energies

MDPI

Article

Multiagent Reinforcement Learning Based on Fusion-Multiactor-Attention-Critic for Multiple-Unmanned-Aerial-Vehicle Navigation Control

Sangwoo Jeon [1], Hoeun Lee [1], Vishnu Kumar Kaliappan [2,*], Tuan Anh Nguyen [2], Hyungeun Jo [1], Hyeonseo Cho [1] and Dugki Min [1,*]

[1] Department of Computer Science and Engineering, Konkuk University, Seoul 05029, Korea
[2] Konkuk Aerospace Design-Airworthiness Research Institute, Konkuk University, Seoul 05029, Korea
* Correspondence: vishnudms@gmail.com (V.K.K.); dkmin@konkuk.ac.kr (D.M.)

Abstract: The proliferation of unmanned aerial vehicles (UAVs) has spawned a variety of intelligent services, where efficient coordination plays a significant role in increasing the effectiveness of cooperative execution. However, due to the limited operational time and range of UAVs, achieving highly efficient coordinated actions is difficult, particularly in unknown dynamic environments. This paper proposes a multiagent deep reinforcement learning (MADRL)-based fusion-multiactor-attention-critic (F-MAAC) model for multiple UAVs' energy-efficient cooperative navigation control. The proposed model is built on the multiactor-attention-critic (MAAC) model, which offers two significant advances. The first is the sensor fusion layer, which enables the actor network to utilize all required sensor information effectively. Next, a layer that computes the dissimilarity weights of different agents is added to compensate for the information lost through the attention layer of the MAAC model. We utilize the UAV LDS (logistic delivery service) environment created by the Unity engine to train the proposed model and verify its energy efficiency. The feature that measures the total distance traveled by the UAVs is incorporated with the UAV LDS environment to validate the energy efficiency. To demonstrate the performance of the proposed model, the F-MAAC model is compared with several conventional reinforcement learning models with two use cases. First, we compare the F-MAAC model to the DDPG, MADDPG, and MAAC models based on the mean episode rewards for 20k episodes of training. The two top-performing models (F-MAAC and MAAC) are then chosen and retrained for 150k episodes. Our study determines the total amount of deliveries done within the same period and the total amount done within the same distance to represent energy efficiency. According to our simulation results, the F-MAAC model outperforms the MAAC model, making 38% more deliveries in 3000 time steps and 30% more deliveries per 1000 m of distance traveled.

Keywords: air logistics; multiagent reinforcement learning; actor-attention-critic; sensor fusion; multiple UAV

Citation: Jeon, S.; Lee, H.; Kaliappan, V.K.; Nguyen, T.A.; Jo, H.; Cho, H.; Min, D. Multiagent Reinforcement Learning Based on Fusion-Multiactor-Attention-Critic for Multiple-Unmanned-Aerial-Vehicle Navigation Control. *Energies* **2022**, *15*, 7426. https://doi.org/10.3390/en15197426

Academic Editors: R. Maheswar, M. Kathirvelu and K. Mohanasundaram

Received: 2 September 2022
Accepted: 5 October 2022
Published: 10 October 2022

1. Introduction

In recent years the usage of unmanned aerial vehicles (UAVs) for various applications has increased spontaneously. Multiple UAVs are deployed for cooperative missions such as passenger transportation, logistics delivery, and surveillance [1]. In order to successfully carry out the mission in limited resources and time, an energy-efficient multiple-UAV navigation control is needed for the cooperative task. Since the energy consumption of a UAV is proportional to operating time, the UAV's energy efficiency is directly related to a high performance [2]. To develop an energy-efficient multiple-UAV control model, control complexity is a typical problem that needs to be resolved. When UAVs perform cooperative missions together, the decision of one UAV affects the decision of other UAVs. Moreover, complexity increases exponentially as the number of UAVs increases [3]. Consequently,

there are clear limitations in solving such problems with existing conventional heuristic-based search algorithms.

Multiagent deep reinforcement learning (MADRL) is a novel model that enables each agent to perform cooperative tasks by interacting with other agents through their own decisions. MADRL is a suitable model compared to a conventional model, which can be applied to various environments where multiple agents exist, such as multirobot controls, multiplayer games, and multiple-UAV control, etc. [4,5]. Unlike a ground vehicle that moves on a 2D plane, the range of a UAV's motion is much broader. As a result, the movement strategy for mission performance is more diverse. Furthermore, UAVs must make appropriate decisions by using their own sensor information and the information retrieved by other UAVs. For these reasons, a suitable MADRL model must be selected for efficient navigation control.

There has been considerable research carried out in reinforcement learning (RL) based on UAV navigation and its application. G. Muñoz et al. [6] developed a DQN-based model applied to a single UAV for navigation with obstacle avoidance. The Airsim-based realistic simulated 3D environment was utilized for training the agent. The author evaluated and demonstrated that the proposed model outperformed other DQN-based algorithms. Similarly, H. Qie et al. [7] proposed a multiagent deep deterministic policy gradient (MADDPG)-based model for multiple-UAV target assignment and path planning. The results showed that agents could be assigned to their targets at a relatively close distance with a clear behavior for avoiding threat areas. Linfei Feng [8] introduced the policy gradient (PG) model, which could be applied to optimize the logistics distribution routes of a single UAV. The results showed that the UAV arranged delivery routes to multiple destinations with the shortest path. Ory Walker et al. [9] developed a framework based on the combination of proximal policy optimization (PPO) and adaptive belief tree (ABT) for multiple-UAV exploration and target finding. The proposed algorithm was verified in both 2D and 3D environments with the physically simulated UAVs using the PX4 software stack. W.J. Yoon et al. [10] utilized the QMIX model for eVTOL mobility in drone taxi applications. The proposed QMIX-based algorithm showed optimal performance when compared with independent DQN (I-DQN) and a random walk in the drone taxi service scenario. Zhou W. et al. [11] proposed a reciprocal-reward multiagent actor-critic (MAAC-R) method and applied it for learning cooperative tracking policies for UAV swarms. The training results demonstrated that the proposed model performed better than the MAAC model in terms of cooperative tracking behaviors of UAV swarms. D. Xu et al. [12] improved the MADDPG-based algorithm and applied it for the autonomous and cooperative control of UAV clusters in combat missions. The proposed algorithm was tested by performing two conventional combat missions. The result showed that the learning efficiency and the operational safety factor were improved when compared with the original MADDPG algorithm. Similarly, Guang Zhan et al. [13] applied multiagent proximal policy optimization (MAPPO) in a Unity based 3D-simulated air combat environment. The proposed algorithm was trained with a Ray based distributed training framework. In the experiment, MAPPO outperformed COMA and BiCNet in average accumulate reward. Table 1 shows a detailed comparison of research activity conducted utilizing MADRL and RL.

Table 1. Comparison of RL-based UAV application.

