drones

Article

Multi-UAV Autonomous Path Planning in Reconnaissance Missions Considering Incomplete Information: A Reinforcement Learning Method

Yu Chen [1], Qi Dong [1,2,*], Xiaozhou Shang [2], Zhenyu Wu [3] and Jinyu Wang [4]

1 Institute of Advanced Technology, University of Science and Technology of China, Hefei 230026, China
2 China Academy of Electronics and Information Technology, Beijing 100049, China
3 School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
4 School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: dongqiouc@126.com

Abstract: Unmanned aerial vehicles (UAVs) are important in reconnaissance missions because of their flexibility and convenience. Vitally, UAVs are capable of autonomous navigation, which means they can be used to plan safe paths to target positions in dangerous surroundings. Traditional path-planning algorithms do not perform well when the environmental state is dynamic and partially observable. It is difficult for a UAV to make the correct decision with incomplete information. In this study, we proposed a multi-UAV path planning algorithm based on multi-agent reinforcement learning which entails the adoption of centralized training–decentralized execution architecture to coordinate all the UAVs. Additionally, we introduced a hidden state of the recurrent neural network to utilize the historical observation information. To solve the multi-objective optimization problem, We designed a joint reward function to guide UAVs to learn optimal policies under the multiple constraints. The results demonstrate that by using our method, we were able to solve the problem of incomplete information and low efficiency caused by partial observations and sparse rewards in reinforcement learning, and we realized kdiff multi-UAV cooperative autonomous path planning in unknown environment.

Keywords: multi-UAV; path planning; incomplete information; multi-objective, reinforcement learning

check for updates

Citation: Chen, Y.; Dong, Q.; Shang, X.; Wu, Z.; Wang, J. Multi-UAV Autonomous Path Planning in Reconnaissance Missions Considering Incomplete Information: A Reinforcement Learning Method. *Drones* **2023**, *7*, 10. https://doi.org/10.3390/drones7010010

Academic Editors: Mou Chen, Xiwang Dong, Xiangke Wang and Fei Gao

Received: 30 November 2022
Revised: 14 December 2022
Accepted: 17 December 2022
Published: 23 December 2022

1. Introduction

Multi-UAV perform well in complex tasks because of their robustness and high efficiency [1]. When multi-UAV perform reconnaissance tasks cooperatively in an unknown environment, they have to perceive the environment through their own sensors and plan the optimal path online according to the current environmental state to reach the target points safely. It is important for UAVs to be capable of autonomous navigation in complex and unknown environments. Moreover, a greater coordination is needed between all UAVs. Thus, we have to consider we can guide multi-UAV to achieve a common goal.

Multi-UAV path planning can be considered as a Multi-Agent Path Planning (MAPF) problem [2], which is a model used to find the optimal path for multi-agents from the starting positions to destinations without conflicts. In fact, MAPF is a relatively complex joint objective optimization problem. The state space of this problem grows exponentially with the number of agents, and it has been proved to be an NP-hard problem [3]. In the reconnaissance tasks, multi-UAV not only have to avoid dangerous areas and reach the target points safely, but they must also cover a larger area in a shorter time. However, the time cost and coverage area are in conflict, as these are multi-objective optimization problems, and we must make a trade-off between two or more conflicting goals to enable optimal decision making. It is impossible to find a solution that can achieve the optimal performance of all objectives; therefore, for the multi-objective optimization problem, we usually use a set of non-inferior solutions called the "Pareto solution set" [4].

Most of the previous research regarding multi-UAV path planning has focused on intelligent optimization algorithms, such as evolutionary algorithms [5] including Particle Swarm Optimization(PSO) [6–8]. Shao et al. [9] proposed a more accurate and faster PSO algorithm to effectively improve the convergence speed and solution optimality, and the proposed PSO was successfully used in UAV formation path planning under terrain, threat, and collision avoidance constraints. Evan et al. [10] proposed a PSO algorithm for use in navigating in an unknown environment, which was able to reach a pre-defined goal and become collision-free. Ajeil et al. [11] proposed a hybridized PSO-modified which was shown to minimize the distance and follow path smoothness criteria to form an optimized path. Evolutionary algorithms based on swarm intelligence can iteratively search for local optimal solutions, but this method is difficult to expand to online and real-time optimization due to its limited speed, and it is not suitable for use in reconnaissance tasks.

In recent years, with the rapid development of Deep Reinforcement Learning (DRL), its powerful representation and learning capabilities have enabled it to perform well in decision-making problems [12]; therefore, researchers are beginning to explore the application of reinforcement learning in multi-UAV path planning and navigation [13–15]. Compared with traditional algorithms, reinforcement learning performs better when the environment is unknown and dynamic. Moreover, the inference speed and generalization of reinforcement learning are advantages in real-time decision-making tasks.

In our research, the perception abilities of multi-UAV were limited, and only partial observations of the environment were made, meaning that it was difficult for the multi-UAV to make the optimal decisions when global states were lacking because the state transitions were unknown. The action of each UAV could change the environment's state, especially in a learning-based algorithm, such as reinforcement learning, the incomplete information will lead to poor efficiency and convergence. Moreover, it is vital to design training architecture to coordinate multi-UAVs to achieve a common goal. It is unwise to adopt a completely distributed training architecture to solve MAPF problems because of the high complexity. The same applies to multi-objective optimization problems. Lowe, firstly, proposed a framework of centralized training with decentralized execution [16], allowing extra information to be used in policies to make training easier. This framework has been proved to be capable of handling collaborative problems, such as multi-agent path planning. For instance, Jose et al. [17] proposed a DRL model with a centralized training and decentralized execution paradigm to solve vehicles routing problem, which was shown to be able to produce near-optimal solutions through cooperative actions. Marc et al. [18] adopted a distributed multi-agent variable framework to solve conflicts between UAVs, and also to train agents using centralized learning. Wang et al. [19] adopted a centralized training and decentralized executing framework to enable dynamic routing, introducing a counterfactual baseline scheme to improve the convergence speed. Moreover, the reward function of reinforcement learning should be reviewed in light of multi-objective optimization problems. On the one hand, a reward function that is too simple maybe cause "Reward Hacking" [20] and exacerbate the difficulties of policy learning due to incomplete information. On the other hand, a reward function that is too complex will lead the worse generalization. The most commonly used solution is to design a reward to satisfy multiple objectives of different weights according to the prior knowledge, in which a multi-objective optimization problem will be changed into a singe-objective optimization problem. In fact, this solution is near-optimal. Li [21] proposed an end-to-end framework for use in solving multi-objective optimization problems using deep reinforcement learning. Xu [22] proposed prediction-guided multi-objective reinforcement learning for use in solving continuous robot control problems. In multi-UAV path planning, some constraints, such as time cost, security, and coverage, must be considered.

