinforcement learning framework and successfully make the trained UCAV achieve a 99.96% target destruction rate, i.e., the game win rate, in a 0.1 s operating cycle.

Keywords: target-searching; cognitive electronic warfare; deep reinforcement learning; continuous control-based navigation optimization; behavior angle

1. Introduction

The cognitive degree of electronic warfare depends on the adaptability of the autonomous decision-making system of combat vehicles to various tasks. Taking an unmanned combat air vehicle (UCAV), which possesses the strongest maneuverability, as an example, target-searching and target-tracking are two key tasks for achieving precise target strikes. Moreover, target-searching is the preliminary task of target-tracking, and its importance is self-evident. This high-level task is facilitated through the integration of radar sensors with limited detection range, efficient autonomous decision-making systems, multiple intelligent navigation policies, and a representation of situational awareness. To reduce the consumption of labor and material resources, a major improvement can be achieved by employing intelligent UCAV systems for autonomous search tasks.

Although the concept of cognitive electronic warfare (CEW) has become popular in recent years, there is still a lack of approaches or frameworks to enable UCAVs to search for unknown targets without prior knowledge. This lack is mainly caused by the following two factors: (1) due to the sensitivity of field test data and the complexity of traditional simulators, it is difficult to obtain adequate training/testing data, and (2) the essence of target-searching is probabilistic path planning. If the mission area is too spacious or the operating cycle of the UCAV is too short, then the high

maneuverability of the UCAV and the sparsity of the battle environment will make it very difficult for the mobile platform to output a feasible end-to-end motion strategy. In fact, even for autonomous robots in 2D space, exploring an unknown workspace and recognizing the target based on its physical properties (location, velocity, volume, and so forth) are a formidable challenge.

For factor 1, we use Explorer as the flight simulator of a UCAV, which is a game based on the versatile CEW (vCEW) framework, so the sensors set by each member of the game have typical radar characteristics, and the interaction between the radar sensors and the game environments is detailed in Reference [1], and to simplify the complexity of the model, no soft/hard-killing weapons (such as jammers and missiles) are introduced in this version of the game. In the original version of this game, a player aims to find the target defender, i.e., an observation station, in a given 3D mission map by controlling a UCAV within a limited perspective. With Explorer, we can randomly generate many trajectory samples. Please note that although Explorer is not a game full of confrontation, we still need its combat vehicles (as control agents) possess excellent maneuverability and certain radar signal processing capability. Therefore, in this paper, the term unmanned combat air vehicle (UAV) is qualitatively described as UCAV for distinction. Actually, the method proposed in this paper for UCAV control is also applicable to conventional UAVs. For factor 2, because all states and behaviors of the UCAV in Explorer can be characterized and the simulation environment can provide rich and direct multiagent interactions, we consider using deep reinforcement learning (DRL) algorithms to obtain the optimal control strategy that is approximately end-to-end for the UCAV. At this time, the input of the system is the UCAV's observation state, and the output is the planned acceleration vector of the UCAV.

In widely used navigation strategies, vector-based navigation (VBN) is primarily used in target-searching/tracking, which is a traditional goal-driven navigation method. For long-term trajectory planning, VBN is designed to simply become closer to the target as soon as possible; thus, other auxiliary algorithms, such as dynamic planning, are required to formulate the UCAV's actions at any time. Unfortunately, when experimenting with the DRL framework that we designed, we found that the ability to complete the Explorer game is quite limited when using the feedback of pure goal-driven navigation as a reward signal.

The organization of this paper is as follows: In Section 2, related works about the development of DRL and CEW are discussed. Section 3 explains the control principle of a UCAV in Explorer. A novel target-searching approach with the DRL framework is proposed in Section 4. Then, different levels of designing experiments are given in Section 5. Finally, Section 6 presents the conclusion and future work.

2. Related Works

We first introduce some advanced technologies involved in CEW and then highlight some new developments in DRL, especially for engineering applications.

2.1. Target-Searching in CEW

For years, the research and popularization of advanced electronic warfare has been in great demand. Novel combat vehicles, intelligent autonomous control systems, and powerful detection sensors simultaneously promote the development of various technologies. Among such technologies, researchers are struggling to optimize the cognitive awareness of UCAVs for circumstances as a key combat agent. At the communication end, the radar transmitter is used as the most basic detection sensor. By combining software and hardware signal processing methods, the interference (jamming) signal in the noisy channel can be distinguished to expand the detection range of combat vehicles [2,3].

A UCAV evaluates the environmental situation depending on its detection results. At this time, the physical states of the flying objects in space, including the spatial position, fuselage posture, and flight speed, are the indicators that are of great importance. The relevant details of the established threat model can be found in [4–6]. Moreover, for a combat platform, due to the intricate mission space, gridding the map can effectively reduce the difficulty of processing the situational information [7].

Furthermore, in combination with the UCAV historical path, the target position is inferred by Bayesian estimation or similar [8–10]. Then, real-time or non-real-time path planning algorithms are developed according to various combat requirements [11–13]. The ultimate goal is to approach the target and confirm the identity of the target as soon as possible.

In fact, completing the target-searching task in partially observable environments has been a hot topic in the field of autonomous robotics [13–15]. However, the lack of a simple and clear three-dimensional (3D) space simulator and the corresponding continuous control algorithms for autonomous navigation is a key factor restricting the development of CEW [16].

2.2. Deep Reinforcement Learning

Based on deep Q learning (DQN), DRL has shown excellent performance in the classic Atari 2600 games and demonstrated the potential to transcend human beings [17]. At the same time, the DeepMind team employed a DRL-based agent, AlphaGo, to play Go and even defeated the current world champion [18]. Since then, research on DRL algorithms and engineering applications has begun to increase in various areas. Double DQN (DDQN) can solve the overestimation problem caused by the greedy strategy used in the DQN [19]. To enhance the learning efficiency, the authors designed a sampling policy named prioritized experience replay instead of using uniform sampling, which can distribute the larger sampling weights to the states of sufficient learning contents [20]. The use of dueling DQN is quite suitable when the specificity of the agent's behavior to the state is weak [21]. Additionally, a series of advanced algorithms, such as normalized advantage functions (NAF) [22], asynchronous advantage actor-critic (A3C) [23,24], deep deterministic policy gradient (DDPG) [25], proximal policy optimization (PPO) [26], and soft actor-critic (SAC) [27], which have been developed to solve continuous control problems, have attracted considerable attention.

The DRL-based approach also exhibits unprecedented superiority in most engineering applications. However, we are more concerned with how reinforcement learning performs in the field of intelligent navigation and the metrics that can be achieved. Banino et al. [28] found that DRL could enable an intelligent agent to learn VBN strategies using grid-like representations from the trained networks. Simultaneous localization and mapping (SLAM) schemes based on visual sensors have achieved amazing results due to the introduction of DRL approaches [29]. In an attack-defense pursuit-warfare of multiple UCAVs, the authors in [30] used DQN to generate air-combat strategies. However, to simplify the dynamic maneuvering model, they sliced the UCAV's actions into a fixed two-group mode and limited the battle space to a two-dimensional plane.

Based on the above, building a framework that adapts to multiple DRL algorithms to guide the UCAV to learn the maneuvering strategies of searching targets in 3D space and analyzing the causes of these actions (or how the UCAV understands these actions) is the core contribution of this work.

3. Problem Formulation Based on Explorer

In this section, considering the model dependence of the DRL framework, we first introduce the game environment used to simulate search targets, i.e., Explorer. Then, the operation of the entire Explorer system, including the states, actions, and game goals of the UCAV, is described. Finally, the planning process of the aircraft's actions is specifically discussed. This work mainly refers to our previous research results [1] and provides many significant improvements.

3.1. Game Environment

The display interface of the Explorer game is presented in Figure 1a. In this game, all we need is to control a UCAV and find an enemy observation station in a mission space of 7.5 km in height (from 0.5 km to 8 km above the ground) and 15 km in length and width (both from -7.5 km to 7.5 km). Explorer assumes that the reconnaissance sensor equipped by each combat vehicle is a radar emitter with a limited detection range. The UCAV's reconnaissance perspective at each operating cycle is illustrated in Figure 1b. Please note that Explorer uses the geocentric coordinate system, $OXYZ$, as the

base reference frame, and in the space, the coordinates of any point are recorded as $\boldsymbol{p} = [x, y, z]^T$ or $\hat{\boldsymbol{p}} = [x, y, h]^T$. The formula for transforming between the height from the ground and the Z-axis coordinate of any point in space is

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

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

The maps in Figure 1a,b are obtained by projecting the game space in the opposite direction of the Z-axis.

