drones

MDPI

Article

Modular Reinforcement Learning for Autonomous UAV Flight Control

Jongkwan Choi [1], Hyeon Min Kim [1], Ha Jun Hwang [1], Yong-Duk Kim [2] and Chang Ouk Kim [1,*]

[1] Department of Industrial Engineering, Yonsei University, Seoul 03722, Republic of Korea; jk-choi@yonsei.ac.kr (J.C.); rlagusals50@yonsei.ac.kr (H.M.K.); hajuny@yonsei.ac.kr (H.J.H.)
[2] Defense Artificial Intelligence Technology Center, Agency for Defense Development, Daejeon 34186, Republic of Korea; ydkim.4653@gmail.com
* Correspondence: kimco@yonsei.ac.kr

Abstract: Recently, research on unmanned aerial vehicles (UAVs) has increased significantly. UAVs do not require pilots for operation, and UAVs must possess autonomous flight capabilities to ensure that they can be controlled without a human pilot on the ground. Previous studies have mainly focused on rule-based methods, which require specialized personnel to create rules. Reinforcement learning has been applied to research on UAV autonomous flight; however, it does not include six-degree-of-freedom (6-DOF) environments and lacks realistic application, resulting in difficulties in performing complex tasks. This study proposes a method of efficient learning by connecting two different maneuvering methods using modular learning for autonomous UAV flights. The proposed method divides complex tasks into simpler tasks, learns them individually, and then connects them in order to achieve faster learning by transferring information from one module to another. Additionally, the curriculum learning concept was applied, and the difficulty level of individual tasks was gradually increased, which strengthened the learning stability. In conclusion, modular learning and curriculum learning methods were used to demonstrate that UAVs can effectively perform complex tasks in a realistic, 6-DOF environment.

Keywords: UAV; autonomous flight control; reinforcement learning; modular learning; curriculum learning; JSBSim

Citation: Choi, J.; Kim, H.M.; Hwang, H.J.; Kim, Y.-D.; Kim, C.O. Modular Reinforcement Learning for Autonomous UAV Flight Control. *Drones* **2023**, *7*, 418. https://doi.org/10.3390/drones7070418

Academic Editors: Zhaokui Wang, Xiumin Diao and Lin Zhang

Received: 6 June 2023
Accepted: 21 June 2023
Published: 23 June 2023

1. Introduction

In recent years, research on unmanned aerial vehicles (UAVs) has increased significantly in various fields, such as aviation, smart agriculture, and medical supply transportation [1,2]. To best utilize the benefits of UAVs, which do not require pilots, it is necessary to study autonomous UAVs that can maneuver themselves to accomplish a desired mission by completely replacing the pilot and without ground operators.

Previous autonomous UAV flight studies used rule-based methods, wherein experts determined the UAV behavior for each situation. However, rule-based autonomous flights require significant expert involvement for defining the rules, and detailed rules are needed to define complex tasks. Additionally, there is a limitation that the UAV cannot maneuver normally in situations beyond the established rules [3–5]. Therefore, recent autonomous UAV flight research has used reinforcement learning based on artificial neural networks for robust flights in external environments [6–8]. In the case of autonomous UAV flight using reinforcement learning, real aerodynamics based on simulated environments can be implemented, and the UAV can fly based on decision-making processes similar to those made by a pilot. A pilot controls an aircraft by responding to the environment in real time to overcome any unexpected events. Reinforcement learning is used to control a UAV in real time using collected data in a similar method and, hence, the response to environmental changes is flexible. However, autonomous UAV flights using reinforcement learning are complex for the following reasons.

- Six-degrees-of-freedom (6-DOF) environment: For a UAV to perform real-world tasks learned through reinforcement learning, a 6-DOF environment, including the 3D position and orientation, must be considered. However, if the state and action values are both continuous and high-dimensional features, it makes it difficult for a reinforcement learning agent in a 6-DOF environment to learn the features of the environment [9]. Thus, existing reinforcement-learning-based autonomous UAV flight research is limited to conducting experiments after simplifying the action and state spaces in the simulation environment [10–15].
- Difficulty in setting the reward: It is difficult to develop reward settings that are suitable for research purposes in the field of reinforcement learning; this difficulty increases as the agent task and environment become increasingly complex. Many researchers have explored reward engineering, which is a process of setting a reward such that the agent can act according to the researcher's intentions [16]. UAVs need to perform tasks by using not only absolute location information, such as the target coordinates and velocity, but also the relative target point location information. This increases the amount of data that is considered when creating a suitable reward for the task, and the number and type of the location data that are used in each task differ. Therefore, setting a suitable autonomous UAV flight reward is challenging.
- Difficulty performing complex tasks: As a UAV task becomes more difficult, the number of parameters required in the artificial neural network increases. This results in a greater amount of computational resources becoming necessary. Additionally, when performing different tasks sequentially, an artificial neural network receives state and action values with distributions different from those in previous tasks; consequently, model training becomes unstable or the task performance decreases. For this reason, previous autonomous UAV flight research has been limited to performing only a single task [17–22]. Thus, research on connecting tasks is necessary for situations in which two tasks must be performed continuously or when complex tasks must be performed.

The contributions of this research are as follows. JSBSim is an open-source 6-DOF dynamic flight model that implements all aircraft characteristics, such as the flight control system and aerodynamics [23]. JSBSim is used as the reinforcement learning environment, where the agent learns flight forms similar to those of an actual aircraft. Additionally, a modular learning method, wherein the UAV can efficiently perform complex tasks, is proposed. Modular learning comprises individual networks for each characteristic task type. Curriculum learning, in which the difficulty of tasks is gradually increased depending on the corresponding method, is applied to successfully connect different maneuvering tasks. Compared to using a single network, this method supports stable and complex maneuvering and learning. Moreover, the experimental results show that the modular learning method is efficient for successfully conducting two continuous tasks, with a small number of learning episodes.

The remainder of this paper is arranged as follows. In Section 2, the soft actor-critic (SAC) algorithm used in this research and reinforcement learning are discussed, and an autonomous UAV flight literature review is provided. In Section 3, the techniques used to connect the modules and the modular learning method proposed in this research are presented. In Section 4, the two task scenarios that are connected in modular learning are explored, and the experimental results are presented. Finally, Section 5 summarizes the conclusions, and future research directions are discussed.

2. Background

2.1. Reinforcement Learning

Reinforcement learning is an method used to train a desired policy (π) using a reward that the agent receives after taking an action in a certain state in a given environment. Rein-

forcement learning aims to maximize the sum of accumulated rewards. For example, the Q-learning algorithm learns the Q-value of action a at a certain state s, which is defined as:

$$Q(s,\ a) = r(s,\ a) + \gamma \max_{a'} Q(s',\ a'). \quad (1)$$

In Equation (1), $r(s, a)$ represents the reward obtained from taking action a in state s, $s\prime$ represents the next state, $a\prime$ represents all possible actions in the next state, and γ is a discount factor related to the importance of future rewards relative to immediate rewards. Thus, the Q-value is the sum of the reward obtained from the current state and action and the maximum Q-value in the next state. This value is updated through learning to find the optimal Q-value. After the update is completed, the agent selects the action that helps yield the maximum Q-value. The basic concept of reinforcement learning is learning to maximize the reward through interaction with the environment, as shown in Figure 1 [24].

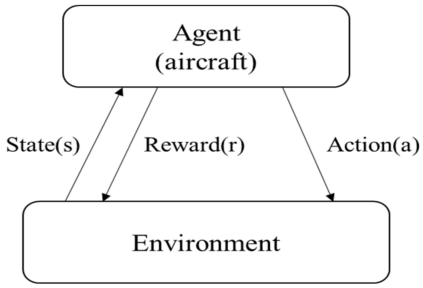
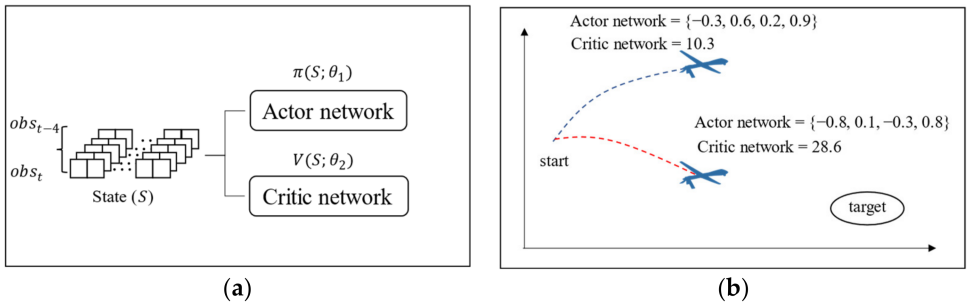


Figure 1. Basic concept of reinforcement learning.

2.2. Actor-Critic Method

A reinforcement learning algorithm that selects an action through values, such as Q-learning, is referred to as a value-based reinforcement learning method. An action that maximizes the value after completing learning is selected from the beginning to the end of an episode; therefore, no policy is needed. Consequently, it is difficult to determine the value of an action for high-dimensional and continuous tasks. In contrast, an algorithm that selects an action according to probability and then conducts policy training is referred to as a policy-based reinforcement learning method. Because learning is sound even for continuous and high-dimensional tasks, this method is suitable for the environment and agent used in this study.

An actor-critic algorithm is a widely used policy-based algorithm that comprises a critic network that approximates the state value and an actor network that estimates the policy. As shown in Figure 2, the actor network learns policies for the agent to take actions and outputs them accordingly. The critic network evaluates the value of the current state and uses this value to train the actor network [25].



(**a**) (**b**)

Figure 2. (**a**) Actor-critic network output. Five timestep states are merged into one observation and used as the input of the networks. (**b**) Well-trained networks predict high values when the agent approaches the target.