Name of the Research	Year	Baseline	Actor Critic	Single/ Multiagent	Centralized/ Decentralized	Applications	Simulated Environment
Multiple-UAV Reinforcement Learning Algorithm Based on Improved PPO in Ray Framework [13]	2022	MAPPO [14]	Yes	Multiagent	Centralized critic with decentralized actor	Distributed decision-making and complete cooperation task	Unity collaborative combat environment 3D
Autonomous and cooperative control of UAV cluster with multi-agent reinforcement learning [12]	2022	MADDPG [15]	Yes	Multiagent	Centralized critic with decentralized actor	Autonomous and cooperative control of UAV clusters	Conventional combat environment
Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning [11]	2021	MAAC [16]	Yes	Multiagent	Centralized critic with decentralized actor	Tracking the perceived targets and searching the unknown targets	Coordinate plane 2D
Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications [10]	2021	QMIX [17]	Yes	Multiagent	Centralized critic with decentralized actor	Computing the optimal passenger transportation routes	200-by-200 grid map 2D
A Framework for Multi-Agent UAV Exploration and Target-Finding in GPS-Denied and Partially Observable Environments [9]	2020	ABT + PPO [18]	Yes	Multiagent	Decentralized actor and critic	Multiple-UAV exploration and target finding	Occupancy map with OpenAI Gym 2D + 3DR Iris and 3DR Solo with Gazebo 3D
Reinforcement Learning to Optimize the Logistics Distribution Routes of Unmanned Aerial Vehicle [8]	2020	PG [19]	Yes	Single Agent	-	Path planning for UAVs in complex surroundings	Coordinate plane 2D
Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning [7]	2019	MADDPG [15]	Yes	Multiagent	Centralized critic with decentralized actor	Multiple-UAV target assignment and path planning	OpenAI's platform 2D
Deep reinforcement learning for drone delivery [6]	2019	DDQN [20]	No	Single Agent	-	Navigation with obstacle avoidance in realistic environment	Realistic neighborhood environment on AirSim 3D

From Table 1, most of the research was carried out using actor-critic-based models. Additionally, based on the previous research related to MADRL, we conclude that centralized training with a decentralized execution methodology is more suitable for real-world situations. In real-world execution, it is difficult for one UAV to obtain data from all other UAVs in real time. A decentralized actor network can be used to infer the action in such a partially observable environment. We paid attention to the multi-actor-attention-critic (MAAC) model, which showed optimal performance among algorithms based on a centralized critic and a decentralized policy, which can be used in environments where information exchange between agents is not guaranteed [16].

This study makes the following significant contributions.

- The development of an MAAC-based model with two significant improvements by applying a sensor fusion layer in the actor network and a dissimilarity layer in the critic network.
- A new feature to calculate the energy efficiency of UAVs is incorporated with the previously developed UAV LDS simulation environment.
- The performance of the existing RL and MADRL models are compared with two energy efficiency indicators.

In this research, we focus on optimizing learning efficiency by efficiently processing the observations of multiple UAVs by adding two features to the MAAC model. First, we introduce a sensor fusion layer in the actor network to extract features from various sensors such as a ray-cast sensor for preventing collision with adjacent obstacles, an inertial navigation system (INS) for the self-awareness of flight status, and a radio detection and ranging (RADAR) system for collecting location data from other UAVs. Second, in the critic network, a dissimilarity layer is added to provide more weight to the information of agents with fewer similarities. By implementing these functions, the efficiency of information processing is increased, and we prove through experiments that it plays a decisive role in achieving the goal of energy-efficient UAV navigation control.

To experiment and validate our proposed MADRL model, the logistic delivery service virtual test bed is adopted from our previous research [21]. The test bed is customized by adding an energy efficiency module for multiple-UAV cooperation specifically for logistic delivery. To find out whether UAVs can cooperatively perform missions well, the environment includes a scenario in which two UAVs cooperate for transport logistics. A function to measure the total travel distance of UAVs has been added to validate the energy efficiency of the UAVs. Our proposed model shows the highest performance in terms of energy efficiency compared to conventional RL algorithms. We measure energy efficiency with the number of trips carried out during the same time, and the number of cargos carried out during the same distance traveled. Our model shows superiority in both indicators.

Our work is structured as follows. Section 2 covers the general background of the RL and MADRL algorithms. In Section 3, we expound on the proposed fusion-MAAC (F-MAAC) method. Section 4, the test bed for the training and evaluation is described in detail. Section 5 shows the results and discusses performance evaluation. Finally, the study concludes with future directions in Section 6.

2. Background

RL is a field that has recently been spotlighted in the field of machine learning. It is a technology that learns a model through the trial and error of an agent in a given environment without any data. RL can be described as a learning process that develops a behavior through trial and error to maximize the cumulative reward in a sequential decision-making problem. The Markov decision process (MDP) can be expressed as a sequential decision-making problem. RL is being utilized in various fields and situations expressed as sequential decision-making problems, such as stock investment, driving, and games.

2.1. Markov Decision Process (MDP)

RL is an optimization method for solving sequential decision-making problems using the Markov decision process (MDP). The MDP is defined as follows.

$$(S, A, P, R, \gamma) \tag{1}$$

Here, S stands for state space and A stands for action space. P is the probability distribution of the next state s' when the agent chooses the action $a \in A$ from the state $s \in S$, and R means the reward received in the next state s'. For the cumulative reward, the future reward is depreciated using the discount rate γ. This reflects future uncertainty and prevents the divergence of the cumulative reward so that learning can be performed stably. Figure 1a exemplifies the basic concept of MDP. When an agent chooses action a, the environments proceed to the next step by action a and return the next state s and reward r.

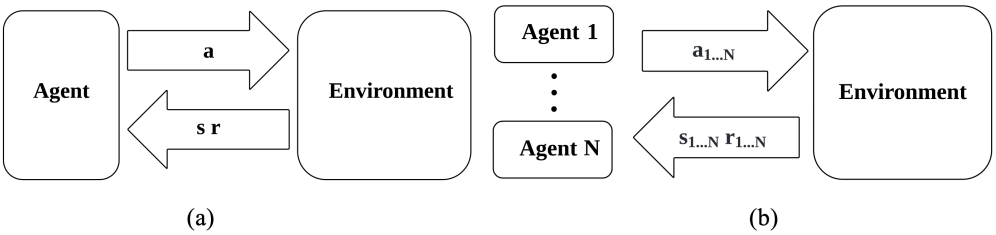

Figure 1. Conceptual diagram: (**a**) Markov decision process and (**b**) Markov game (a—action, s—state, r—reward).

A Markov game is a multiagent extension of the MDP [22]. A Markov game is defined as a set of states and actions for N agents. A probability distribution for the next state is given through the current state and action of each agent. The reward function for each agent depends on the global state and action of all agents. Observation O_i is a partial state that agent i can observe and includes some information of the global state. Each agent learns the policy $\pi : O_i \rightarrow P(A_i)$ that maximizes the expected sum of rewards. Figure 1b shows the multiple-agent interaction with the environment to update the rewards. Multiple agents $\{A_1 \ldots A_N\}$ send the action command $a_N = \{A_1\{a_1\}, A_2\{a_2\}, \ldots, A_N\{a_N\}\}$ to the environment. The environment returns the following set of state $s_N = \{A_1\{s_1\}, A_2\{s_2\}, \ldots, A_N\{s_N\}\}$ and reward $r_N = \{A_1\{r_1\}, A_2\{r_2\}, \ldots, A_N\{r_N\}\}$.

2.2. Bellman Equation

Solving the MDP is divided into prediction and control problems. Prediction is the problem of evaluating the value of each state given a policy. Control is the problem of finding the optimal policy. The policy and value need to be expressed through the Bellman equation to solve these problems. Bellman's equation is defined using the recursive relationship between the present time step t and the next time step t + 1. The value function $V(S)$ and the action value function $Q(S, A)$ can be expressed as the Bellman expectation equation and the Bellman optimal equation [23].

The expected reward G_t is derived using the following equation:

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T \tag{2}$$

where R is the reward, G_t is the sum of the rewards received from time step t + 1 to the final time step T.

Since immediate rewards are more important than the future reward, the discount factor γ is multiplied by Equation (2) to redefine G_t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = R_{t+1} + \gamma G_{t+1} \tag{3}$$

- *Bellman's expectation equation*—The value function $v_\pi(s_t)$ is calculated using the expected value G_t in Equation (4).

$$v_\pi(s) = E_\pi[G_t|S_t = s] \tag{4}$$

where E_π is the expectation when following the policy π .
Now to derive the value function $v_\pi(s_t)$, Equation (3) is substituted in Equation (4).

$$v_\pi(s_t) = E_\pi(R_{t+1} + \gamma v_\pi(s_{t+1})] \tag{5}$$

The action value function $q_\pi(s, a)$ is calculated using the expected value $E_\pi(G_t)$.