To solve these problems, we proposed an improved multi-agent reinforcement learning algorithm based on centralized training and decentralized execution architecture. The policy learning algorithm is proximal policy optimization (PPO) [23]. It is a model-free reinforcement learning algorithm, which can adapt to a dynamic environment and provide good generalization. The critic network of PPO is used to coordinate all the UAVs to maximize team returns through centralized training by receiving joint observations, and the actor network of PPO is used to output actions. We also added a recurrent neural network to the actor–critic network to gather the historical information from the hidden state of the recurrent neural network [24], which solves the problem of incomplete information caused by partial observations. In addition, we designed a joint reward function to guide multi-UAV to learn optimal policies. When the training stage is completed, each UAV can execute an action based on its local observations in the reference stage. The contributions of our research are as follows:

1. We solved the problem caused by multi-UAV path planning with incomplete information through reinforcement learning based on the centralized training and decentralized execution architecture. We deeply explored the reasons why centralized training and decentralized execution architecture improves model performance, and we explained the benefits of centralized training compared to fully distributed methods.

2. When designing the reward function, we decomposed the multi-objective optimization problem into multiple sub-problems based on the idea of decomposition, solving the multi-objective optimization problem through reinforcement learning.

Experiments show that by using our method, the performance was significantly improved compared with baselines, and we demonstrated the high application value of reinforcement learning in multi-UAV path planning. In the execution stage, our method could be used to plan paths online, far exceeding the speed of heuristic algorithms. Section 2 introduces the backgrounds of our research. Section 3 describes our methodologies in details. Section 4 introduces the experimentation setup and results. We provide a conclusion in Section 5.

2. Background

2.1. Problem Description

When multi-UAV perform reconnaissance missions, they need to make real-time decisions based on current state information, and a collision-free path to reach the target points must be planned. In addition, time cost and coverage need to be considered. Therefore, multi-UAV autonomous path planning is a online decision-making problem under the constraints of incomplete information. It has three characteristics: distributed decision-making, partial observation and multi-objective optimization. Multi-UAV autonomous path planning is considered to be a fully cooperative task. The objective of all the participants in such a task is to obtain the maximum team returns. Therefore, we could establish a multi-agent real-time sequential decision-making model by the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) theoretical framework [25].

Dec-POMDP is a general model used to solve multi-agent objective optimization problems in cooperative environments, which generalizes the Partially Observable Markov Decision Process (POMDP) to multi-agent environments. It allows for the distributed control of multiple agents which may not be able to observe global states of environment. In every step, each agent chooses an action based on local observations (all agents in parallel), and then obtains its own reward from the environment, and all of agents cooperate to obtain common long-term benefits and maximize returns. Generally, a Dec-POMDP model is described by a tuple: $\langle I, S, \{A_i\}, T, R, \{\Omega_i\}, O, h \rangle$

- I is a set of N agents.
- S is a set of states of the environment, and S_0 is the initial state.
- $\{A\}$ is a set of actions for the agents. It is an action tuple $A_1, A_2, \ldots, A_i$.
- T is the state transition probability function $P(S'|S, A)$.
- R is the reward when agents take actions $\{A\}$ in state S, it depends on all the agents.
- $\{\Omega\}$ is a set of observations for the agents.
- O is a table of the observation probabilities, where $O(o_1, o_2, \ldots, o_i|S', A)$ is the probability that $(o_1, o_2, \ldots, o_i)$are observed by all the agents, respectively.
- h is the maximum number of steps in an episode which is called "horizon".

However, the complexity of the optimal solution of this distributed model is

$$O\left[\left(|\mathcal{A}|^{\frac{|o|^h-1}{|o|-1}}\right)^n\right] \tag{1}$$

which is double exponential [26]; it is hard to compute directly, and reinforcement learning is usually used to obtain the approximate solution.

2.2. Actor–Critic Algorithm

In reinforcement learning, an agent interacts with the environment continuously to optimize the policy through the feedback (reward) given by the environment. Reinforcement learning is mainly divided into value-based methods and policy-based methods. A policy-gradient algorithm can easily select the appropriate action in the continuous action space, while value-based algorithm cannot. However, the limitation of the policy-gradient algorithm is its poor learning efficiency. Therefore, researchers proposed a method that combines the policy-gradient and value-based algorithms, called the actor–critic algorithm [27]. The architecture of actor–critic is shown in Figure 1. Actor–critic uses a value-based network and policy-based network as the critic network and the actor network, respectively. The critic network can realize single-step updates to overcome the poor learning efficiency, and the actor network outputs actions according to the current observation, while the critic network can judge whether the current action is good or bad, which can lead the actor network to output a better action. Currently, the algorithms based on the actor–critic framework, such as DDPG, PPO, and A3C, are very popular.