2.3. Soft Actor-Critic Algorithm

An SAC algorithm encourages exploration through an entropy regularization term, and it can reuse previously collected experiences through off-policy learning [26]. Numerous existing algorithms that use actor-critic networks cannot reuse the training data and have the disadvantage of reduced learning efficiency; therefore, an SAC algorithm samples the learning data, which are reused in learning [26]. In addition, approximating and presenting the actor and critic networks through functions leads to effective learning for UAVs in continuous action space reinforcement learning problems. The optimal policy π^* in an SAC algorithm is defined as shown in Equation (2). Equation (2) adds an entropy term to the traditional reinforcement learning algorithm, which learns to maximizing the sum of the expected rewards of the subsequent policy π and the entropy of the action probabilities for the current policy. A high value of action probability entropy indicates that the action is randomized, so high entropy encourages exploration. α is the weight of the entropy term, and increasing α further encourages exploration.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t E_{(s_t,\, a_t)\sim\pi} \gamma[r(s_t,\, a_t) + \alpha H(\pi(\cdot|s_t)]. \tag{2}$$

An SAC algorithm is a reinforcement learning algorithm that performs well in a continuous action space, reuses training data for efficient learning, and adds an entropy term to encourage exploration, making it suitable for UAV reinforcement learning, which requires learning in a high-dimensional state space.

2.4. Autonomous UAV Maneuvering

Research on autonomous UAV maneuvering has been conducted for years, with varying studies applying rule-based methods for performing the designated maneuvering of an aircraft according to previously given conditions and others applying reinforcement learning methods using artificial neural networks. Among them, example studies using reinforcement learning are as follows.

Yang et al. [10] resolved the problem of selecting 15 basic maneuvers, such as straight deceleration, straight acceleration, turning left, and turning right, in a close combat problem using a deep Q-network. Wang et al. [11] proposed a learning method in which the aircraft could react to various enemy maneuvers in a 2D simulation environment in which the aircraft velocity was limited. Lee and Kim [12] proposed a method that could avoid a dangerous object approaching the UAV in a 2D grid simulation environment. Yan et al. [13] proposed continuous actor-critic learning automation in an environment where the action space was limited by the velocity and roll and succeeded in training an aircraft following another aircraft. Bohn et al. [14] used a proximal policy optimization algorithm to show that the position of a fixed-wing UAV could approximate a similar existing proportional–integral–derivative controller method, even under severe turbulence conditions. Tang and Lai [15] used a deep deterministic policy gradient algorithm to support UAV landing along a consistent route; however, the experiment was conducted in an abstract simulation scenario different from the actual environment. Notably, most reinforcement learning-based research has been performed in simplified simulation environments, and action space maneuvering (e.g., left turn, right turn, etc.) was simplified or involved the selection of optimal maneuvering in downscaled environments [18–20,22].

However, there has been a lack of research on autonomous UAV flight based on reinforcement learning methods that support complex maneuvering in high-dimensional states and action spaces in realistic simulation environments.

3. Proposed Method

3.1. Modular Learning

Modular learning is used to divide a complex task into several simple modules for learning. This approach is effective because learning a complex task from scratch can be difficult, but learning simple modules does not require as many resources because high per-

formance can be achieved with relative ease [27]. This study proposes a modular learning method that enables complex maneuvering for autonomous UAV flight in continuous and high-dimensional states and action spaces in realistic simulation environments. Furthermore, the experimental environment is based on physical laws, eliminating the need for new learning by leveraging transfer learning to transfer information from already trained modules to others [27]. Using this method, other modules can be trained with limited resources, and better performance can be achieved compared to learning from scratch [28]. For new UAV types with varying operational methods, previous training information can be transferred to facilitate learning. In conclusion, the modular learning method provides numerous advantages to autonomous UAV flight, including improved learning efficiency, high performance, and the easy accomplishment of complex tasks. Moreover, by breaking down complex tasks into smaller and simpler modules, an autonomous UAV flight algorithm that is both more effective and powerful can be developed. Figure 3 shows a training scenario for identifying and tracking an unidentified UAV after arriving at a point that needs to be explored for Type 1 and Type 2 UAVs; this task is divided into two parts: Mission 1, an approach maneuver to reach the target point, and Mission 2, a tracking maneuver to follow the target. After each module is trained using an individual network, if the Module #1 agent satisfies the completion condition of arriving at the target location, Module #2 will take over the task and track the target.

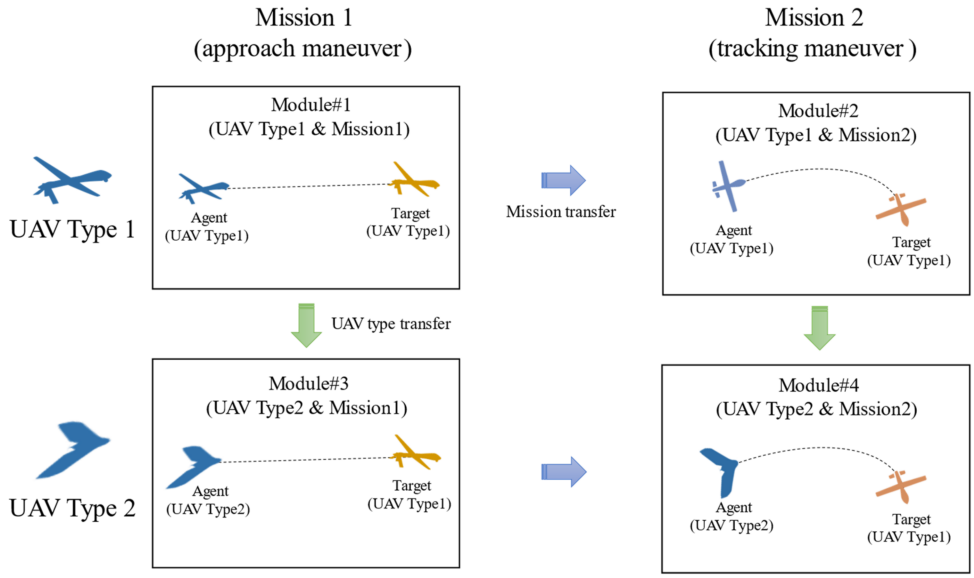


Figure 3. Concept of modular learning.

The modular learning method proposed in this paper has the advantage of enabling efficient learning with limited resources. If a single network is used to train the UAV, the entire network must be retrained each time a new mission is performed. For example, let us consider a new compound task by adding a new maneuver, namely, Mission 3, which needs to be performed after Missions 1 and 2. To learn the composite task with a single network, the entire network, which has already learned the basic features of Missions 1 and 2, must be retrained to perform all missions. However, in this case, the existing network may not be able to fully learn the characteristics of the new mission and may fail to perform the complex mission or require a larger network size.

In conclusion, learning with a single network requires constructing and retraining a new network each time a new task must be performed. However, modular learning can utilize network knowledge learned in existing missions through transfer learning and learn the required tasks through fine-tuning for new tasks. By connecting the network to the existing learned modules, learning can effectively occur without wasting computational resources or discarding existing knowledge. In this paper, we demonstrate through exper-

iments that, compared to single networks, a network based on modular learning can be successfully applied to perform complex missions. Additionally, the performance of this approach is good in cases with single missions and complex missions decomposed into modules.

3.2. Smooth Module Connection

In modular learning, different modules individually learn and are subsequently connected. If the task of Module #1 is accomplished and completed, the completed condition is transferred to Module #2 as the starting condition. Module #1 must reach its destination regardless of the starting point, and each module must learn from various starting points to perform robust autonomous flight. Therefore, noise was added to the starting condition of each module to enable robust learning. If significant noise is present initially, the learning process may not be efficient; therefore, only a small amount of noise was added initially and increased during each round of learning until a task was successfully completed. This approach is illustrated in the pseudocode in Algorithm 1. Consequently, we allowed the target tracking maneuver regardless of the starting condition when Module #2 took over the task from Module #1.

Algorithm 1: Soft Actor-Critic and Noise

Set : $R_s,\ I,\ G$ ▷ Set success criteria (R_s), initial condition (I) and final goal (G)
Initialize $\theta_1,\ \theta_2,\ \phi$ ▷ Initial parameters of actor & critic networks
$\theta_1 \leftarrow \theta_1,\ \theta_2 \leftarrow \theta_2$ ▷ Initialize *critic* networks $(target\ Q\ and\ Q)$ weights
$\mathcal{B}_\mathrm{R} \leftarrow \varnothing$ ▷ Initialize replay buffer $(\mathcal{B}_\mathrm{R})$
$R,\ N \leftarrow 0$ ▷ Initialize episode reward (R) and success counter (N)
For *each episode* **Do**
  $\mathcal{B}_\mathrm{E} \leftarrow \varnothing$ ▷ Initialize an empty episode buffer
  $I \leftarrow I + noise \times N$ ▷ Add noise to the initial condition
  while *not Done*
    $a_t \sim \pi_\phi(a_t|s_t)$ ▷ Sample an action from the policy
    $s_{t+1} \sim p(s_{t+1}|s_t,\ a_t)$ ▷ Sample a transition from the environment
    $r_t \sim (s_t, a_t)$ ▷ Reward from environment
    $Done \sim (s_t, a_t)$ ▷ Done from environment
    $\mathcal{B}_\mathrm{R} \leftarrow \mathcal{B}_\mathrm{R} \cup \{(s_t,\ a_t,\ r_t,\ s_{t+1})\}$ ▷ Store the transition in the replay buffer
    $\mathcal{B}_\mathrm{E} \leftarrow \mathcal{B}_\mathrm{E} \cup \{(s_t,\ a_t,\ r_t, s_{t+1})\}$ ▷ Store the transition in the episode buffer
    $R \leftarrow R + r_t$ ▷ Calculate the value
  End while
  IF $R \geq R_s$ **Then**
    $N \leftarrow N + 1$ ▷ Add success counter
  End IF
  For *each gradient* step **Do**
    $\theta_1 \leftarrow \theta_1 - \lambda_\mathrm{Q} \hat{\nabla}_{\theta_1} J_\mathrm{Q}(\theta_1)$ ▷ Update Q − *network* parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\theta_2 \leftarrow \tau\theta_1 + (1-\tau)\theta_2$ ▷ Update target $Q-$ network weights
  End For
End For
Output : $\theta_1,\ \theta_2,\ \phi$ ▷ Optimized parameters

3.3. Curriculum Learning

Curriculum learning begins with easy tasks and gradually progresses to difficult tasks [29,30]. By applying this approach to a reinforcement learning agent, it is possible to acquire knowledge by starting with relatively low-difficulty but similar-purpose tasks at the beginning of the learning process and gradually increasing the difficulty until the agent finally performs the desired task. In this study, curriculum learning was applied for the robust training of each module. The reinforcement learning agent must update its policy using only the reward as a scalar value; therefore, a significant amount of learning data is necessary to achieve the desired goal, which can sometimes be very difficult if the

task is complex. Curriculum learning can be used to acquire task knowledge and achieve goals through performing easier tasks. In this research, the success condition of the task was initially broadly set, as shown in Figure 4. Then, as the agent successfully reached the target point, the target point was gradually narrowed until the agent was able to reach the final point. This process is shown in Algorithm 2. The hyperparameters for Algorithm 2 are described in Section 4.