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] \tag{6}$$

Now, to action value function $q_\pi(s_t, a_t)$, Equation (3) is substituted in Equation (6).

$$q_\pi(s_t, a_t) = E_\pi[R_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})] \tag{7}$$

- *Bellman's optimal equation*—The optimal value $v_*(s)$ and $q_*(s, a)$ is calculated as follows:

$$v_*(s) = \max_\pi v_\pi(s_t) = \max_a E[R_{t+1} + \gamma v_\pi(s_{t+1})] \tag{8}$$

where $\max_\pi$ is the maximum cumulative rewards and $\max_{a'}$ is the best action a′ out of all actions a_{t+1} that provides a maximum reward.

$$q_*(s, a) = \max_\pi q_\pi(s_t, a_t) = E[R_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a')] \tag{9}$$

2.3. Multiagent Deep Reinforcement Learning

Multiagent Deep Reinforcement Learning (MADRL) is one of the most popular and effective models for solving more complex problems where multiple agents collaborate to perform specific tasks. For example, playing soccer games with multiple robots where the team of robots collaborates to achieve the mission. One of the key challenges in such an environment is that the environment is more dynamic to the perceptive of each agent, which may affect the individual learning rate as a team.

- *Multiagent deep deterministic policy gradient (MADDPG)*— MADDPG [15] is a multiagent extension of DPG [24], which combines DDPG [25] with DQN [26] approaches such as replay buffer and target separation. Each agent has its own actor and critic. In the MADDPG method, centralized training with a decentralized execution approach is used. The architecture of the MADDPG model is shown in Figure 2. A centralized critic network $Q_{1...N}$ is used for centralized training with observations $o_{1...N}$ and actions $a_{1...N}$ from all other agents as input. In the decentralized execution, agents use an actor network $\pi_{1...N}$ to choose an action by only using local information. By this approach, the MADDPG model can be applied even in a partially observable environment where communication between agents is limited.
- *Multiactor-attention-critic (MAAC)*—MAAC was developed by [16] and adopted from the MADRL model. The model trains the decentralized policies in multiagent environments by utilizing centrally computed critics with an attention mechanism. It chooses relevant information for each agent at every time step. The multiattention head layer consists of multiple attention heads. The attention function in the attention head can be described as mapping a query and a set of key–value pairs to an output [27]. The attention function is calculated as Equation (10), where query Q has the corresponding key K and value V and d_k is a scaling factor. As shown in Figure 3, encodings of the agent's state and action denotes the state action encodings (SAE_i) are the key and value. The encodings of the other agent's state encoder (SE_j), $j \in \backslash i$ are the query. In

each attention head N, different attention head values (AHVs) are derived according to the influence of the query, key, and value extractors. The final output attention value (AV) is achieved with the combination of AHVs. The final output Q_i(o,a) is derived through fully connected layers FC_1 and FC_2 with the input of AV and SE_i. In the multiattention head layer, the agent updates the weighted value which is more similar to other agents. This attention mechanism enables a more effective and flexible learning in complex multiagent environments compared to MADDPG.

$$AttentionFuntion(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

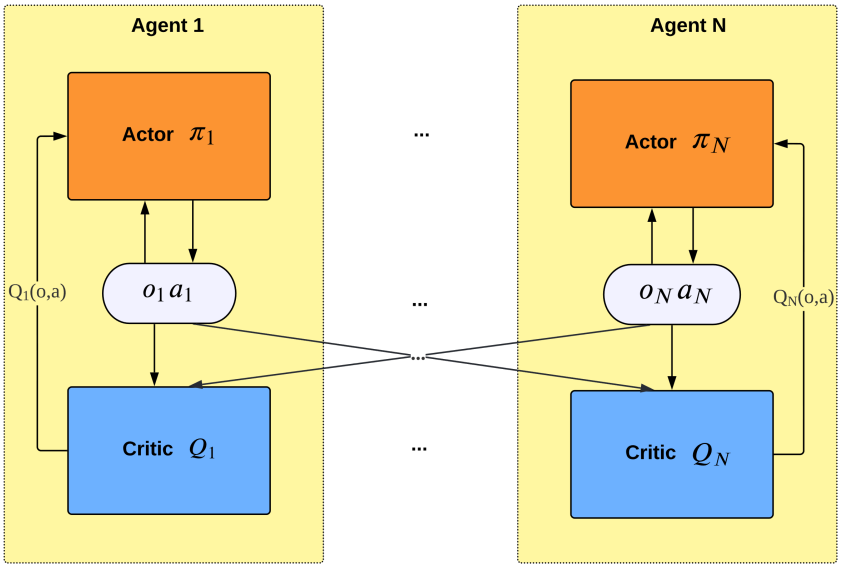

Figure 2. Overall architecture of DDPG.

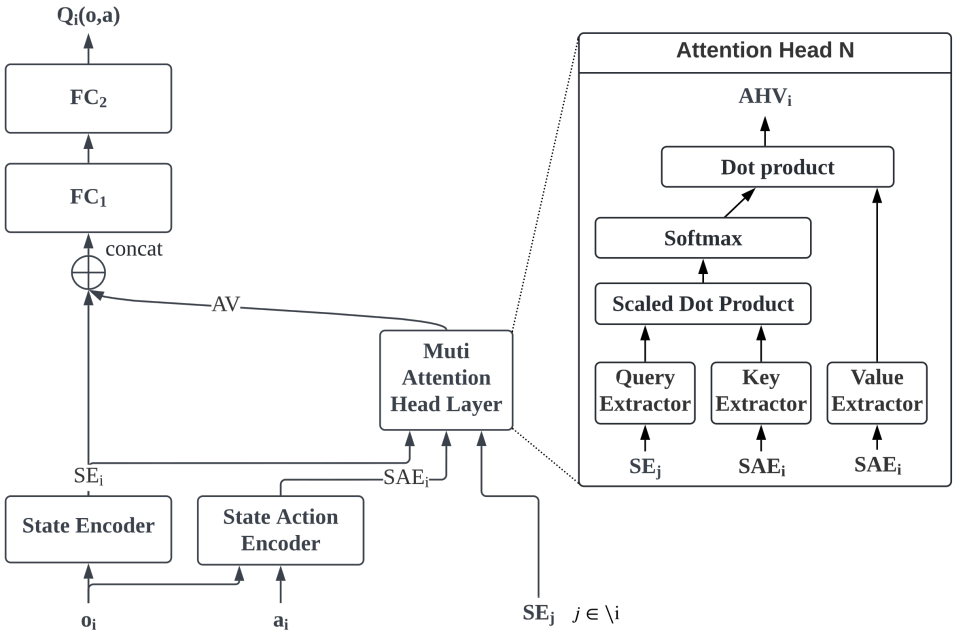

Figure 3. Critic network of MAAC.

3. Fusion-Multiactor-Attention-Critic (F-MAAC) Model

In this section, the F-MAAC model is discussed for the application of multiple-UAV cooperative navigation. To increase the learning efficiency of the agent, we used a new

sensor fusion layer with MAAC. The sensor fusion layer was used for the UAV's local observation, and another layer named cosine dissimilarity was added to utilize global information obtained by other UAVs efficiently. The overall architecture of the proposed F-MAAC model is exemplified in Figure 4.