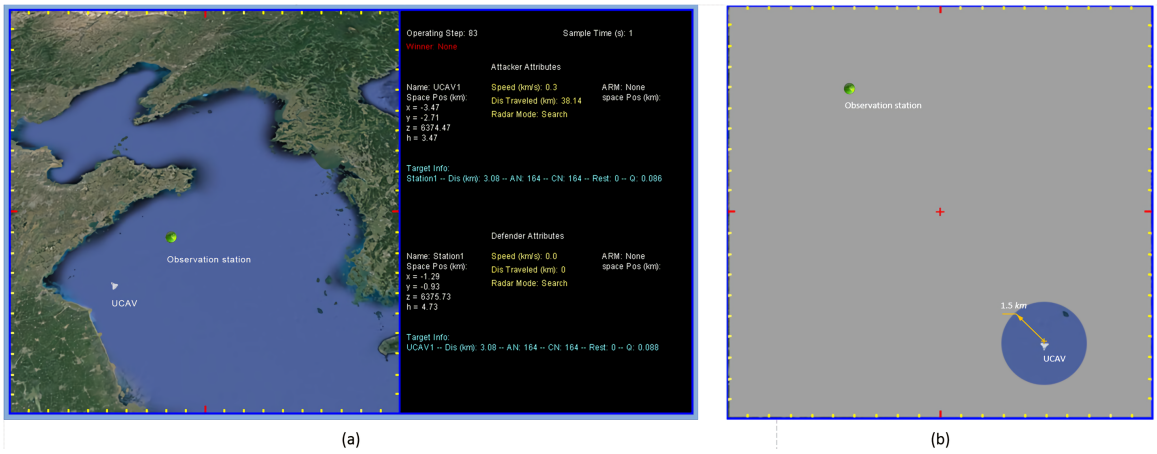

Figure 1. (**a**) displays the interface of the Explorer game, with a total pixel size of 600 × 1050; the 600 × 600 panel on the left side displays the map from the global perspective, and the 600 × 450 panel on the right side displays the status information of the UCAV and the observation station. (**b**) shows the maximum detection range of the UCAV (reconnaissance perspective). The ratio with respect to the real environment is 1:25, i.e., 1 pixel corresponds to 25 m.

3.1.1. Motion State

At sampling time t, the motion state of the UCAV is defined as $\boldsymbol{U}_t = [CON(\boldsymbol{p}_{u,t}^T, \boldsymbol{v}_{u,t}^T, \boldsymbol{a}_{u,t}^T)]^T$, where $\boldsymbol{v}_{u,t} = [v_{ux,t}, v_{uy,t}, v_{uz,t}]^T$ and $\boldsymbol{a}_{u,t} = [a_{ux,t}, a_{ay,t}, a_{uz,t}]^T$ represent the velocity and acceleration of the UCAV, respectively, and CON represents a function used to concatenate vectors. Because the observation station in Explorer remains static, it can sufficiently be described using the position, $\boldsymbol{p}_{a,t}$. For convenience, in the remainder of this paper, the variables with subscripts a and u are defined as belonging to the target observation station and UCAV, respectively.

Please note that the position of the target can only be captured or predicted by the UCAV's radar, we use $\boldsymbol{p}_{a,t}^o$ or $\hat{\boldsymbol{p}}_{a,t}^o$ to indicate the estimated observation of the UCAV, and the specific estimation methods will be detailed in Section 3.2.

3.1.2. Action

For each operating cycle, the manipulation of the UCAV's maneuvering in Explorer can be simply divided into the following three steps:

(1) Input the expected direction of motion, A_t, which is defined by the UCAV's pitching and rotation angles, i.e., $A_t = [\varphi_t, \vartheta_t]$.

(2) By using the approach in [1], A_t and certain physical constraints, including the maximum normal overload, maximum radial overload, and maximum velocity of the UCAV, are used to calculate the maximum acceleration that the UCAV can produce in the given direction; we denote it as $\hat{\boldsymbol{a}}_{u,t}$.

(3) By using the uniform acceleration matrix, $\boldsymbol{\Phi}_{CA}$, as the motion control matrix, the next motion state of the UCAV, $\boldsymbol{U}_{t+1}$, can be obtained by multiplying $\boldsymbol{U}_t$ and $\boldsymbol{\Phi}_{CA}$:

$$\boldsymbol{U}_{t+1} = \boldsymbol{\Phi}_{CA}\boldsymbol{U}_t \Leftrightarrow \begin{bmatrix} \boldsymbol{p}_{u,t+1} \\ \boldsymbol{v}_{u,t+1} \\ \boldsymbol{a}_{u,t+1} \end{bmatrix} = \boldsymbol{\Phi}_{CA} \begin{bmatrix} \boldsymbol{p}_{u,t} \\ \boldsymbol{v}_{u,t} \\ \hat{\boldsymbol{a}}_{u,t} \end{bmatrix}, \tag{2}$$

where $\boldsymbol{\Phi}_{CA} = \begin{bmatrix} \boldsymbol{I} & \tau\boldsymbol{I} & \frac{\tau^2}{2}\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \tau\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$, 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.

3.1.3. Reward Shaping

For a DRL framework, the exquisite shaping of the game rewards plays an important role in the target model. In Explorer, we consider reward-shaping from a behavior-driven perspective, which involves the following three key factors:

(1) Because the target observation station must be within a given task space, it should be punished when the UCAV tends to escape this space. Thus, the following penalty function can be defined:

$$\begin{aligned} {}_{escape}R_t = &- clip((x_{u,t} - 7.5)/l + 1, [0,1]) \\ &- clip((y_{u,t} - 7.5)/l + 1, [0,1]) \\ &- clip((h_{u,t} - 8)/l + 1, [0,1]) \\ &- clip(1 - (x_{u,t} + 7.5)/l + 1, [0,1]) \\ &- clip(1 - (y_{u,t} + 7.5)/l + 1, [0,1]) \\ &- clip(1 - (h_{u,t} - 0.5)/l + 1, [0,1]) \end{aligned}, \tag{3}$$

where $clip(a, [0,1])$ is a clipping function used to control the parameter a to satisfy the boundary constraint, i.e., not less than 0 and not greater than 1. Define V_{max} as the maximum speed of the UCAV, and $l = V_{max}\tau$ is the maximum travel distance of the UCAV in one cycle.

(2) When the UCAV is not equipped with a hard-killing weapon, it is worth encouraging the use of a suicide attack that collides with the observation station. Therefore, we directly use a reward signal with a value of 100 to motivate this particular offensive behavior.

(3) A continuous reward signal needs to be employed to guide the UCAV to learn efficient navigation strategies when training the DRL framework. Restricted by the detection range of radar, the relative displacement's normalization is adopted to describe the tracking and approaching of the target by the UCAV, which can be computed as follows:

$${}_{tracking}R_t = \begin{cases} 1 - ||\oslash(\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t})|| & \text{if } ||\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}|| < \rho_{max} \\ -||\oslash(\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t})|| & \text{else} \end{cases}. \tag{4}$$

where $\oslash$ indicates the normalization operator and $\oslash(\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}) = [(x_{a,t} - x_{u,t})/15, (y_{a,t} - y_{u,t})/15, (h_{a,t} - h_{u,t})/7.5]$ denotes the normalized relative displacement between the target and the UCAV. ρ_{max} is the maximum distance that the UCAV's radar can reach.

The united reward of the UCAV at any time is equal to the sum of the results of the three rewards defined above, i.e., $R_t = {}_{escape}R_t + {}_{tracking}R_t + (0 \; or \; 100)$.

3.1.4. Supplement

Explorer is based on the vCEW framework, which includes many unique designs about radar equations, such as using the line-of-sight (LOS) angle and the equivalent detection distance (related

to the radar cross section) to determine whether the station is within the field of view of the UCAV, assessing the collision caused by objects, and analyzing the stage progression of the radar based on a parametric data processing system (PDPS). The implementation of these model details can be found in [1].

3.2. Observation Estimation and Prediction

As a typical partially observable Markov decision process (POMDP), the UCAV receives feedback signals from partially observable environments during flight, and after a successful target search, it stores the information of the target in its memory bank. In Explorer, we allowed the PDPS to filter and predict the target's motion states, but when the radar sensor cannot perceive the target, other approaches are considered, such as variational Bayesian and neural networks (NNs), to predict the point where the target is most likely to appear.

