IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 3, 2020, accepted December 13, 2020, date of publication December 21, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3046284

Autonomous Control of Combat Unmanned Aerial Vehicles to Evade Surface-to-Air Missiles Using Deep Reinforcement Learning

GYEONG TAEK LEE AND CHANG OUK KIM
Department of Industrial Engineering, Yonsei University, Seoul 03722, South Korea
Corresponding author: Chang Ouk Kim (kimco@yonsei.ac.kr)

This work was supported by the Agency for Defense Development under Grant UD170043JD.

ABSTRACT This paper proposes a new reinforcement learning approach for executing combat unmanned aerial vehicle (CUAV) missions. We consider missions with the following goals: guided missile avoidance, shortest-path flight and formation flight. For reinforcement learning, the representation of the current agent state is important. We propose a novel method of using the coordinates and angle of a CUAV to effectively represent its state. Furthermore, we develop a reinforcement learning algorithm with enhanced exploration through *amplification of the imitation effect* (AIE). This algorithm consists of self-imitation learning and random network distillation algorithms. We assert that these two algorithms complement each other and that combining them amplifies the imitation effect for exploration. Empirical results show that the proposed AIE approach is highly effective at finding a CUAV's shortest-flight path while avoiding enemy missiles. Test results confirm that with our method, a single CUAV reaches its target from its starting point 95% of the time and a squadron of four simultaneously operating CUAVs reaches the target 70% of the time.

INDEX TERMS Deep reinforcement learning, combat unmanned aerial vehicle, deep learning, autonomous flight management system, path planning, exploration.

I. INTRODUCTION

Combat unmanned aerial vehicles (CUAVs) will be an important resource in future military systems because they can replace humans in performing dangerous or important tasks. Thus, CUAVs save human resources. In the future, it is anticipated that CUAVs will have the ability to determine reasonable actions by recognizing and evaluating changes in a military environment, such as enemy surface-to-air threats, in real time and without human intervention. In addition, such CUAVs will be able to carry out tasks such as reconnaissance and target attacks [1].

The aim of reinforcement learning (RL) is to learn an optimal agent policy for a control problem by maximizing the expected return. RL has shown high performance in dense reward environments such as games [2]. However, in many real-world problems, the rewards are sparse, and it is necessary to explore the environment. The RL literature has suggested various exploration methods to address this challenge, such as count-based exploration [3], [4], entropy-based exploration [5], [6] and curiosity-based exploration [7]–[10]. In recent years, researchers have added an exploration bonus, often called a curiosity or intrinsic reward, which is calculated as the difference between the predicted state and the actual next state. The intrinsic reward approach is efficient for exploration because the network that predicts the next state can drive the agent to behave differently from its previous action.

This paper focuses on combining self-imitation learning (SIL) [11] and random network distillation (RND) [12]. SIL is an algorithm that indirectly leads to deep exploration by exploiting only the best historical decisions. The authors proposed a method of evaluating whether an action is a good decision by comparing the return values obtained through actions taken by the agent in the past with the current value. Thus, the authors demonstrated how the agent can seek a good policy by exploiting good decisions. However, in hard exploration environments, it does not make sense only to exploit good past decisions by means of the SIL mechanism. In other words, an exploration bonus is also required.

RND solves the hard exploration problem by awarding an exploration bonus using a deterministic prediction

The associate editor coordinating the review of this manuscript and approving it for publication was Santhosh Kumar Gopalan.

error approach. The RND bonus is based on the deterministic prediction error of a neural network predicting features of observations. The authors proposed the creation of a fixed target network and a predictor network that learns the output of the target network; then, the difference between the outputs of the two networks can be used as an exploration bonus. In this way, the authors demonstrated significant performance gains in some hard exploration Atari games. However, under the RND approach, catastrophic forgetting can occur during learning because the predictor network learns only about the states that the agent has visited most recently. Consequently, the prediction error aggregates over time, and the exploration bonus increases for previously visited states. We will describe SIL and RND in detail in Section 3 and describe catastrophic forgetting in detail in Section 4.3.

This paper introduces a new RL approach for executing CUAV missions. We define the CUAV's mission as follows: evade guided missiles, find the shortest path, and reach the target destination. We have developed RL models for a single CUAV and for a CUAV formation. For efficient RL in a multidimensional space, this paper proposes *amplification of the imitation effect* (AIE) based on a combination of SIL and RND to drive deep exploration. In addition, we introduce techniques to enhance the strength of the proposed network. Adding an intrinsic penalty to a state that is repeatedly visited by the agent leads to deviations from the current converged policy. Moreover, to avoid catastrophic forgetting, we use a pool of stored samples to update the predictor network during imitation learning such that the predictor network can uniformly learn the visited states. We experimentally demonstrate that these techniques lead to deeper exploration. We verify the exploration performance of the proposed algorithm based on experiments in a two-dimensional grid environment and in a CUAV mission environment.

We constructed the experimental environment by simulating the flight maneuvers of a CUAV in a three-dimensional (3D) space. The objective of the RL agent is to learn the maneuvers that will allow the CUAV to reach a target point while avoiding missiles from an enemy air defense network. The remainder of this study is organized as follows. Section 2 reviews the relevant studies on RL and maneuvering for CUAVs. Section 3 defines the problems we wish to address through RL. Section 4 presents the proposed algorithm. Section 5 presents the experimental results of the proposed algorithm. Section 6 concludes the study and discusses future research topics.

II. RELATED WORK

Experience replay [13] is a technique for exploiting past experiences. The Deep Q-Network (DQN) algorithm has achieved human-level performance in Atari games using this technique [2], [14]. Prioritized experience replay [15] is a method for sampling prior experiences based on temporal differences. ACER [16] and Reactor [17] each utilize replay memory in an actor-critic algorithm [18], [19]. However, this method is not efficient when the past policy is too different from the current policy [11]. SIL is immune to this disadvantage because it exploits only past experiences that garnered higher returns than the current value.

Exploration has been a primary challenge in RL, and many studies have proposed methods of exploration enhancement. A count-based exploration bonus [20] is an intuitive and effective mechanism for encouraging exploration in which an agent receives a bonus when it visits a novel state, and the bonus decreases if the agent visits a frequently visited state. In some studies, the density of states has also been estimated to generate bonuses in a large state space [3], [4], [21], [22]. Recent studies have introduced the concept of the prediction error (curiosity), which is the difference between the next predicted state and the actual next state during exploration [7]–[10], [23]. In these studies, the prediction error was designed to function as an exploration bonus (i_t) that gives agents greater rewards when they perform unexpected behaviors.

However, the prediction error is stochastic in nature because the target function is stochastic. In addition, the architecture of the predictor network tends to be too limited to effectively generalize the state of the environment. To solve these problems, RND [12] was proposed, in which the target network is made deterministic by fixing its weights to randomized values and the predictor network has the same architecture as the target network. Other methods for efficient exploration include adding parameter noise within the network [20], [24], maximizing entropy policies [5], [6], adversarial self-play [25] and learning diverse policies [26], [27].

SIL can indirectly lead to deep exploration through the imitation of good decisions made in the past [11]. To exploit such past decisions, the authors used a replay buffer $\mathcal{D} = \{(s_t, a_t, R_t)\}$, where s_t and a_t are the state and action, respectively, at the t-th step and $R_t = \Sigma_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted sum of the reward at the t-th step with a discount factor γ. The authors proposed the following off-policy actor-critic loss:

$$\mathcal{L}^{sil} = \mathbb{E}_{s,a,R\in\mathcal{D}}[\mathcal{L}^{sil}_{policy} + \beta^{sil}\mathcal{L}^{sil}_{value}], \tag{1}$$

$$\mathcal{L}^{sil}_{policy} = -log\pi_\theta(a|s)(R - V_\theta(S))_+, \tag{2}$$

$$\mathcal{L}^{sil}_{value} = \frac{1}{2} \parallel (R - V_\theta(S))_+ \parallel^2, \tag{3}$$

where $(\cdot)_+ = max(\cdot, 0)$; π_θ and $V_\theta(s)$ are the policy (i.e., the actor) and the value function, respectively, parameterized by θ; and $\beta^{sil} \in \mathbb{R}^+$ is a hyperparameter that controls the value loss. Intuitively, for the same state, if the past return value is greater than the current value ($R > V_\theta$), then it can be inferred that the behavior in the past was a good decision. Therefore, imitating this behavior is desirable. However, if the past return is less than the current value ($R < V_\theta$), then imitating this behavior is not desirable. The authors focused on combining SIL with the advantage

actor-critic (A2C) framework [28] and reported significant performance gains in experiments based on hard exploration Atari games.