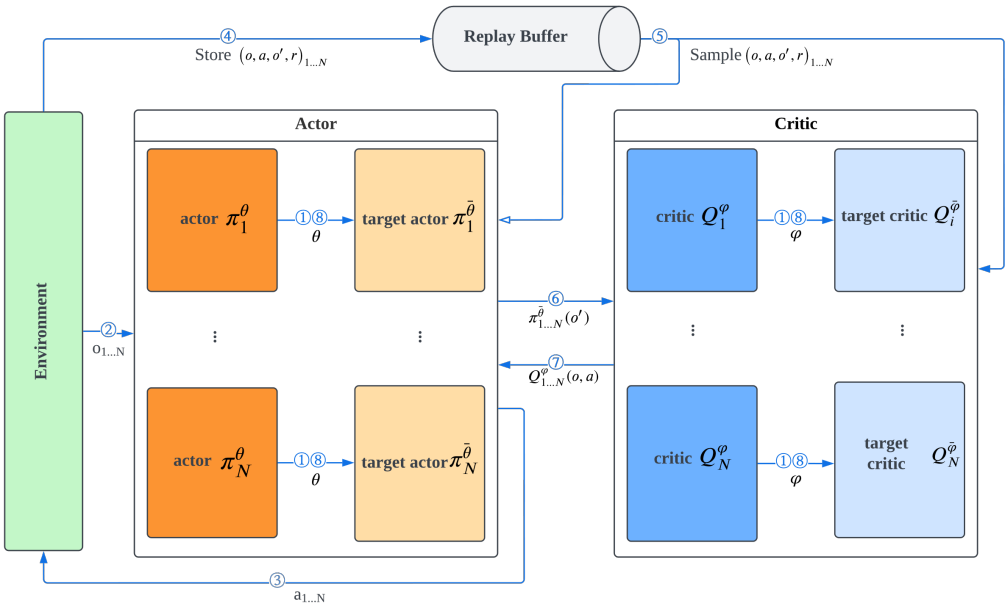

Figure 4. F-MAAC model's overall architecture with training flow.

The overall flow of the F-MAAC model follows the basis of the MAAC model, including a loss function and the gradients of the objective function. Each agent has its own independent actors and critics following a centralized training with a decentralized execution. In the training phase, all agents' observations are entered as inputs of each agent's critic network. In the execution phase, the decentralized actor network is used to choose the action as inference by using only its own observation for input data. This general F-MAAC model can be applied to N agents equipped with M types of sensors. The step-by-step training procedure of the F-MAAC model is as follows:

Step 1: Initialize the critic network $Q_{1...N}^{\psi}$ and actor network $\pi_{1...N}^{\theta}$ with random parameters and synchronize the parameters of target critics $Q_{1...N}^{\overline{\psi}}$ with critics $Q_{1..N}^{\varphi}$, and target actors $\pi_{1...N}^{\bar{\theta}}$ with actors $\pi_{1..N}^{\theta}$.

Step2: Get observation $o_{1...N}$ from the environment, feed-forward to actors $\pi_{1...N}^{\theta}(o)$, and select action $a_{1...N}$.

Step 3: Proceed to the next time step with actions $a_{1...N}$ and get the next observations $o_{1...N}^{'}$ and rewards $r_{1...N}$ from the environment.

Step 4: Push the obtained set of $(o, a, o^{'}, r)_{1...N}$ to the replay buffer.

Step 5: Repeat step 2 to 4 until the number of E data is collected.

Step 6: Sample $B = (o, a, o^{'}, r)_{1...N}$ from the replay buffer,

Step 7: Perform a gradient descent by using B to minimize the loss function in Equation (11) with respect to the network parameter φ

$$L_Q(\varphi) = \sum_{i=1}^{N} E[(Q_i^{\varphi}(o, a) - y_i^2)] \tag{11}$$

where $y_i = r_i + \gamma E_{a' \sim \pi_\theta(o')}\left[Q_i^{\bar{\psi}}(o', a') - \alpha \log\left(\pi_{\theta_i}\left(a_i' \mid o_i'\right)\right)\right]$

Step 8: Perform a gradient ascent by using $o_{1...N}$ in B to maximize the gradient of the objective function in Equation (12) with respect to the network parameter θ

$$\nabla_{\theta_i} J(\pi_\theta) = E_{o \sim D, a \sim \pi} \nabla_{\theta_i} \log \pi_{\theta_i}(a_i|o_i)\left(-\alpha \log\left(\pi_{\theta_i}(a_i|o_i)\right) + A_i(o, a)\right) \tag{12}$$

Step 9: Update the parameters of target critics $Q^{\bar{\psi}}_{1...N}(o,a)$ with Equation (13) and target actors $\pi^{\bar{\theta}}_{1...N}(a,s)$ with Equation (14) using an update rate of $\tau = 0.005$

$$\overline{\psi} = \overline{\psi} * (1.0 - \tau) + \psi * \tau \tag{13}$$

$$\overline{\theta} = \overline{\theta} * (1.0 - \tau) + \theta * \tau \tag{14}$$

Step 10: Steps 2 to 9 should be repeated until the end of the episode.

3.1. Deep Fusion Layer in Actor Network

As illustrated in Figure 5, we propose a deep fusion layer in the actor network to increase efficiency. Observations are separated into the M types of sensors to extract features from each sensor. For instance, three different types of sensors are used for UAVs in our virtual UAV LDS environment: a ray-cast sensor for preventing collision with surrounding obstacles, an INS for the self-awareness of flight status, and a RADAR for retrieving coordinates of other UAVs and hubs. Each sensor's data pass through the sensor encoder. The encoded sensor data are concatenated and pass through two fully connected layers. The output of the deep fusion layer can be expressed by Equation (15). FC_1, FC_2, and sensor encoders($SNE_{1...3}$) are fully connected layers.

$$Output = FC_2(FC_1(Concat(SNE_1(sensor_1),\ SNE_2(sensor_2),\ SNE_3(sensor_3)))) \tag{15}$$

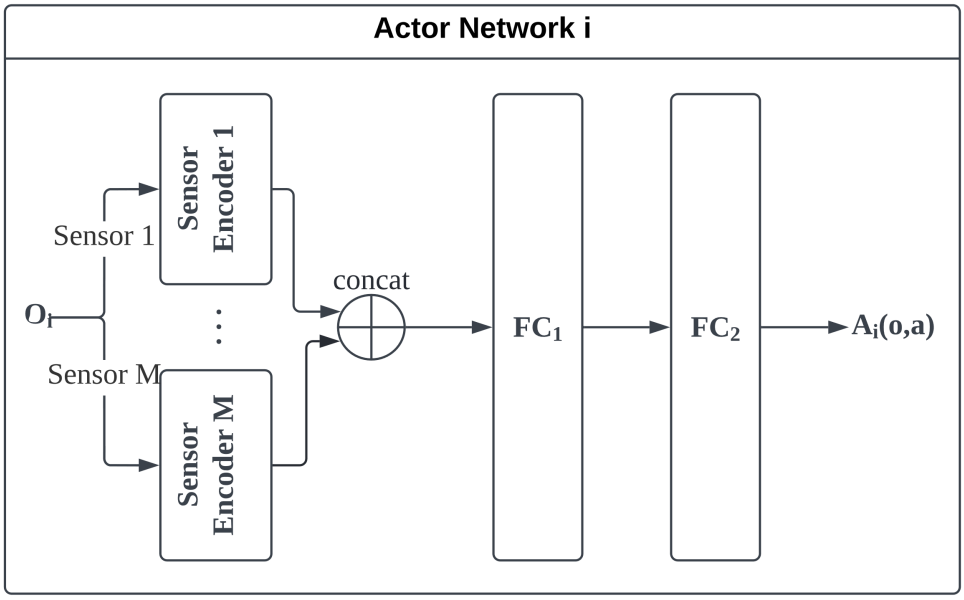

Figure 5. Actor network of F-MAAC model.