The configuration parameters of the original version of Explorer are shown in Table 1. The game duration of each round is fixed at 400 s, based on different lengths of the operating cycle, and the difficulty of the search task can be divided into four levels: (1) the operating cycle corresponding to level 0 is 1 s, and the maximum number of operational steps is 400; (2) the operating cycle corresponding to level 1 is 0.5 s, and the maximum number of operational steps is 800; (3) the operating cycle corresponding to level 2 is 0.2 s, and the maximum number of operational steps is 2000; and (4) the operating cycle corresponding to level 3 is 0.1 s, and the maximum number of operational steps is 4000. Clearly, the shorter the operating cycle sets, the more complicated the motion planning of the autonomous navigation becomes.

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

The introduction of noise causes a deviation between the observed state and the actual state of the UCAV, which will affect the estimation/prediction accuracy and further affect the action decision-making of the UCAV.

As mentioned in Section 3.1.1, the prediction state can be represented by the station's location $\hat{\boldsymbol{p}}^o_{a,t}$, which differs from our previous work in that it employed a generative network to learn the probability distribution of the feedback state and estimated the prediction based on a variational Bayesian network. However, as verified by a large number of simulations [28], as long as the navigation goal is accurate (reaching the target field), we can consider directly using a NN to generate grid cell prediction agents, which supports route planning across unexplored spaces. Please note that before the UCAV begins to learn the state representation based on the grid network, arranging supervised learning experiments to implement path integration is essential.

3.2.1. Path Integration

Based on the long short-term memory (LSTM) (denoted as "grid LSTM"), the grid cell network of the agent was implemented as in the supervised learning setup by using the cumulative trajectories. The simulated UCAV started at a uniformly sampled position and motion direction within their ranges. For each task's UCAV, the motion model based on Equation (2) is used to obtain trajectories that cover the entire environment, where boundary clipping is necessary to avoid persistent collisions with environmental boundaries.

3.2.2. Cell Activations

Ground-truth place cell distribution and head-direction cell distribution are designed to create grid cells using a linear combination. For a given position $\boldsymbol{p}_u \in \mathbb{R}^3$, place cell activations, $\boldsymbol{c} \in [0,1]^N$, were simulated by the posterior probability of each component of a mixture of three-dimensional isotropic Gaussians,

$$c_i = \frac{e^{-\frac{||\boldsymbol{p}_u - \boldsymbol{\mu}_i^{(c)}||^2}{2(\sigma^{(c)})^2}}}{\sum_{j=1}^{N} e^{-\frac{||\boldsymbol{p}_u - \boldsymbol{\mu}_j^{(c)}||^2}{2(\sigma^{(c)})^2}}} \tag{5}$$

where the place cell centers $\boldsymbol{\mu}_i^{(c)}$ are N three-dimensional vectors selected uniformly at random before training, and the place cell scale $\sigma^{(c)}$ is a positive scalar specified by experiments.

Similarly, for a given motion angle A, head-direction cell activations, $\boldsymbol{h} \in [0,1]^M$, were represented by the posterior probability of each component of a mixture of von Mises distributions,

$$h_i = \frac{e^{\kappa^{(h)} \cos(\vec{A} - \boldsymbol{\mu}_i^{(h)})}}{\sum_{j=1}^{M} e^{\kappa^{(h)} \cos(\vec{A} - \boldsymbol{\mu}_j^{(h)})}} \tag{6}$$

where the M head-direction centers $\boldsymbol{\mu}_i^{(h)}$ are chosen uniformly from $[-\pi, \pi]$ at random before training, and the concentration parameter $\kappa^{(h)}$ is a positive scalar specified by experiments.

3.2.3. Grid Cell Network Architecture

As shown in Figure 2, the grid cell network architecture consists of three layers: a recurrent layer (an LSTM with 128 hidden units), a linear layer, and an output layer. The input of the LSTM is a composite vector constructed by the normalized position and velocity of the UCAV, which can be expressed as $CON(\oslash\hat{\boldsymbol{p}}_{u,t}, \oslash\boldsymbol{v}_{u,t})$, where $\oslash\hat{\boldsymbol{p}}_{u,t} = [(x_{u,t}+7.5)/15, (y_{u,t}+7.5)/15, (h_{u,t}-0.5)/7.5]$ is the normalized UCAV's position in Explorer space, and $\oslash\hat{\boldsymbol{v}}_{u,t} = [v_{ux,t}, v_{uy,t}, v_{uz,t}]/V_{max}$ denotes the normalized velocity. The linear layer is regularized by dropout [31] and is used to map place and head-direction units, and the output layer is used to generate predictions for place cells and head-direction cells.

The output of the LSTM (hidden state, $\vec{m}_t$) is connected to the input of a linear decoder, which is used to map the linear layer activations, $\boldsymbol{g}_t \in \mathbb{R}^{512}$. The output of the decoder maps $\boldsymbol{g}_t$ to the predicted place cells, $\boldsymbol{c}_{p,t}$, and the predicted head directions, $\boldsymbol{c}_{h,t}$, via SoftMax functions, and dropout [31] with a drop probability of 0.5 was applied to each $\boldsymbol{g}_t$ unit. Thus, there are three sets of weights and deviations that need to be optimized.

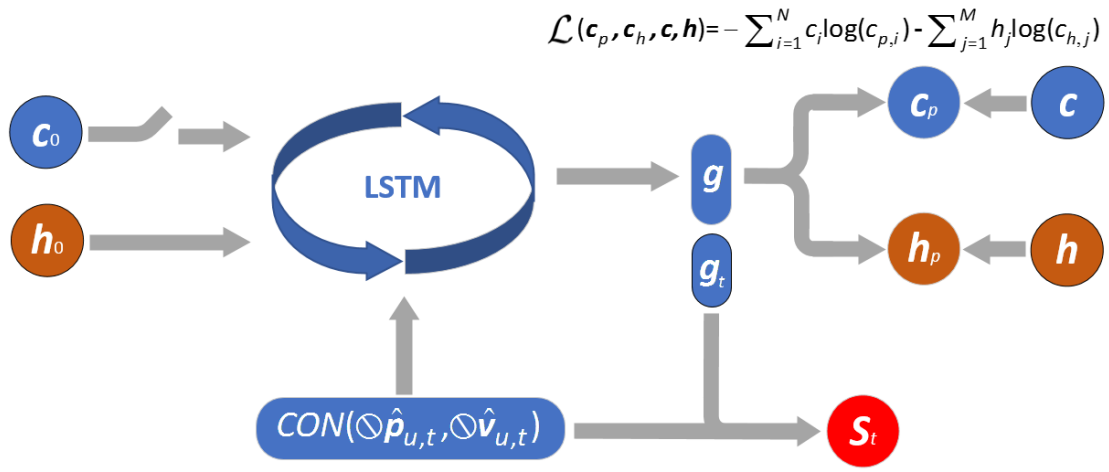

Figure 2. Grid network architecture in the supervised learning experiment. The recurrent layer is an LSTM with 128 hidden units. $\boldsymbol{c}$ and $\boldsymbol{h}$ represent the place cell activations and head-direction cell activations, respectively.

3.2.4. Objective Function

At each time step t, the place and head-direction cell ensemble activations can be predicted via training the grid network. For each task, the network is trained in a fixed environment where the place cell center remains unchanged. The objective function, $\mathcal{L}$, for optimizing the parameters of the grid cell network is to minimize the cross-entropy between the network place cell predictions, $\vec{c}_{p,t}$, and the synthetic place-cells targets, $\vec{c}_t$, and the cross-entropy between the head-direction predictions, $\vec{c}_{h,t}$, and their targets, $\vec{h}_t$:

$$\mathcal{L}(\boldsymbol{c_p}, \boldsymbol{c_h}, \boldsymbol{c}, \boldsymbol{h}) = -\sum_{i=1}^{N} c_i \log(c_{p,i}) - \sum_{j=1}^{M} h_j \log(c_{h,j}). \tag{7}$$

Through time, backpropagation is used to calculate the gradients of the objective function relative to the network parameters, and these parameters are updated using stochastic gradient descent. The network is unrolled into blocks of 200 time steps. Finally, in the UCAV's observation state, S_t, the position prediction of the estimated target can be replaced by $\boldsymbol{g}_t$, i.e.,

$$S_t(pre) = CON(\oslash\hat{\boldsymbol{p}}_{u,t}, \oslash\boldsymbol{v}_{u,t}, \oslash(\hat{\boldsymbol{p}}_{u,t} - \hat{\boldsymbol{p}}^{o}_{a,t})) \Rightarrow S_t = CON(\oslash\hat{\boldsymbol{p}}_{u,t}, \oslash\boldsymbol{v}_{u,t}, \boldsymbol{g}_t). \tag{8}$$

Please note that this paper mainly draws on the method in Reference [28] when training the grid cell network. Equations (5)–(7) have not considerably changed except for rewriting the symbols.