The authors of RND proposed a fixed target network (f) with randomized weights and a predictor network ($\widehat{f}$) trained using the output of the target network. The predictor neural network is trained via gradient descent to minimize the expected mean squared error $\| \widehat{f}(x;\theta) - f(x) \|^2$, and an exploration bonus (i_t) of $\| \widehat{f}(x;\theta) - f(x) \|^2$ is introduced. In other words, the difference between the outputs of the predictor network and the target network for the state feature is provided as an exploration bonus, and the two networks are modules whose only purpose is to generate an exploration bonus. Intuitively, the prediction error will increase for a novel state and decrease for a state that has been frequently visited. However, if the agent converges to a local policy, then a nonzero prediction error (i_t) may no longer arise. Furthermore, using RND can cause catastrophic forgetting. The predictor network may learn only a state that the agent has visited frequently and forget about previously visited states. Consequently, the prediction error will subsequently increase for these past states, and the agent may begin to follow a past policy.

Recently, some studies have used visibility graphs [29], Voronoi diagrams [30] and RL [31]–[34] in UAV path planning. Furthermore, some studies have applied RL to CUAV maneuvers [35]–[39]. However, the authors of those studies simply defined the state and action and conducted experiments in a dense reward environment. By contrast, we verify our algorithm based on the execution of CUAV missions in a sparse reward environment.

The main contributions of this paper are as follows:

- We show that SIL and RND are complementary and that combining these two algorithms is a highly efficient approach to exploration.
- We present several techniques for amplifying the imitation effect.
- We propose a novel method of constructing state vectors that represent coordinates and angles.
- The performance of the proposed RL approach when applied to the CUAV control problem is excellent. In addition to a single CUAV, we apply our algorithm to a flight formation. Our learning method results in reasonable CUAV maneuvers in a sparse reward environment.

III. PROBLEM DEFINITION

A. ENVIRONMENT

Fig 1 shows a visualization of the CUAV's mission environment. This virtual environment is an area of 100 × 200 × 30 km in height, width, and depth, respectively, and includes four air defense networks, each with a radius of 30 km. The goal is to train the CUAV to reach the target from the starting point within a limited time period while avoiding missiles. We conducted experiments in the following two environmental settings.

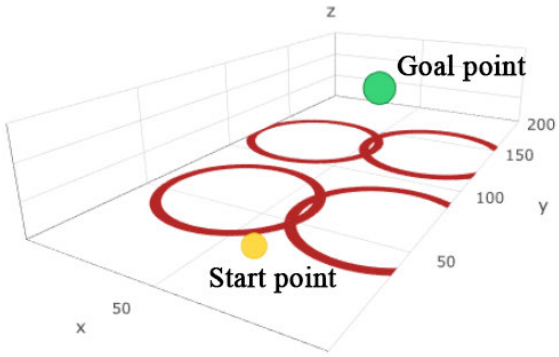

FIGURE 1. CUAV mission environment. The dimensions of the mission environment are 100, 200, and 30 km along the x, y, and z axes, respectively. A total of four air defense networks are deployed, two of which overlap each other.

1) SINGLE CUAV ENVIRONMENT

In this environment, the goal is to for a single CUAV to travel along the shortest path from the starting point to the target while avoiding enemy missiles.

2) FLIGHT FORMATION ENVIRONMENT

In the flight formation environment, we assume that the formation consists of four CUAVs. Similarly, the goal is for the squadron to travel from the starting point to the target point along the shortest distance for each CUAV. In this case, we utilize four networks and add the CUAV formation information to the state representation.

To model the dynamics of a CUAV, we apply the following equations of motion for a 3-degree-of-freedom (3-DOF) point mass model [40]:

$$\begin{aligned} \dot{x} &= V\cos\gamma\cos\psi, \\ \dot{y} &= V\cos\gamma\sin\psi, \\ \dot{z} &= V\sin\gamma, \\ \dot{V} &= \frac{T-D}{m} - g\sin\gamma, \\ \dot{\psi} &= \frac{gn\sin\phi}{V\cos\gamma}, \\ \dot{\gamma} &= \frac{g}{V(n\cos\phi - \cos\gamma),} \end{aligned} \tag{4}$$

where (x, y, z) denotes the position of the CUAV, V is its velocity, ψ is its heading angle and γ is its flight path angle. D denotes air resistance, m is the mass of the CUAV, and g is the gravitational acceleration. T, n and ϕ are the control inputs of the CUAV and denote the engine thrust, load factor and bank angle, respectively. We use these control inputs to implement the actions determined by our RL framework. Fig 2 shows the CUAV's bank angle, flight path angle, and heading angle. The engine thrust affects the velocity of the CUAV. The bank angle and load factor affect the heading angle and flight path angle.

For the missiles, we apply proportional navigation induction to enable them to chase the CUAV [41]. This algorithm works by separating the 3D space into three perpendicular planes: xy, xz and yz. If a missile can recognize the distance

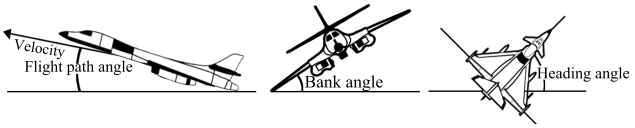

FIGURE 2. Bank angle, flight path angle and heading angle of a CUAV. These 3 DOFs control the maneuvering of the CUAV. In other words, the CUAV's action is determined by these three factors.

and relative speed between itself and its target, then it can be steered to accelerate and hit the target. We assume that when a CUAV enters the radius of a missile, the missile is always able to recognize the CUAV, and if the distance between the CUAV and the missile is less than 0.5 km, then the CUAV is unable to avoid the missile.

B. STATE

In general, in an environment such as an Atari game, the game image is preprocessed and used as the state, and a convolutional neural network (CNN) structure is employed. In this study, however, the CUAV's coordinate information and the CUAV's radar information for missile detection are vectorized to acquire the state for the CUAV control problem. Thus, a multilayer perceptron (MLP) is more appropriate for this problem than a CNN, which is generally adopted when the state of an arcade game is represented as an image.

1) COORDINATE REPRESENTATION

In a coordinate system, the coordinate points do not have a linear value relationship. For example, the two-dimensional (2D) coordinates (10, 10) are not ten times more important than the coordinates (1, 1). However, if the coordinates (10, 10) are used as the input to an MLP, the output value will be approximately 10 times larger than that produced when using the coordinates (1, 1). Therefore, placing coordinates into a state representation using real numbers is unreasonable and will cause learning instability. One way to represent the coordinates in the learning environment is to use a one-hot encoding vector. However, in one-hot encoding, the dimensionality of the vector increases as the coordinate range increases; moreover, this approach is usable only for integer coordinates. In this study, we introduce a new method of efficiently representing a spatial coordinate system.

In the proposed method, the coordinate for each axis is converted into a one-hot encoding vector, and the axis vectors are then concatenated. The basic one-hot encoding method requires 40,000 (200 $\times$ 200) rows to represent (1, 1) when x and y each range from 1 to 200; by contrast, in our method, the representation $c_{(1,1)} = [(1, 0, \cdots, 0)(1, 0, \cdots, 0)]'$ requires only 400 rows (200+200). Using this approach, we can dramatically reduce the dimensionality of the vector representing the coordinates.

