Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4327380, 15 pages
https://doi.org/10.1155/2022/4327380

Research Article

Federated Reinforcement Learning-Based UAV Swarm System for Aerial Remote Sensing

Woonghee Lee

Department of Applied Artificial Intelligence, Hansung University, Republic of Korea

Correspondence should be addressed to Woonghee Lee; whlee@hansung.ac.kr

Received 10 January 2022; Revised 25 February 2022; Accepted 23 March 2022; Published 29 April 2022

Academic Editor: Mohammad R Khosravi

In recent years, due to the development of technologies for unmanned aerial vehicles (UAVs), also known as drones, UAVs have developed rapidly. Because of UAVs' high mobility and computational capability, UAVs have a wide range of applications in Industrial Internet of Things (IIoT), such as infrastructure inspection, rescue, exploration, and surveillance. To accomplish such missions, it is more proper and efficient to utilize multiple UAVs in a swarm, rather than a single UAV. However, it is difficult for an operator to understand and control numerous UAVs in different situations, so UAVs require the significant level of autonomy. Artificial intelligence (AI) has become the most promising combination with UAVs to ensure the high autonomy of UAVs by establishing swarm intelligence (SI). However, existing learning methods for building SI require continuous information sharing among UAVs, which incurs repeated data exchanges. Thus, such techniques are not suitable for constructing SI in the UAV swarm, in which communication resources are not readily available on unstable UAV networks. To overcome this limitation, in this paper, we propose the federated reinforcement learning- (FRL-) based UAV swarm system for aerial remote sensing. The proposed system applies reinforcement learning (RL) to UAV clusters to establish the SI in the UAV system. Furthermore, by combining federated learning (FL) with RL, the proposed system constructs the more reliable and robust SI for UAV systems. We conducted diverse evaluations, and the results show that the proposed system outperforms the existing centralized RL-based system and is more suited for UAV swarms from a variety of perspectives.

1. Introduction

These days, the performance of the hardware and software needed for computing and artificial intelligence (AI) has become remarkably advanced, so AI is being used in a wide variety of fields including Industrial Internet of Things (IIoT). In particular, the development of deep learning has allowed computers to perform various complex operations previously performed only by humans. Unsupervised learning is used in many areas by developing from supervised learning with tagging data, and reinforcement learning (RL), in which machines learn by themselves, has already surpassed people in many areas. Since the development of deep Q network (DQN) by Google DeepMind [1], RL has been applied to Atari Games in 2015 [2], Go in 2016 [3], and StarCraft II in 2019 [4], and many studies have drawn attention to solving various problems in IIoT.

Unmanned aerial vehicles (UAVs), also known as drones, are useful in that they can be put into difficult environments for people to perform the given missions. Thus, they are used in various applications in IIoT, such as infrastructure inspection, traffic patrol, rescue, exploration, environmental monitoring, remote sensing, and surveillance [5]. To accomplish such missions, UAVs are controlled by radio from a remote controller or are self-judged by a system that has already been designed by an operator. However, it is difficult for the operator to clearly understand the situation in which UAVs exist over long distances and to control the UAVs' behaviors elaborately. In addition, it is impossible to come up with all the countermeasures for various unpredictable situations. Moreover, in recent years, a number of UAVs, rather than a single UAV, are simultaneously utilized in a cluster to perform more diverse missions of IIoT more efficiently, but it is hard to control all of these UAVs in a

centralized manner. Thus, UAVs require the significant level of autonomy and should have the ability to perform tasks in unexpected situations without human intervention.

To ensure the sufficiently high autonomy of UAV, a number of studies were conducted to enable UAV clusters to perform common missions more efficiently and intelligently by utilizing AI algorithms. However, despite a lot of interest in AI, the collaboration with swarm intelligence (SI) in IIoT has not been considered deeply. It is because that it is not easy to satisfy the concept of SI systems in which each object has to decide on an action based on local and partial information obtained from its own environment.

RL is performed by an agent repeating an action based on a state in a given environment and maximizing a reward. Therefore, even for learning with the same goal of a certain application, the results of the learning can be substantially different as the environment changes. In addition, the action is chosen stochastically, so different results can be produced each time even in the same environment. Thus, even if the same learning is performed, it can result in biased results depending on the agent, which increases difficulty in establishing swarm intelligence in IIoT. To overcome this, many studies on multiagent RL have been proposed, and the studies simultaneously utilize multiple agents to perform RL. However, such methods require sharing information of agents, which incurs continuous data exchanges. Thus, it is not easy for them to be applied to the environments such as UAV systems in IIoT, in which communication resources are not readily available on unstable UAV networks.

Federated learning (FL) is a new approach to training machine learning (ML) models that decentralizes the training process, and it was first introduced in the paper published by Google [6]. In FL, each agent receives an initial common global model, which is not trained, from a server, and each agent performs independent learning. After that, the server collects the trained local models, creates a global model, and returns it back to the agents. These operations are repeated to achieve a fully trained global model. By using FL, each agent has an advantage in terms of communication resources in that it does not need to repeatedly share the data required for learning. Fusing FL with RL allows multiple agents to compose the global and unbiased model based on many agents' diverse actions in different environments without exchanging data for learning. Thus, due to these advantages, federated reinforcement learning (FRL) is suited for UAV swarms in IIoT, but only few studies have yet been applied to UAV systems.

Motivated by the fact described above, in this paper, we propose the FRL-based UAV swarm system for aerial remote sensing. To show the application of our proposed system, we take a gas detection as an application example and propose the FRL-based gas sensing system using UAV swarm. However, since the proposed system is not designed to be specialized in specific applications, the system can be applied to any IIoT applications using UAVs.

To summarize the contributions of this paper:

(i) We propose the FRL-based UAV system that outperforms the existing centralized RL-based system

(ii) We establish the swarm intelligence in UAV system by applying RL to UAV clusters

(iii) By combining FL with RL, we construct the more reliable and suitable swarm intelligence for UAV systems

(iv) We conducted diverse performance evaluations considering various factors to analyze the proposed system from a variety of perspectives

The remainder of this paper is organized as follows. In Section 2, we introduce related work and describe our research's novelties and advantages against the related work. We describe preliminary knowledge related to our research in Section 3. After that, in Section 4, we explain our proposed system and give detailed explanations about the learning algorithm and implementation. In Section 5, we describe the experiments and performance evaluation results. Finally, Section 6 concludes this paper with explaining remarks and future directions.

2. Related Work

In this section, we firstly introduce several researches which tried to apply RL or FL to UAV systems. Then, we describe some studies focusing on utilizing FRL for various systems in IIoT. After that, we explain our research's novelties and advantages in comparison with the relevant studies.

Several studies have been conducted that present a variety of techniques using RL to perform path planning tasks or address some of the subtasks. Pham et al. proposed a deep reinforcement learning (DRL) algorithm which enables UAVs to learn their paths autonomously and to pass through changing environments without collisions [7]. Lin et al. proposed a combination of DRL and long short-term memory (LSTM) [8] network that allows UAVs to interact with their surroundings directly and continuously [9]. Lillicrap et al. proposed an improved deep deterministic policy gradient (DDPG) [10] algorithm for object avoidance and target tracking [11]. The proposed algorithm uses reward functions and penalty actions to achieve smoother trajectories. Koch et al. investigated the performance and accuracy of the inner control loop providing attitude control when using autonomous flight control systems trained with various RL algorithms [12].

Using traditional DL-enabled approaches, data needs to be transmitted and stored at a central server. This can be a significant problem because it generates massive network communication overhead to send raw data to centralized entities, which can lead to network usage and energy inefficiency of UAVs. The transferred data can also include personal data such as location and identity of UAVs that can directly affect privacy issues. As a solution, FL was introduced for privacy and low communication overhead. Considering the advantages of FL, FL is much better suited for many UAV-enabled wireless applications in IIoT than the existing DL methods [13], so some researches tried to apply FL to UAV systems in IIoT. Chhikara et al. proposed an FL algorithm within a drone swarm that collects air quality data

using built-in sensors [14]. Using the proposed scheme, a UAV swarm composes the SI to find the area with the highest air quality index value effectively. Awada et al. introduced an FL-based orchestration framework for a federated aerial edge computing system [15]. The authors proposed a federated multioutput linear regression model to estimate multitask resource requirements and execution time to find the optimal drone deployment.