3.3. Behavior-Angle-Based Reward

For autonomous navigation, some special indicators are used to assess the learning behavior of the combat vehicle, such as the behavior angle (BHA). As shown in Figure 3, in Explorer, BHA is the angle between the UCAV's maneuvering direction and the LOS direction $\boldsymbol{n}_t$, which can be represented by $\vartheta_t = \langle \hat{\boldsymbol{a}}_{u,t}, \boldsymbol{n}_t \rangle$. The BHA can be calculated to observe the movement trend of the UCAV. For convenience of analysis, ϑ is further normalized to $\oslash\vartheta = \varrho/\pi \in [0, 1]$; when it is greater than 0.5, the UCAV is considered to be moving away from the target, and vice versa.

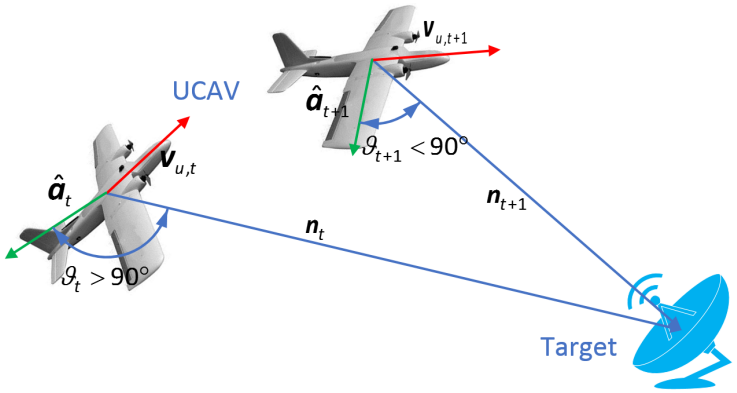

Figure 3. Schematic diagram of the behavior angle of UCAV in motion.

Section 3.1.3 defines rewards for different attributes according to the different navigation purposes in the reward-shaping and then weights them together. Based on the conclusion drawn from reference [1], the behavioral reward of approaching (or colliding with) the target allows the UCAV to learn that the navigation policy is to maintain a BHA close to the target field. Although this design award enables the UCAV to perform the task well in an Explorer game with a difficulty level of 2 (the definition of the game difficulty can be found in Section 3.2), it failed in the task with a difficulty level of 3, i.e., the highest level. After a thorough analysis and experiments, the following two aspects are considered to improve the learning system:

(1) Reference [1] uses a sparse positive reward to induce the UCAV to collide with the target and uses a continuous negative reward to prevent the UCAV from being too far from the target. However, we find that distance-based reward-shaping is information-redundant for guiding UCAV navigation. This is because the UCAV's velocity is the first-order difference of the distance; in VBN, the planned acceleration can achieve the most effective control of speed and head direction. Therefore, for the UCAV to extract the behavior of adjusting acceleration from the behavior of adjusting distance, the DRL algorithm is required to possess a great sample use.

(2) Because the motion model of the object in Explorer is based on vCEW, only the direction of the maneuver needs to be planned to obtain the desired acceleration. To fully use this advantage, we hope to reduce the information redundancy of training samples by modifying the reward function and then generally improve the performance of learning algorithms operating in Explorer.

For (1), we introduced the SAC algorithm in the DRL framework; for (2), as long as the BHA is less than 90° and the speed is greater than 0, then there is always an approach for the agent to arrive in the mission area. Using the facing angle $\langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \rangle \in [0, \pi]$ to indicate the motion state of the UCAV relative to the target, we modify the tracking reward function represented by Equation (4) to the following:

$$_{tracking}R'_t = \begin{cases} 1 - \langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \rangle / \pi & \text{if } ||\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}|| < \rho_{max} \\ -\langle \boldsymbol{v}_{u,t}, \boldsymbol{n}_t \rangle / \pi & \text{else} \end{cases}, \tag{9}$$

and the composite reward function becomes $R'_t = {}_{escape}R_t + {}_{tracking}R'_t + (0 \text{ or } 100)$. Simultaneously, we supplement a pure reward function to create a set of comparison simulations, i.e., $R''_t = {}_{escape}R_t + {}_{tracking}R''_t + (0 \text{ or } 100)$, where ${}_{tracking}R''_t$ is

$$_{tracking}R''_t = \begin{cases} 1 & \text{if } ||\hat{\boldsymbol{p}}_{a,t} - \hat{\boldsymbol{p}}_{u,t}|| < \rho_{max} \\ 0 & \text{else} \end{cases}. \tag{10}$$

Finally, the incentive process of navigation behavior using different reward functions (R_t, R'_t, and R''_t) can be analyzed via Figure 4a–c:

(1) When using the facing angle shown in Figure 4a to evaluate the accuracy of the search/tracking, the UCAV will directly understand the effect of acceleration from the changes in angular velocity. It is very easy to learn a VBN policy when the facing angle is less than 90°.

(2) As shown in Figure 4b, when using the relative displacement between the target and the UCAV as a motivating factor, the UCAV will understand the effect of velocity on navigation from the short-term changes in distance, while the resulting policies of motion should be only a by-product of acceleration planning (although the direction of the planned acceleration is what we most expect). Therefore, the training information sampled in this way is insufficient.

(3) Figure 4c shows that the UCAV is awarded tracking and collision rewards only after entering the target field, and there is no continuous feedback signal to guide the UCAV's action. In this way, although the UCAV can explore more behavior strategies, the generated training samples will carry considerable redundant information. Although the UCAV can explore more behavior strategies in this way, the generated training samples will carry substantial redundant information. Therefore, this reward-shaping method has high-performance requirements for the DRL algorithm, particularly in the case of a large amount of sampled data.

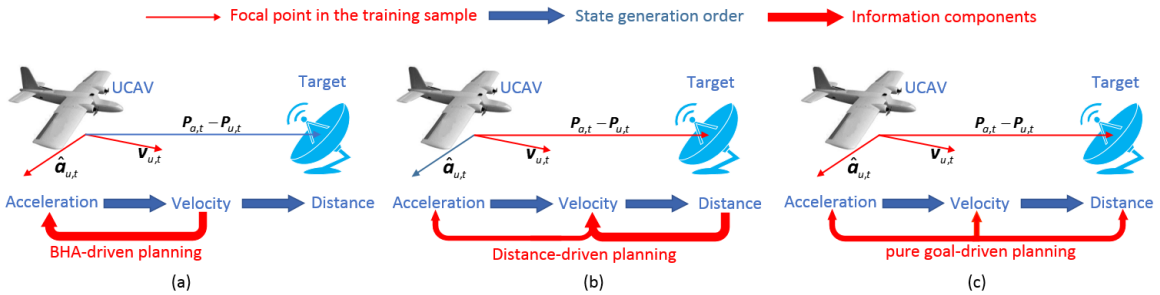

Figure 4. The behavior incentives provided by different reward functions: (**a**–**c**) correspond to R_t, R'_t, and R''_t, respectively. The thickness of the red arrow indicates the proportion of information dissemination.

4. Deep Reinforcement Learning Framework for Continuous Control

DRL can carefully choose the feature representations through autonomous learning based on deep learning; thus, the agent can learn more effectively to take actions to maximize the cumulative returns through interaction with the environment. In the domain of continuous control, where actions are continuous and often high-dimensional, practical engineering, such as intelligent navigation, requires a high capability of DRL algorithms, and we argue that there is no unique DRL algorithm that is efficient for all scenarios. With recent progress, based on the actor-critic (AC) framework shown in Figure 5, the DRL algorithms with a policy gradient (PG) as their core have advantages in convergence and efficiency. Therefore, this work integrates DDPG, PPO, A3C and SAC into a DRL framework for the Explorer game, in which the implementation of the PPO algorithm is divided into two types: penalty-based and probability-clipping-based.

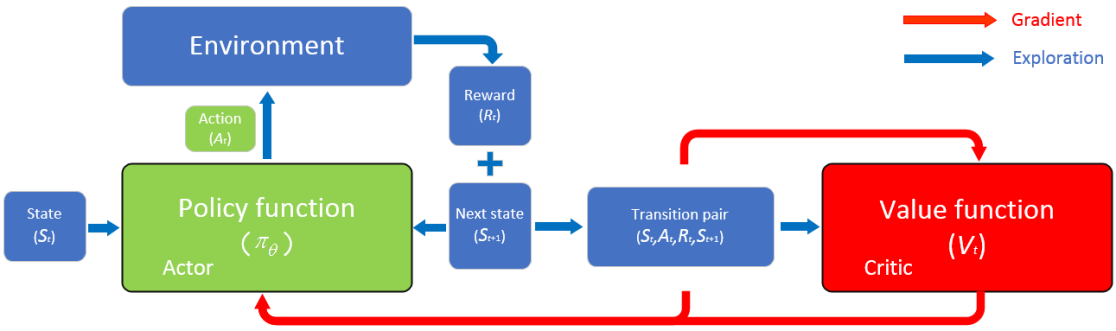

Figure 5. The actor-critic (AC) framework.