We extend this method to a real-valued coordinate system as follows. Real-valued coordinates can be represented by introducing weights within the vector. For example, 1.3 can be regarded as a value consisting of approximately 70% of 1 and approximately 30% of 2; in other words, the number

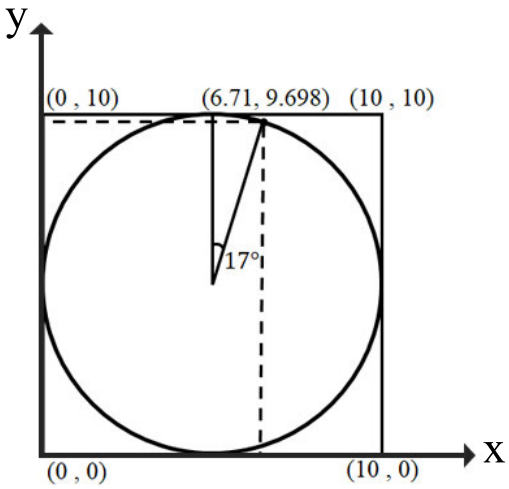

FIGURE 3. Example of angle representation. Suppose that there is a circle with a radius of 5 in a 2D space. The coordinates corresponding to a 17° rotation on the circle are (6.71, 9.698) in the Cartesian space. In the ECV representation, these coordinates are transformed into $c_{(17°)} = (0, \cdots, 0.71, 0.29, 0, \cdots, 0, 0.302, 0.698)'$ (20 rows).

1.3 is a number with a weight of 70% in 1 and 30% in 2. Thus, 1.3 can be represented as $c_{(1.3)} = (0.7, 0.3, \cdots, 0)'$ (200 rows). Moreover, the resulting vector can be reduced to smaller dimensions by reducing this coordinate to 1/10 of its original size. Accordingly, the number 1.3 can be represented as $c_{(1.3)} = (0.13, 0, \cdots, 0)'$ (20 rows). This method allows real-valued coordinates to be represented within limited dimensions. We call this method the efficient coordinate vector (ECV) representation.

2) ANGLE REPRESENTATION

Formulating a state representation of an angle for RL is also difficult because angles characteristically rotate through a complete period of 360°. For example, suppose that an angle changes from 10° to 350°. If we use either the real values or the ECV method, the agent will perceive the result as a 340° change. However, this difference (340°) is simultaneously equivalent to only 20°. Consequently, such an angle representation will tend to confuse the RL agent. We solve this problem by applying the ECV method to a polar coordinate system. First, the polar coordinates r and θ are transformed into the Cartesian coordinates x and y using the corresponding trigonometric functions. Then, a state representation of the transformed coordinates can be constructed using the ECV method. In other words, the angle is converted into a position on the upper part of a circle using the polar coordinate system, and then it is represented as a state through the ECV method. For example, as shown in Fig 3, the point on the circle corresponding to 17° can be represented as $c_{(17°)} = (0, \cdots, 0.71, 0.29, 0, \cdots, 0, 0.302, 0.698)'$ (20 rows) in the ECV representation.

3) FINAL STATE

Finally, we use the following information to represent the state for the CUAV control problem.

a: SINGLE CUAV ENVIRONMENT

- Flight path of the CUAV, consisting of the five most recent steps
- Path angle, heading angle and bank angle of the CUAV for the two most recent steps
- Velocity and load factor of the CUAV
- Distance between the CUAV and a missile
- Horizontal and vertical angles between the CUAV and a missile

b: FLIGHT FORMATION ENVIRONMENT

- States of the corresponding single CUAV environments
- Distance between the center of the formation and each of the CUAVs

C. ACTION

An action consists of a combination of changes to the input parameters used in equation 4, namely, the engine thrust, bank angle and load factor. For each input parameter, there are three possible changes: increase, hold, and decrease. In addition, we add an action that initializes all input parameters to their default values (a bank angle of 0°, a load factor of $1g$, and an engine thrust of 50 kN). This action allows the CUAV to cruise. The total number of actions is 28.

D. REWARD

The default reward is zero, except for the following specific situations:

c: SINGLE CUAV ENVIRONMENT

- As a result of a missile skirmish
- When the CUAV arrives at the target point
- Entering the cruise condition

d: FLIGHT FORMATION ENVIRONMENT

- Rewards for the corresponding single CUAV environments
- Distance between the center of the formation and each of the CUAVs

We reward the cruise condition because the CUAV cannot maintain its maximum speed during cruising. We impose a penalty of -0.01 if the speed reaches the maximum speed. In the formation flight environment, our goal is to maintain certain preferred distances among the CUAVs in the formation during flight. Thus, we penalize an agent when it is too far from the center of the formation or too close to another CUAV.

IV. AMPLIFICATION OF THE IMITATION EFFECT (AIE)

A. COMBINING SIL AND RND

In this section, we explain why combining SIL and RND amplifies the imitation effect and leads to deep exploration. In SIL, an update is performed only when the past R is greater than the current V_θ; then, the agent imitates the past decision. Intuitively, if we combine SIL and RND, then the value of $R - V_\theta$ will be larger than the corresponding value in SIL alone because of the exploration bonus. During the process of optimizing the actor-critic network to maximize $R_t = \Sigma_{k=t}^{\infty} \gamma^{k-t}(i_t + e_t)_k$, where i_t is the intrinsic reward and e_t is the extrinsic reward, an increase in the i_t value generated by the predictor network will cause R to increase. In other words, learning progresses through appropriate weighting of past good decisions. In this type of learning, the learning history is comprehensively considered. If the policy starts to converge as learning progresses, then i_t will be lower for more frequently visited states. One might think that learning would be slower because $R_t - V_\theta > R_{t+k} - V_\theta$, where $k > 0$ for the same state and i_t decreases. However, SIL exploits past good decisions, which leads to deep exploration. With the addition of the exploration bonus, the agent will be encouraged to further explore novel states. Consequently, a nonzero exploration bonus is likely to continue to be earned. In addition, when prioritized experience replay is used [15], the sampling probability is determined by $R - V_\theta$; thus, there is a high probability that a previous transition will be exploited in SIL even if i_t decreases. In other words, the two algorithms are complementary, and SIL is immune to the phenomenon in which a nonzero prediction error i_t no longer occurs.

B. INTRINSIC PENALTY

Adding an exploration bonus to a novel state that the agent visits is clearly an effective incentive encouraging exploration. However, when the policy and predictor networks converge, no exploration bonus will be earned for a novel state. In other words, the exploration bonus method provides a reward when the agent itself performs an unexpected action but not when the agent is induced to take the unexpected action. Therefore, an exploration method that entices the agent to take unexpected actions is necessary. We propose a method in which an intrinsic penalty is applied to an action that causes the agent to revisit a frequently visited state in addition to simply rewarding the agent when it makes an unexpected action. This intrinsic penalty drives the agent to escape from a converged local policy and helps it to experience more diverse policies. Specifically, we apply the penalty by transforming the current intrinsic reward into $\lambda log(i_t)$, where λ is a penalty weight parameter if the current intrinsic reward is less than a certain quantile α of the past N intrinsic rewards. This reward mechanism prevents the agent from remaining stuck in the same policy. In addition, adding a penalty to the intrinsic reward indirectly amplifies the imitation effect. Because $R_t - V_\theta$ decreases due to the penalty, the probability of the corresponding transition being sampled from the replay memory is smaller than that of a nonpenalized transition. Thus, the SIL updates are more likely to exploit nonpenalized transitions. Even if $R_t - V_\theta < 0$ due to a penalty, this does not affect SIL because it is not updated according to the SIL objective presented in equation 3. In other words, the intrinsic penalty allows the policy network to deviate from