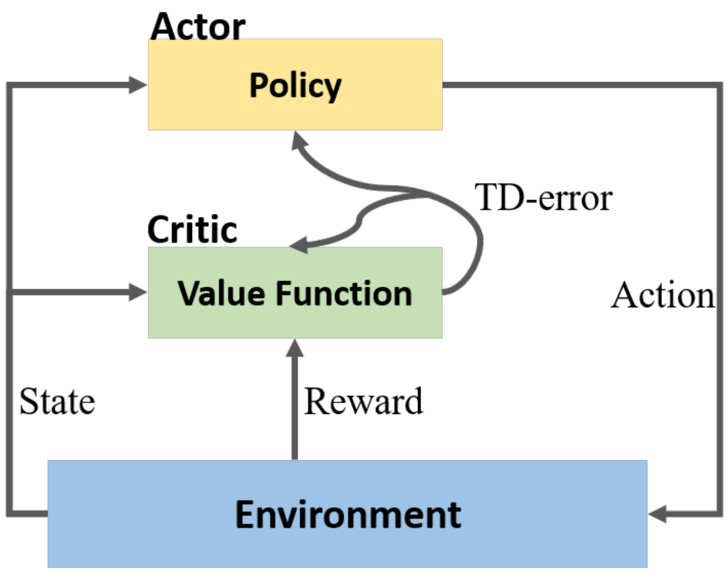

Figure 1. Actor–critic algorithm.

2.3. Centralized Training and Decentralized Execution Architecture

When there are multiple agents in a completely cooperative environment, we can establish a Dec-POMDP framework. Reinforcement learning is a great method to seek the optimal solution of a model. However, if we directly use single-agent reinforcement learning algorithms to train agents independently, it is hard to converge them, because the actions of each agent will change the environment, meaning the environment will be unstable for each agent and lead to learning difficulties. The MADDPG trains multi-agents through a centralized critic network, providing a good solution for the training of multi-agent systems. As shown in Figure 2, the input of the critic network is the joint observation of all of the agents in the environment in the training stage, and the actor network only inputs its own local observations and output actions according to the observations in the inference stage. This architecture enables each agent's actions in the environment to be observed by other agents, ensuring the stability of the environment. Therefore, multi-agent reinforcement learning is mostly based on centralized training and decentralized execution (CTDE) architecture, such as COMA [28].

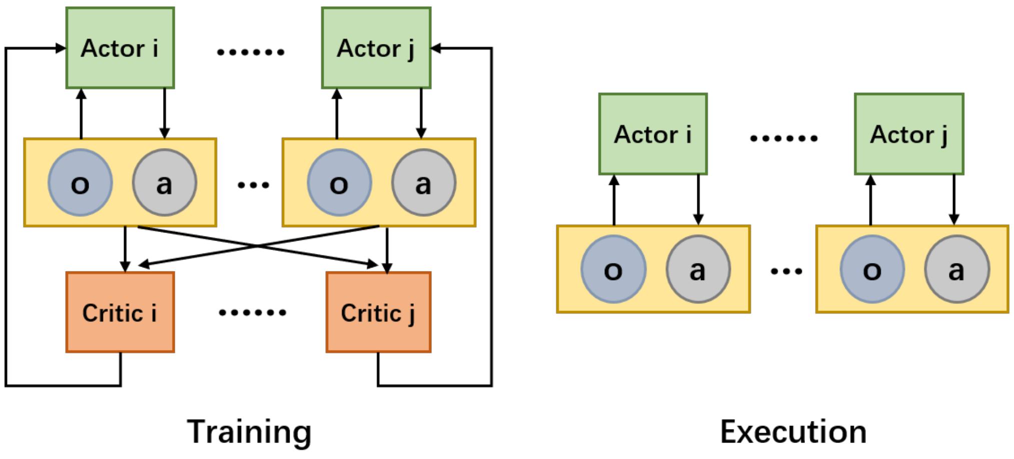

Figure 2. Centralized training (**left**) and decentralized execution (**right**).

3. Methodology

3.1. Proximal Policy Optimization with CTDE

We choose proximal policy optimization (PPO) to guide UAVs in learning policies. The PPO algorithm is based on the actor–critic architecture, which can more effectively achieve continuous control in high-dimensional space, and it is also an on-policy reinforcement learning algorithm. The learning approach of PPO is policy gradient. However, the policy-gradient algorithm is unstable, and this makes it difficult to choose an appropriate steps. If the difference between the old policy and new policy is too great during the training process, it is not conducive to learning. Using the PPO algorithm, a new objective function was proposed which can be updated in small batches in multiple training steps, which solves the problem of steps being difficult to determine in the policy-gradient algorithm. The algorithm takes into account the difference between an old network and an new network when updating parameters. In order to avoid the difference being too great, a clip is introduced to limit:

$$\nabla\overline{\mathcal{R}}(\tau) = E_{\tau\sim\pi\theta(\tau)}[A^{\pi}(s_t,a_t)\nabla log p_{\theta}(a_t|s_t)] \tag{2}$$

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{3}$$

$$\mathbb{E}_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] \tag{4}$$

$$\mathcal{L}^{CLIP}(\theta) = \hat{E}_t\left\{min\left[\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\right)\hat{A}_t, clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right]\right\} \tag{5}$$

where A is an advantage function, indicating the return of the action a in the current state.