FRL, the combination of FL and RL, is a relatively recently proposed technique, and a few researches tried to apply FRL to applications of IIoT. Lim et al. proposed an FRL architecture to allow multiple RL agents to learn optimal control policy on their own IoT devices of the same type but with slightly different dynamics [16]. Abdel-Aziz et al. proposed a RL-based cooperative perception framework and introduced an FRL approach to speed up the training process across vehicles [17]. Xu et al. proposed a multiagent FL-based incentive mechanism to capture the stationarity approximation and learn the allocation policies efficiently [18]. Xue et al. proposed an FRL framework which extracts the knowledge from electronic medical records across all edge nodes to help clinicians make proper treatment decisions [19].

This paper has novelty and advantages compared to the related studies. As explained before, a few researches utilized FRL for applications of IIoT, but among them, there are few studies that tried to apply FRL to UAV systems. However, in this paper, we propose the FRL-based UAV system for aerial remote sensing. We establish the SI in UAV system by applying RL to UAV clusters. Furthermore, by combining FL with RL, we constructed the more reliable and suitable SI for UAV systems.

3. Preliminary

This section describes preliminary knowledge related to our research. We first explain DRL, and then give a description of FL and FRL.

3.1. Deep Reinforcement Learning. RL is a mathematical framework for experience-driven autonomous learning [20], and the main base of RL is learning through interaction with environments [21]. In RL, the agent observes state, s_t, in the environment at time t. The state is statistics containing the information, such as sensor values and the agent's position, and it is necessary for the agent to select the action. In a given state, the policy returns an action, and the agent takes the selected action. After that, the state transitions to the new state, s_{t+1}, and the agent gets the reward, r_t, from the environment as feedback. The best order of action is determined by the rewards provided by the environment, and the optimal policy is one that maximizes the reward expected in the environment. Thus, using RL algorithms, the agent tries to learn a policy that maximizes expected returns.

DRL was introduced to accelerate the development of RL [22], and DRL uses neural networks to deliver innovative ways to obtain more optimal policy [1]. DL allows RL to deal with intractable decision-making problems in high-dimensional states and environments [2]. There are a variety of DRL algorithms, such as DQN, DDPG, proximal policy optimization (PPO) [23], trust region policy optimization (TRPO) [24], soft actor-critical (SAC) [25], and asynchronous advantage actor-critic (A3C) [26].

3.2. Federated Learning. Without data, model learning cannot be performed. Data often exists in the form of data islands, and the direct solution is to process the data in a centralized manner, requiring training data to be concentrated on the same server. FL shifts the focus of research on ML with data islands. In comparison to centralized learning methods, FL belonging to distributed learning methods allows individual devices in different locations to collaborate with others to learn ML models. The concept of FL was introduced by Google in 2016 and first applied to Google keyboards for joint learning on multiple Android phones [27]. Given that FL can be applied to all edge devices in IoT, there is the potential to revolutionize various IIoT areas, such as healthcare, transportation, and finance [28].

FL offers new research directions on AI in IIoT, and FL provides a new way of learning to build a personalized model without exchanging raw data. With the advancement of computing technologies, the computing resources of IoT devices have become more powerful. Training for AI is also gradually moving from central servers to edge devices. FL provides a privacy mechanism that can effectively use the computing resources of the device to train the model, preventing the leakage of personal information during data transmission. In various areas, numbers of wireless devices exist and there are a large amounts of valuable data, so FL can take full advantage of them. FL is the collection of training information from distributed devices to learn the model, and it includes the following basic steps [29, 30]. Firstly, the server sends the initial model to all of the devices, and then, each device trains its own local model using local data. After that, the devices send local model parameters back to the server, and the model parameters are aggregated into the global model. The aggregated global model is delivered to the devices again, and the above procedures are repeated.

3.3. Federated Reinforcement Learning. The combination of RL and FL was first studied in [31]. Unlike traditional FL, the authors proposed a new FL framework based on RL [2, 20, 32], i.e., FRL. In the study, the authors demonstrated that the FRL approach can take full advantage of the joint observations in the environment and perform better than simple DQNs with partial observations in the same environment. FRL was also applied to autonomous driving, and all participant agents perform steering control actions with knowledge learned by others, even when acting in very different environments [33]. In robot system control, FRL was used to fuse robot agent models and communicate experience effectively using prior knowledge and quickly adapting to new environments [34]. However, there are few studies that applied FRL to UAV systems.

4. System Design and Implementation

In this section, we explain the details of our proposed system and implementation. We first explain the concept of the

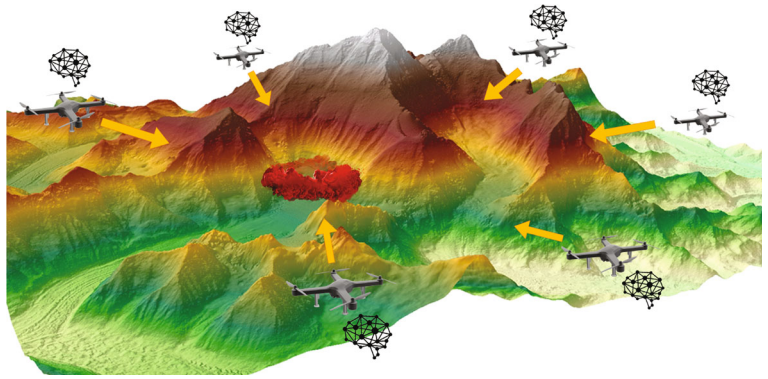


Figure 1: The application concept of the proposed system.

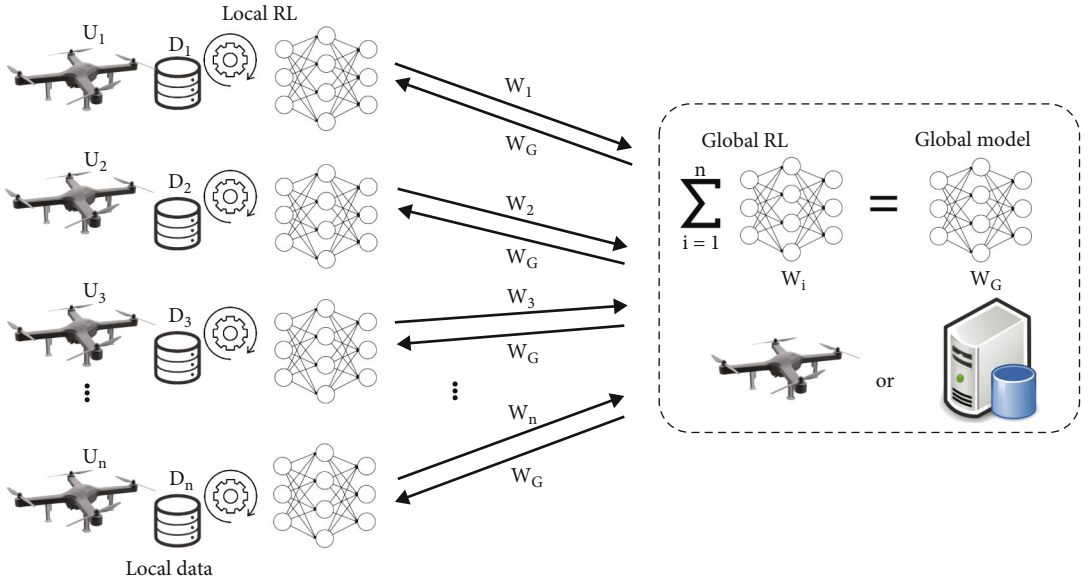


Figure 2: The overall operations of FRL in the proposed system.

proposed system. Then, we give descriptions of our FRL system, the RL algorithm used in the system, and the environment constructed for learning. After that, we describe the system implementation.