3.2. Dissimilarity Layer in Critic Network

In the critic network, state encodings ($SE_{1...N}$) are shared with other agents, as shown in Figure 6. The attention head in the multiattention head layer selects relevant information from other agents' observations. The attention head is constructed with a scaled dot product [27] which calculates the degree of similarity between encoded observations of agent i (SE_i, SAE_i) and the encoded observations of the other agents $j \in \backslash i$ (SE_j). The UAVs at adjacent distances will have similar observation data. When more weights are provided to similar observations, the UAVs will have a wider field of view and less chance of colliding with each other.

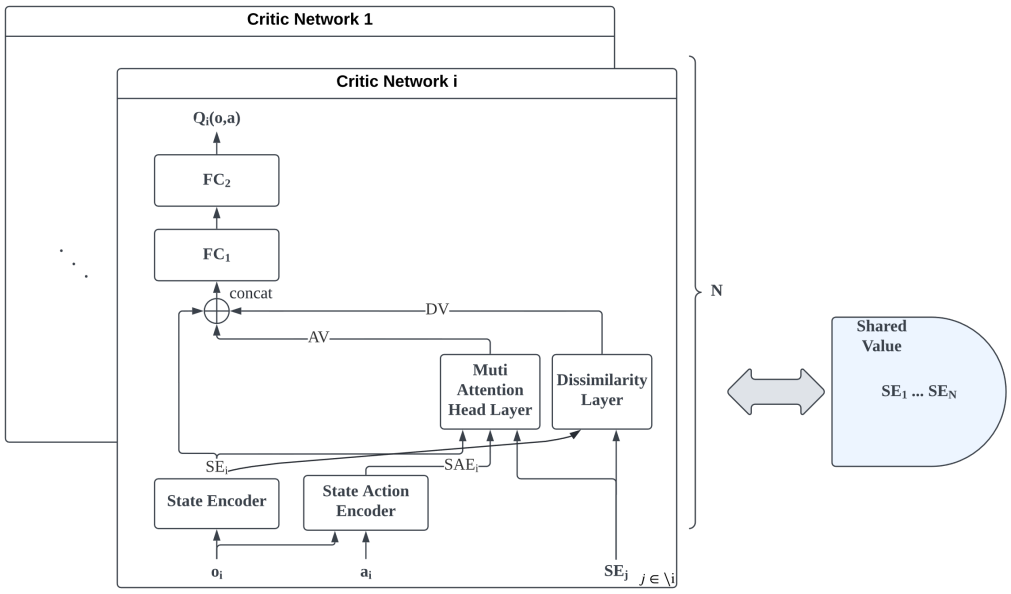

Figure 6. Critic network of F-MAAC model.

However, there are also drawbacks derived from the multiattention head layer. For example, when an agent's observation at a distance that is dissimilar from the current agent's observation plays an essential role in performing its mission, it can lead to serious performance degradation. More specifically, the observation from an agent at long distances near a target point may provide helpful information. For these reasons, a dissimilarity layer was added to prevent performance degradation due to attention and to improve learning stability. In the previous study, we verified the effect of adding a dissimilarity layer to the MAAC model in a simple 2D cooperative navigation environment [28].

Cosine similarity refers to the similarity between two vectors obtained by using the cosine angle between the two vectors. The additional use of the observation multiplied by the dissimilarity value may offset the effect of attention. The dissimilarity value is calculated with the encoded observations of agent i (SE_i) and the encoded observations of other agents $j \in \backslash i$ (SE_j). The value passed through the dissimilarity layer's dissimilarity value (DV) is concatenated with the value from the multiattention head layer's attention value (AV) and SE_i. Then, the concatenated value is sent to the fully connected layers FC_1 and FC_2 to calculate the critic value Q_i.

Figure 7 shows the detailed process of the dissimilarity layer. The dissimilarity weight between the agent's observations is calculated by multiplying the cosine similarity value by a negative number as in Equation (16).

$$CosineDisimilarity(SE_i, SE_n) = -1 \cdot \frac{SE_i \cdot SE_n}{max(\| SE_i \|_2 \cdot \| SE_n \|_2, \varepsilon)} \quad (16)$$

where, $\varepsilon = 1 \times 10^{-8}$.

The negative dissimilarity values are replaced with 0 to focus on the agents' information with different patterns. Then, the observations of each agent are multiplied by the cosine dissimilarity weight and concatenated. The concatenated value is entered as the input value of the fully connected layer. The output value DV from the dissimilarity layer, the output value AV from the multiattention head layer, and the encoded value SE_i are concatenated as an input of the fully connected layers.

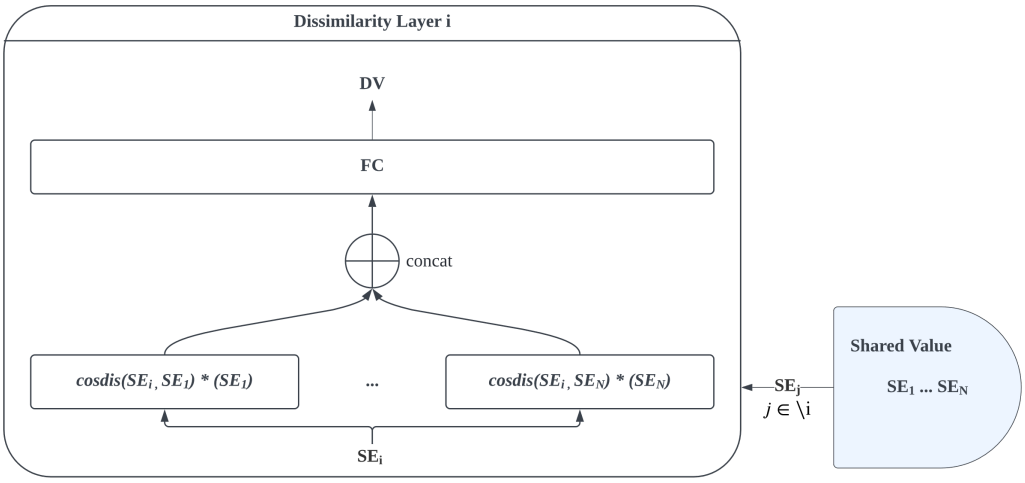

Figure 7. Dissimilarity layer in critic network.

4. Test bed

4.1. UAV LDS Environment

In our previous work [21], we developed a UAV logistic delivery service (UAV LDS) environment for evaluating MADRL-based models. To calculate energy efficiency for this research, we added the feature of calculating the total movement of all agents. The UAV LDS environment is a virtual environment designed to reflect simplified logistics delivery scenarios in the real world and implemented through the Unity platform equipped with the 3D physics engine. The modified source was updated in the following repository (https://github.com/leehe228/LogisticsEnv, accessed on 3 October 2022). The environment follows the Open AI Gym API [29] design which provides standard communication between learning algorithms and environments. In LDS, multiple UAVs act as an air transportation system which is used to carry cargo in the three-dimensional city sky that connects land and air. To implement this as a simulated environment, we constructed blocks representing obstacles such as buildings, warehouses, and cargo to be transported. In the scenario, UAVs delivered big cargo and small cargo from hubs to the destination. What was unique about this environment was that two UAVs had to collaborate to move a big cargo. The reason for including this scenario was that it was possible to check whether the cooperation of UAVs worked well directly. In addition, such cooperative situations could occur any time in the real world, such as when multiple UAVs need to move together to load multiple cargos. Figure 8 shows the UAV carrying cargo in the UAV LDS environment. The gray box indicates the buildings in the real world, the blue box is the small cargo, and the red box is the big cargo. Cargos are generated from the hubs, colored blue on the ground. The destination of the big cargo is colored pink and that of the small cargo is colored green.