Algorithm 2: Soft Actor-Critic and Noise and Curriculum Learning

Set : R_s, T, G
▷Set success criteria (R_s), initial condition (I), Target Zone(T), and final goal (G)
Initialize θ_1, θ_2, ϕ ▷ Initial parameters of actor & critic networks
$\theta_1 \leftarrow \theta_1$, $\theta_2 \leftarrow \theta_2$ ▷ *Initialize critic networks (target Q and Q) weights*
$\mathcal{B}_R \leftarrow \varnothing$ ▷ Initialize replay buffer $(\mathcal{B}_R)$
R, $N \leftarrow 0$ ▷ Initialize episode reward (R) and success counter (N)
For *each episode* **Do**
$\mathcal{B}_E \leftarrow \varnothing$ ▷ Initialize an empty episode buffer
IF $T > G$ **Then**
$I \leftarrow I + noise \times N$ ▷ Add noise to initial condition
$T \leftarrow T - \delta \times N$ ▷ Narrow the target zone T(curriculum learning)
END IF
while *not Done*
$a_t \sim \pi_\phi(a_t|s_t)$ ▷ Sample an action from the policy
$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ ▷ Sample a transition from the environment
$r_t \sim (s_t, a_t)$ ▷ Reward from environment
$Done \sim (s_t, a_t)$ ▷ Done from environment
$\mathcal{B}_R \leftarrow \mathcal{B}_R \cup \{(s_t, a_t, r_t, s_{t+1})\}$ ▷ Store the transition in the replay buffer
$\mathcal{B}_E \leftarrow \mathcal{B}_E \cup \{(s_t, a_t, r_t, s_{t+1})\}$ ▷ Store the transition in the episode buffer
$R \leftarrow R + r_t$ ▷ Calculate the value
End while
IF $R \geq R_s$ **Then**
$N \leftarrow N + 1$ ▷ Add success counter
End IF
For *each gradient* step **Do**
$\theta_1 \leftarrow \theta_1 - \lambda_Q \hat{\nabla}_{\theta_1} J_Q(\theta_1)$ ▷ Update Q − *network* parameters
$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
$\theta_2 \leftarrow \tau\theta_1 + (1 - \tau)\theta_2$ ▷ Update target Q − network weights
End For
End For
Output : θ_1, θ_2, ϕ ▷ Optimized parameters

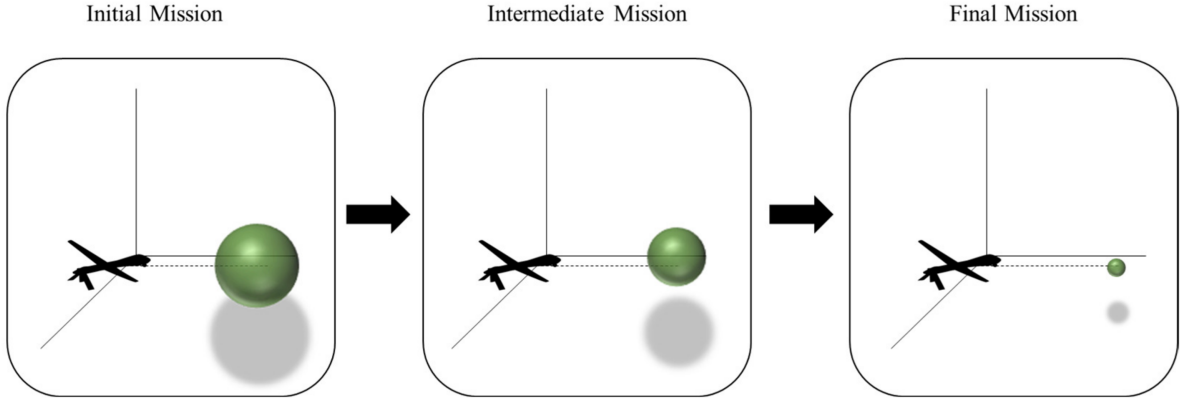


Figure 4. Concept of curriculum learning.

4. Experiments

4.1. Simulation Environment Definition

Reinforcement learning involves learning through the repetitive performance of suitable action mapping tasks to maximize the reward at given states [31]. However, in reality, there are temporal and spatial restrictions that affect this repetitive performance. Therefore, many studies of reinforcement learning have been conducted in simulated environments. In this section, we define the environment, states, actions, and rewards for learning in simulated environments.

4.1.1. Environment

In this research, the open-source aerodynamics model JSBSim was used for learning. JSBSim uses a 6-DOF aerodynamics model created with C++, commonly used in engineering simulations and design for various aircraft, spacecraft, and rockets. It was previously used in the air combat evolution program of the Defense Advanced Research Projects Agency of the United States [32,33]. Rennie [34] developed Gym-JSBSim for the application of JSBSim, which was developed with C++ in a Python environment and revised to fit the learning objectives of this research, as shown in Figure 5.

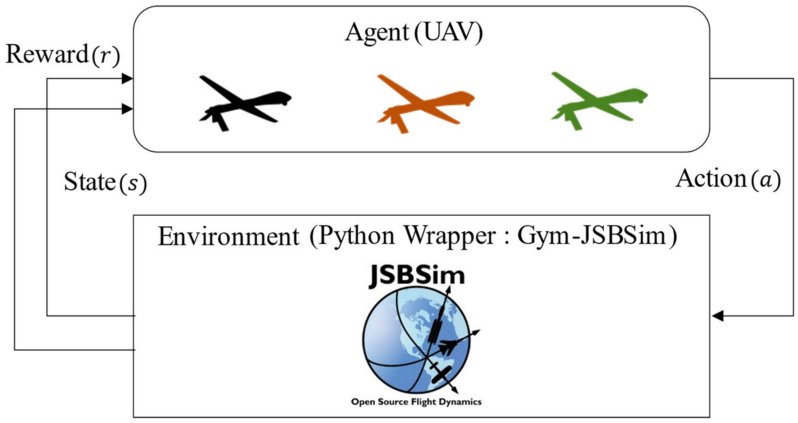


Figure 5. Conceptual diagram of the simulation environment.

4.1.2. States

The state is represented in a finite set $S = \{s^1, \ldots, s^N\}$ of size N ($N = 20$), all states must be acquired from the environment, and each must include the necessary information for the agent to perform the learning [35]. All states S in this research were distinguished and included in S_{basic} and $S_{relative}$ as follows:

$$S = \{S_{basic}, S_{relative}\}, \tag{3}$$

$$S_{basic} = \{x, y, z, \psi, \theta, \varphi, u, v, w, p, q, r\},$$

$$S_{relative} = \{d^x, d^y, d^a, p^t, h^t, D, AA, HCA\}.$$

As shown in Figure 6, S_{basic} represents the state information of the agent that can be directly acquired from JSBSim, and it comprises the location data of the agent $\{x, y, z\}$, 3-axis velocity $\{u, v, w\}$, angle of each axis $\{\varphi, \theta, \psi\}$, and acceleration along each axis $\{p, q, r\}$. The definition of each state is shown in Table 1.

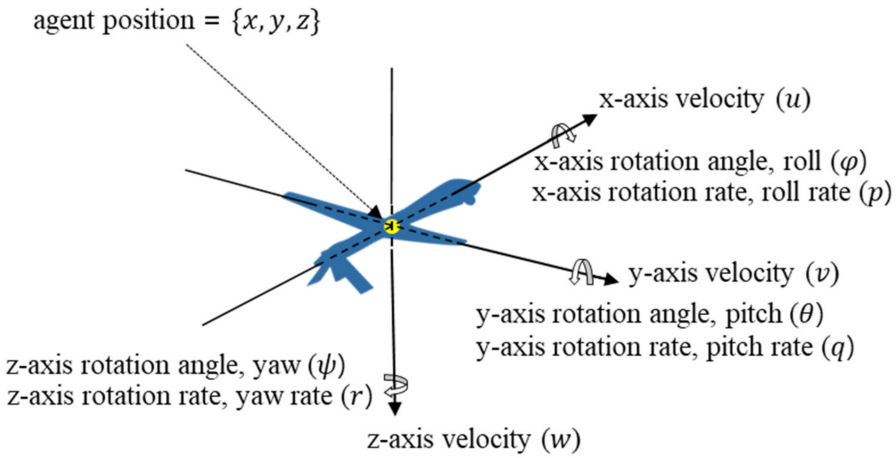


Figure 6. Basic UAV agent states.

Table 1. Basic state definitions.