4.1. System Concept. We propose the FRL-based UAV swarm system for aerial remote sensing. As we mentioned before, to show the application of our proposed system, we take gas sensing as an application example, and Figure 1 shows the proposed system's application concept. Initially, a UAV swarm consisting of multiple UAVs is arranged in an area where a gas source is expected to exist. In this situation, the mission of the UAV swarm is to find the origin of the gas source, marked as red smoke in the figure, with avoiding collisions not only between UAVs but also with other obstacles, such as tall trees. The UAVs continually move without any predetermined guidance or programmed function. At the same time, they repeatedly perform local learning based on their own actions and data collected from gas sensors and ranging sensors, such as LiDAR or radar. After that, the UAVs share only their locally trained models with each other periodically. During the mission, the UAVs repeat such moving, learning, and occasional sharing to build SI.

4.2. Federated Reinforcement Learning System. In the proposed system, the neural network of UAVs is trained using FRL, and Figure 2 shows the overall learning procedures in the system. To explain the FRL operations in our system, we assumes n UAVs, U_1, …, U_n with their own data D_1, …, D_n. The proposed FRL scheme includes the following main steps. First, a server (a ground control system) or a header UAV in our system sends initial global models to all UAVs, and each UAV trains their own local model using local information including states, actions, and rewards. We will describe the detailed explanation about the learning algorithm in Section 4.3. The UAVs send the local model parameters, W_1, …, W_n, back to the server, and then, the server aggregates the model parameters into the global model as follows:

$$W_G = \sum_{n=1}^{n} W_i. \tag{1}$$

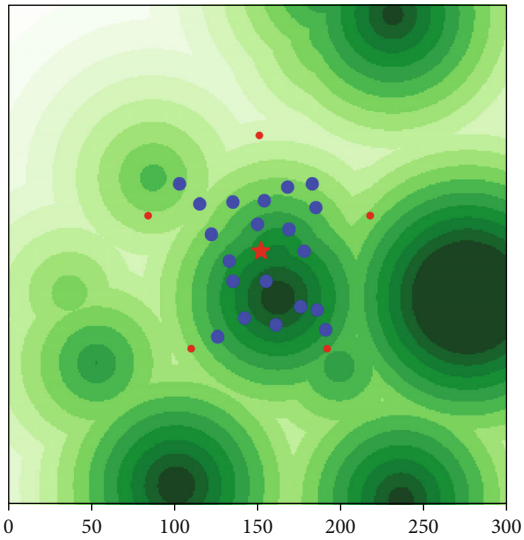


Figure 3: An example of map used for the proposed FRL system.

The global model's parameters, W_G, are distributed back to the UAVs and the above procedures are repeated until the global model is sufficiently trained.

4.3. Reinforcement Learning Algorithm. This subsection describes the RL algorithm used in our proposed system. The PPO algorithm is based on the actor-critic concept and utilizes two separate networks [23]. The actor network determines the agent's optimal behavior, whereas the critic network evaluates policies and trains the actor using rewards. The PPO algorithm was inspired by the TRPO algorithm [24], and the PPO algorithm provides a more direct approach to implementing and coordinating tasks for learning. Compared to TRPO, PPO is also known to provide simpler and better performance in many applications in IIoT [35]. The UAV system prefers algorithms requiring a small amount of computation, so PPO is suitable for various tasks performed by UAVs [5]. In fact, many studies used PPO as the RL algorithm for UAV systems, and many results have shown that PPO is superior to other algorithms in various aspects [5]. For these reasons, we chose PPO as the RL algorithm of our FRL system.

We describe the detailed explanation about the learning algorithm for the proposed system with reference to [16, 23, 34]. In training, an agent observes a state, s_t, in the environment at time step t. The actor model, π_θ, with its model parameters, θ, is used to determine an action, a_t, to be taken in the given state, s_t. The agent takes the selected action, the state transitions to the new state, s_{t+1}, and the agent gets the reward, r_{t+1}. For every time step, the agent stores the trajectory segment, $<s_t, \cdot a_t, \cdot r_{t+1}, \cdot s_{t+1}>$ in the trajectory memory. The critic model, V_μ, with its model parameters, μ, evaluates whether the action led the agent to a better state, and the critic model's feedback is used to optimize the actor model. Whenever a determined number of steps proceed, based on the PPO algorithm, the gradients for the optimization of the actor and critic models are calculated using the

Input: sensing information, distance information
Output: state
1: state = **zeros**(n_{state})
Calculating state values regarding sensing:
2: $s_{\text{sum}} \longleftarrow 0$
3: **for** each sensing value, s, in sensing value set, S **do.**
4: $s_{\text{sum}} \longleftarrow s_{\text{sum}} + s$
5: **end for.**
6: $s_{\text{average}} \longleftarrow s_{\text{sum}}/n_{\text{sensor.}}$
7: $s_{\text{max}} \longleftarrow 0$
8: **for** i, s in **enumerate**(S) **do**
9: state[i] $\longleftarrow s - s_{\text{average}}$
10: **if** $s_{\text{max}} <$ **abs**(state[i]) **then** $s_{\text{max}} \longleftarrow$ **abs**(state[i])
11: **end for**
12: **for** i in range(n_{sensor}) **do**
13: state[i] $\longleftarrow$ state[i]/s_{max}
Calculating state values regarding distance:
15: $o \longleftarrow$ **NearestObj** (O)
16: dist $\longleftarrow$ **CalDist** (o)
17: **if** dist $\leq$ size$_{\text{uav}}$ **then** state[−4] $\longleftarrow -1$
18: **else** state[−4] $\longleftarrow 1$
19: $\vec{o} \longleftarrow$ **CalVec** (o)
20: state[−3] $\longleftarrow \vec{o}_x$
21: state[−2] $\longleftarrow \vec{o}_y$
22: state[−1] $\longleftarrow \vec{o}_z$
23: **return** state

Algorithm 1: Algorithm for getting the state.

Table 1: Variables used for getting the state.

Notation	Description
state	Set of state values
n_{state}	Number of state values in state
S	Set of sensing values
s	Each sensing value in S
s_{sum}	Sum of sensing values in S
s_{average}	Average of sensing values in S
s_{max}	The maximum of the absolute values of S
n_{sensor}	Number of sensors attached onto the UAV
O	Set of nearby objects
o	The nearest object
dist	The distance to o
$\vec{o}$	Normalized vector to o

trajectory segments in the trajectory memory. The objective function, L^{PG}, in a general policy gradient RL is as follows:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t|s_t)\hat{A}_t\right], \tag{2}$$

where $\hat{\mathbb{E}}_t[\cdots]$ means the empirical average over a finite batch of samples and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Utilizing the generalized advantage

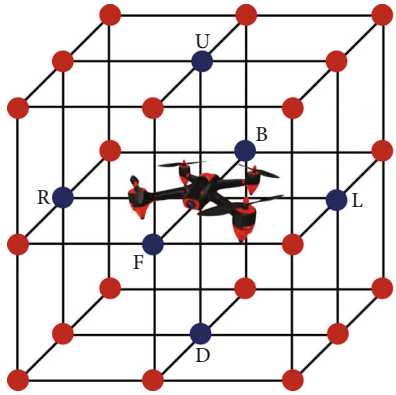


Figure 4: The sensors' position and possible movement actions of UAV in the proposed FRL.