4.2. Observation, Action, and Reward Design

This section describes the state, action, and reward of the environment which are essential elements of MDP. The state is the observation received by the agent, the action is the type of movement that can be selected, and the reward is the compensation according to the UAV's action.

- *Observation*—The UAVs received three different sensor data such as ray-cast for preventing collision with adjacent obstacles, INS for the self-awareness of flight status, and RADAR used to find the location of the other UAVs and hubs. In Table 2, a detailed description of the sensor data is provided.
- *Actions*—The UAVs could perform seven types of actions: ascend, descend, forward, backward, left, right, and not move.
- *Driving reward* —To make the UAVs deliver cargo in the shortest path, a driving reward was given at every step. The reward was calculated with the difference between the

distance of the previous time step d_{pre} and the distance of the current time step d_{curr}. Each distance was calculated with the distance to the target point. Before picking up the cargo, the nearest cargo was the target point. After picking up the cargo, the delivery point was the target point. If the UAV was not closer to the target point in the current time step than in the previous time step, a negative reward was given as $(d_{pre} - d_{curr}) \times 0.5$.

- *Delivery rewards*—The values in Table 3 were designed to make UAVs deliver cargo efficiently. For training numerous UAVs to work together to carry cargos, we delicately designed the rewards related to the delivery.
- *Collision penalty*—The UAVs must avoid buildings and other UAVs with ray-cast observations. A negative reward of −10 was given when a collision occurred.

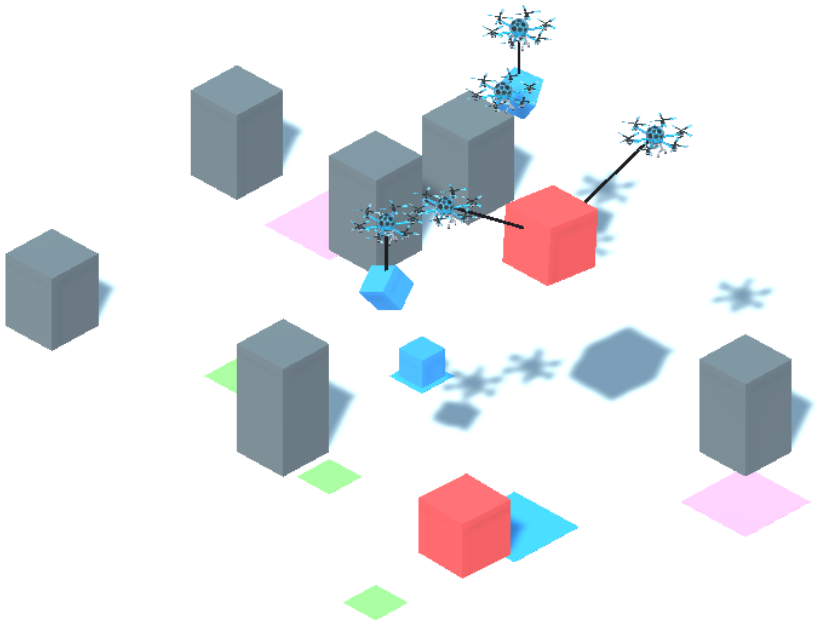

Figure 8. UAV logistic delivery service virtual environment.

Table 2. Summary of observations.

Sensor Type	Size	Description
Ray-cast	1×9	Distance of 9 directions of ray-cast sensor
	2×9	One-hot encoding of the detected object (nothing, building) of 9 direction of ray-cast sensor
INS	3	(x, y, z)—coordinates of UAV_i.
	3	(x, y, z)—velocity of UAV_i.
	3	One-hot encoded cargo type (not holding, small cargo, and big cargo).
RADAR	6	(x, y, z, x, y, z)—coordinates of a big cargo hub and a small cargo hub.
	2	Distance from UAV to big and small cargo hubs.
	6	(x, y, z, x, y, z)—each nearest big and small cargo coordinates.
	2	Distances from UAV_i to the nearest big and small cargos.
	4	(x, y, z, d) if UAV_i holds any cargo, the coordinates and distance of the destination are given.
	7×4	Coordinates of UAV_j (size 3), cargo type of UAV_j (size 3), and distance from UAV_i to UAV_j (size 1). *

* UAV_i is the current, and UAV_j are the rest of all UAVs except UAV_i.

Table 3. Summary of delivery rewards.

Action	Collaborative	First UAV	Second UAV
UAV picks up a small cargo	No	+20.0	-
Small cargo delivery completed	No	+20.0	-
First UAV picks up a big cargo	Yes	+10.0	-
The second UAV picks up a big cargo	Yes	+ 10.0	+20.0
Big cargo delivery completed	Yes	+30.0	+30.0
First UAV drops a big cargo	Yes	−8.0	-
Both UAVs drop a big cargo	Yes	−15.0	−15.0

4.3. Environmental Setup

The UAV environment provided custom settings for the environmental setup. In this research we used the default values in Table 4 for training and evaluation.

Table 4. Summary of environmental setup.

Parameter	Description	Default (Training)	Default (Execution)
NumAgent	Total number of UAVs	5	5
width	Width of the Unity window	480 pixels	1280 pixels
height	Height of the Unity window	270 pixels	720 pixels
timescale	The multiplier for the time	20×	1×
mapsize	Size of the map	13 m	13 m
numbuilding	Number of buildings	3 units	3 units
MaxSmallbox	Total number of small cargos that can be generated	100 units	100 units
MaxBigbox	Total number of big cargos that can be generated	100 units	100 units

5. Experimental Simulation and Results

The proposed F-MAAC model was validated using the environment proposed in Section 4. For more efficient evaluations of the proposed F-MAAC model, we first compared the mean episode rewards of the MAAC, MADDPG, and DDPG models with a training of 20k episodes. Then, the two models with the highest performance, F-MAAC and MAAC, were selected for the training of 150k episodes. To evaluate the trained model to achieve a meaningful scale length, the episode length was replaced with 3000 from 1000 time steps. The timescale of the environment was decreased in the evaluation phase to observe and analyze the strategies of UAVs. The total number of deliveries during one episode and the same distance traveled were evaluated to verify the energy efficiency.

The hyperparameters for training the RL models are shown in Table 5.

Table 5. Hyperparameter settings of RL models.

	DDPG	MADDPG	MAAC	F-MAAC
Number of episodes	1000	1000	1000	1000
Steps per update	100	100	250	250
Batch size	1024	1024	1024	1024
Number of attention heads	-	-	4	4
Policy hidden dimension	128	128	128	128
Learning rate of critic	0.01	0.01	0.001	0.001
Learning rate of policy	0.01	0.01	0.001	0.001

5.1. Comparison of Performance of RL Models

Two MADRL models (MAAC, MADDPG) and one single agent RL model (DDPG) were compared with the proposed F-MAAC model. Each model was trained for 20k episodes with 1000 steps per episode in the proposed UAV LDS simulation environment.

According to Figure 9, the DDPG's mean episode reward value showed the worst performance because it did not increase significantly tableuntil 20k episodes. Although it rose slightly higher from 5k episodes to 20k episodes when compared to DDPG, the increase in the mean episode reward value in the MADDPG model was also minor. The F-MAAC and MAAC models, on the other hand, displayed an impressive performance and successfully conveyed some quantities of both large and small cargos. Between 10k and 20k training episodes, the F-MAAC model demonstrated a more significant value than the MAAC model in the mean episode reward. At the end of 20k training sessions, the F-MAAC model demonstrated greater mean episode rewards than the MAAC model by more than 30%.