4.1. Preliminaries

In this section, we define the notation used in the subsequent sections.

Commonly, the navigation task model in Explorer conforms to a finite-horizon discounted Markov decision process (MDP), which can be defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma,)$, where $\mathcal{S}$ denotes the agent's state space of size $|S|$, $\mathcal{A}$ represents the action space of size $|A|$, $\mathcal{P}$ represents the transition probability distribution of the state, $\mathcal{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor that weights future rewards.

Most of the algorithms implemented in our DRL framework optimize a stochastic policy $\pi_\theta \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}\geqslant 0}$ via a network θ (or networks). For a continuous task, at time step t, the DRL-based agent first obtains the observed state, $S_t \in \mathcal{S}$, the agent then selects the optimal or sub-optimal action $A_t \sim \pi(A_t|S_t)$ from the policy function, and finally receives the reward R_t from the environmental feedback and prepares to enter the next state $S_{t+1} \sim P(S_{t+1}|S_t, A_t)$ after interacting with the environment. Multiple state transition pairs are concatenated to obtain the whole trajectory of the agent, which can be denoted as $\phi = \{...(S_{t-1}, A_{t-1}, R_{t-1}, S_t), (S_t, A_t, R_t, S_{t+1})...\}_{t=0}^{T}$. Therefore, the discount sum of future rewards is used to define the expected state-value function for π from S_t, as follows:

$$V^\pi(S_t) = \mathbb{E}_\phi[R_t|S_t] = \mathbb{E}_\phi[\sum_{t=0}^{T} \gamma^t R_t|S_t]. \tag{11}$$

Similarly, the state-action value function, Q, is defined as follows:

$$Q^\pi(S_t, A_t) = \mathbb{E}_\phi[\sum_{t=0}^{T} \gamma^t R_t|S_t, A_t]. \tag{12}$$

To output continuous actions, the following PG theorem can be used to update the network:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(A_t|S_t) V_t, \tag{13}$$

where V_t represents the unbiased sample of $Q^\pi(S_t, A_t)$, and α is the learning scale.

Denote the parameters of actor and critic networks in the AC as θ^μ and θ^Q, respectively. Please note that for the DRL algorithms of AC structure, the action A_t is chosen by a policy network μ_θ, i.e., $A_t = \mu_\theta(S_t) + \mathcal{N}_t$, where $\mathcal{N}$ is the artificially introduced exploration noise.

4.2. Algorithms

In this section, we briefly summarize the algorithms implemented in our Explorer environment and note any techniques for applying them to general parameterized modules, including

Memory replay: For each iteration, the UCAV generates N trajectories $\{\phi_n\}_{n=0}^{N}$, which contain state transition pairs sufficient for DRL training. However, due to the continuity of the trajectory, the sampled data generated are highly correlated. Thus, if the on-policy training is adopted, the DRL neural network will output an unstable action decision [17], and a divergent accumulative reward. A memory pool plays an important role in decorrelating them by storing and sampling these transfer pairs. Unique sampling methods such as [20] are generally used.

Dual networks: In the training process, using NNs to fit V^π or Q^π for DRL is very unstable. Reference [32] introduces a target network $\theta^{\mu'}$ (or $\theta^{Q'}$) to make the evaluated network of the DRL training independent of the target network. It also recommends that the target network be updated by delay update or soft update. The formula for the soft update is

$$\begin{cases} \theta^{\mu'} \leftarrow \sigma\theta^\mu + (1-\sigma)\theta^{\mu'} \\ \theta^{Q'} \leftarrow \sigma\theta^Q + (1-\sigma)\theta^{Q'} \end{cases} \quad \sigma \ll 1. \tag{14}$$

Action clipping: The physical state involved in the engineering model is generally constrained, and the DRL's actor network tends to output a value that exceeds the boundary of the agent's motion state. There are two corresponding solutions: clipping the action according to the boundary constraints or designing an ideal transformation function at the output of the network.
Cross-entropy method (CEM) [33]: Unlike the previous method of exploring through random actions, CEM explores directly in the policy parameter space. First, the outputs μ_k and σ_k of the actor network are designed to form the Gauss distribution space $\theta_n^\mu \sim \mathcal{N}(\mu_k, \sigma_k^2)$. Then, in each iteration, the output is sampled according to θ_n^μ, and the current policy distribution of the action is evaluated. Finally, the network is optimized by maximizing the entropy.
Advantage function: The basic idea of the PG method in updating the policy is to increase the probability action with a large reward and reduce the probability action with a small reward. Suppose that there is a scenario agent such that the reward is positive at all times and that the reward is set to be 0 for the actions that are not sampled. In this scenario, if a good action has not been sampled and the sampled action is bad and obtains a small positive reward, then the probability of the good action that is not sampled will become increasingly smaller, which is clearly inappropriate. Thus, creating a reward baseline and balancing the positive and negative rewards is necessary. This baseline can work through the following advantage function:

$$\hat{A}_t = Q^\pi(S_t, A_t) - V^\pi(S_t), \tag{15}$$

where $V^\pi(S_t)$ can be evaluated by a critic network.

4.2.1. Deep Deterministic Policy Gradient (DDPG)

DDPG [25] 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. Moreover, the DDPG applies gradient descent to the policy with minibatch data sampled from a replay pool, where the gradient for the critic network is computed as follows:

$$\frac{1}{B}\sum_t (R_t + \gamma Q'(S_{t+1}, \mu'(S_{t+1}|\theta^{\mu'})|\theta^{Q'}) - Q(S_{t+1}A_{t+1}|\theta^Q))^2, \tag{16}$$

and the loss function that should be backpropagated through the deep actor network is

$$\nabla_{\theta^\mu}\mu|_{S_t} \approx \frac{1}{B}\sum_t \nabla_A Q(S, A|\theta^Q)\Big|_{S=S_t, A=\mu(S_t)} \nabla_{\theta^\mu}\mu(S|\theta^\mu)|_{S_t}, \tag{17}$$

where B is the batch size.

4.2.2. Proximal Policy Optimization (PPO)

PPO [26] is an improved algorithm based on trust region policy optimization (TRPO) [34]. Because the maximization of the objective function in TRPO is limited by the size of the policy update, the PPO considers replacing the constraint with a penalty term, so Schulman et al. propose an unconstrained policy optimization scheme:

$$\underset{\theta}{\text{maximize}}\mathbb{E}_t\left[\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}\hat{A}_t - \beta\text{KL}[\pi_{\theta_{old}}(\cdot|S_t)||\pi_\theta(\cdot|S_t)]\right], \tag{18}$$

where θ_{old} is the vector of policy parameters before the update and β is the penalty coefficient.

Denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}$. Considering that the selection of coefficient β is difficult, Equation (18) can be modified as

$$\underset{\theta}{\text{maximize}}\mathbb{E}_t\left[\min(r_t(\theta)\hat{A}_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right], \tag{19}$$

where ϵ is a hyperparameter less than 1. For distinction, we denote the PPO algorithms based on Equations (18) and (19) as PPO (KL) and PPO (CLIP), respectively.

4.2.3. Asynchronous Advantage Actor-Critic (A3C)

A3C [23,24] is a multi-threaded algorithm based on the AC framework. By asynchronously executing multiple agents and taking different states experienced by parallel agents as training samples, the correlation between state transition samples generated in a single training process is removed. This algorithm can be implemented with only one standard multi-core CPU (very suitable for micro-labs) and is superior to traditional methods in terms of efficiency, time, and resource consumption. The update formula of the actor network in A3C is

$$\theta^{\mu} \leftarrow \theta^{\mu} + \alpha_{\mu} \nabla_{\theta_i^{\mu}} \log \pi_{\theta_i^{\mu}}(A_t|S_t) \hat{A}(S_t|\theta_i^{Q}), \tag{20}$$

where θ^{μ} (same as θ^{Q}) represents the global shared parameters, and θ_i^{μ} and θ_i^{Q} are the i-th thread-specific parameters.