Input: action, sensing information, distance information
Output: reward
reward ⟵ 0
Detecting a collision with any other objects:
o ⟵ **NearestObj** (O)
if $o \neq t$ **then**
  dist_{obj} ⟵ **CalDist** (o) **then**
  if $\text{dist}_{\text{obj}} \leq \text{size}_{\text{uav}}$ **then**
    reward ⟵ −2
    return reward
  end if
end if
When the agent reaches the target:
if $o == t$ **then**
  dist_t ⟵ **CalDist** (t)
  if $\text{dist}_t \leq \text{th}_{\text{succ}}$ **then**
    if action == 'staying' **then**
      reward ⟵ 1
      return reward
    end if
  end if
end if
Calculating the reward in the other cases:
$\vec{t}_x$ ⟵ $s_{\text{right}} - s_{\text{left}}$
$\vec{t}_y$ ⟵ $s_{\text{front}} - s_{\text{back}}$
$\vec{t}_z$ ⟵ $s_{\text{up}} - s_{\text{down}}$
$\vec{t}$ ⟵ $v_t / \|v_t\|$
$\vec{a}$ ⟵ **NorVec**(action)
reward ⟵ **InnerProd**$(\vec{d}, \vec{a})$ abs(reward) < val_{clip}
if reward ≥ 0 **then**
  if reward ⟵ val_{clip}
  else reward ⟵ $-\text{val}_{\text{clip}}$
end if
return reward

Algorithm 2: Algorithm for determining the reward value.

Table 2: Variables used for determining the reward.

Notation	Description
O	Set of nearby objects
o	Nearest object
t	Target object
dist_{obj}	Distance to o
size_{uav}	Radius size of UAV
dist_t	Distance to the target
th_{succ}	Threshold of distance to the target where the agent is deemed to arrive at the target
$\vec{t}$	Normalized vector to target
$\vec{a}$	Normalized vector of action
val_{clip}	Clip value for determining reward

Table 3: Hyperparameters and values used for learning.

Hyperparameter	Value
Actor network dimension	16∗256∗256∗256∗5
Critic network dimension	16∗256∗256∗256∗5
Minibatch size	5
Number of epochs	4
Learning rate	0.0003
Horizon value	20
Generalized advantage estimator	0.95
Discount factor gamma	0.99
Clipping parameter	0.2
Value function coefficient	0.5
Optimizer algorithm	Adam

estimator (GAE) [36], $\hat{A}_t$ can be calculated as follows:

$$\hat{A}_t = \delta_t^V + (\gamma\lambda)\delta_{t+1}^V + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}^V, \tag{3}$$

where γ is the discount factor $(\gamma \in [0, 1])$, λ is the GAE parameter $(\lambda \in [0, 1])$, T is the size of mini-batch samples, and $\delta_t^V = r_t + \gamma V_\mu(s_{t+1}) - V_\mu(s_t)$. The objective function, L^V, is as follows:

$$L^V(\mu) = \hat{\mathbb{E}}_t\left[\left|\hat{V}_\mu^{\text{target}}(s_t) - V_\mu(s_t)\right|\right], \tag{4}$$

where $\hat{V}_\mu^{\text{target}}$ is the target value of time-difference error (TD-error), and $\hat{V}_\mu^{\text{target}}(s_t) = r_{t+1} + \gamma V_\mu(s_{t+1})$. Using a stochastic gradient descent (SGD) algorithm (i.e., Adam optimization [37]), the parameters of V_μ are updated as follows:

$$\mu = \mu - \eta_\mu \nabla L^V(\mu), \tag{5}$$

where η_μ is the learning rate for the critic model optimization.

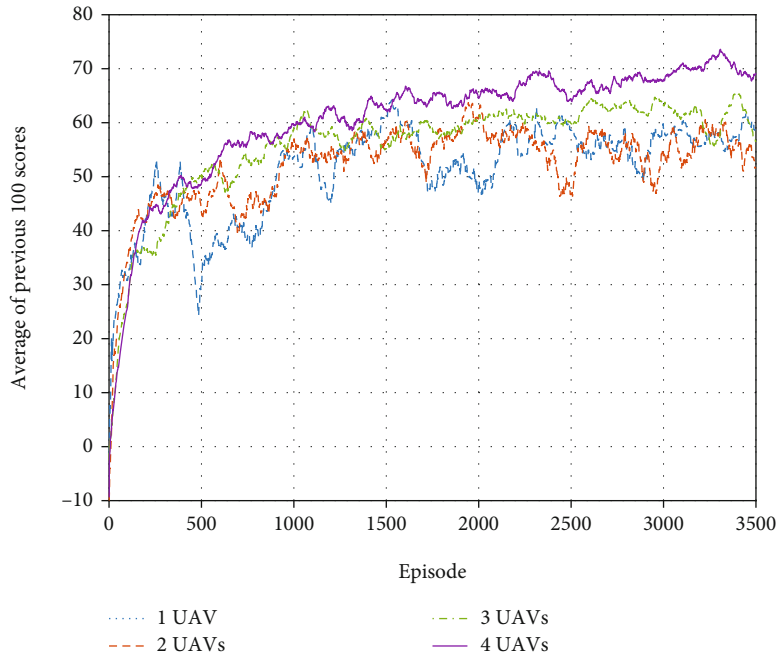


Figure 5: The average of the score values as the episode goes by.

In the actor model of TRPO, the importance sampling is used to obtain the expectation of samples gathered from the old policy, $\pi_{\theta_{\text{old}}}$, under the new policy, π_θ. The TRPO algorithm maximizes the surrogate objective function, L^{CPI}, presented in

$$L^{\text{CPI}}(\theta) = \hat{\mathrm{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t = \hat{\mathrm{E}}_t\left[R_t(\theta)\hat{A}_t\right]\right] \quad (6)$$

where CPI refers to conservative policy iteration [38] and $R_t(\theta)$ denotes the probability ratio. The TRPO algorithm optimizes L^{CPI} subject to the constraint on the amount of the policy update as follows:

$$\hat{\mathrm{E}}_t\left[KL\left[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right]\right] \le \delta, \quad (7)$$

where KL refers to the Kullback-Leibler divergence [39]. As we explained before, the PPO algorithm was inspired by the TRPO algorithm, and the objective function of PPO, L^{CLIP}, is as follows:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(R_t(\theta), \text{clip}(R_t(\theta), 1-\varepsilon, 1+\varepsilon)\right)\hat{A}_t\right], \quad (8)$$

where ε is the clipping parameter. The parameters of π_θ are updated by the SGD algorithm with the gradient, ∇L^{CLIP}, as follows:

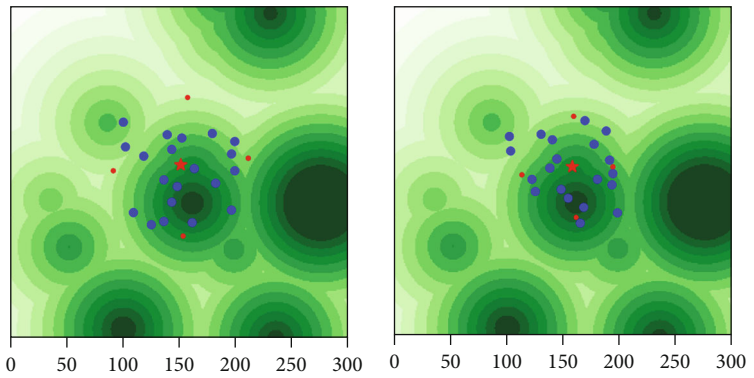
$$\theta = \theta - \eta_\theta \nabla L^{\text{CLIP}}(\theta), \quad (9)$$

where η_θ is the learning rate for the actor model optimization.