State	Definition	State	Definition
x	x-axis position	y	y-axis position
z	z-axis position	u	x-axis velocity
v	y-axis velocity	w	z-axis velocity
φ	x-axis rotation angle (roll)	θ	y-axis rotation angle (pitch)
ψ	z-axis rotation angle (yaw)	p	x-axis rotation rate (roll rate)
q	y-axis rotation rate (pitch rate)	r	z-axis rotation rate (yaw rate)

Through S_{basic}, the information associated with the corresponding agent and target in 3D space can be obtained; however, the relative target information cannot be acquired from JSBSim and, hence, the relative target location and geometric information is expressed through $S_{relative}$, as shown in Table 2.

Table 2. Relative state definitions.

State	Definition	State	Definition
d^x	x-axis distance between the UAV and target	d^y	y-axis distance between the UAV and target
d^z	z-axis distance between the UAV and target	p^t	Pitch angle to target
h^t	Heading angle to target	D	Distance to target
AA	Aspect angle	HCA	Heading cross-angle

As shown in Figure 5, $\{d^x, d^y, d^a\}$ expresses the distance between the agent and the target in the x, y, and z planes, whereas the distance (D) represents the Euclidian distance between the agent and the target in 3D space. The pitch angle (p^t) and heading angle (h^t) to the target indicate the up and down and left and right angles, respectively, which allow the agent to gaze at the target. The aspect angle (AA) and heading cross-angle (HCA) comprehensively express the location information for the agent and the target. AA refers to the angle from the tail of the target to the agent, which varies from 0° to 180° depending on whether the agent is located directly behind or in front of the target, respectively. HCA refers to the heading difference between the agent and the target, which varies from 0° to 180°, depending on whether they face the same or opposite directions, respectively. When the AA and HCA states between two aircraft are given, the 2D location data can be ascertained, as shown in Figures 7 and 8.

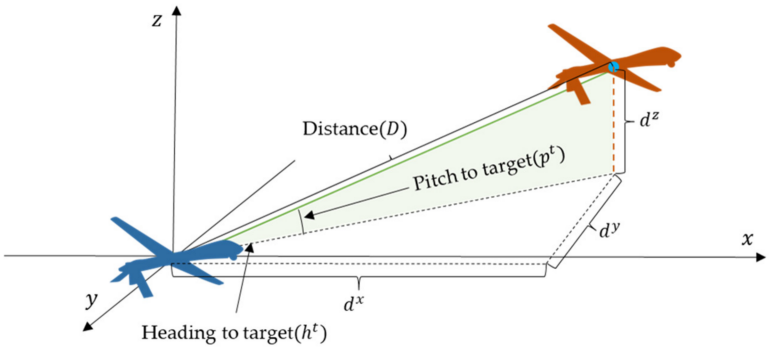


Figure 7. States for the relative position of the agent and target.

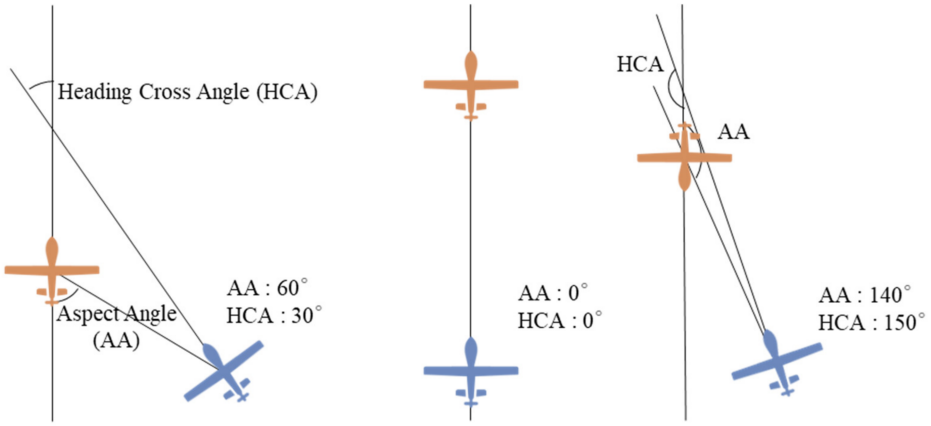


Figure 8. AA and HCA examples under different conditions.

4.1.3. Action

An action is represented as a finite set $A = \{A^1, \ldots, A^K\}$ with size K $(K = 4)$, and the state can be controlled through this action. In this research, the agent was set as a fixed-wing unmanned aircraft, and the action was designed to be identical to that of a general fixed-wing aircraft. The fixed-wing aircraft was controlled in 3D space with a stick and rudder combination that moved the control surfaces of the main and tail wings, as well as a throttle for engine thrust. Therefore, the agent action was constructed as a combination of stick, rudder, and throttle adjustment, and the stick action was distinguished as $\{s_x, s_y\}$ to express the stick movement in 2D space. Therefore, including the stick, rudder, and throttle adjustments, the action was defined as shown in Equation (4). The definition and range of each action are shown in Table 3.

$$A = \{s_x, s_y, \tau, \rho\}. \tag{4}$$

Table 3. Action definitions.

Action	Definition	Action	Definition
s_x	X-axis stick position (-1 to 1)	τ	Throttle angle (0 to 1)
s_y	Y-axis stick position (-1 to 1)	ρ	Rudder pedal angle (-1 to 1)

4.1.4. Reward

The reward defines the aim of reinforcement learning. The agent receives an assessment of the action performed in the corresponding state as a reward and learns in the direction that maximizes the total sum of the rewards. Therefore, defining a proper reward is important for successful learning [35]. In this research, rewards were divided into episode

and timestep rewards. The episode reward was received when the episode was terminated following task failure or success, whereas the timestep reward was received according to the agent state for each simulation step.

The reward must be appropriately set so that each task can be successfully completed. Therefore, the reward for each mission was set as described in Section 3 and explained in detail in the Experimental Design section.

4.2. Experimental Design

In this section, we define the reward function and the start and end conditions for each mission in the modular learning experiment. Furthermore, we describe the transfer learning process for each module and the connection experiment for the trained modules.

4.2.1. Mission 1 (Approach Maneuver) Definition

The objective of Mission 1 is to learn the approach maneuver for a target that flies straight and steady at a distance. As shown in Figure 9, the task is completed starting at a distance of 70 km from the target and reaches a distance of 8 km from the rear of the target.

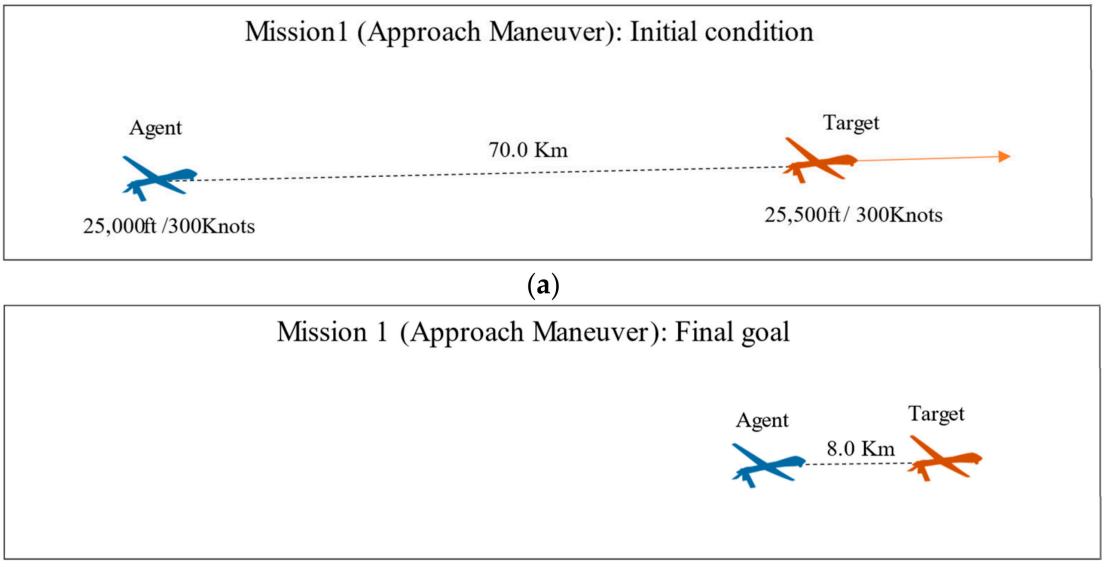


Figure 9. Overview of Mission 1 (approach maneuver): (**a**) initial condition and (**b**) final goal.

The reward in this study was defined as a mission-ending condition based on episode and timestep rewards given once for each episode and timestep, respectively. Mission 1 comprises four episode rewards and two timestep rewards.

Success reward: This reward is given when the designated target is achieved. The task is completed after receiving the reward when the desired location is reached from the target rear. Here, the target objective comprises a combination of four conditions, as shown in Equation (5).

$$\begin{gathered} IF\ (D < 8.0km)\ and\ (d^z < 400m)\ and\ \left(\left|h^t - \psi\right| < 20^\circ\right)\ and\ (|\theta| < 5^\circ) \\ THEN\ \text{done}\ and\ reward = 5 \end{gathered} \tag{5}$$

Crash reward: The task is completed and a penalty is received if the agent does not reach the target and collides with the ground surface.

$$IF\ (z < 100m)\ THEN\ \text{done}\ and\ reward = -10 \tag{6}$$

Overtake reward: If the agent overtakes the target's wing line, a penalty is received and the task ends.

$$IF\ (AA > 90^{\circ})\ THEN \text{ done } and\ reward = -10 \tag{7}$$

Timeout reward: If the task is not successfully performed after 10,000 steps based on the simulation time, a penalty is received and the task ends.

$$IF\ (timesteps > 10,000)\ THEN \text{ done } and\ reward = -10 \tag{8}$$

Line of sight (LOS) score: A reward is given when the difference between the altitude and heading between the agent and the target is small for each timestep and the agent approaching from the rear is oriented in the same direction as the target trajectory.