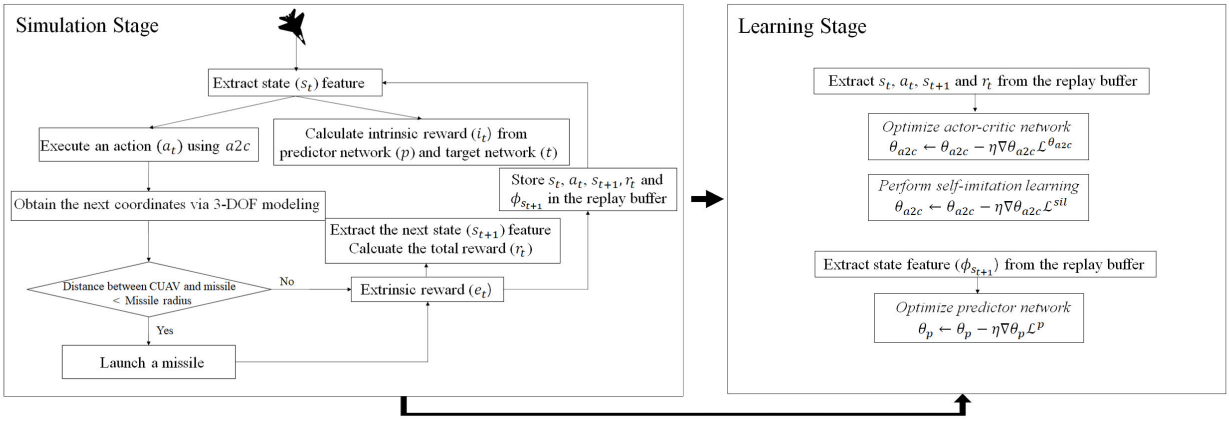

FIGURE 4. Procedural overview of the proposed algorithm in the CUAV environment. In the simulation stage, the CUAV maneuvers using the trained networks and stores the results of the episodes in the replay buffer. In the learning stage, the networks are further trained based on the collected data. The simulation stage and learning stage are repeated until a specified termination condition is met.

states that are repeatedly visited by the agent and indirectly amplifies the imitation effect for SIL.

C. CATASTROPHIC FORGETTING IN RND

The predictor network in RND mainly learns about the states that the agent has recently visited; thus, a catastrophic forgetting process can occur that is similar to the phenomenon in continual task learning in which knowledge learned from previous tasks is forgotten. If the prediction error increases for a state that the agent has visited before, the agent might assume that the previous state is a novel state, which would prevent the agent from exploring effectively. The method of mitigating this phenomenon is simple but effective. We store the output of the target network and the corresponding state feature in the memory of the predictor network, similar to using a replay memory, so that the predictor network will not forget past state features, and we train the predictor network in batch mode. The use of such a predictor memory reduces the prediction error for states that the agent has previously visited, making the agent more likely to explore novel states. Even if the agent returns to a past policy, the prediction error of the state visited by the policy will be low, an intrinsic penalty will be applied to the state, and the probability of escaping from that state will be high.

D. PROPOSED ALGORITHMS FOR THE CUAV ENVIRONMENT

Fig 4 describes the interaction between the simulation and learning stages in the CUAV environment. In the simulation stage, the CUAV state composed of the coordinates and the relationship information between the CUAV and a missile is fed into the A2C network. In addition, the output difference between the predictor network and the target network for the current state feature is calculated to obtain the intrinsic reward. The next position of the CUAV is determined from the actor-critic output. If the distance between the CUAV and a missile is within the missile radius, the missile is launched, and the extrinsic reward and the next state of the CUAV are derived. Once the next coordinates of the CUAV are known, the step ends, and the state, action, reward, next state and state feature are stored in replay buffers. An episode ends when the CUAV reaches the target point from the starting point, leaves the mission environment, is shot down by a missile, or does not reach the target point within 600 seconds. This state transition process continues until the end of the current episode, and multiple episodes are conducted. Once the simulation stage ends, the learning stage begins. In this stage, the A2C network is optimized, and SIL is performed by randomly sampling an instance (consisting of a state, an action, the reward and the next state) from the replay buffer. Then, the RND predictor network is optimized using the state features stored in the replay buffer. After the learning stage ends, the simulation stage starts again. Each stage is performed repeatedly. In the current study, a total of 60,000 episodes were executed in the single CUAV environment, and 80,000 episodes were performed in the flight formation environment. The notations used in Algorithm 1 are defined in Table 1.

In Algorithm 1, we provide the pseudocode corresponding to this interactive procedure. Three RL algorithm variants are proposed for use in the learning stage. The first proposed AIE algorithm (AIE1) combines A2C-based SIL (ASIL) and RND, as described by Algorithm 1 without lines 15-18 and lines 33-38. The second (AIE2) is equivalent to AIE1 with the addition of the intrinsic penalty, which is described in lines 15-18. The third (AIE3) is equivalent to AIE2 with the further addition of the replay memory for the predictor network, as described in lines 36-38.

V. EXPERIMENT

A. TEST ALGORITHMS

We chose DQN, DQN with prioritized experience replay (PerDQN), A2C and ASIL as the algorithms for comparison, where ASIL denotes the combination of the A2C framework

Algorithm 1 Amplification of the Imitation Effect (AIE)

Initialize A2C network parameters θ_{a2c}
Initialize predictor network parameters θ_p
Initialize and fix target network parameters θ_t
Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
Initialize episode buffer $\mathcal{E} \leftarrow \emptyset$
Initialize feature buffer $\mathcal{F} \leftarrow \emptyset$
Input State
Output Action
procedure AIE
  for episode = 1, M **do**
    \\ Simulation stage.
    for each step **do**
      Execute an action $s_t, a_t, e_t, s_{t+1} \approx \pi_\theta(a_t|s_t)$
      Extract the state feature of s_{t+1} to obtaion $\phi_{s_{t+1}}$
      Calculate the intrinsic reward i_t
      if $i_t <$ penalty condition threshold **then**
        $i_t \leftarrow \lambda log(i_t)$
      end if
      $r_t = r_t + e_t + i_t$
      Store transition $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, s_{t+1}, a_t, r_t)\}$
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\phi_{s_{t+1}}, \theta_t(\phi_{s_{t+1}}))\}$
    end for
    if s_{t+1} is terminal **then**
      Compute returns $R_t = \Sigma_k^\infty \gamma^{k-t} r_k$ in $\mathcal{E}$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t)\}$
      Clear episode buffer $\mathcal{E} \leftarrow \emptyset$
    end if

    \\Learning stage.
    $\theta_{a2c} \leftarrow \theta_{a2c} - \eta \nabla_{\theta_{a2c}} \mathcal{L}^{a2c}$
      ▷ Optimize actor-critic network
    for k= 1, M **do**
      Sample a minibatch $\{(s, a, R)\}$ from $\mathcal{D}$
      $\theta_{a2c} \leftarrow \theta_{a2c} - \eta \nabla_{\theta_{a2c}} \mathcal{L}^{sil}$
        ▷ Perform SIL
      Sample a minibatch $\{(\phi_{s_{t+1}}, \theta_t(\phi_{s_{t+1}}))\}$ from $\mathcal{F}$
      $\theta_p \leftarrow \theta_p - \eta \nabla_{\theta_p} \mathcal{L}^p$
        ▷ Optimize predictor network
    end for
  end for
end procedure

and SIL. In addition, we tested the three AIE algorithms introduced in the previous section.