Based on the PPO algorithm, we adopted the training method of the CTDE architecture and designed a multi-agent PPO algorithm in a multi-agent environment. Compared with the single-agent environment, the critic network's input is the joint observation of multi-UAV, which is equivalent to a central controller, each drone can obtain more information. The actor network is updated to maximize the objective:

$$\mathcal{L}(\theta) = \frac{1}{\mathcal{B}n}\sum_{i=1}^{\mathcal{B}}\sum_{k=1}^{n}\left[min\left(r_{\theta,i}^k \mathcal{A}_i^k, clip\left(r_{\theta,i}^k, 1-\epsilon, 1+\epsilon\right)\mathcal{A}_i^k\right) + \sigma * \mathcal{S}_\pi\right] \tag{6}$$

where $r_{\theta,i}^k = \frac{\pi_\theta\left(a_i^k|o_i^k\right)}{\pi_{\theta_{old}}\left(a_i^k|o_i^k\right)}$. The critic network is updated to minimize the value loss:

$$\begin{gathered}\mathcal{L}(\phi) = \frac{1}{\mathcal{B}n}\sum_{i=1}^{\mathcal{B}}\sum_{k=1}^{n} max\left[\left(\mathcal{V}_\phi\left(s_i^k\right) - \widehat{\mathcal{R}}_i\right)^2,\right. \\ \left.\left(clip\left(\mathcal{V}_\phi\left(s_i^k\right), \mathcal{V}_{\phi_{old}}\left(s_i^k\right) - \varepsilon, \mathcal{V}_{\phi_{old}}\left(s_i^k\right) + \varepsilon\right) - \widehat{\mathcal{R}}_i\right)^2\right]\end{gathered} \tag{7}$$

The weights of two networks are updated in every episodes, the process is shown in Figure 3.

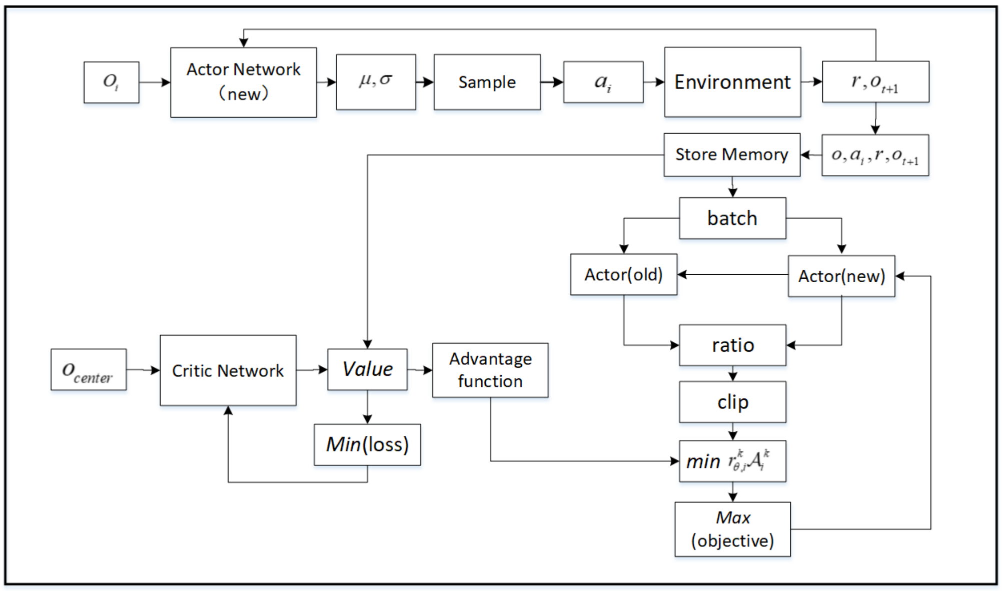

Figure 3. The weights of actor–critic network are updating in every episodes.

3.2. Adding RNN Layer For Incomplete Information

One of the difficulties within multi-UAV autonomous path-planning tasks is partial observation, which leads to limited information being obtained by UAVs. A solution to this is the utilization of the previous state to avoid falling into a local optimum.

Recurrent neural networks can memorize the previous information and apply it to the calculation of the current output. The nodes between the hidden layers are connected, and the input of the hidden layer includes the current input and the previous output, as well as the output of the hidden layer at the moment. The study of deep recurrent Q-learning (DRQN) was the first to combine an RNN with reinforcement learning [29]. As shown in Figure 4, DRQN essentially turns one of the linear layers of DQN into an RNN layer. Due to the addition of RNN, DRQN has short-term memory, and it can achieve similar scores to DQN in "Atari Games" without frame stack technology. Taking inspiration from this idea, we added the RNN layer to the PPO network, and we used the RNN layer to process historical information to solve the problem of incomplete information in the training process.

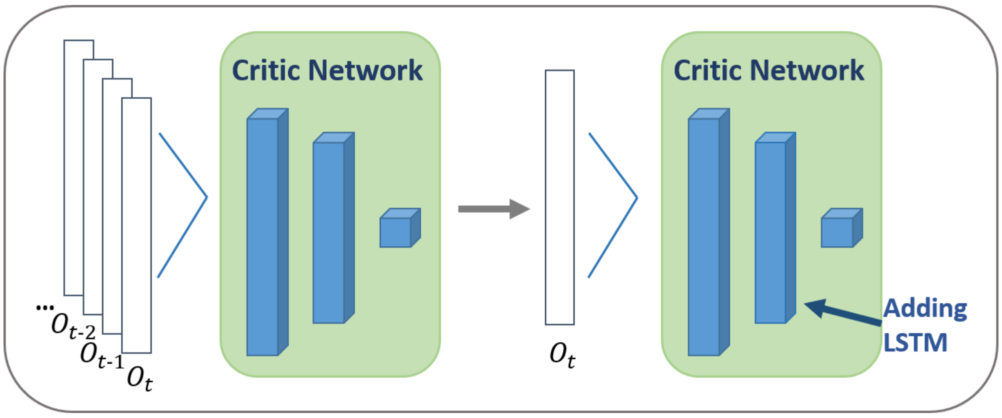

Figure 4. Adding LSTM layer have the same effect compared with frame stack, and reduce the dimension of input.