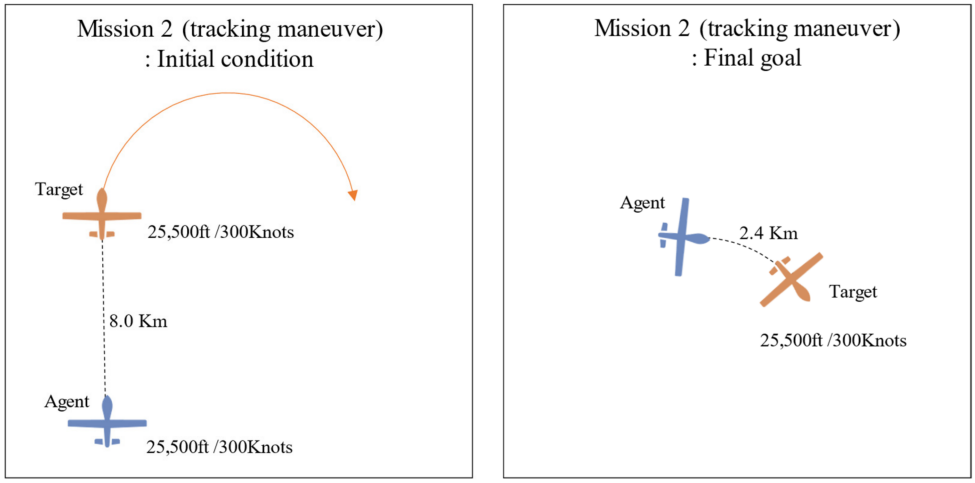
$$\text{LOS score} = \left(0.666 \times \exp(-0.5 \times \left(\frac{d^z}{500m}\right)^2)\right) + \exp\left(-0.5 \times \left(\frac{h^t - \psi}{30^{\circ}}\right)^2\right) \tag{9}$$

Close score: This reward refers to the value of the distance from the target at point t subtracted from the distance at point t − 1, indicating the distance between the agent and target.

$$\text{Close score} = D_{t-1} - D_t \tag{10}$$

4.2.2. Mission 2 (Tracking Maneuver) Definition

The objective of Mission 2 is to learn the approach maneuver for a target that turns around. This task begins from a location that is 8 km from the target, as shown in Figure 10, and the task is complete when the agent reaches a distance of 2.4 km from the target rear.



(**a**) (**b**)

Figure 10. Overview of Mission 2 (tracking maneuver): (**a**) initial condition and (**b**) final goal.

The reward for Mission 2 was transformed from the reward of Mission 1 to fit the relevant objective, and the desired zone score for inducing the approach to the desired point was added, and four episode rewards and three timestep rewards were created.

Success reward: This is a reward given when the desired location at the rear of the target is reached and the mission is completed. The aim of the task includes the aspect angle, which was not considered in Mission 1, and d^z and θ are removed to create a combination of three conditions.

$$IF\ (D < 2.4km)\ and\ (\text{AA} < 50^{\circ})\ and\ \left(\left|h^t - \psi\right| < 5^{\circ}\right)\ THEN \text{ done } and\ reward = 5 \tag{11}$$

Crash reward: The task is completed and a penalty is received if the agent does not reach the target and collides with the ground surface.

$$IF\ (z < 100m)\ THEN \text{ done } and\ reward = -10 \tag{12}$$

Overtake reward: If the agent overtakes the target wing line, a penalty is given, and the characteristics are slightly adjusted in this case, compared to those in Mission 1, to allow for a slight overtake.

$$IF\ (AA > 110^{\circ})\ THEN \text{ done } and\ reward = -10 \tag{13}$$

Timeout reward: If the task does not succeed after reaching 3000 timesteps, a penalty is given, and the task ends.

$$IF\ (timesteps > 3000)\ THEN \text{ done } and\ reward = -10 \tag{14}$$

Line of sight (LOS) score: In precise short-distance maneuvering, the pitch p^t toward the target is used rather than the altitude difference d^z between the agent and target, unlike in Mission 1, to determine the LOS.

$$\text{LOS score} = \left(0.3 \times exp(-0.5 \times \left(\frac{p^t - \theta}{3^{\circ}}\right)^2)\right) + \left(0.3 \times exp(-0.5 \times \left(\frac{h^t - \psi}{3^{\circ}})\right)^2)\right) \tag{15}$$

Close score: The reward is the value of the distance from the target at point t subtracted from the distance at point t − 1, which is the decrease in the distance between the agent and target.

$$\text{Close score} = 5 \times (D_{t-1} - D_t) \tag{16}$$

Desired zone score: The desired zone condition is more relaxed than the task success condition, and a reward is given when the location is within a set boundary.

$$IF\ (|AA| < 30^{\circ})\ and\ (|HCA| < 40^{\circ})\ and\ (D < 5km)\ THEN \text{ Desired zone score,}$$
$$\text{Desired zone score} = \frac{40 - |\text{AA}|}{30^{\circ}} \tag{17}$$

4.2.3. Hyperparameter Settings

For the experiment, the Algorithm 2 hyperparameters described in Section 3 are shown in Table 4. R_s is the episode success criterion; if it is greater than the set value of R_s, which is the cumulative reward sum, the corresponding episode is a success. I refers to the initial value of the agent in the simulated environment, and it represents the location and velocity of the agent in 3D space. Additionally, noise is added to the initial value as the episode proceeds and learning occurs, making it robust to changes in the starting condition and facilitating the connections between modules in the future. T represents the ending conditions of each episode, and the level of difficulty is increased by δ each time the episode narrows the target zone until the final objective G is achieved.

Table 4. Algorithm 2 hyperparameters.

	R_s	I	Noise	T	δ	G
Mission 1 (approach maneuver)	2500	x : 3.5 y : 4.5 z : 25,000 θ : 0 u : 500 v : 0 w : 0	x : $random(\pm 0.0001)$ y : $random(\pm 0.0002)$ z : $random(\pm 5)$ θ : $random(\pm 0.05)$ u : $random(-0.1, +2)$ v : $random(\pm 0.2)$ w : $random(\pm 0.2)$	15.0 km	0.02	8.0 km
Mission 2 (tracking maneuver)	500			4.6 km	0.02	2.4 km

4.2.4. Experiment #1: Learning for Each Module

Experiment #1 was carried out to identify whether a single task could be successfully performed by the UAV through reinforcement learning. The network weights of the UAV for each task learned in the experiment were used to proceed with modular learning for complex task performance.

Experiment #1 involved learning using curriculum learning and the SAC algorithm for the four modules shown in Figure 3. Each experiment was conducted from scratch without transfer learning from existing trained models, and identical rewards were established for the same task performance, even when the UAV type varied. The experiment proceeded as shown in Table 5 based on UAV Types 1 and 2 in the simulated environment structured through JSBSim. The first type of UAV was a high-altitude UAV with a single engine. The second type of UAV was a high-altitude UAV with two engines. These UAVs had clearly different flight characteristics, making them suitable for verifying the proposed methodology. Both UAVs were built in JSBSim, and factors such as weight and drag were defined with the help of experts.

Table 5. Experiment #1.

	Description
Expt. 1-1	Train Module #1 (UAV Type 1 & Mission 1) from scratch
Expt. 1-2	Train Module #2 (UAV Type 1 & Mission 2) from scratch
Expt. 1-3	Train Module #3 (UAV Type 2 & Mission 1) from scratch
Expt. 1-4	Train Module #4 (UAV Type 2 & Mission 2) from scratch

4.2.5. Experiment #2: Transfer Learning Based on the Mission and UAV Type

Experiment #2 was performed to identify whether efficient task learning was possible through transfer learning for all cases when performing different tasks using the same UAV type and the same task using different UAV types. Therefore, in the case of learning a new task network from modular learning in this experiment, efficient learning was achieved through transfer learning. In Experiment 2-1, as shown in Table 6, the weight of Module #1, which completed learning in Experiment 1-1, was transferred to Module #2, which performed a task different from that in Module #1. In the case of Module #3, the weight of Module #1 was transferred to a different UAV type, which performed an identical task to proceed with learning.

Table 6. Experiment #2.

	Description
Expt. 2-1	Transfer learning by applying the weights from Experiment 1-1 to Module #2
Expt. 2-2	Transfer learning by applying the weights from Experiment 1-1 to Module #3

4.2.6. Experiment #3: Module Connection

Experiment #3 was conducted to verify the effect of the modular learning approach proposed in this research, and it was completed as a single task by connecting Modules #1 and #2 from the previous experiment, as shown in Table 7. Therefore, in this case, modular learning involved using different networks for each task. For comparison, we concatenated the two maneuvers into a single mission and trained a single network for learning, as shown in Figure 11. Modular learning allows efficient learning through transfer learning for each module. To compare the learning results of a single network with those of modular learning under identical conditions, the weight of the single Module #1 was transferred for learning.

Table 7. Experiment #3.

	Description
Expt. 3-1	Connecting the approach maneuver and tracking maneuver learned in each network
Expt. 3-2	Learning the approach maneuver and tracking maneuver with a single network

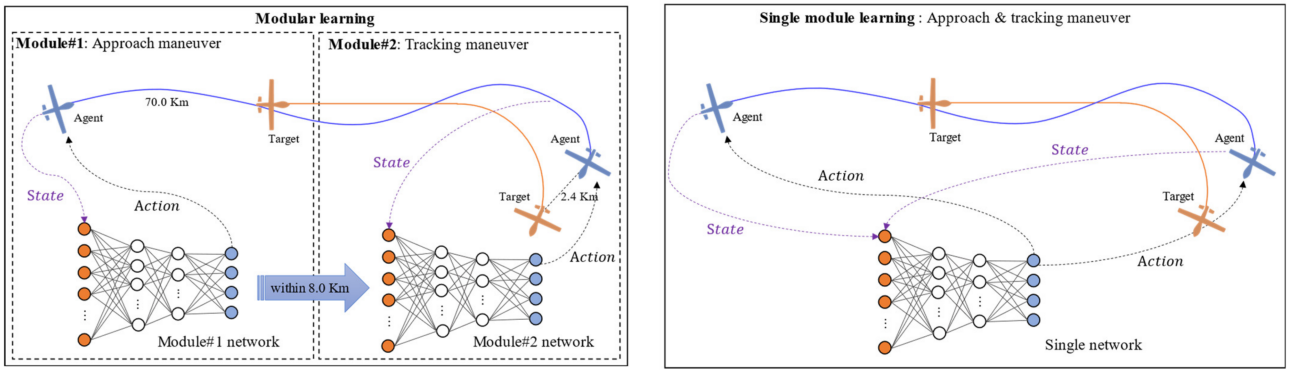


Figure 11. Concepts of Experiments 3-1 and 3-2: (**a**) modular learning and (**b**) single network learning.