B. HARD EXPLORATION IN A 2D ENVIRONMENT

1) SPARSE REWARD SETTING

We conducted a simple experiment to determine the effectiveness of the proposed algorithms in promoting exploration. We constructed a 2D grid environment in which the goal is for the agent to learn a sequence of simple stepwise movements (up, down, left, or right) beginning from a starting point and ending at a goal point. The reward was set to zero except when the agent reached the target point (a reward of 30) or left the environment (a reward of -30). RL was performed for a total of 10,000 episodes for each algorithm. Fig 5 shows a visualization of the agent's movement paths. Because the reward was too sparse, the agent implementing ASIL failed to reach the target point. In contrast, all of the proposed algorithms successfully reached the target point because of the exploration bonus. For AIE1, the results show that the agent quickly reached the target point. However, AIE2 and AIE3 (which include the intrinsic penalty) conducted deeper explorations than AIE1, as seen from the fact that these two algorithms arrived at the target point via more diverse paths.

TABLE 1. Parameter notations.

Notation	Definition
s	state
a	action
r	reward
e	extrinsic reward
i	intrinsic reward
θ	network parameters
$a2c$	actor-critic network
p	predictor network
t	target network
sil	self-imitation learning
$\mathcal{D}$	replay buffer used to perform SIL
$\mathcal{E}$	episode buffer used to train $a2c$
$\mathcal{F}$	feature buffer used to train the predictor network
p	predictor network

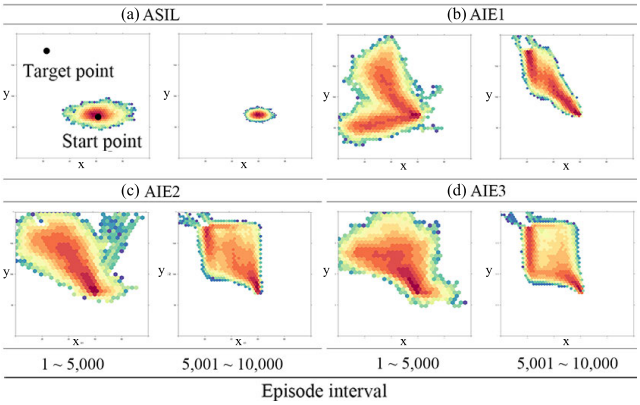

FIGURE 5. Path visualization for each algorithm in a 2D grid environment (x-axis: latitude, y-axis: longitude). The color changes from blue to red as the agent visits a state more frequently. (a) ASIL fails to reach the target point. (b) The AIE1 results show that the agent reaches the target point. (c), (d) AIE2 and AIE3 explore larger areas than AIE1.

2) NO-REWARD SETTING

We performed an additional experiment in the same environment but with no target point. Hence, the agent conducted only exploration in each episode. We argue that catastrophic

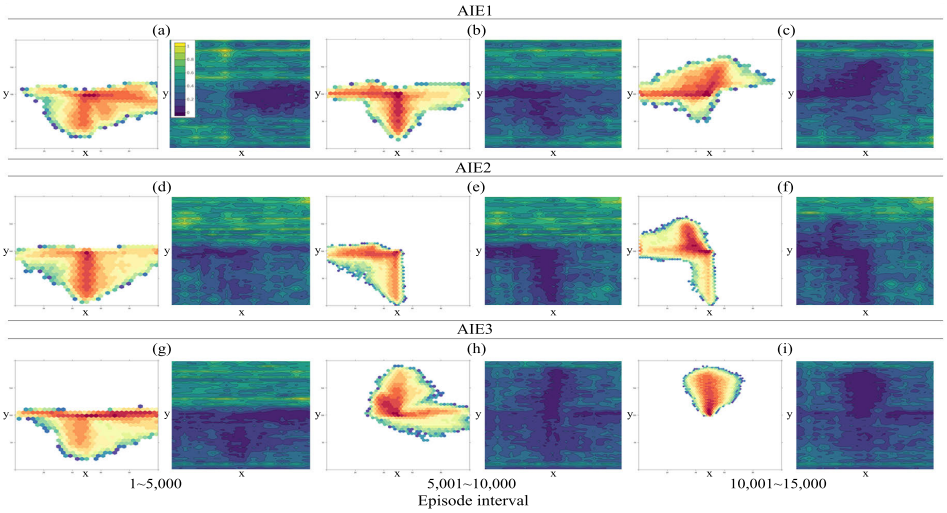

FIGURE 6. Visualization of the paths of the agent and the losses of all coordinate states for each algorithm in the no-reward 2D grid environment (x-axis: latitude, y-axis: longitude). In the plots showing the agent's paths, the color gradient from blue to red indicates areas that the agent visits more frequently. In the loss plots, the color gradient from blue to yellow indicates areas where the loss is larger. Panels (a) - (c) show the exploration of the AIE1 agent. (a) The dark blue region on the right side of the loss plot corresponds to the region that the agent has recently explored very frequently. (b) As the agent further explores the left region, the dark blue loss region on the right fades. (c) As the agent explores the upper left, the dark blue loss regions on the lower left and right fade. Panels (d) - (f) show the exploration of the AIE2 agent. (d) The agent intensively explores the lower left region. (e) As the agent explores the lower left region, the dark blue loss region on the right side fades. (f) As the agent explores the upper left region, the dark blue loss region in the lower left does not lighten, but the right-side loss is considerably lessened. Panels (g) - (i) show the exploration of the AIE3 agent. (g) The agent explores mainly the lower right region. (h) The agent mainly explores the upper right region, but the color of the loss plot in the unexplored lower region remains blue. (i) The agent explores only the upper region, but the color of other areas of the loss plot remains blue. These results illustrate that AIE3 allows the agent to remember more information about areas explored farther in the past than AIE1 and AIE2 do.

forgetting hinders effective exploration when an exploration bonus is implemented because the agent has less chance of searching for a novel state if the prediction error remains high for previously searched states. Furthermore, we argue that using a replay memory for the predictor network (AIE3) enables more efficient exploration because the memory mitigates catastrophic forgetting.

Fig 6 shows visualizations of the movement paths of the agent in 5,000 episodes (left-hand plots) and the losses of the predictor network at all coordinates (right-hand plots). In the right-hand plots, the color gradient from blue to yellow indicates where the loss is larger. We observe that the losses in the regions explored by the agent are lower than the losses in other regions. As the number of elapsed episodes increases, the agent tends to explore novel spaces with higher prediction errors. At this time, the losses in the regions that the agent has explored in recent episodes are increased compared to the losses in the regions explored during previous episodes. For instance, in the first loss figure for AIE1, it is apparent that the agent has visited the blue region on the right more frequently, while in the second figure, the agent has recently visited the blue region on the left more frequently than the right region. In addition, the blue color on the right is paler than that in the first plot. This is because the agent has most recently visited the left region more frequently and has begun to forget about the right region.

By contrast, with AIE3, the losses in the previously explored regions remain relatively low. From these plots, we can confirm that AIE3 is less susceptible to catastrophic forgetting than AIE1 and AIE2. In the sparse reward environment, the agents implementing DQN, PerDQN, and A2C remains almost stationary, and the ASIL agent explored only a small area, circulating throughout this area repeatedly with an increasing number of episodes; by contrast, the three proposed algorithms explored many different locations. Table 2 presents scores quantifying the extent to which

TABLE 2. The exploration area score for each algorithm in the 2D no-reward grid environment, representing the average area explored by the agent in 30 repeated experiments.

Algorithm	Exploration area
DQN	5.2±1.04
PerDQN	5.4±1.21
A2C	6.5±1.47
ASIL	11.2±1.25
AIE1	40.5±2.06
AIE2	43.2±2.36
AIE3	46.7±2.19

each algorithm uniformly explored the four quadrants of the 2D grid space throughout 30,000 episodes. The formula used to calculate the exploration area score is

$$score = mean(EQ_q) \times \sigma_{EQ} \times 100, \tag{5}$$

where EQ_q is the explored portion of the total area of each quadrant. From the results, we can confirm that the proposed algorithms (particularly AIE3) are very effective in promoting exploration.