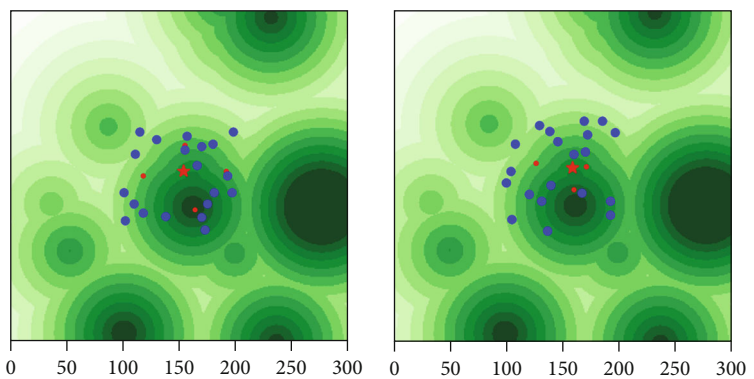
Using the above algorithm, each agent in our system performs RL repeatedly, and the agents send the updated model parameters to the server periodically as we explained in Section 4.2.

4.4. Environment. The agents continually interact with the environment while performing learning, so it is important to construct an appropriate environment for proper learning. We constructed the environment for agents to perform learning well to accomplish the mission described in Section 4.1. This subsection provides a detailed description of the environment, especially about map, state, action, and reward.

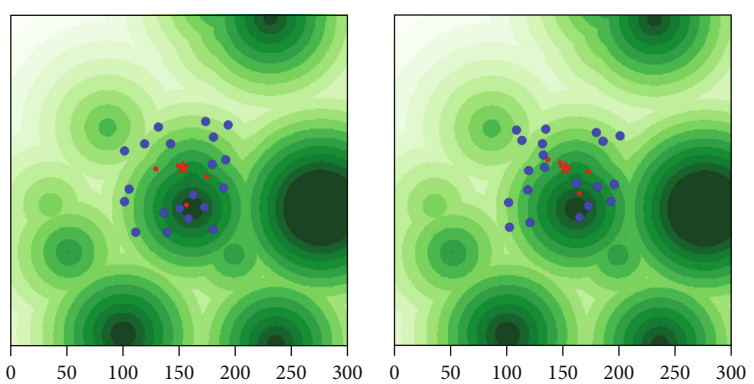
4.4.1. Map. Figure 3 shows an example of map which is used for FRL of the proposed system. In the map, green circle lines mean contour lines. In other words, an area marked as darker green means a higher area. Red and blue dots represent UAVs and obstacles, respectively. The red star in the middle means the gas source that the UAVs should find. We set the map to change every a certain period so that the UAVs can experience various environments. At each change, both the position of the obstacles and the height of the terrain change. The UAVs are initially placed evenly between UAVs outside a certain range from the gas source since it is efficient and reasonable to spread them as much as possible. The obstacle is assumed to be a very tall object, such as a transmission tower, so that the UAVs cannot avoid the obstacle by flying higher but should move horizontally to avoid the obstacle. Considering collisions not only with obstacles but also between UAVs, if a UAV gets closer to another object than a certain distance, it is considered as a collision.



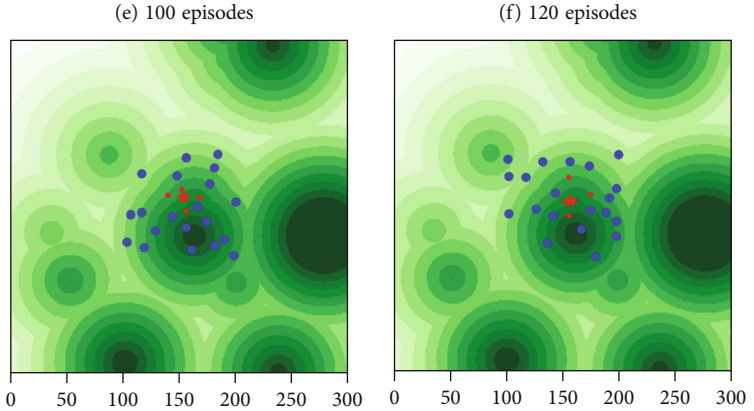
(a) 20 episodes (b) 40 episodes



(c) 60 episodes (d) 80 episodes



(e) 100 episodes (f) 120 episodes



(g) 140 episodes (h) 160 episodes

Figure 6: Final position of UAVs as the episode goes by.

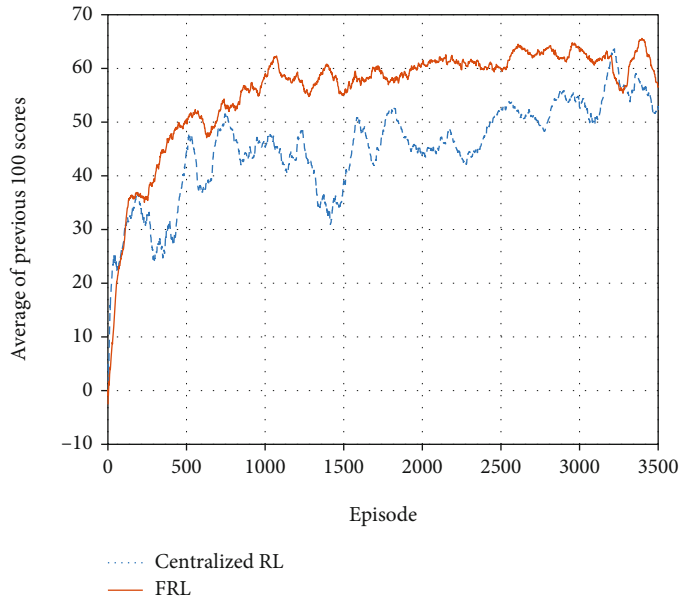


Figure 7: The learning performance comparison between centralized RL- and FRL-based systems.

4.4.2. State. In order for an agent to take an appropriate action, the state should consist of appropriate values. Algorithm 1 shows the pseudocode for getting the state, and Table 1 lists the variables used in Algorithm 1. The state is composed of two value sets, one set regarding sensing values and the other set containing distance information, so the algorithm for obtaining state is also composed of two parts.

Lines 2 to 14 in Algorithm 1 are relevant to calculating state values regarding sensing. The UAV has multiple sensors, gas sensors in our application example scenario, and blue dots in Figure 4 present the position of sensors attached onto the UAV. Each sensor continuously collects sensor data. In real-world environments, there is always noise in sensor values obtained from real sensors. Therefore, in order to consider noise in a real environment, we added different Gaussian noise to sensor values. We will give the detailed explanations about the noise values and the performance evaluation considering the sensor noise in Section 5.3. Using the sensor data, the agent finds the sum of the collected values and calculates the average of them. After that, the agent subtracts the mean value from each sensor value, and in this process, the agent finds and memorizes the maximum absolute value of the result values. The agent performs normalization using this maximum value, and the agent takes these final results as values of the state's first set.

Lines 15 to 22 in Algorithm 1 are relevant to the state's second value set, state values regarding distance information. First, the agent finds the nearest object, a UAV or an obstacle, from the agent, and then calculates the distance to the object. If the distance is smaller than the size of the UAV, there is a collision, so -1 is stored in the state, and if not, 1 is stored. After that, the agent calculates the normalized vector directed towards the nearest object, and the values of x, y, and z axes of the vector are stored in the state.

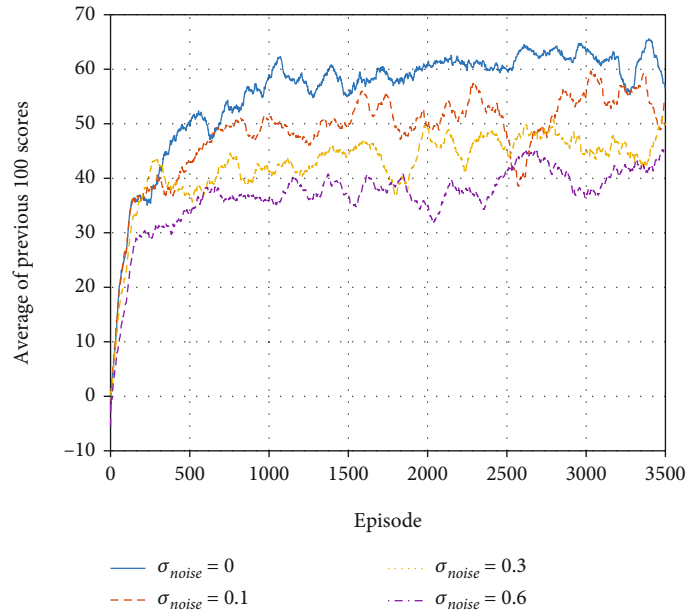
4.4.3. Action. Figure 4 shows the movement actions that the UAV can choose. UAVs in real world can move in more diverse directions, but in order to reduce the complexity of learning, we assumed that UAVs can perform only 27 actions, moving in 26 directions and staying. Red and blue dots in the figure indicate the 26 directions, and blue dots also show the position of sensors attached onto the UAV as explained before.