4.2.7. Experiment #4: Agent Test

Experiment #4 was designed to verify that the agent trained with modular learning in Experiment 3-1 could successfully perform the task after training. In other words, the purpose of this experiment was to test the robustness of the performance of the agent trained with modular learning. Since a trained agent may experience overfitting and not be able to perform the task consistently, the experiment was tested with different initial conditions. The noise used to establish the different conditions was adjusted as described in Table 4. Experiment 4-1 was performed with an agent that was stable after learning over 1000 episodes in Experiment 3-1. To assess the modular learning approach under the same conditions, the single-network agent trained in Experiment 3-2 was tested under the conditions in Experiment 4-2, as shown in Table 8.

Table 8. Experiment #4.

	Description
Expt. 4-1	Testing the agent trained with modular learning in Experiment 3-1
Expt. 4-2	Testing the agent trained with a single network in Experiment 3-2

4.3. Experimental Results

4.3.1. Results of Experiment #1

Experiment 1 involved four modules with varying UAV types and missions. The success score standard for the approach maneuver in Experiments 1-1 and 1-3 was 2500, and for the tracking maneuver in Experiments 1-2 and 1-4, it was 500. In the approach maneuver, a long flight distance was used compared to that for the tracking maneuver, resulting in a high standard for the success reward. All modules began without initial transfer learning, and 1500 episodes were explored in the experiment for each module. The results are shown in Table 9.

Table 9. Results of experiment #1.

	Min Score	Max Score	Mean Score	Cumulative Successes
Expt. 1-1: Module #1 (UAV Type 1 & Mission 1)	−2595.25	6383.08	1127.48	684
Expt. 1-2: Module #2 (UAV Type 1 & Mission 2)	−583.51	1371.83	233.15	908
Expt. 1-3: Module #3 (UAV Type 2 & Mission 1)	−975.43	7364.83	1285.82	304
Expt. 1-4: Module #4 (UAV Type 2 & Mission 2)	−1194.26	1349.59	143.26	797

In Experiments 1-1 and 1-2, which were based on UAV Type 1, the cumulative number of successful episodes was higher than that in Experiments 1-3 and 1-4, which involved training based on UAV Type 2. Therefore, the learning difficulty for UAV Type 1 was lower. To efficiently learn with UAV Type 2, we utilized the weights of UAV Type 1 to perform transfer learning between UAVs. Figure 12 shows the training results for each module based on the output values. Learning for the approach maneuver in Experiments 1-1 and 1-3 takes a longer time than that for the tracking maneuver.

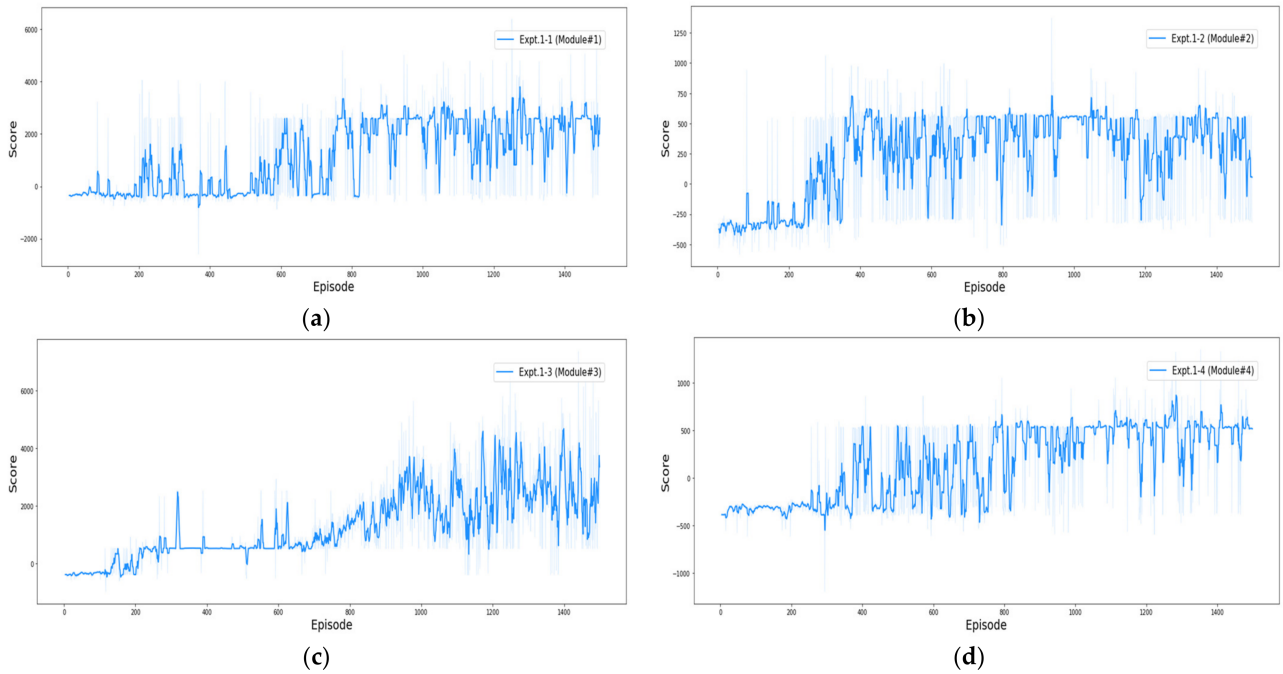


Figure 12. Results of experiment #1 (score plot): (**a**) Experiment 1-1, (**b**) Experiment 1-2, (**c**) Experiment 1-3 and (**d**) Experiment 1-4.

The results of Experiment #1 show that a single network can effectively learn for single tasks that are relatively easy compared to multiple tasks.

4.3.2. Results of Experiment #2

Experiment #2 was conducted to identify the effect of transfer learning on various UAV types and tasks using the network weights of each trained module from Experiment #1. Experiment 2-1 aimed to identify the effect of transfer learning when identical UAVs performed different tasks, and the network weight of UAV Type 1, trained based on the approach maneuver, was transferred to the network that performed the tracking maneuver. To identify the effect of transfer learning, we compared the agent learning results of Experiment 1-2, which involved training from scratch without transfer learning, with those of the current experiment.

The experimental results showed that the performance of Experiment 2-1 with transfer learning was superior to that of Experiment 1-2 without transfer learning, as shown in Table 10. Additionally, as shown in Figure 13, there was an advantage for the initial learning results of Experiment 2-1, as indicated by the blue line. Through this experiment, we identified that transfer learning was effective for tasks with reward functions based on different maneuvering types. In the case of modular learning, there were networks for each task and UAV type. Therefore, when performing a new task, the previously trained module weight could be transferred, increasing the efficiency of learning.

Table 10. Results of experiment 2-1.

	Min Score	Max Score	Mean Score	Cumulative Successes
Expt. 2-1 (transfer learning)	−283.89	1620.16	452.16	957
Expt. 1-2 (from scratch)	−583.51	1371.83	233.15	908

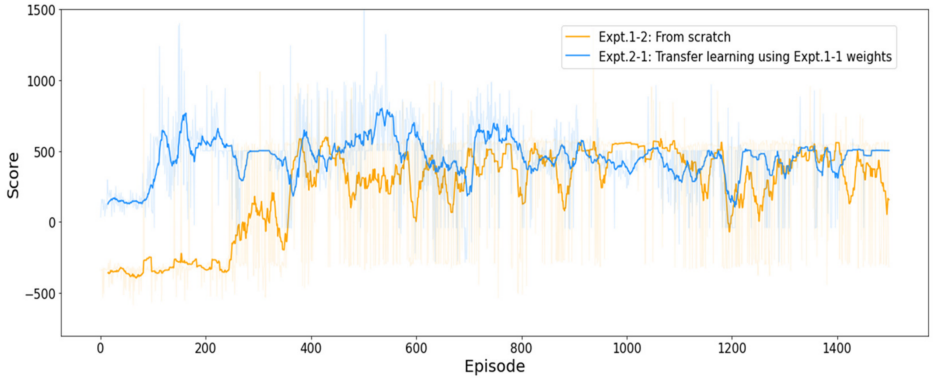


Figure 13. Results of experiment 2-1 (score plot).

Experiment 2-2 was conducted to assess the transfer learning effects for the UAV types, and weight transfer was conducted for identical Module #3 (UAV Type 2, Mission 1). For this experiment, the weight from Experiment 1-1 was transferred, and comparisons with Experiments 2-2 and 2-3, which involved training with and without weight transfer, respectively, were performed. In Table 11, Experiment 2-2 displays better performance across all indices, excluding the max value, compared to Experiment 1-3. The graph in Figure 14 shows the improvement in learning speed.

Through this experiment, the effect of transfer learning was found to be similar to that in Experiment 2-1. The results of Experiments 2-1 and 2-2 indicate that the modular learning method can be effectively applied to new UAV types and by utilizing transfer learning. Additionally, even cases in which complex forms of tasks were not performed could be divided into modules, and a single task could be performed without additional learning.