However, RNNs suffer from short-term memory. If a sequence is too long, it is difficult to transfer information from an earlier time step to a later time step. During back propagation, the gradient easily vanishes. Long Short-Term Memory (LSTM) is a variant of the RNN. It can select the information to be remembered or forgotten through the gate mechanism. As shown in Figure 5, the forget gate determines which relevant information in the previous step needs to be retained; the input gate determines which information in the current input is important and needs to be added; the output gate determines what the next hidden state should be. These "gates" can keep the important information in the sequence and discard the useless information, preventing the gradient from vanishing.

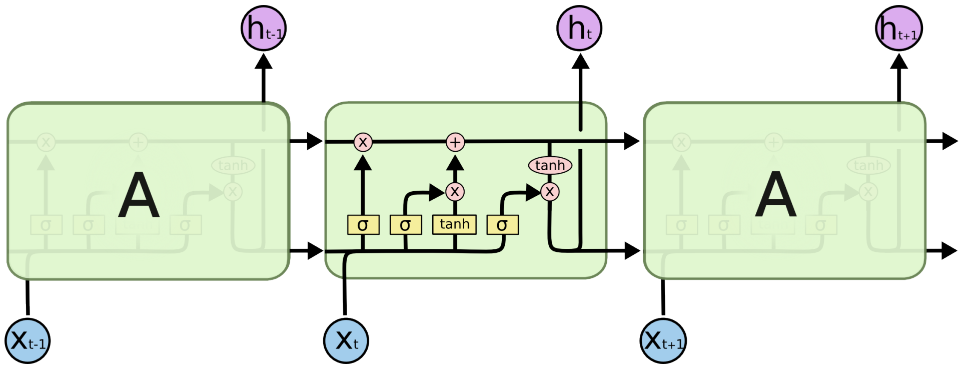

Figure 5. LSTM structure: at timestep t, X_t is input, C_t is cell state, and h_t is hidden state.

$$\begin{cases} f_t & = \sigma\left(W_f \bullet [h_{t-1}, x_t] + b_f\right) \\ i_t & = \sigma(W_i \bullet [h_{t-1}, x_t] + b_i) \\ \widetilde{C}_t & = \tanh(W_c \bullet [h_{t-1}, x_t] + b_c) \\ C_t & = f_t * C_{t-1} + i_t * \widetilde{C}_t \\ o_t & = \sigma(W_o \bullet [h_{t-1}, x_t] + b_o) \\ h_t & = o_t * \tanh(C_t) \end{cases} \tag{8}$$

where C_t is the cell state, and h_t is the current hidden state. Therefore, we added LSTM layers to both the actor and critic networks of PPO. After adding the LSTM layer, the historical information was remembered by updating the cell state and hidden state at each timestep T, and the problem of incomplete information caused by the local observations was alleviated.

Batches of τ are used to update the parameters of actor and critic networks to maximize $\mathcal{L}(\theta)$ and minimize $\mathcal{L}(\phi)$ through gradient descent. $\tau = [s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}, a_{t+1} \ldots]$ Add an LSTM layer to the network, two elements $h_{t,\pi}$ and $h_{t,V}$ are added to τ, which changed into $[s_t, o_t, h_{t,\pi}, h_{t,V}, a_t, r_t, s_{t+1}, o_{t+1}, h_{t+1,\pi}, h_{t+1,V}, a_{t+1} \ldots]$, $h_{t,\pi}, h_{t,V}$ are the hidden state of timestep t in an LSTM layer of the actor network and critic network.

$$\mathcal{L}(\theta) = \frac{1}{\mathcal{B}n} \sum_{i=1}^{\mathcal{B}} \sum_{k=1}^{n} \left[min\left(r_{\theta,i}^{k} \mathcal{A}_i^k, clip\left(r_{\theta,i}^{k}, 1-\epsilon, 1+\epsilon \right) \mathcal{A}_i^k \right) + \sigma * \mathcal{S}_\pi \right] \tag{9}$$

Moreover, we sought to establish the different roles of the critic network and actor network in the multi-agent reinforcement learning algorithm based on the CTDE architecture. We believe that the critic network acts as a central controller to process all observation information, meaning that adding an RNN layer to the critic network can, theoretically, greatly enhance the performance of the model in partially observable environments. The actor network acts as a policy network for each agent, and adding the RNN layer to the actor network has less of an effect on its performance. The experimental results prove our analysis, it also confirms that the CTDE architecture is effective in this task.

3.3. Multi-Objective Joint Optimization

Multi-UAV autonomous path planning is a multi-objective optimization. The time cost, coverage area, and security of multi-UAV systems need to be considered in Figure 6. Multi-objective optimization is the optimal selection of decision variables in a discrete decision space.

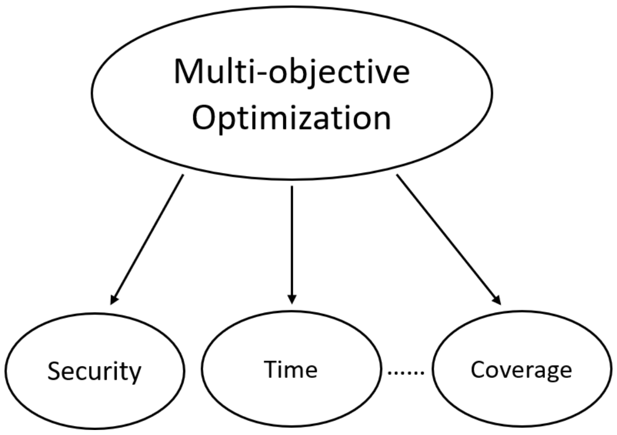

Figure 6. Multi-objective optimization seeks optimal solutions under the constraints of security, time, and coverage.

The mathematical expression is as follows:

$$max_\pi F(\pi) = max_\pi [f_1(\pi), f_2(\pi), \ldots, f_m(\pi)] \tag{10}$$

where m is the number of the objectives,and π is the policy.