Additionally, the critic network is updated as follows:

$$\theta^{Q} \leftarrow \theta^{Q} + \alpha_{Q} \partial (R_{\phi} - V_t(S_t|\theta_i^{Q}))^2 / \partial \theta_i^{Q}, \tag{21}$$

4.2.4. Soft Actor-Critic (SAC)

SAC [27] is an off-policy and model-free DRL algorithm with sample efficiency that is sufficient to solve the problems of real-world robot learning in hours. In addition, SAC's hyperparameters are quite robust, requiring only a single hyperparameter set to perform well in different simulation environments. The SAC is an AC framework with maximum entropy objective:

$$J(\pi) = \sum_{t=0}^{T-1} E_{t \sim \rho_{\pi}} [R_t - \delta \log \pi(A_t|S_t)], \tag{22}$$

where ρ_{π} denotes the state and state-action margins of the trajectory distribution induced by a policy $\pi(A_t|S_t)$. δ is the temperature parameter that determines the relative importance of the entropy term against the reward and thus controls the stochasticity of the optimal policy.

Furthermore, Haarnoja et al. [27] proposed a practical multimodal representation based on a mixture of K Gaussians to provide a distribution in medium-dimensional action spaces. This distribution is more expressive and easier to handle, and it can endow SAC with more effective exploration and robustness in the framework of entropy maximization.

5. System Implementation and Simulation Details

This section first describes the system framework based on multiple DRL algorithms for testing the Explorer game and configures the core parameters; then, it introduces the simulation environment of hardware and software in the experiment. Finally, based on different goal-driven rewards, the proposed system is run in games with different difficulty levels, and the corresponding metrics are obtained and analyzed.

5.1. System Configurations

The complete system framework is shown in Figure 6. The left blue box is used to interact with the Explorer environment and generate the estimated state of the agent. The right grey box is a DRL-based decision framework to guide the behavior of the agent (UCAV). There are five algorithms in the DRL framework, and the detailed hyperparameters of each algorithm are presented in Table 2. Please note that in Table 2, 2 × 2FC-(520, 1024, 1) means that there are two networks, and each network consists of two fully connected (FC) layers, which have 520 input units, 1024 connection units, and 2 output units.

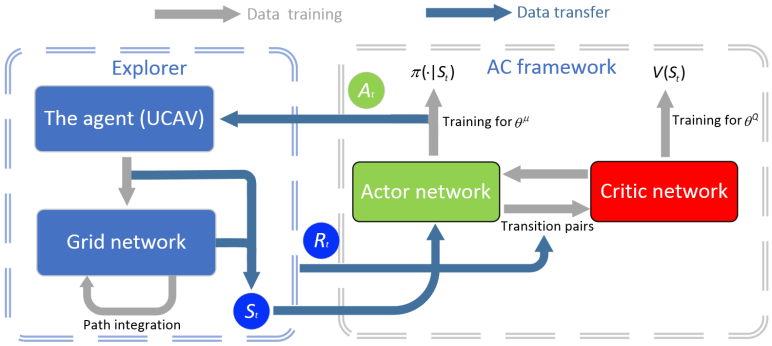

Figure 6. DRL-based cognitive system for Explorer. Please note that the S_t is the cognitive state of the UCAV, and the R_t is the real reward induced by the real state of the Explorer game.

Table 2. Detailed network configurations for the five algorithms in the DRL framework.

Algorithm	Layers	Modularization Techniques					Hyperparameters
		Memory Replay	Dual Networks	Action Clipping	Advantage Function	CEM	
DDPG	Actor: 2 × 2FC-(518, 1024, 2) Critic: 2 × 2FC-(520, 1024, 1)	✓	✓	✓	×	×	Learning rate for actor: 2×10^{-5} Learning rate for critic: 10^{-4} Discount factor γ: 0.9 Soft replacement factor σ: 5×10^{-3} Memory capacity: 3×10^{5} Batch size: 128 Optimizer: Adam
SAC	Actor: 2 × 2FC-(518, 1024, 2) Critic: 2 × 2FC-(520, 1024, 1)	✓	✓	×	✓	✓	Learning rate for actor: 5×10^{-3} Learning rate for critic: 10^{-3} Discount factor γ: 0.99 Soft replacement factor σ: 5×10^{-3} Memory capacity: 3×10^{6} Batch size: 128 Optimizer: Adam
A3C	Actor: 2FC-(518, 1024, 2) Critic: 2FC-(520, 1024, 1)	×	×	✓	✓	✓	Learning rate for actor: 2×10^{-5} Learning rate for critic: 10^{-4} Discount factor γ: 0.9 Global update steps: 10 Optimizer: RMSPro
PPO (KL)	Actor: 2FC-(518, 1024, 2) Critic: 2FC-(520, 1024, 1)	×	×	✓	✓	✓	Learning rate for actor: 2×10^{-5} Learning rate for critic: 10^{-4} Discount factor γ: 0.9 Penalty coefficient β: 0.5 Update steps of critic: 10 Update steps of actor: 10 Optimizer: Adam
PPO (CLIP)	Actor: 2FC-(518, 1024, 2) Critic: 2FC-(520, 1024, 1)	×	×	✓	✓	✓	Learning rate for actor: 2×10^{-5} Learning rate for critic: 10^{-4} Discount factor γ: 0.9 Clipping coefficient ϵ: 0.2 Update steps of critic: 10 Update steps of actor: 10 Optimizer: Adam

5.2. Simulation Platform of Software and Hardware

The software and hardware versions of the simulation platform clearly limit the learning efficiency of the DRL system. We tested the five DRL algorithms horizontally on a single platform. The hardware parameters of the platform are a PC running Windows 8 with a XeonE5-1620 v43.50 GHz CPU and 16 GB of DDR4 RDIMM memory. The software parameters of the platform are listed as follows: (1) Python v3.5.4 is used as the programming language; (2) TensorFlow v1.7.0 is used as the machine learning framework; and (3) Pyglet v1.3.2 is used to realize visualization of maneuvering physical models.

5.3. Simulation Details and Metrics

Based on the combination of the game environment of four levels and three reward functions, we design 12 groups of comparison simulations for the DRL framework. At each level, 10,000 episodes of

the game are run with a specific random seed used for randomly generating the training data; for each episode, to enhance the generalization ability of the DRL framework, the initial states of the UCAV and target observation station require a random reinitialization at the beginning of the game. To observe the behavior changes of the UCAV before and after training, we introduce a set of random actions as the benchmarking strategy for game agents in each episode.

In the comparison of the simulation results, we divided the total episodes of the game into two parts: the previous 7500 episodes are used to train the DRL framework, and the last 2500 episodes are used to test the performance of the DRL framework after training. Additionally, although each episode of the game has a fixed number of termination steps (the maximum operational steps corresponding to the current task), if the UCAV successfully tracks the target for 50 cycles or collides with the target, then the episode's game will be terminated prematurely.

Four metrics, the win rate (WR), tracking rate (TR), mean accumulated reward (MAR), and computational time (CT), were employed in the experiments, where WR refers to the percentage of episodes in which the UCAV ends the game by destroying the target observation station and TR refers to the percentage of episodes in which the UCAV successfully locates the target and keeps track until the end of the game in the current mission. For the desired combat strategies, WR and TR represent the UCAV's ability to find the optimal strategy and the sub-optimal strategy, respectively. Because the physical constraints make the UCAV need to constantly adjust its attitude when approaching the target, the UCAV at the end of the path planning is more worthy of attention than that at the beginning of the game. Define the calculation formula for MAR as follows:

$$MAR = \frac{1}{\wedge(|\phi|, \zeta)} \sum_{t=-1}^{\vee(-|\phi|,-\zeta)} R_t \quad , \tag{23}$$

where $|\phi|$ denotes the total step size of the current task trajectory and $\vee$ and $\wedge$ are the maximum and minimum operators, respectively. t=−1 means the reciprocal first step of the trajectory, and ζ is the cumulative step size of MAR after completing the task. The MAR in Equation (23) can be used to observe how the UCAV understands the behavior of "target-searching" [1].

Please note that for each level in Explorer, WR and TR are statistical values that are calculated after the complete test set has been run, and MAR is the statistical value that is returned immediately at the end of each episode's game. Although colliding with a target can bring a great reward, the distribution of this behavior's policies is sparse. Therefore, the agent tends to fall into a local optimum when learning in a continuous state space. Considering the above phenomenon, the DRL-based agents are always trained with a large variance, which results in an unstable oscillation of the MAR value even when the algorithm converges. Therefore, the method of multi-episode averaging to smooth the data of the MAR is adopted: $\overline{MAR} = \frac{1}{\zeta_e} \sum_{I=1+(i-1)\zeta_e}^{i\zeta_e} MAR_I$, where $i = 1, 2, 3, ..., (10,000/\zeta_e)$ and ζ_e is the smoothing scale. To adaptively observe the performance of the DRL framework at different game levels (from level 0 to level 3), we set ζ to be $1/40$ of the maximum operational steps of the current task and set ζ_e to 40.

5.4. Simulation Results

In this section, simulation results are analyzed from the three aspects of reward function, behavior strategy, and algorithmic robustness.