4.4.4. Reward. An appropriate reward should be given for an agent to perform well in learning. Algorithm 2 shows the detailed process of determining the reward value, and Table 2 lists the variables used in Algorithm 2.

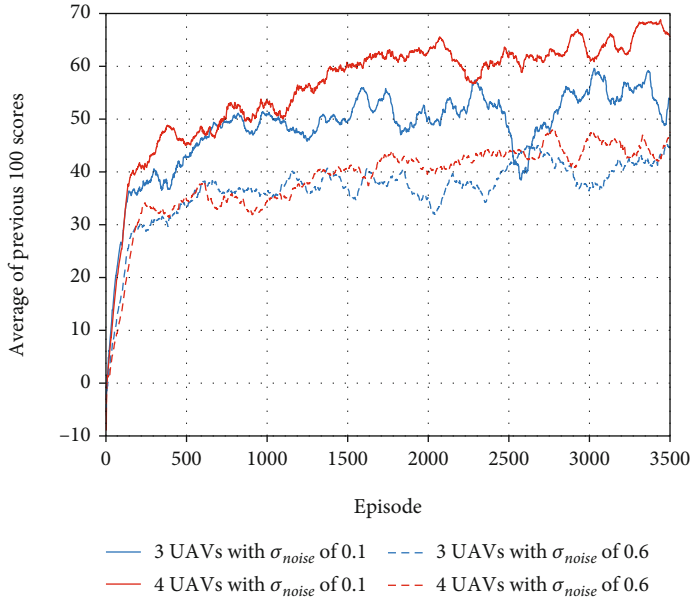
The UAV should not collide with other UAVs or obstacles while moving. Lines 2 to 9 in Algorithm 2 are relevant to detecting a collision with any other objects. First, the agent finds the nearest object among nearby objects. If the nearest object is not the target, the agent calculates the distance to the object. If the distance is less than the radius of UAV, in other words, if a collision occurs, the reward is set to -2 to train the agent not to do such action causing the collision in the future.

If the agent arrives at the target, it is reasonable for the agent to be located there without moving, and lines 10 to 18 in Algorithm 2 are relevant to this case. Firstly, the agent calculates the distance to the target. When the distance is shorter than the determined distance for judging whether the agent arrives at the target, if the agent takes the action of staying there, the agent gains 1 as compensation.

In the other cases, the agent calculates the reward, and lines 19 to 28 in Algorithm 2 are relevant to these cases. The principle of determining the reward is that the better the agent moves in the direction of the target, the larger the reward the agent receives. The shorter the distance



(a) The evaluation on learning performance with different noise



(b) The impact of packet loss depending on the number of UAVs participating in learning

Figure 8: The performance evaluation considering sensor noise.

between the sensor and the target is, the larger or smaller the sensing value is, depending on the characteristics of sensors. In the case of gas sensor, the shorter the distance, the larger the sensing value [40]. Therefore, a larger sensing value means that the sensor is closer to the target. The agent obtains a normalized vector, on x, y, and z axes, directed toward the target using values of sensors marked with blue circles in Figure 4. After that, the agent calculates the normalized vector for the action and obtains the inner product of the two vectors. If the absolute value of the reward is too small, learning may not be performed well, so the reward is adjusted based on the clipping value.

4.5. Implementation. As explained in Section 4.3, we used PPO as the RL algorithm, and we implemented the RL model of the proposed system by using the PyTorch library [41] with reference to [42]. Table 3 shows hyperparameters used in the algorithm. By adding FL to the RL model, we constructed the FRL model with reference to [43]. We implemented the FRL system on Ubuntu 20.04 LTS using a desktop with AMD Ryzen™ 7 5800X and 32 GB RAM. For faster learning, we trained the learning model by using NVIDIA's compute unified device architecture (CUDA) on the NVIDIA GeForce RTX 3070 8 GB GDDR6 PCI Express 4.0 graphic card. In addition, we constructed a map,

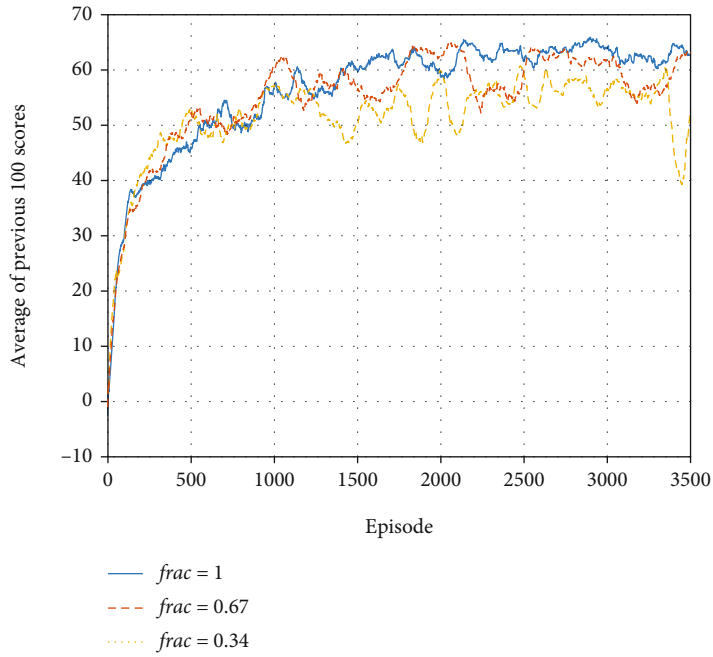


Figure 9: The learning performance depending on the participation ratio.

explained in Section 4.4.1, referring to the 2D Gaussian grid map introduced in [44].

5. Performance Evaluation

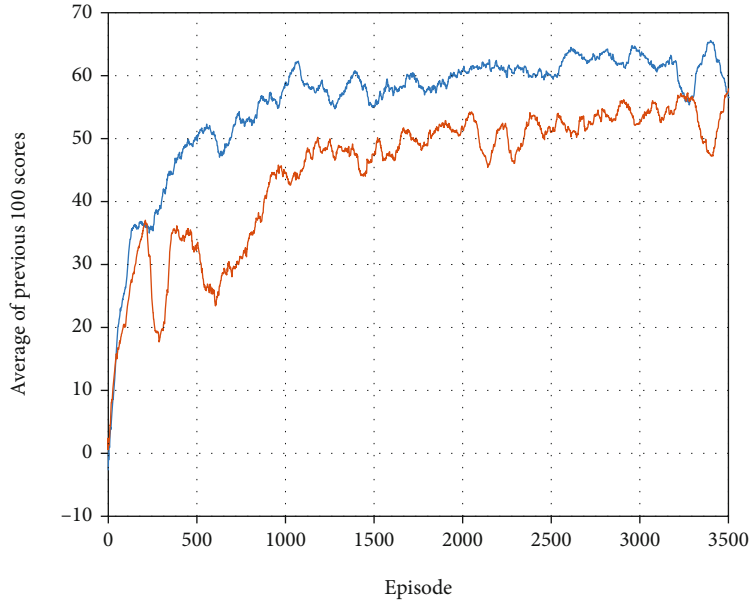
In this section, we explain the various experiments and evaluation results. We first explain the performance evaluation of the proposed FRL system and then show the result of performance comparison between RL- and FRL-based systems. After that, we describe diverse evaluations considering various factors, such as sensor noise, participation ratio, packet loss, and duplication sending.