4.3.3. Results of Experiment #3

Experiment #3 was conducted to directly assess the effect of modular learning proposed in this study. The results of Experiment 3-1 (modular learning), which connected Module #1, trained with individual networks, and Module #2 as a single task, and Experiment 3-2 (single network learning), in which an identical task was learned with a single network, are shown in Table 12 and Figure 15. The experimental results indicate that the modular learning method in Experiment 3-1 initially succeeded in completing the task by connecting the previously trained modules through smooth connections and noise addition. However, Experiment 3-2, which simultaneously performed approach and tracking maneuvers with a single network, was not successful after approaching success at approximately episode 900.

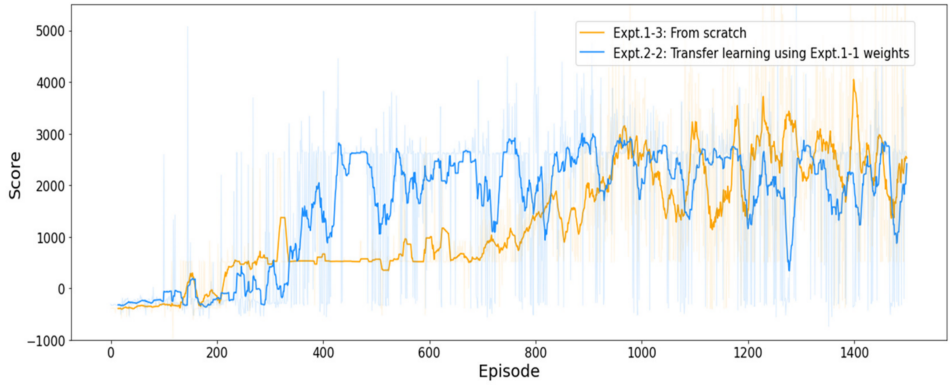


Figure 14. Results of experiment 2-2 (score plot).

Table 11. Results of experiment 2-2.

	Min Score	Max Score	Mean Score	Cumulative Successes
Expt. 2-2 (transfer learning)	−738.35	5772.40	1622.16	855
Expt. 1-3 (from scratch)	−975.43	7364.83	1285.82	304

Table 12. Results of experiment #3.

	Min Score	Max Score	Mean Score	Cumulative Successes
Expt. 3-1 (modular learning)	258.95	1788.80	685.07	1357
Expt. 3-2 (single network learning)	−37.51	1034.33	18.16	11

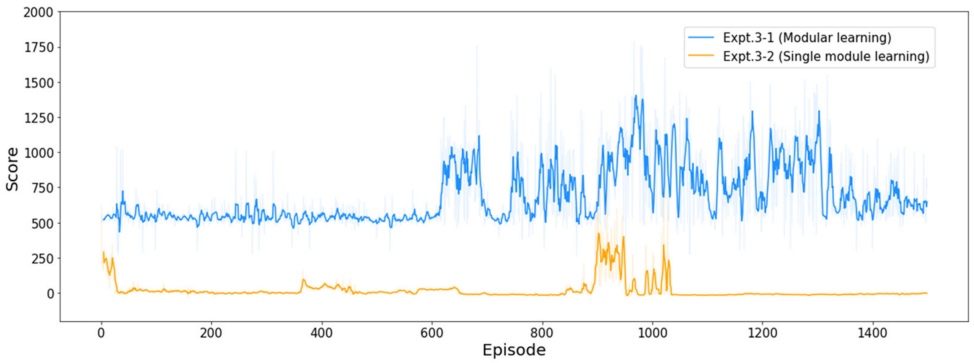


Figure 15. Results of experiment #3 (score plot).

These results show that, although learning simple maneuvering with a single network was successful, learning the maneuvers of different characteristics was inefficient. In contrast, modular learning based on identical rewards displayed stable success for complex tasks compared to a single network.

The results of Experiments 3-1 and 3-2, at the 100th and 1000th episodes, are plotted in 3D in Figure 16. Figure 16a,c show the results of modular learning, and the light blue line represents the approach maneuver, or Module #1. Here, the agent maneuvers through Module #1's network, and if Module #1 is successfully complete, the task switches to Module #2, the tracking maneuver, indicated in blue. Furthermore, if the task is transformed, the agent uses the network trained for the tracking maneuver. As shown in the figure, when modular learning was applied, we can see that the agent tracks the target successfully even

after the target turned and succeeded the task. Figure 16b,d show the results of Experiment 3-2, which involved learning with a single network. As shown in Figure 16b, at the onset of learning, the agent did not perform normal flight and collided with the ground, ending the task. As shown in Figure 16d, the agent performed an approach maneuver in the target direction but failed to normally track the target after the target turned, and the task ended.

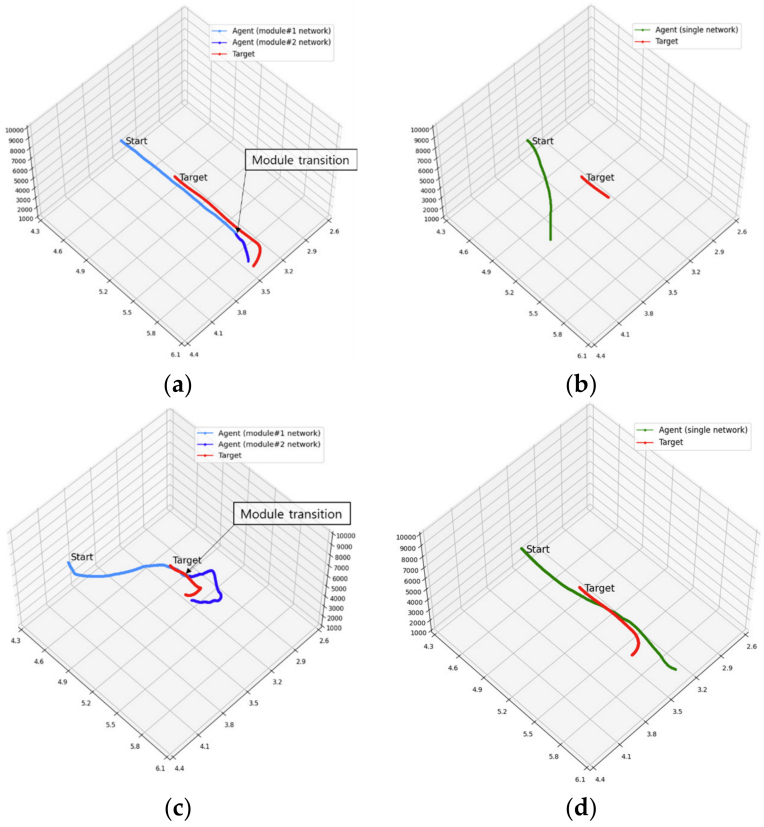


Figure 16. Results of experiment #3 (3D plot): (**a**) Expt. 3-1 (modular learning) episode 100, (**b**) Expt. 3-2 (single network) episode 100, (**c**) Expt. 3-1 (modular learning) episode 1000 and (**d**) Expt. 3-2 (single network) episode 1000.

4.3.4. Results of Experiment #4

Experiment #4 was conducted to evaluate the performance stability of the agent trained with modular learning in the previous Experiment 3-1 and the agent trained without modular learning in Experiment 3-2. This experiment was designed to determine if the trained agent could successfully continue to perform the task after being trained. The agents trained and stabilized in Experiment 3-1 and Experiment 3-2 were tested with slightly different starting conditions without additional training, in other words, without changing the weights. Furthermore, we tested whether both agents could perform well in situations in which they started with a significant altitude difference. The results demonstrate the effectiveness of modular learning.

The results of Experiment 4-1 with modular learning are better than those for Experiment 4-2 with a single network, as shown in Table 13. Additionally, as shown in Figure 17, the agent trained with modular learning reliably performed the task with comparatively less variability. Additionally, even when the initial conditions were set with an altitude difference of 4000 ft, as shown in Figure 18, the agent trained with the modular learning method successfully performs the task, and the agent trained with the single network failed to perform the task. These experiments confirm that modular learning makes the agent robust. Therefore, applying modular learning when performing complex tasks has advantages in terms of robustness and performance.

Table 13. Results of experiment #4.

	Min Score	Max Score	Mean Score	Cumulative Successes
Expt. 4-1 (modular learning)	42.39	1882.44	1514.98	1435
Expt. 4-2 (single network learning)	−42.56	1845.80	1130.81	696

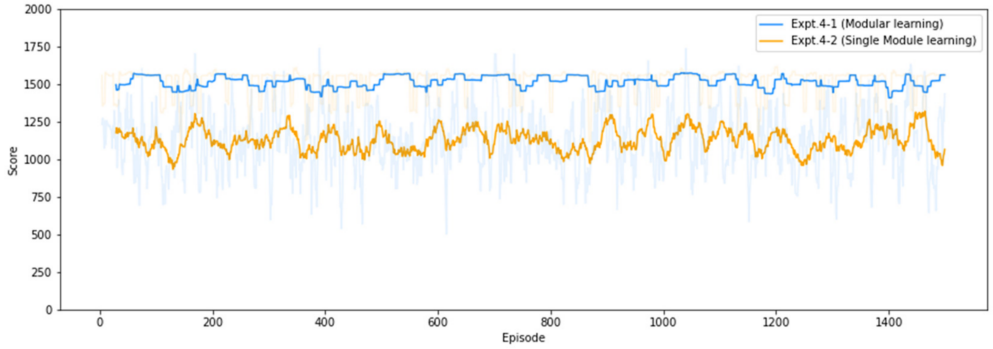
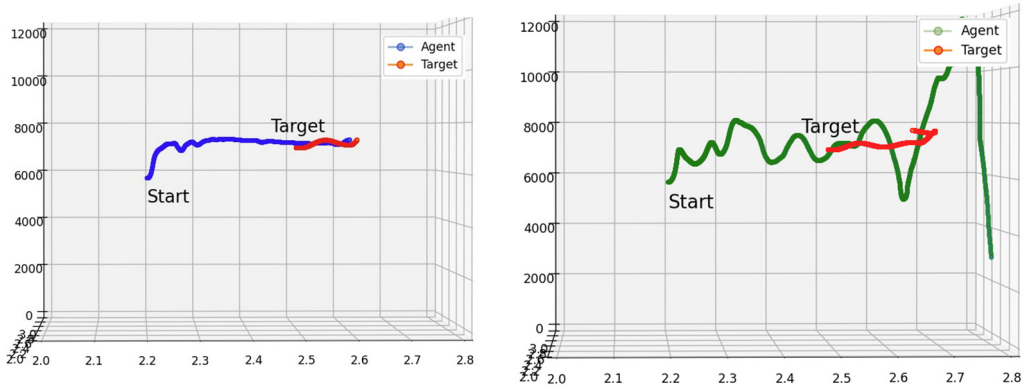


Figure 17. Results of experiment #4 (score plot).