C. EXPERIMENTS ON CUAV MISSION EXECUTION

1) SINGLE CUAV ENVIRONMENT

We performed an experiment to investigate CUAV control in a sparse reward environment and compare the performances of the algorithms. In addition, we analyzed how a CUAV can maneuver to avoid missiles. Fig 7 (a) shows the performance curves of ASIL and the three proposed algorithms in an experiment consisting of 60,000 episodes. The curves displayed in light colors and normal colors represent the worst and average performances, respectively, of the compared algorithms. Table 3 shows the performance scores for each algorithm

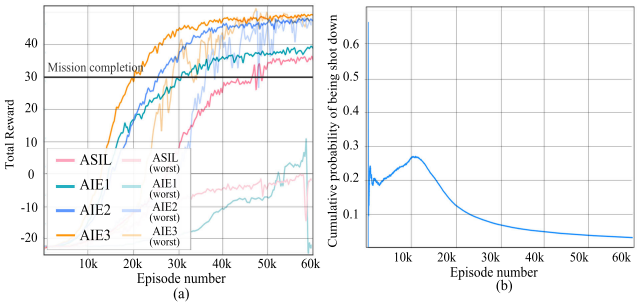

FIGURE 7. (a) Learning curves in the CUAV mission execution environment. The x- and y-axes represent the episode number and the average reward, respectively. The plot shows the average reward over 10 experiments for each algorithm. The lighter-colored curves represent the worst performance results for each algorithm. (b) Cumulative probability of being shot down by a missile. In the early learning stage, as the CUAV moved forward, the probability increased. However, as learning progressed, the CUAV learned a control policy that could effectively avoid missiles, and as a result, the probability of being shot down decreased.

TABLE 3. Performance scores for each algorithm in the CUAV mission environment. We conducted ten experimental trials with different simulation seeds for each algorithm.

Algorithm	Episodes to Mission Completion	Rate of Convergence
DQN	No convergence	0%
PerDQN	No convergence	0%
A2C	No convergence	0%
ASIL	45,000	50%
AIE1	30,000	60%
AIE2	26,000	100%
AIE3	20,000	100%

in the CUAV mission environment. The first column lists the algorithms used in the experiment. The second column reports the number of episodes elapsed to complete a mission in the CUAV environment, and the third column reports the percentage of the trial that each algorithm converged in 10 repeated experiments.

First, since our experimental environment had a sparse reward structure, DQN, PerDQN, and A2C failed to converge to a policy that could generate the shortest path from the origin to the target point while avoiding enemy missiles. The results show that AIE2 and AIE3 succeeded in converging to the desired policy, while ASIL and AIE1 fell into a local minimum in one in two trials and one in three trials, respectively. In particular, AIE3 outperformed the other algorithms in terms of convergence speed and stability, as shown in Fig 7. Similar to the previous exploration experiment, we confirmed that the three proposed algorithms showed better performance than ASIL (the baseline model) in the CUAV control environment. Fig 8 presents snapshots of the learning process (an animation is available here[1]). During early learning episodes, the CUAV performed random actions and occasionally left the battlefield. As the number of episodes increased, it tended to move forward gradually but eventually be shot down by a missile. This result is confirmed by the plot of the cumulative probability of the CUAV being shot down presented in Fig 7 (b). As the learning process continued, the CUAV learned how to avoid the missiles and began to move to new coordinates (attempting to increase the intrinsic reward). The CUAV attempted to reach the target via various paths. After training was complete, in 1000 test trials, the CUAV could reach its target with a 95% probability. Fig 9 shows a 3D representation of the path taken by a single CUAV to reach the target while avoiding missiles. When the CUAV passes through the center of an air defense network, its probability of being shot down by a missile increases. Therefore, the CUAV learned a safe path that passes through the overlapping areas of the air defense networks at a low altitude.

2) FLIGHT FORMATION ENVIRONMENT

In the flight formation environment, we utilized four actor networks (analogous to four single CUAV environments) and additionally considered the state of the formation and the reward for the formation. We adopted the AIE3 algorithm, which performed best in the single CUAV environment. The goal was for the squadron to move from the starting point to the target point while maintaining a certain distance between the CUAVs. We trained a total of 10 networks, including the actor and critic networks for each CUAV and the target and predictor networks for RND.

Fig 10 presents snapshots of the learning process, and Fig 9 (b) presents a 3D representation of the paths taken by the CUAVs in the formation. Similar to the behavior in the single CUAV environment, each CUAV in the squadron initially appeared to fly randomly and then gradually began

[1] https://youtu.be/7R5lZAsCs2c

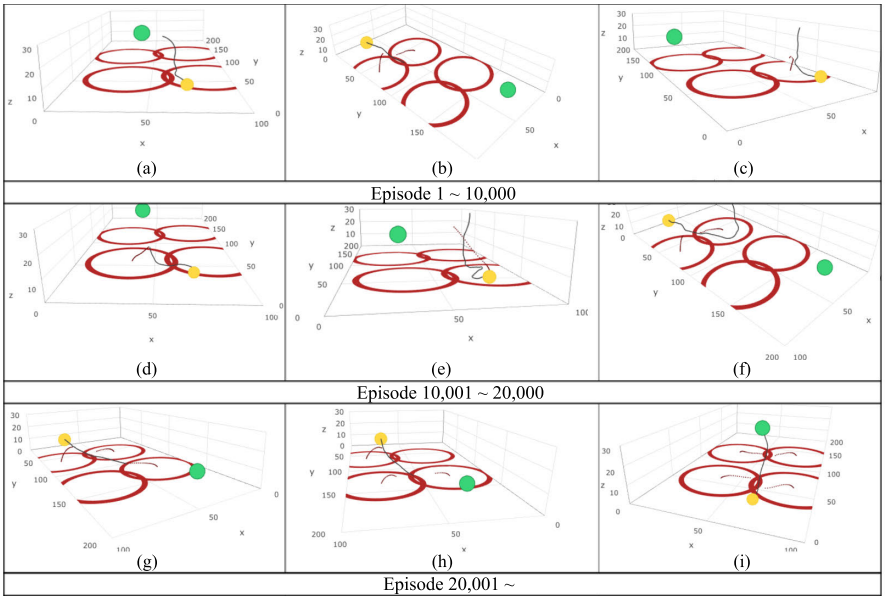

FIGURE 8. 3D visualization of the learning process of a single CUAV (x-axis: latitude, y-axis: longitude, z-axis: altitude). Each red circle represents an air defense network, the black solid lines represent the movement paths of the CUAV, and a red dotted line represents a missile's movement path. From panels (a) - (c), it can be seen that early in the learning process, the CUAV performed almost random maneuvers. Panels (d) - (f) show the CUAV gradually moving towards its destination and learning maneuvers for evading missiles. In later episodes, as shown in panels (g) - (i), the CUAV had learned an effective control policy for finding the best path to the destination through the air defense networks while avoiding missiles.