5.1. Evaluation on Learning Performance. An episode is a unit of learning, and each episode ends after a determined number of steps proceed. To evaluate the learning performance, we recorded the sum of the reward values gained by the agent in the episode as the score of the episode, and we investigated the sum of scores from the last 100 episodes. We conducted the evaluation by varying the number of agents, and the four lines with different colors in Figure 5 show the results. As shown in the figure, the average of the score values increases as the episode goes by, which means that the agent performed the mission well as the learning was repeated. The average value continues to increase up to about 3000 episodes and reaches the saturation point. In terms of the number of agents, the result shows that the more agents participate in learning, the better the learning performance is. In other words, the average score increases higher and the range of fluctuation is smaller in cases where the more agents participate in learning. This is because the more UAVs learn together, the more diverse experiences are collected, which not only makes learning better but also causes unbiased learning to be performed. However, it is not easy for many UAVs to continuously send raw data to centralized entities, which can lead to massive communication overhead and energy inefficiency of UAV systems. Thus, our FRL-based system is suited for UAV swarms because FRL has an advantage in terms of communication resources in that it does not need to repeatedly share the raw data for learning.

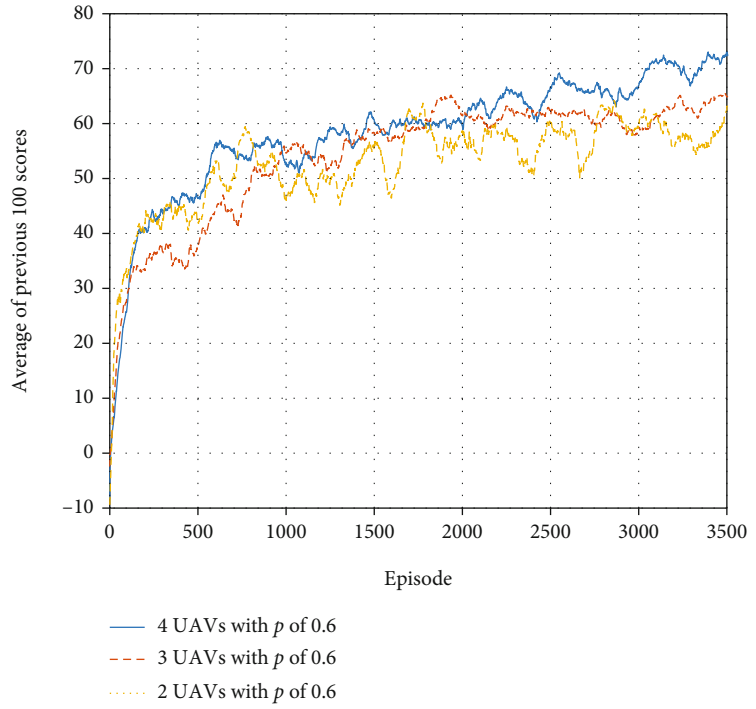
As shown in Figure 5, the learning progresses rapidly in the early stage. To analyze this in more detail, Figure 6 shows the final positions of UAVs every 20 episodes. In the figure, after only 20 episodes, in other words, when the sufficient learning was not performed, the UAVs could not find the target. However, as the episode went by, the more UAVs moved closer to the target, which means that the learning was performed well.

5.2. Performance Comparison between RL- and FRL-Based Systems. In existing RL approaches, it is common to collect data and perform learning in a centralized manner. In UAV systems, it is not easy to continuously send all raw data to the central entity in real time, so the learning can be performed by transferring data to the server after the flight of all UAVs is over. We compared the results of learnings performed using such centralized RL-based method and our FRL-based method. As shown in Figure 7, the FRL-based method performed learning better and reached the saturation point faster than the centralized RL-based method. The reason for this result is that the FRL-based method does not require raw data transmission so that learnings can be performed more frequently, resulting that agents can be trained faster and more stably.

5.3. Learning Performance considering Noise. As explained in Section 4.4.2, there is always noise in sensor values obtained from real sensors. Therefore, to evaluate the performance



(a) Learning performance in difference communication conditions



(b) The impact of packet loss depending on the number of UAVs participating in learning

Figure 10: The performance evaluation considering packet loss.

considering noise, we analyzed the learning performance by adding a different Gaussian noise of $\mathcal{N}(\mu, \sigma^2)$ to sensor values. We performed FRL with 3 agents by using the zero mean and different variance values from 0 to 0.6 with reference to the values obtained from real gas sensors [40]. As shown in Figure 8(a), the higher the noise, the lower the learning performance. However, even when there was noise, a certain level of learning was sufficiently performed. Thus, this result shows that the proposed FRL system can be utilized in a real environment with noise.

As shown in the result above, the noise degrades the learning performance. However, as the number of UAVs increases, the more experience the UAVs have and share, which mitigates the degradation caused by noise. As shown

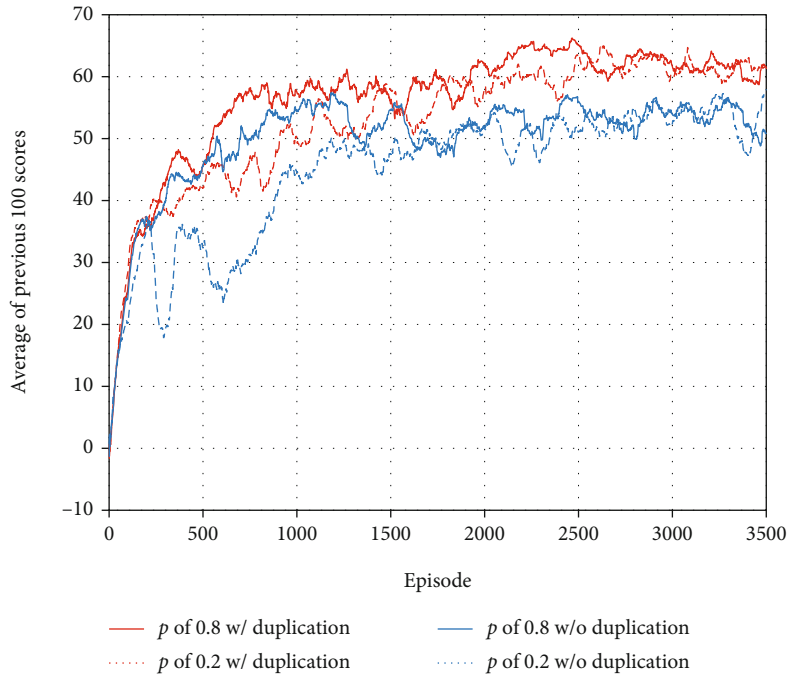


Figure 11: The learning performance depending on the use of the duplication sending technique.

in Figure 8(b), when there was little noise, the more UAVs participated in learning together, the better the learning performance. Similarly, even when there was severe noise, the fluctuation was smaller when more UAVs participated in learning although the overall learning performance was relatively low. In summary, using the FRL-based system, the more UAVs participate in building SI, the impact of noise can be alleviated as well as the overall performance can be improved.

5.4. Performance Evaluation considering Participation Ratio. In real situations, all devices may not be always able to participate in learning on all rounds due to diverse causes, such as the situation of devices, communication, and network problems. Therefore, in the actual FL, the ratio of devices participating in learning is determined, some of the devices are chosen every round according to the participation ratio, and the selected devices participate in learning. Thus, we evaluated the performance by changing the participation ratio in the learning, and Figure 9 shows the result. Naturally, the higher the participation ratio, the more stable and better performance, but for this to occur, many devices should participate in the learning of every round. As shown in the figure, even when 0.67 was selected as the participation ratio, there is the little degradation in performance compared to the case with the participation ratio of 1. Thus, this result shows that it is possible to obtain not only efficient learning but also acceptable performance by using the proper participation ratio.