This is very similar to the "action selection" of reinforcement learning, and the "offline training, online decision-making" characteristic of deep reinforcement learning make it possible for an online, real-time solution of a multi-objective optimization problem to be achieved. Therefore, deep reinforcement learning methods are a good choice when used to solve traditional multi-objective optimization problems, and the learning-based model has a good generalizability.

We designed the reward function based on multi-objective optimization and combined it with prior knowledge regarding navigation, decomposing the multi-objective optimization problem into multiple sub-problems. We shaped a joint reward function by considering the constraints of security, time, and coverage. There are several ways for multi-UAV to obtain feedback.

$$r_{total} = \alpha * r_{timecost} + \beta * r_{security} + \gamma * r_{coverage} \tag{11}$$

where $r_{security} = \sum |distance(UAV_i - UAV_j)| - |distance(UAV_i - target_i)|$, which guides the multi-UAV to reach the target points and avoid each other. The purpose of this design is to achieve larger coverage by distributing all of the UAVs, ensuring that the drone does not collide with other drones or obstacles. $r_{coverage} = \sum new\ area_{UAVi}$, it means the UAV will be rewarded if new areas are explored; this reward encourages multi-UAV to explore an environment, not simply reach the required points. $r_{timecost}$ guides the drone to reach the target point with the shortest possible number of steps. These three different rewards constitute the reward function that guides the multi-UAV autonomous path planning under constraints.

Usually, we give these rewards different weights to change a multi-objective problem into single-objective problem and to seek a solution. The advantage of this design is that the aggressiveness of the agent's learning strategy can be changed amending intended meaning has been retrained the manually set rules, but this is a near-optimal solution under the constraint. A multi-objective optimization problem can be solved through multi-objective reinforcement learning, as shown in Figure 7.

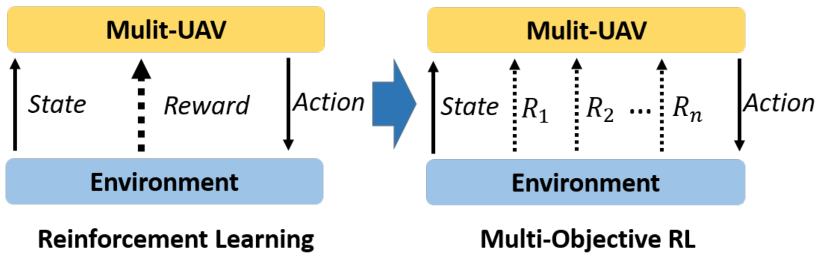

Figure 7. The difference between reinforcement learning and multi-objective reinforcement learning.

A set of solutions called "Pareto front" can represent the optimal solutions in all of the different weights. The differences between two methods are as shown in Figure 8.

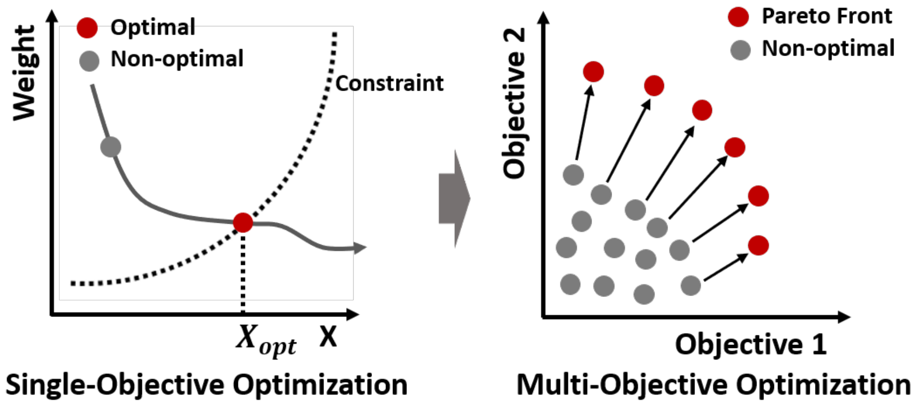

Figure 8. Multi-objective optimization: obtaining the optimal solution under the constraints of security, time, and coverage.

A multi-objective gradient optimizes the policy to maximize the weight–sum reward, where w is the weight of every objective, meaning that the policy gradient has changed.

$$\mathcal{J}(\theta,\omega) = \omega^T F(\pi) = \sum_{i=1}^{m} \omega_i f_i(\pi) = \sum_{i=1}^{m} \omega_i J_i^{\pi} \tag{12}$$

$$\begin{aligned}\nabla_\theta \mathcal{J}(\theta,\omega) &= \sum_{i=1}^{m} \omega_i \nabla_\theta J_i(\theta) \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \omega^T A^{\pi}(s_t,a_t) \nabla_{\pi\theta}(a_t \mid s_t)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} A_{\omega}^{\pi}(s_t,a_t) \nabla_{\pi\theta}(a_t \mid s_t)\right]\end{aligned} \tag{13}$$

4. Experiment

4.1. Experimental Setup

We built a simulation platform based on Unreal Engine 4 to support quad-rotor dynamics simulation. In this platform, we created a scenario to simulate multi-UAV reconnaissance missions, as shown in Figure 9. The reconnaissance area was 2 km × 2 km, and the scenario contained four movable anti-drone devices. Once the drone entered the coverage area of these devices, it would be destroyed. The perception radius of a drone was 200 m × 200 m, and all of the drones communicated with each other by default.

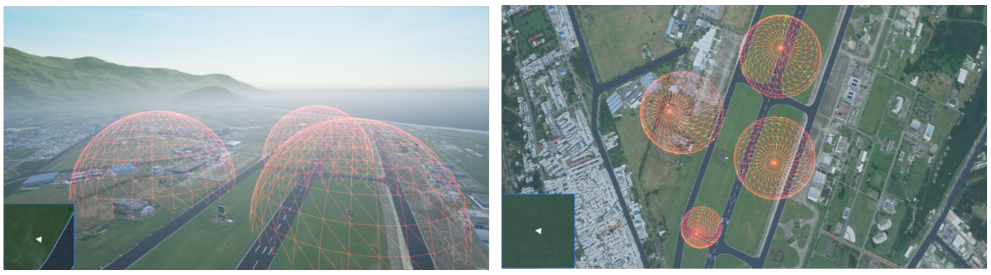

Figure 9. Multi-UAV simulation platform.