Analysis of reward-shaping effects: the convergence performance of five algorithms in the DRL framework for different level tasks is shown in Figure 7. The long-term stability of the mean value of MAR indicates that the trained UCAV successfully masters the policy of target-searching, and the higher the peak of the MAR curve jitters, the more wins the UCAV has, i.e., the stronger the ability of the UCAV to find the global optimal solution. As demonstrated by Figure 7, the DRL framework with a pure goal-driven incentive can maintain better performance in simple tasks (such as level 0 and level 1), but for high-level tasks (such as level 2 and level 3), the quality of the behavior generated by

the framework decreases dramatically, which leads to the divergence of MAR values. When using distance-based rewards, the SAC algorithm has a certain improvement in the WR of all levels of tasks, and the performance of the PPO (KL) and PPO (CLIP) algorithms changes drastically, while the performances of the A3C and DDPG algorithms hardly change. Fortunately, the BHA-driven incentive enabled the performance of the DRL framework to be state-of-the-art because in the most difficult game, algorithms including SAC show an obvious convergence trend, and the PPO algorithm even reaches 88.68% WR in 8 hours.

Analysis of behavior strategies: from Figure 8, we can clearly understand which maneuvering strategy is well performed for the UCAV to search and destroy the target. Through careful observation, we conclude that the algorithm that performs well in searching targets aims to output a low and stable BHA value in motion planning. Clearly, the BHA value obtained by the DRL framework using R'_t as the incentive is smaller and more stable than the BHA values of using the other two reward functions. Therefore, it is feasible to optimize the convergence efficiency of the DRL algorithm by reducing the learning of redundant control information.

Analysis of the algorithmic robustness: based on the detailed performance values reported in Table 3, the robustness of the DRL framework can be comprehensively evaluated. The DRL framework always maintains outstanding performance in low-difficulty tasks: currently, the A3C algorithm not only has the best searching ability but also has a satisfactory time-consuming ability. The combination of DRL algorithms and the reward function R'_t has a good effect on solving the tasks of high levels, among which PPO (CLIP)'s performance is the most commendable. Additionally, the high use rate of SAC algorithm for data samples makes its MAR value possible to converge quickly and achieve 63.48%/23.44% WR even when using R_t/R''_t as the reward function in the most difficult task (up to 30 million total samples). However, one unsatisfactory point is that the SAC algorithm is more likely to fall into a local optimum (a high TR rather than a high WR), which makes it often take more time to find the global optimal strategy in simple tasks.

Table 3. Performance of the proposed DRL framework at different game levels and reward functions, metrics includes the WR (%), TR (%), and CT (*h*).

Reward	Level	Random			DDPG			SAC			A3C			PPO(KL)			PPO(CLIP)		
		WR	TR	CT	WR	TR	CT	WR	TR	CT	WR	TR	CT	WR	TR	CT	WR	TR	CT
R_t	0	2.24	4.90	1.01	99.76	46.73	2.14	98.96	64.64	4.07	99.76	62.83	0.75	73.56	55.10	0.99	98.80	65.35	0.52
	1	2.04	4.45	1.73	99.08	44.20	3.27	98.96	65.70	5.97	96.12	55.25	1.59	8.92	28.44	2.31	97.72	56.07	1.15
	2	2.04	4.25	4.86	16.60	14.03	26.76	87.08	71.21	31.95	2.04	3.12	9.14	1.72	3.39	9.10	19.08	19.72	7.91
	3	1.08	3.86	7.00	2.76	6.24	54.49	63.48	71.07	83.29	0.84	3.17	20.45	1.20	2.81	15.88	5.40	7.85	12.47
R'_t	0	2.24	4.90	1.01	99.96	47.20	2.07	99.96	56.13	4.19	99.96	62.24	0.64	99.20	66.67	0.52	99.20	68.52	0.48
	1	2.04	4.45	1.73	99.72	43.58	2.81	99.96	52.07	4.70	100	61.42	0.78	99.68	61.68	0.61	99.68	63.65	0.72
	2	2.04	4.25	4.86	53.84	34.44	12.53	99.96	48.62	7.86	100	56.53	1.67	69.88	44.93	10.51	98.36	53.81	1.80
	3	1.08	3.86	7.00	15.96	19.96	60.90	99.96	44.25	25.40	0.48	2.70	16.67	0.19	0.27	18.53	88.68	46.76	7.64
R''_t	0	2.24	4.90	1.01	99.92	46.79	2.04	95.96	69.95	5.12	99.80	62.43	0.77	87.24	72.23	0.78	94.64	69.81	0.61
	1	2.04	4.45	1.73	97.56	44.88	3.07	41.72	86.13	11.60	88.96	60.52	1.97	74.92	75.56	1.72	57.92	78.92	1.87
	2	2.04	4.25	4.86	5.92	8.60	17.79	25.52	88.40	35.40	1.40	3.33	6.08	65.88	58.14	4.49	72.84	65.04	3.81
	3	1.08	3.86	7.00	2.52	5.53	39.46	23.44	59.11	125.59	1.08	3.14	18.62	26.84	78.20	22.11	7.88	12.45	33.83

To summarize, the application of reward-shaping based on BHA in the DRL framework can enable the UCAV to automatically and quickly complete fine- and long-term navigation tasks, including intelligent target-searching and target tracking.

Videos of all Explorer experiments are available at the following website: https://github.com/youshixun/vCEW.

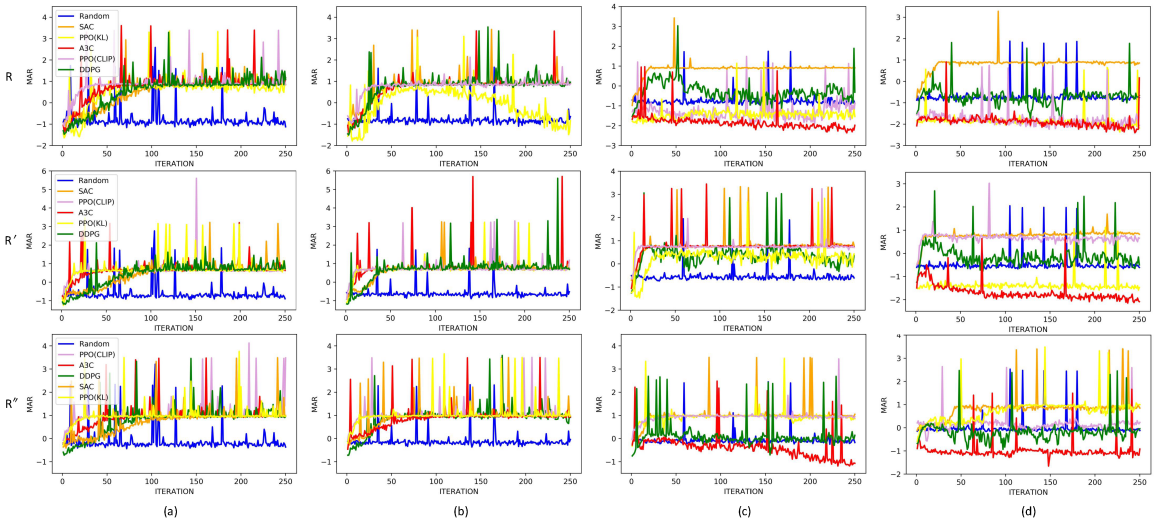

Figure 7. Evaluating the convergence performance of the DRL framework in Explorer. The twelve subfigures from left to right correspond to the results for game levels 0 to 3, and from top to bottom correspond to the reward functions R_t, R'_t, and R''_t, respectively.