5.5. Performance Evaluation considering Packet Loss. In UAV systems, due to the high mobility of UAV and continuous changes in network topology, wireless data communications are frequently unstable, which can lead to packet loss. When packet loss occurs or communication situation is poor, some of trained local models cannot be transferred. In consideration of this situation, we evaluated the performance by changing the packet loss probability, and Figure 10 shows the result. Figure 10(a) shows learning performance in cases where there was no packet loss in a stable communication situation and where a lot of packet losses occurred due to poor network condition. In the case of severe network condition, since the packet loss occurred frequently, the trained models could not be transferred well, so learning was performed unstably at the beginning of learning. However, as shown in Figure 10(b), the learning performance can be improved if more agents participate in learning even when the communication situation is unstable. In conclusion, if FRL is utilized in a UAV system composed of a number of UAVs, it is possible to perform learning even in poor communication situations.

5.6. Performance Evaluation considering Duplication Sending. These days, it is not difficult for UAVs to transmit packets through multiple paths by leveraging multiple interfaces simultaneously. In our previous work [45], to improve the reliability and stability of controlling UAVs, we proposed a scheme that selectively duplicates only important packets and then transfers the originals and copies of them through different paths. Such technique and other similar ones can increase the success rate of transmitting trained models, which in turn improves learning performance. Figure 11 shows the results of learning performance depending on the use of the technique when the packet transmission probability is 0.8 or 0.2. As shown in the result, we can get better learning performance when using the duplication sending technique. This means that the reliable communication and network in UAV systems are

critical for improving the FRL system's learning performance to build SI.

6. Conclusion

Nowadays, UAVs are widely used in various fields of IIoT due to the many advantages of the UAVs. In order to carry out today's complicated and complex missions, it is more appropriate and efficient to use multiple UAVs together, so many people utilize UAVs in the form of swarm. However, it is not easy to control multiple UAVs from a distance at the same time. Thus, UAVs are required to have the high autonomy, and AI is the most promising technique to provide the intelligence to UAVs. However, to secure SI using existing techniques, raw data should be continuously exchanged between UAVs, which is not suitable for UAV systems operating on unstable networks. Motivated by the fact described above, in this paper, we proposed the novel FRL-based UAV swarm system for aerial remote sensing. The proposed system utilizes RL to ensure the high autonomy of UAVs, and moreover, the system combines FL with RL to construct the more reliable and robust SI for UAV swarms. Through the performance evaluations, we showed that the proposed system outperformed the existing centralized RL-based system. Furthermore, we conducted various analyses considering the diverse factors, such as sensor noise, participation ratio, packet loss, and duplication sending, and the results proved that our proposed system is more suited for UAV swarms from a variety of perspectives.

We have several directions as future work. We will implement our FRL algorithm on UAV devices and apply the proposed system to UAV systems in a real environment. In order to do this, we will construct the more complex state and devise the more sophisticated reward algorithm. In addition, we plan to elaborate our system to include additional techniques, such as more efficient model exchange and adaptive participation ratio, which results in the better SI development.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon reasonable request and with permission of funders.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1G1A1092939).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, arXiv preprint arXiv: 1312.5602.

[2] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[5] A. T. Azar, A. Koubaa, N. Ali Mohamed et al., "Drone deep reinforcement learning: a review," *Electronics*, vol. 10, no. 9, 2021.

[6] B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics. PMLR*, pp. 1273–1282, 2017.

[7] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous uav navigation using reinforcement learning," 2018, arXiv preprint arXiv: 1801.05086.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Y. Lin, M. Wang, X. Zhou, G. Ding, and S. Mao, "Dynamic spectrum interaction of UAV flight formation communication with priority: a deep reinforcement learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 3, pp. 892–903, 2020.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, arXiv preprint arXiv: 1509.02971.

[11] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.

[12] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[13] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53841–53849, 2020.

[14] P. Chhikara, R. Tekchandani, N. Kumar, M. Guizani, and M. M. Hassan, "Federated learning and autonomous UAVs for hazardous zone detection and AQI prediction in IoT environment," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15456–15467, 2021.

[15] U. Awada, J. Zhang, S. Chen, and S. Li, *Air-to-Air Collaborative Learning: A Multi-Task Orchestration in Federated Aerial Computing*, 2021.

[16] H. K. Lim, J. B. Kim, J. S. Heo, and Y. H. Han, "Federated reinforcement learning for training control policies on multiple IoT devices," *Sensors*, vol. 20, no. 5, p. 1359, 2020.

[17] M. K. Abdel-Aziz, C. Perfecto, S. Samarakoon, M. Bennis, and W. Saad, "Vehicular cooperative perception through action branching and federated reinforcement learning," arXiv preprint arXiv: 2012.03414 2020.

[18] M. Xu, J. Peng, B. Gupta et al., "Multi-agent federated reinforcement learning for secure incentive mechanism in

intelligent cyber-physical systems," *IEEE Internet of Things Journal*, 2021.

[19] Z. Xue, P. Zhou, Z. Xu et al., "A resource-constrained and privacy-preserving edge-computing-enabled clinical decision system: a federated reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9122–9138, 2021.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, 2018.

[21] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, arXiv Preprint arXiv: 1708.05866.

[22] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *Journal of Machine Learning Research*, vol. 10, 2009.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv preprint arXiv: 1707.06347.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[26] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a GPU," 2016, arXiv preprint arXiv: 1611.06256.

[27] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, "Federated optimization: distributed machine learning for on-device intelligence," 2016, arXiv preprint arXiv: 1610.02527.

[28] P. M. F. Mammen, "Learning: opportunities and challenges," 2021, arXiv Preprint arXiv: 2101.05428.

[29] H. B. McMahan, E. Moore, and D. Ramage, "Federated learning of deep networks using model averaging," 2016, arXiv preprint arXiv: 1602.05629.

[30] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, article 106775, 2021.

[31] H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," 2019, arXiv preprint arXiv: 1901.08277.

[32] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent trajectory autoencoder: hierarchical reinforcement learning with trajectory embeddings," in *International Conference on Machine Learning*, pp. 1009–1018, PMLR, 2018.

[33] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated transfer reinforcement learning for autonomous driving," 2019, arXiv preprint arXiv: 1910.06001.

[34] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[35] M. Chen, H. K. Lam, Q. Shi, and B. Xiao, "Reinforcement learning-based control of nonlinear systems using Lyapunov stability concept and fuzzy reward scheme," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, pp. 2059–2063, 2019.

[36] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, arXiv preprint arXiv: 1506.02438.

[37] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, arXiv preprint arXiv: 1412.6980.

[38] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *In Proc. 19th International Conference on Machine Learning*, Citeseer, 2002.

[39] S. Kullback, *Information Theory and Statistics*, Courier Corporation, 1997.

[40] S. Lee, S. Park, and H. Kim, "Gas detection-based swarming: deterministic approach and deep reinforcement learning approach," *The Journal of Supercomputing in submission.*.

[41] A. Paszke, S. Gross, F. Massa et al., "Pytorch: an imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[42] P. Tabor, *Youtube-Code-Repository*, 2020, https://github.com/philtabor/Youtube-Code-Repository/tree/master/ReinforcementLearning/PolicyGradient/PPO/torch.

[43] A. R. Jadhav, *Federated-Learning (PyTorch)*, 2021, https://github.com/AshwinRJ/Federated-Learning-PyTorch.

[44] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "PythonRobotics: a Python code collection of robotics algorithms," 2018, arXiv preprint arXiv:1808.10703.

[45] W. Lee, J. Y. Lee, H. Joo, and H. Kim, "An MPTCP-based transmission scheme for improving the control stability of unmanned aerial vehicles," *Sensors*, vol. 21, no. 8, p. 2791, 2021.