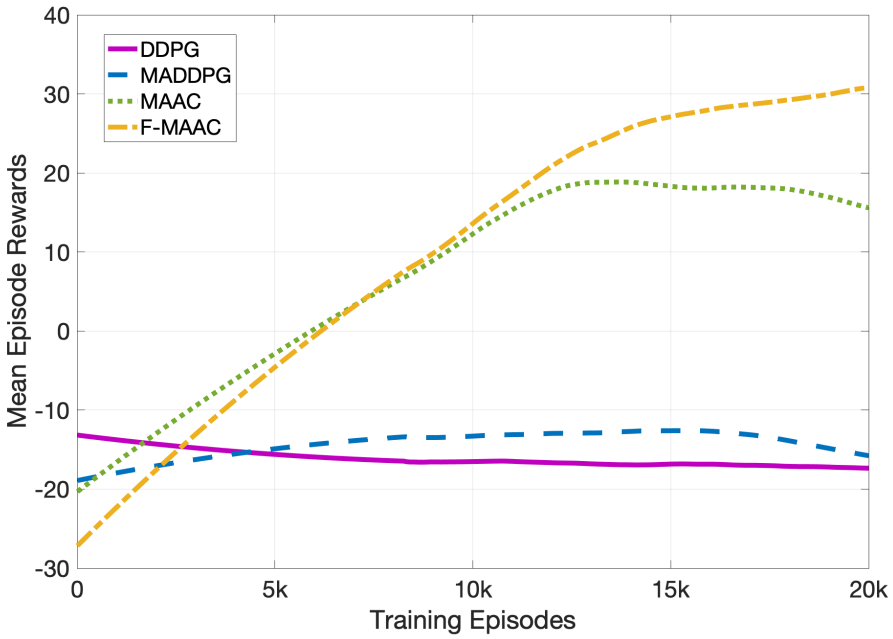

Figure 9. Mean episode rewards comparison of different models for 20k episodes.

5.2. Comparison of Performance between F-MAAC and MAAC Models

We retrained the F-MAAC and MAAC models with 150k episodes, which took about six days with two GPU machines. The detailed specifications of the machine are listed below in Table 6.

Table 6. Specifications and environmental setup of the GPU machine.

CPU	Intel i7 8700 k
GPU	Nvidia RTX 3080
RAM	64 GB
OS	Ubuntu 20.04 LTS
Deep Learning Framework	Pytorch 1.8.2

Figure 10 shows the mean episode rewards of the MAAC and F-MAAC models for 150k training episodes. The mean episode reward value of the F-MAAC and MAAC models increased noticeably in this experiment compared to the previous Section 5.1. The difference between them with training episodes until 40k was unnoticeable. After the training of 40k episodes, the F-MAAC model started to outperform the MAAC model. From 80k to 150k, the mean episode reward of the MAAC model decreased while that of the F-MAAC

model constantly increased. At the end of the training, the F-MAAC model obtained 50% more rewards than the MAAC model. The randomness and instability of the complex 3D environment produced different learning patterns compared with the previous training, since the maps of the UAV LDS environment were generated randomly for every episode. However, both results showed that the F-MAAC model outperformed the MAAC model. The result of this experiment showed a more reliable comparison since it was trained longer, until 150k episodes.

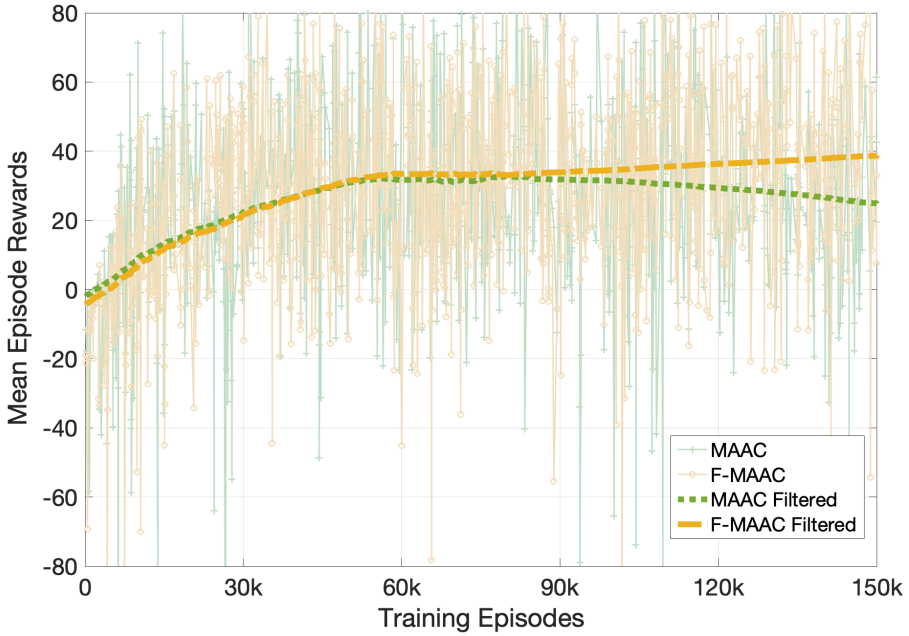

Figure 10. Mean episode rewards of the MAAC and F-MAAC models for 150k episodes.

5.3. Comparison of Energy Efficiency between F-MAAC and MAAC Models

For energy efficiency evaluation, we executed the trained model of F-MAAC and MAAC with 150k episodes. Each model was executed for 100 episodes with 3000 time steps of each episode. The average performance per episode is shown with a box plot in Figure 11. We show the number of successful deliveries of small cargo and big cargo. Furthermore, the total performance was evaluated with $Score = NumberOfSmallCargo + 1.5 * NumberOfBigCargo$. The weight of 1.5 was multiplied by the number of big cargos since we gave 50% more rewards to the big cargos in the training phase.

The result showed that the number of deliveries in both small and big cargos with the F-MAAC model was higher than in the MAAC model. Table 7 shows that the score of the F-MAAC model was 38% higher than that of the MAAC model during one episode, indicating that the F-MAAC model was more energy efficient.

Table 7. Overall comparison of MAAC and F-MAAC models.

	MAAC	**F-MAAC**
Score	13.29	18.31
Movement	1200 m	1270 m
Collision	9.2	8.4
Score_movement	11.08	14.42

We also provided the energy efficiency with Score_movement, which is the performance per 1000 m distance moved. We recorded the total movements of the UAV during execution. The Score_movement was calculated with $\frac{Score}{Movement} \times 1000$. The results showed that the F-MAAC model was 30% more efficient compared to the MAAC model.

In addition, the number of collisions of the F-MAAC model was about 9% less than that of the MAAC model. The improvement of the F-MAAC model's sensor processing efficiency can be interpreted as having a positive effect on the obstacle avoidance performance of the UAV.