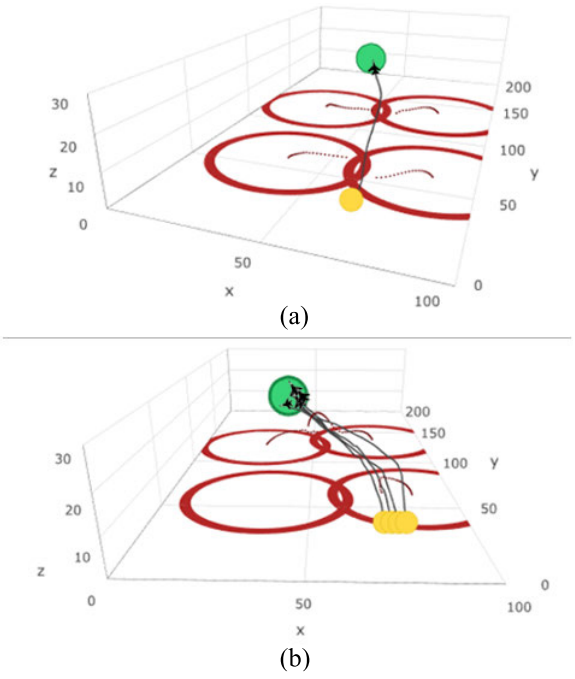

FIGURE 9. 3D views of CUAV paths after learning (x-axis: latitude, y-axis: longitude, z-axis: altitude): (a) a single CUAV and (b) a formation of CUAVs. The CUAV agents take paths that pass through the overlapping areas of the air defense networks to minimize the risk of being shot down by missiles and the flight distance to the target point.

to move towards the target point. Fig 11 shows the overall average distance between CUAVs by episode. This plot confirms that the distance between the CUAVs decreased as the learning process progressed. In this environment, we trained the networks for 60,000 episodes over three weeks. In addition to the flight paths of each CUAV, the information on the relationships between the CUAVs in the formation needed to be learned; therefore, the learning time had to be increased. Fig 12 shows the mission completions rate for different flight formations in the test experiment. The third bar indicates that there was a 99% chance that three or more of the CUAVs in the squadron would be successful. Furthermore, the probability of all four CUAVs reaching the target from the starting point was approximately 70%.

3) TEST ENVIRONMENT

We conducted test experiments using the pretrained models in a more difficult environment than the learning environment. Table 4 shows the experimental results for various scenarios in the single CUAV environment and the flight formation environment. Scenario 1 refers to an environment in which the missile performance is better than that in the learning environment. Scenario 2 refers to an environment in which the locations of the air defense networks have been randomly adjusted relative to the learning environment in addition to improved missile performance. In the flight formation environment, conditions 1, 2, and 3 refer to 1, 2, and 3 of the 4 CUAVs reaching the target point, respectively, and condition 4 means that all CUAVs reach the target point. For both the single CUAV environment and the flight formation environment, a slight decrease in performance is evident in the test environment compared with the learning environment.

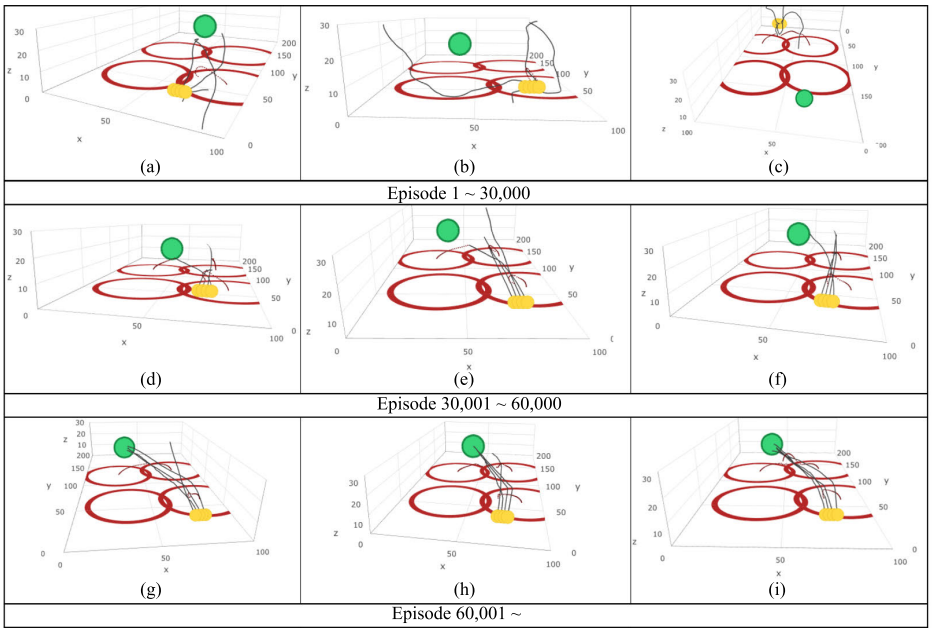

FIGURE 10. 3D visualization of the CUAV learning process in the formation environment (x-axis: latitude, y-axis: longitude, z-axis: altitude). The red circles represent the air defense networks, the black solid lines represent the movement paths of the CUAVs, and a red dotted line represents a missile's movement path. From panels (a) - (c), it can be seen that the CUAVs in the formation performed random maneuvers early in the learning process, but by the episodes depicted in panels (d) - (f), the CUAVs had been trained to find ways to move towards their destination. Finally, as shown in panels (g) - (i), they learned the best control policy to reach the destination while minimizing the missile risk.

TABLE 4. Experimental results for various scenarios in the single CUAV environment and the flight formation environment. In Scenario 1, the proportional constant for the launching of a missile is adjusted from 3 to 5, corresponding to improved shooting performance of the missiles compared with the learning conditions. In Scenario 2, the missile performance is improved, and the locations of the air defense networks are also changed. Specifically, we randomly adjusted the locations of the air defense networks by 3 km. We conducted 1,000 tests for each scenario. Conditions 1, 2, and 3 are fulfilled when 1, 2, and 3 of the 4 CUAVs reach the target point, respectively, and similarly, condition 4 means that all CUAVs reach the target point.

Single CUAV environment		Flight formation environment							
		Scenario 1				Scenario 2			
Scenario 1	Scenario 2	condition1	condition2	condition3	condition 4	condition1	condition2	condition3	condition4
95.5%	91.5%	99.7%	98.5%	96.5%	84.5%	96.5%	98.0%	94.3%	69.8%

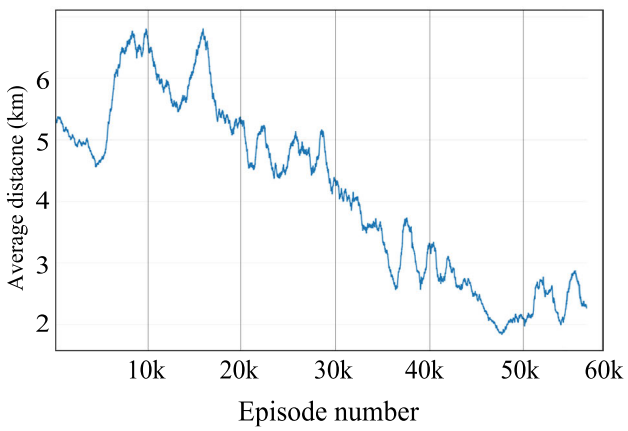

FIGURE 11. Plot of the overall average distance between CUAVs by episode. The x- and y-axes represent the episode number and the average distance, respectively. As the number of learning episodes increased, the average distance between the CUAVs decreased.

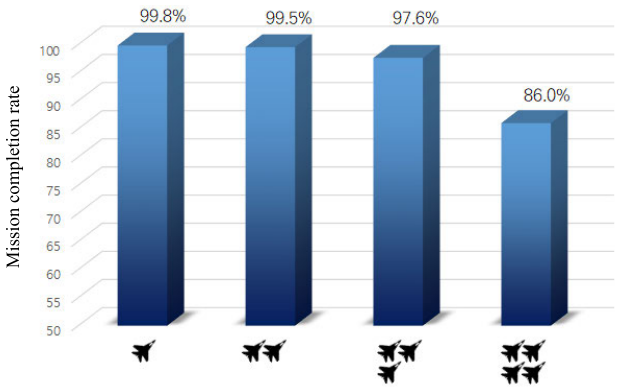

FIGURE 12. Plot of the mission completion rates for the flight formation during testing. The first bar indicates a 99% probability that one or more of the CUAVs in the squadron will succeed in the mission.