Three UAVs started from the starting points and planed a collision-free path to three target points online. The area covered by all of the UAVs was the final total coverage area, and the total path length of the UAVs was the path cost. Figure 10 shows that three UAVs started from different points, and there were four threat areas in the environment. By default, the drones could only perceive dangerous areas within their capability radius.

In this simulation platform, low-level and high-level commands were used to control the motion of a UAV. As shown in Figure 11, in order to simulate a real flight, we choose to control the motion of the UAV through the underlying control method. The policy network outputs (*pitch*, *roll*, *yaw_rate*, *throttle*, and *duration*) a five-dimension vector in every step, where *pitch*, *roll*, and *yaw_rate* controlled the attitude and direction of a UAV, and *throttle* and *duration* made the UAV to accelerate or decelerate for a period of time.

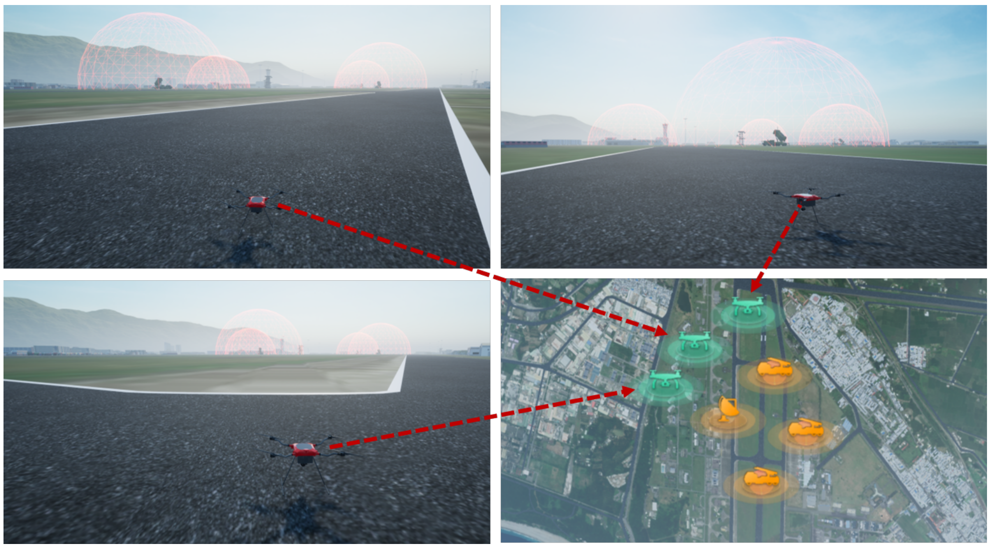

Figure 10. Initial state when multi-UAV perform a reconnaissance task.

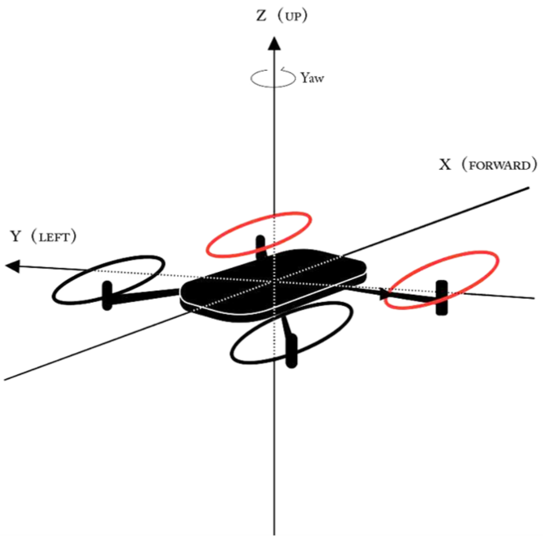

Figure 11. Kinematics of a quad-rotor.

4.2. Network Architecture

We used PyTorch to build a three layers neural network for the actor and critic networks of PPO, respectively. We used a centralized training and decentralized execution architecture to coordinate all of the UAVs; the intuitive difference between centralized training and independent training is the input of the value network. In this experiment, we connected the local observations of all of the UAVs into a high-dimensional vector as the joint observations, and then input the value network, called O_{center}, and the actor network input was the observation O_i of each UAV itself as shown in Figure 12. We set up four control experiments to compare the performance between CTDE and independent training in this task. The first and third layers of the networks were fully connected layers, and the second layer was an LSTM layer. In order to validate whether adding lstm was effective, we use the same network architecture to build a network without an LSTM layer as a comparison with the specific aim of verifying which one of critic and value was more dependent on historical information, thus confirming the role of CTDE architecture.

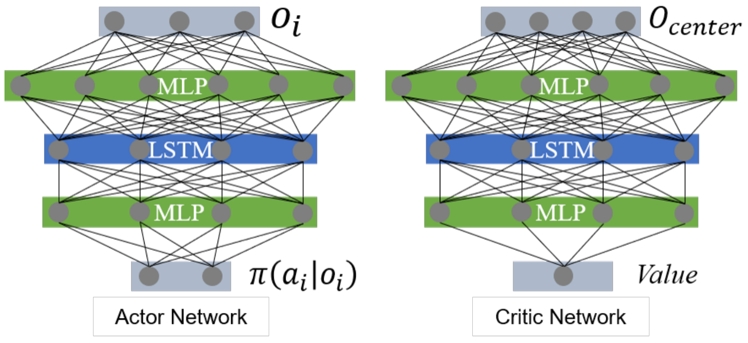

Figure 12. Actor network and critic network, and all of the agents that share the networks.