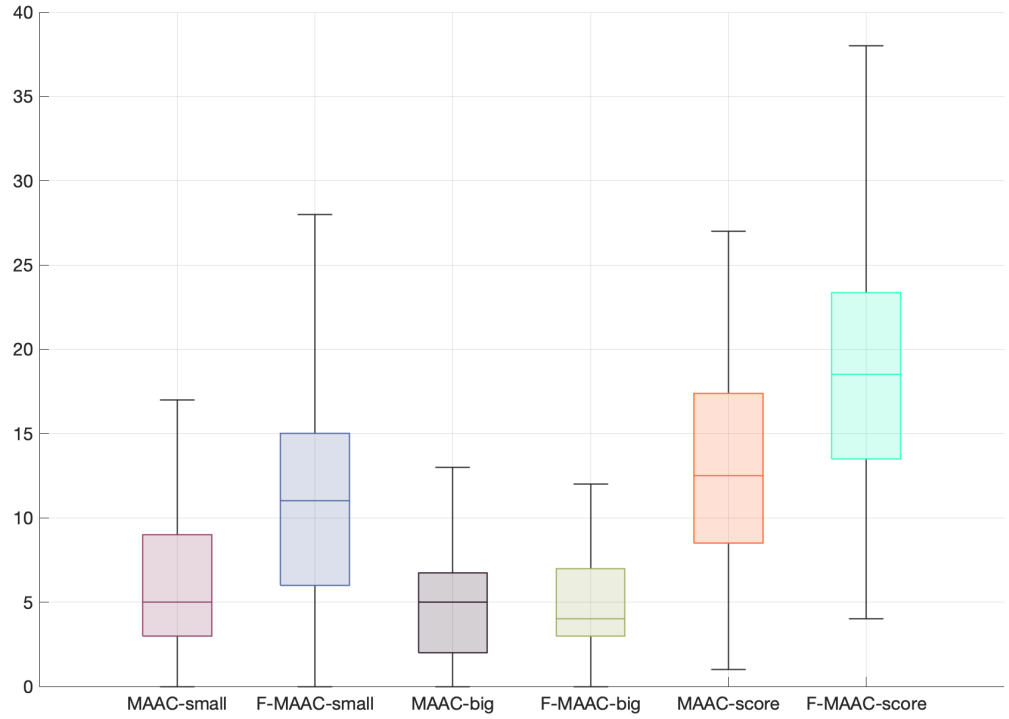

Figure 11. Comparison of delivery performance.

6. Conclusions

This study proposed an MAAC-based multiple-UAV navigation control model that improved energy efficiency through efficient data processing of the UAVs. The following significant findings were obtained.

(a) In the proposed model, the sensor fusion layer was adapted in the actor network, and the dissimilarity layer was utilized for the critic network. When applied to the UAV LDS simulation environment, it outperformed the conventional RL model in terms of energy efficiency.

(b) The sensor fusion layer extracted features from each sensor enabling the UAVs to use various sensor data efficiently. The dissimilarity layer compensated for the loss derived from the attention layer by providing data with high dissimilarity to other agents.

(c) The F-MAAC-applied UAVs transported more cargo than the MAAC in the same amount of time and distance with greater cooperation and fewer collisions.

The feature of measuring the total movement of UAVs was added to the existing UAV LDS environment to calculate energy efficiency. We provided two indicators that calculated the energy efficiency of UAVs. The proposed model showed the best performance in both types of energy efficiency indicators out of various RL models, including the original MAAC model. In future studies, further verification and development are needed for the model in a more sophisticated environment, including realistic sensors and dynamic flight models. Furthermore, the scalability should be verified in a broader environment where more agents exist.

Author Contributions: Conceptualization, S.J.; Investigation, S.J. and H.C.; Methodology, S.J. and H.J.; Project administration, V.K.K. and D.M.; Software, H.L.; Supervision, V.K.K. and D.M.; Validation, V.K.K. and T.A.N.; Visualization, S.J.; Writing—original draft, S.J.; Writing—review & editing, V.K.K. and T.A.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (no. 2020R1A6A1A03046811). This work was supported by the National Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT (MIST)) (no. 2021R1A2C209494311).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

UAV	Unmanned aerial vehicle
RL	Reinforcement learning
MADRL	Multiagent reinforcement learning
LDS	Logistic delivery service
MAAC	Multiactor-attention-critic
F-MAAC	Fusion-multiactor-attention-critic
DDPG	Deep deterministic policy gradient
MADDPG	Multiagent deep deterministic gradient
MDP	Markov decision process
INS	Inertial navigation system
RADAR	Radio detection and ranging

References

1. Roldán, J.J.; Cerro, J.D.; Barrientos, A. A proposal of methodology for multi-UAV mission modeling. In Proceedings of the 2015 23rd Mediterranean Conference on Control and Automation (MED), Torremolinos, Spain, 16–19 June 2015; pp. 1–7.
2. Abeywickrama, H.V.; Jayawickrama, B.A.; He, Y.; Dutkiewicz, E. Comprehensive energy consumption model for unmanned aerial vehicles, based on empirical studies of battery performance. *IEEE Access* **2018**, *6*, 58383–58394. [CrossRef]
3. Zhang, J.; Jiahao, X.I.N.G. Cooperative task assignment of multi-UAV system. *Chin. J. Aeronaut.* **2020**, *33*, 2825–2827. [CrossRef]
4. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef] [PubMed]
5. Chang, H.; Chen, Y.; Zhang, B.; Doermann, D. Multi-UAV mobile edge computing and path planning platform based on reinforcement learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *6*, 489–498. [CrossRef]
6. Muñoz, G.; Barrado, C.; Çetin, E.; Salami, E. Deep reinforcement learning for drone delivery. *Drones* **2019**, *3*, 72. [CrossRef]
7. Qie, H.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; Wang, L. Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning. *IEEE Access* **2019**, *7*, 146264–146272. [CrossRef]
8. Feng, L. Reinforcement learning to optimize the logistics distribution routes of unmanned aerial vehicle. *arXiv* **2020**, arXiv:2004.09864.
9. Walker, O.; Vanegas, F.; Gonzalez, F. A framework for multi-agent UAV exploration and target-finding in GPS-denied and partially observable environments. *Sensors* **2020**, *20*, 4739. [CrossRef] [PubMed]
10. Yun, W.J.; Jung, S.; Kim, J.; Kim, J.H. Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications. *ICT Express* **2021**, *7*, 1–4. [CrossRef]
11. Zhou, W.; Li, J.; Liu, Z.; Shen, L. Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning. *Chin. J. Aeronaut.* **2022**, *35*, 100–112. [CrossRef]
12. Xu, D.; Chen, G. Autonomous and cooperative control of UAV cluster with multi-agent reinforcement learning. *Aeronaut. J.* **2022**, *126*, 932–951. [CrossRef]
13. Zhan, G.; Zhang, X.; Li, Z.; Xu, L.; Zhou, D.; Yang, Z. Multiple-UAV Reinforcement Learning Algorithm Based on Improved PPO in Ray Framework. *Drones* **2022**, *6*, 166. [CrossRef]
14. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv* **2021**, arXiv:2103.01955.
15. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; 30p.
16. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 2961–2970.

17. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.
18. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
19. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the Advances in Neural Information Processing Systems 12 (NIPS 1999), Denver, CO, USA, 29 November–4 December 1999; Volume 12.
20. Hasselt, H.V.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
21. Jo, H.; Lee, H.; Jeon, S.; Kaliappan, V.K.; Nguyen, T.A.; Min, D.; Lee, J.W. Multi-Agent Reinforcement Learning-based UAS Control for Logistics Environments. In Proceedings of the Asia-Pacific International Symposium on Aerospace Technology, Jeju, Korea, 15–17 November 2021.
22. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*; Morgan Kaufmann: Waltham, MA, USA, 1994; pp. 157–163.
23. Glorennec, P.Y. Reinforcement learning: An overview. In Proceedings of the European Symposium on Intelligent Techniques (ESIT-00), Aachen, Germany, 14–15 September 2000; pp. 14–15.
24. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 387–395.
25. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
26. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; 30p.
28. Jeon, S.; Kaliappan, V.K. Dissimilarity Multi Actor Attention Critic based model for robot navigations in cooperative disaster recovery applications. In Proceedings of the International Virtual Conference on Industry 4.0, Amsterdam, The Netherlands, 22–24 September 2022.
29. OpenAi Gym. Available online: https://github.com/openai/gym (accessed on 3 October 2022).