In particular, when the locations of the air defense networks have changed, there is a high probability that a CUAV will not reach its target point. The reason is that in the learning environment, the CUAVs are effectively seeking routes to avoid rather than learning evasive maneuvers against missiles. This phenomenon is particularly apparent in the flight formation environment. Accordingly, Scenario 2 shows significantly lower mission completion rates than Scenario 1.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an RL algorithm for guiding a CUAV to achieve multiple goals through actions in a manner similar to human behavior. We defined the mission goals as finding the shortest path for the CUAV while avoiding threats and using the thrust, bank angle and load factor, which a human pilot uses to control a fighter, to execute actions. In addition, we conducted RL experiments for missions involving both a single CUAV and a squadron of four CUAVs. We set three goals in the flight formation environment: missile avoidance, finding the shortest path to the target point, and adjusting the distance between the CUAVs in the formation. To improve the efficiency of RL, we developed a novel method of representing the coordinates and angle of a CUAV. Furthermore, we proposed several AIE algorithm variants by combining SIL and RND for deep exploration. In the AIE2 algorithm, an intrinsic penalty is imposed on states that the agent has frequently visited, which prevents the agent from falling into a local optimal policy. The AIE3 algorithm additionally adopts a replay memory to mitigate catastrophic forgetting by the predictor network. These two algorithms amplify the imitation effect, leading to deep exploration and thereby enabling the policy network to quickly converge to the desired policy. In a 2D grid environment, we experimentally demonstrated that the proposed AIE algorithms could successfully explore wide areas of the grid space. In particular, AIE3 explored an area that was more than 4 times larger than the area explored by ASIL in the 2D grid environment. In addition, for the CUAV control problem, we observed that the proposed algorithms quickly converged to the desired policy both for a single CUAV and in a squadron environment. AIE3 showed more than twice the learning convergence speed and more than twice the convergence success rate of ASIL. In both experimental environments, it was confirmed that the proposed AIE approach is superior to existing algorithms in terms of both convergence speed and learning stability.

In future work, more efficient ways to train a squadron should be addressed. In this study, we trained a total of 10 networks for the flight formation environment, but the learning process required a total of three weeks. Thus, it will be necessary in future work to consider using transfer learning based on a pretrained network trained for a single CUAV to achieve more efficient learning. Furthermore, RL does not always guarantee an optimal policy because it learns only from the experience of the agent. In our experiments, when the air defense networks were tightly overlapped, the CUAV learned to fly closer to the ground. If the CUAV could be trained based on human guidelines, it would be possible for it to learn an optimal policy as desired by humans.

REFERENCES

[1] M. Franklin, "Unmanned combat air vehicles: Opportunities for the guided weapons industry," Occasional Papers, Roy. United Services Inst. (RUSI), London, U.K., Sep. 2008.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[3] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1471–1479.

[4] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," 2017, *arXiv:1703.01310*. [Online]. Available: http://arxiv.org/abs/1703.01310

[5] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," 2017, *arXiv:1702.08165*. [Online]. Available: http://arxiv.org/abs/1702.08165

[6] B. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2010.

[7] P. J. Silvia, "Curiosity and motivation," in *The Oxford Handbook of Human Motivation*, R. M. Ryan, Ed. Oxford, U.K.: Oxford Univ. Press, 2012, pp. 157–167.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 16–17.

[9] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," 2018, *arXiv:1808.04355*. [Online]. Available: http://arxiv.org/abs/1808.04355

[10] N. Haber, D. Mrowca, L. Fei-Fei, and D. L. K. Yamins, "Learning to play with intrinsically-motivated self-aware agents," 2018, *arXiv:1802.07442*. [Online]. Available: http://arxiv.org/abs/1802.07442

[11] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," 2018, *arXiv:1806.05635*. [Online]. Available: http://arxiv.org/abs/1806.05635

[12] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018, *arXiv:1810.12894*. [Online]. Available: http://arxiv.org/abs/1810.12894

[13] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, 1992.

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

[15] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952

[16] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," 2016, *arXiv:1611.01224*. [Online]. Available: http://arxiv.org/abs/1611.01224

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

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

[19] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[20] A. L. Strehl and M. L. Littman, "An analysis of model-based interval estimation for Markov decision processes," *J. Comput. Syst. Sci.*, vol. 74, no. 8, pp. 1309–1331, 2008.

[21] L. Fox, L. Choshen, and Y. Loewenstein, "Dora the explorer: Directed outreaching reinforcement action-selection," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[22] M. C. Machado, M. G. Bellemare, and M. Bowling, "Count-based exploration with the successor representation," 2018, *arXiv:1807.11622*. [Online]. Available: http://arxiv.org/abs/1807.11622

[23] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," 2015, *arXiv:1507.00814*. [Online]. Available: http://arxiv.org/abs/1507.00814

[24] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," 2017, *arXiv:1706.01905*. [Online]. Available: http://arxiv.org/abs/1706.01905

[25] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," 2017, *arXiv:1703.05407*. [Online]. Available: http://arxiv.org/abs/1703.05407
[26] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," 2018, *arXiv:1802.06070*. [Online]. Available: http://arxiv.org/abs/1802.06070
[27] T. Gangwani, Q. Liu, and J. Peng, "Learning self-imitating diverse policies," 2018, *arXiv:1805.10309*. [Online]. Available: http://arxiv.org/abs/1805.10309
[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[29] A. Majeed and S. Lee, "A fast global flight path planning algorithm based on space circumscription and sparse visibility graph for unmanned aerial vehicle," *Electronics*, vol. 7, no. 12, p. 375, 2018.
[30] H. Tong, W. W. Chao, H. C. Qiang, and X. Y. Bo, "Path planning of UAV based on Voronoi diagram and DPSO," *Procedia Eng.*, vol. 29, pp. 4198–4203, Jan. 2012.
[31] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen, "Reinforcement learning for autonomous UAV navigation using function approximation," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Aug. 2018, pp. 1–6.
[32] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 1–21, 2019.
[33] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of uav by using real-time model-based reinforcement learning," in *Proc. 14th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, 2016, pp. 1–6.
[34] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.
[35] P. Liu and Y. Ma, "A deep reinforcement learning based intelligent decision method for UCAV air combat," in *Proc. Asian Simulation Conf.* Singapore: Springer, Aug. 2017, pp. 274–286.
[36] Y. Zhang, W. Zu, Y. Gao, and H. Chang, "Research on autonomous maneuvering decision of UCAV based on deep reinforcement learning," in *Proc. CCDC*, 2018, pp. 230–235.
[37] C. Minglang, D. Haiwen, W. Zhenglei, and S. QingPeng, "Maneuvering decision in short range air combat for unmanned combat aerial vehicles," in *Proc. Chin. Control and Decis. Conf. (CCDC)*, 2018, pp. 1783–1788.
[38] Q. Yang, J. Zhang, G. Shi, J. Hu, and Y. Wu, "Maneuver decision of UAV in short-range air combat based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 363–378, 2019.
[39] H. Zhang and C. Huang, "Maneuver decision-making of deep learning for UCAV thorough azimuth angles," *IEEE Access*, vol. 8, pp. 12976–12987, 2020.
[40] S. Kim and Y. Kim, "Three dimensional optimum controller for multiple UAV formation flight using behavior-based decentralized approach," in *Proc. Control, Autom. Syst., Int. Conf. (ICCAS)*, 2007, pp. 1387–1392.
[41] I. Moran and T. Altilar, "Three plane approach for 3D true proportional navigation," in *Proc. AIAA Guid., Navigat., Control Conf. Exhibit*, 2005, p. 6457.

GYEONG TAEK LEE received the B.S. degree in statistics from Sungkyunkwan University, South Korea, in 2016. He is currently pursuing the Ph.D. degree in industrial engineering with Yonsei University. His current research interests include machine learning and reinforcement learning for manufacturing and unmanned aerial vehicle combat systems.

CHANG OUK KIM received the Ph.D. degree in industrial engineering from Purdue University, West Lafayette, IN, USA, in 1996. He is currently a Professor with the Department of Industrial Engineering, Yonsei University, South Korea. He has published more than 100 papers in journals and conference proceedings. His current research interests include data science for manufacturing and defense analysis.