(**a**) (**b**)

Figure 18. A 3D plot of different initial altitudes: (**a**) modular learning and (**b**) single network.

5. Conclusions

In this study, a modular learning method was proposed to improve the efficiency of complex UAV agent tasks in environments with continuous state and action spaces. The proposed method involved training individual task networks and connecting them to perform complex tasks depending on their type and requirements. To evaluate the proposed method, experiments were conducted using a JSBSim 6-DOF aerodynamic simulation for moving targets. The experiments consisted of 1500 episodes, and the results showed that modular learning resulted in high rewards with a relatively small number of learning episodes, thereby verifying the effectiveness of the proposed method. This was a significant improvement over using a single network for all tasks, as it allowed for more efficient learning and better task performance. Moreover, the study indicated that transfer learning between modules can enable the efficient learning of new tasks and UAV types. This means that the knowledge gained from training one module can be transferred to other modules, enabling them to learn faster and with fewer data. This is a key advantage of modular learning over other approaches. The experiments also showed that agents trained with modular learning display better stability and robustness than those trained using a single network and can perform tasks effectively, even when starting at different altitudes. In summary, by applying reinforcement learning and modular learning methods to study autonomous flight control for UAVs performing complex tasks, we show that UAVs can

effectively perform complex tasks in a realistic 6-DOF environment. The implications of this research are important in the field of UAV technology and reinforcement learning. By enabling UAVs to autonomously perform complex tasks, the modular learning method provides new possibilities for use in various applications. The ability to efficiently adapt to different missions and perform intermission transfers demonstrates the potential for UAVs to effectively handle dynamic and changing environments.

However, the study was limited in scope, as only two tasks were connected. In the future, the algorithm will be improved to overcome this limitation, and the number of connected modules will be increased to perform complex tasks more efficiently in more challenging environments. Additionally, the generalizability of the learned policies across different UAV platforms and environments should be explored.

Author Contributions: Conceptualization, J.C.; methodology, J.C.; software, H.M.K. and H.J.H.; validation, J.C., H.M.K. and H.J.H.; formal analysis, J.C.; investigation, Y.-D.K.; resources, Y.-D.K.; data curation, J.C.; writing—original draft preparation, J.C.; writing—review and editing, J.C.; visualization, H.J.H.; supervision, C.O.K.; project administration, C.O.K.; funding acquisition, Y.-D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Agency for Defense Development under Grant UD200015RD.

Data Availability Statement: Not applicable.

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

References

1. Kim, J.; Kim, S.; Ju, C.; Son, H.I. Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications. *IEEE Access* **2019**, *7*, 105100–105115.
2. Eichleay, M.; Evens, E.; Stankevitz, K.; Parker, C. Using the Unmanned Aerial Vehicle Delivery Decision Tool to Consider Transporting Medical Supplies via Drone. *Glob. Health Sci. Pract.* **2019**, *7*, 500–506. [PubMed]
3. Teng, T.H.; Tan, A.H.; Tan, Y.S.; Yeo, A. Self-Organizing Neural Networks for Learning Air Combat Maneuvers. In Proceedings of the International Joint Conference on Neural Networks, Brisbane, Australia, 10–15 June 2012.
4. Ernest, N.; Carroll, D. Genetic Fuzzy Based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions. *J. Def. Manag.* **2016**, *6*, 1000144. [CrossRef]
5. NLR-Netherlands Aerospace Centre. *Rapid Adaptation of Air Combat Behaviour*; Netherlands Aerospace Centre: Amsterdam, The Netherlands, 2016.
6. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone Deep Reinforcement Learning: A Review. *Electronics* **2021**, *10*, 999.
7. Mudrov, M.; Ziuzev, A.; Nesterov, K.; Valtchev, S. Power Electrical Drive Power-Hardware-In-the-Loop System: 2018 X International Conference on Electrical Power Drive Systems (ICEPDS). In Proceedings of the 10th International Conference on Electrical Power Drive Systems (ICEPDS), Novocherkassk, Russia, 3–6 October 2018; ISBN 9781538647134.
8. Chen, H.; Wang, X.M.; Li, Y. A Survey of Autonomous Control for UAV. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence (AICI), Shanghai, China, 7–8 November 2009; Volume 2, pp. 267–271.
9. Rocha, T.A.; Anbalagan, S.; Soriano, M.L.; Chaimowicz, L. Algorithms or Actions? A Study in Large-Scale Reinforcement Learning. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 2717–2723.
10. Yang, Q.; Zhang, J.; Shi, G.; Hu, J.; Wu, Y. Maneuver Decision of UAV in Short-Range Air Combat Based on Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 363–378. [CrossRef]
11. Wang, Z.; Li, H.; Wu, H.; Wu, Z. Improving Maneuver Strategy in Air Combat by Alternate Freeze Games with a Deep Reinforcement Learning Algorithm. *Math. Probl. Eng.* **2020**, *2020*, 7180639. [CrossRef]
12. Lee, G.T.; Kim, C.O. Autonomous Control of Combat Unmanned Aerial Vehicles to Evade Surface-to-Air Missiles Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 226724–226736. [CrossRef]
13. Yan, C.; Xiang, X.; Wang, C. Fixed-Wing UAVs Flocking in Continuous Spaces: A Deep Reinforcement Learning Approach. *Rob. Auton. Syst.* **2020**, *131*, 103594. [CrossRef]
14. Bohn, E.; Coates, E.M.; Moe, S.; Johansen, T.A. Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019, Atlanta, GA, USA, 11–14 June 2019; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2019; pp. 523–533.
15. Tang, C.; Lai, Y.C. Deep Reinforcement Learning Automatic Landing Control of Fixed-Wing Aircraft Using Deep Deterministic Policy Gradient. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems, ICUAS 2020, Athens, Greece, 1–4 September 2020; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2020; pp. 1–9.

16. Dewey, D. Reinforcement Learning and the Reward Engineering Principle. In Proceedings of the Association for the Advancement of Artificial Intelligence, AAAI 2014, Palo Alto, CA, USA, 24–26 March 2014; pp. 13–16.
17. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement Learning for UAV Attitude Control. *ACM Trans. Cyber-Phys. Syst.* **2019**, *3*, 1–21. [CrossRef]
18. Imanberdiyev, N.; Fu, C.; Kayacan, E.; Chen, I.M. Autonomous Navigation of UAV by Using Real-Time Model-Based Reinforcement Learning. In Proceedings of the 2016 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016, Phuket, Thailand, 13–15 November 2016; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2017.
19. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
20. Liu, P.; Ma, Y. A Deep Reinforcement Learning Based Intelligent Decision Method for UCAV Air Combat. In *Communications in Computer and Information Science*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 751, pp. 274–286.
21. Zhang, H.; Huang, C. Maneuver Decision-Making of Deep Learning for UCAV Thorough Azimuth Angles. *IEEE Access* **2020**, *8*, 12976–12987. [CrossRef]
22. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Van Nguyen, L. Reinforcement Learning for Autonomous UAV Navigation Using Function Approximation. In Proceedings of the 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR, Philadelphia, PA, USA, 6–8 August 2018; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2018.
23. Berndt, J.S. JSBSim: An Open Source Flight Dynamics Model in C++. In Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit, Providence, RI, USA, 16–19 August 2004.
24. Pack Kaelbling, L.; Littman, M.L.; Moore, A.W.; Hall, S. Reinforcement Learning: A Survey. *J. Artiicial Intell. Res.* **1996**, *4*, 237–285.
25. Bhatnagar, S.; Sutton, R.S.; Ghavamzadeh, M.; Lee, M. Natural Actor-Critic Algorithms. *Automatica* **2009**, *45*, 2471–2482. [CrossRef]
26. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning (PMLR), Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
27. Transfer, M.; Devin, C.; Darrell, T.; Abbeel, P. Learning modular neural network policies for multi-task and multi-robot transfer. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
28. Zhu, Z.; Lin, K.; Zhou, J. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv* **2020**, arXiv:2009.07888.
29. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum Learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
30. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *J. Mach. Learn. Res.* **2020**, *21*, 7382–7431.
31. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2015.
32. Cereceda, O. A Simplified Manual of the JSBSim Open-Source Software FDM for Fixed-Wing UAV Applications. Memorial University. 2019. Available online: https://research.library.mun.ca/13798/1/Tech_Report_JSBSim.pdf (accessed on 5 June 2023).
33. Pope, A.P.; Ide, J.S.; Micovic, D.; Diaz, H.; Rosenbluth, D.; Ritholtz, L.; Twedt, J.C.; Walker, T.T.; Alcedo, K.; Javorsek, D. Hierarchical Reinforcement Learning for Air-to-Air Combat. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2021; pp. 275–284.
34. Rennie, G. Autonomous Control of Simulated Fixed Wing Aircraft Using Deep Reinforcement Learning. Master's Thesis, The University of Bath, Bath, UK, 2018.
35. Wiering, M.A.; Van Otterlo, M. *Reinforcement Learning: State-of-the-Art*; Springer: Berlin/Heidelberg, Germany, 2012; ISBN 9783642015267.