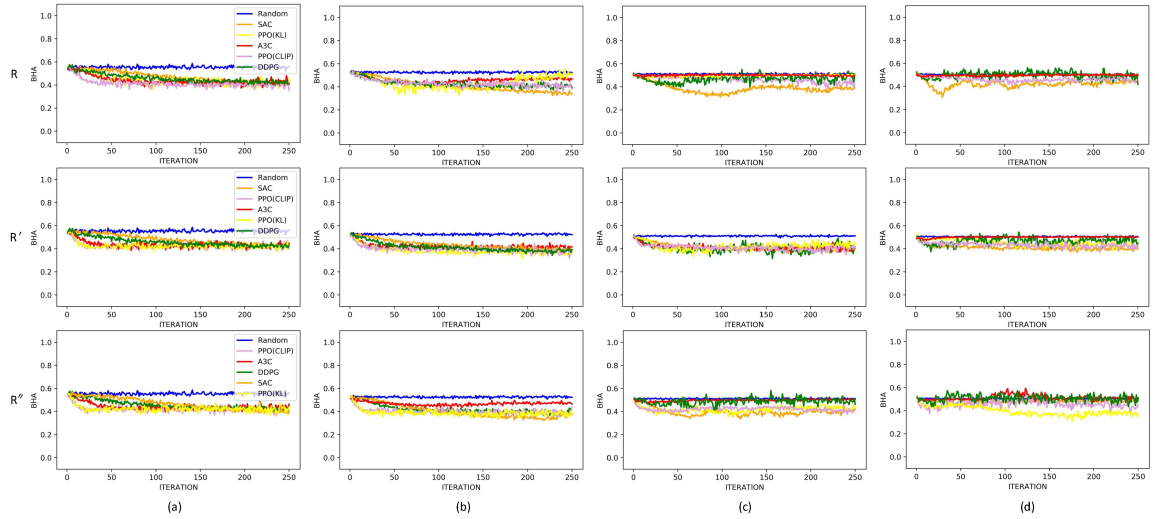

Figure 8. Investigating the potential behavior policies learned by the UCAV in Explorer. The twelve subfigures from left to right correspond to the results for game levels 0 to 3, and from top to bottom correspond to the reward functions R_t, R'_t, and R''_t, respectively.

6. Conclusions

In the framework, five DRL algorithms are adapted to the Explorer game, aiming to solve long-term intelligent target search tasks and other well-characterized electronic warfare tasks, including target-tracking or target striking. First, to enable the UCAV to accurately estimate the target position in 3D space, we use path integration to generate a large number of data samples to perform supervised training on the designed mesh unit network to obtain the predicted features of VBN rather than the traditional relative displacement. Second, by analyzing the motion state of the continuous control model and the information redundancy existing between different states, a reward-shaping method based on the BHA is designed. Finally, the network configuration details of the five DRL algorithms are refined and combined with the three goal-driven incentives to form a complete DRL framework. It has been proven through sufficient experiments that the combination of the BHA-driven incentive and A3C algorithm is suitable for the autonomous motion planning of low-precision short-term navigation tasks; for any difficulty of the Explorer game, the SAC algorithm has excellent performance when using BHA-based rewards.

Our work not only pioneered verifying the ability of popular DRL algorithms to perform target-searching tasks in 3D continuous action space, but also achieved breakthrough results in optimizing the end-to-end action policies. However, in the future, there are still many aspects worthy of in-depth research, among which we are most concerned about solving navigation problems in the environments of multiagent cooperation and competition, such as real-time obstacle avoidance and moving target-tracking. In addition, when introducing weapon models such as missiles and jammers into the vCEW framework, a visual physics engine software is needed to better evaluate the interaction between these models and various game environments. At the same time, with the increasing complexity of engineering models, the development of high-performance DRL algorithms is imminent.

Author Contributions: Conceptualization, M.D., S.Y. and L.G.; Funding acquisition, L.G.; Resources, S.Y. and M.D.; Software, S.Y.; Writing, S.Y.

Funding: This work was supported in part by the Equipment Pre-research Fund under Grant 61404150101, in part by the Shanghai Aerospace Science and Technology Innovation Fund under Grant SAST2017-068.

Acknowledgments: We thank all those who have provided valuable advice for the writing and experiment design of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. You, S.; Diao, M.; Gao, L. Deep Reinforcement Learning for Target Searching in Cognitive Electronic Warfare. *IEEE Access* **2019**, *7*, 37432–37447. [CrossRef]
2. Fitelson, M.M. Cryogenic electronics in advanced sensor systems. *IEEE Trans. Appl. Supercond.* **2004**, *5*, 3208–3213. [CrossRef]
3. Ernest Potenziani, I.I. Current and Future Trends in Military Electronic Warfare Systems and the Role of Thin Films and Related Materials. *Ferroelectrics* **2006**, *342*, 151–161. [CrossRef]
4. Das, S. Situation assessment in urban combat environments. In Proceedings of the 2005 7th International Conference on Information Fusion, Philadelphia, PA, USA, 25–28 July 2005.
5. Xue, S.; Jiang, G.; Tian, Z. Using Fuzzy Cognitive Maps to Analyze the Information Processing Model of Situation Awareness. In Proceedings of the Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 26–27 August 2014; pp. 245–248.
6. Ma, S.; Zhang, H.; Yang, G. Target threat level assessment based on cloud model under fuzzy and uncertain conditions in air combat simulation. *Aerosp. Sci. Technol.* **2017**, *67*, 49–53. [CrossRef]
7. Chung, T.H.; Burdick, J.W. Analysis of Search Decision Making Using Probabilistic Search Strategies. *IEEE Trans. Robot.* **2012**, *28*, 132–144. [CrossRef]
8. Lavis, B.; Furukawa, T.; Whyte, H.F.D. Dynamic space reconfiguration for Bayesian search and tracking with moving targets. *Auton. Robot.* **2008**, *24*, 387–399. [CrossRef]
9. Chen, X.; Li, Y.; Pang, Y.; Chen, P. Underwater Terrain Navigation Based on Improved Bayesian Estimation. In Proceedings of the Third International Conference on Digital Manufacturing and Automation, Guilin, China, 31 July–2 August 2012; pp. 1002–1005.
10. Portugal, D.; Rui, P.R. Cooperative multi-robot patrol with Bayesian learning. *Auton. Robot.* **2016**, *40*, 929–953. [CrossRef]
11. You, S.; Gao, L.; Diao, M. Real-Time Path Planning Based on the Situation Space of UCAVs in a Dynamic Environment. *Microgravity Sci. Technol.* **2018**, *30*, 899–910. [CrossRef]
12. Zhu, L.; Cheng, X.; Yuan, F.G. A 3D collision avoidance strategy for UAV with physical constraints. *Measurement* **2016**, *77*, 40–49. [CrossRef]
13. Noh, S.; Jeong, U. Intelligent Command and Control Agent in Electronic Warfare Settings. *Int. J. Intell. Syst.* **2010**, *25*, 514–528. [CrossRef]
14. Dadgar, M.; Jafari, S.; Hamzeh, A. A PSO-based multi-robot cooperation method for target searching in unknown environments. *Neurocomputing* **2016**, *177*, 62–74. [CrossRef]

15. Braca, P.; Goldhahn, R.; Lepage, K.D.; Marano, S.; Matta, V.; Willett, P. Cognitive Multistatic AUV Networks. In Proceedings of the 17th International Conference on Information Fusion, Salamanca, Spain, 7–10 July 2014; pp. 1–7.
16. Fan, Q.; Wang, F.; Shen, X.; Luo, D. Path planning for a reconnaissance UAV in uncertain environment. In Proceedings of the IEEE International Conference on Control and Automation, Kathmandu, Nepal, 1–3 June 2016; pp. 248–252.
17. Volodymyr, M.; Koray, K.; David, S.; Rusu, A.A.; Joel, V.; Bellemare, M.G.; Alex, G.; Martin, R.; Fidjeland, A.K.; Georg, O. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529.
18. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
19. Van Hasselt, H.; Guez, A.; Hessel, M.; Mnih, V.; Silver, D. Learning values across many orders of magnitude. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 5–10 December 2016.
20. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2016**, arXiv:1511.05952.
21. Wang, Z.; de Freitas, N.; Lanctot, M. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1511.06581.
22. Gu, S.; Lillicrap, T.; Sutskever, I.; Levine, S. Continuous Deep Q-Learning with Model-based Acceleration. In Proceedings of the International Conference on International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
23. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the International Conference on International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
24. Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; Kautz, J. Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU. *arXiv* **2016**, arXiv:1611.06256.
25. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *Comput. Sci.* **2015**, *8*, A187.
26. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
27. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.
28. Banino, A.; Barry, C.; Uria, B.; Blundell, C.; Lillicrap, T.; Mirowski, P.; Pritzel, A.; Chadwick, M.J.; Degris, T.; Modayil, J. Vector-based navigation using grid-like representations in artificial agents. *Nature* **2018**, *557*, 429–433. [CrossRef]
29. Bhatti, S.; Desmaison, A.; Miksik, O.; Nardelli, N.; Siddharth, N.; Torr, P.H.S. Playing Doom with SLAM-Augmented Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1612.00380.
30. Ma, X.; Xia, L.; Zhao, Q. Air-Combat Strategy Using Deep Q-Learning. In Proceedings of the 2018 Chinese Automation Congress (CAC), Xi'an, China, 30 November–2 December 2018; pp. 3952–3957.
31. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
32. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
33. Szita, I.; Lorincz, A. Learning Tetris using the noisy cross-entropy method. *Neural Comput.* **2006**, *18*, 2936–2941. [CrossRef] [PubMed]
34. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. In Proceedings of the 31st International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.