PPO outputs random policies, meaning that the outputs of the actor network are μ, σ, which are the expectation and variance of a Gaussian distribution, and the output action is randomly sampled by this Gaussian distribution. In the experiment, all of the agents shared common networks parameters as shown in Table 1.

Table 1. Network parameter table.

Episode	Episode length	Rollout thread	Clip	Discount	Entropy coefficient
625	200	16	0.2	0.99	0.1
Buffer size	**Batch size**	**FC layer dim**	**RNN hidden dim**	**Activation**	**Optimizer**
500	32	128	64	Relu	Adam

4.3. Results

After 1,000,000 steps of training, by analyzing the experimental results, we came to the conclusion that the PPO algorithm based on the centralized training decentralized execution architecture performed better compared to independent training in multi-UAV autonomous path planning tasks. As the results show in Figure 13, it is difficult for a completely independent and distributed training method to perform well in multi-UAV tasks. The adoption of CTDE architecture obviously and significantly improved the performance, the reward became positive, and the performance was even improved when the number of UAVs was larger. This proves that CTDE architecture is effective in such distributed tasks. A center controller can coordinate all of the UAVs. However, it does not indicate the number of UAVs, which can be unlimited. In fact, we found when there was more than six UAVs, the center controller could not effectively handle it, due to the dimension of joint observation being too high.

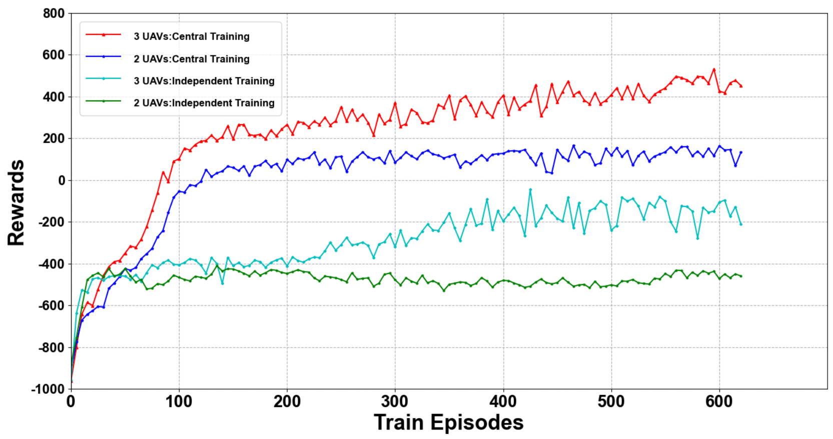

Figure 13. Comparison of the CTDE and independent architecture.

In addition, we carried out a set of control experiments to verify whether adding the RNN layer could solve the problem of multi-UAV learning difficulties with incomplete information. According to the experimental results in Figure 14, we found that adding the RNN layer to both the actor and critic networks significantly improved the performance of the model. Adding the RNN layer to the critic network also achieved practically the same effect, with the convergence speed being slower. The method of only adding the RNN layer to the actor network did not significantly improve the model performance, and it failed to solve the problem caused by partial observations of multi-UAVs. This result also verified our analysis: in the CTDE architecture, the critic network is the central controller, it coordinates all of the UAVs to complete common goals through the input of joint observations. The addition of the RNN layer to the critic network is effective, and the problem of incomplete information is solved through the hidden state.

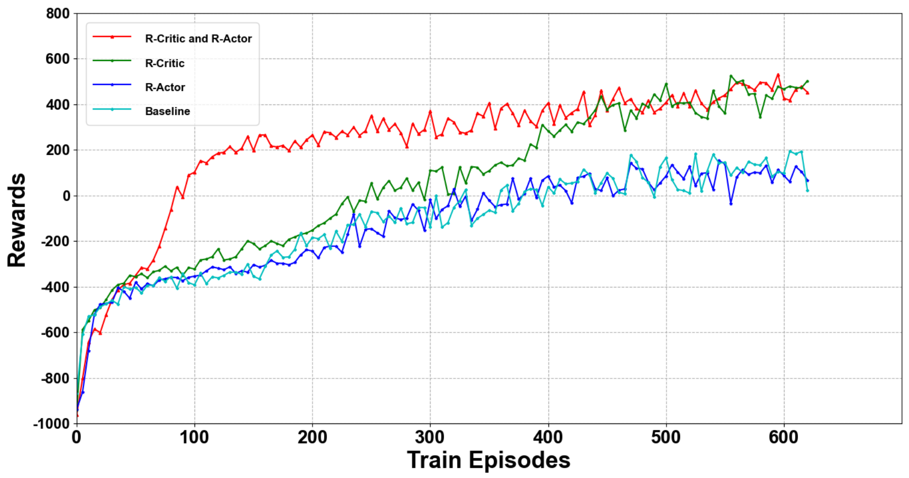

Figure 14. After the addition of the LSTM layer, better performance in an average reward was achieved.

In order to solve the decision-making problem with incomplete information, we chose CTDE architecture and added RNN layer to utilize historical information. In model-free reinforcement learning algorithms, reward represent an important evaluation criterion. Similarly, value loss, policy loss, and action entropy are also key components to evaluating algorithm performance. Value loss evaluates a value output of critic network and determines whether the prediction is accurate, and the action entropy reflects the randomness of the actor network strategy output. Here, we hoped that the action entropy would be larger enough to facilitate adequate exploration. The experimental results prove that our algorithm significantly improved the performance. As shown in Figure 15, after about 300 episodes, the loss function begin to stabilize, and the rapid convergence of value loss also showed that the value predicted by the critic network was more accurate. Similarly, after adding an LSTM layer, the critic loss was decreased, and the policy entropy value descended smoothly, which was a good performance and meant that the agents did not fall into a local optimum. We did not want this value to descend too rapidly or too slowly. Policy entropy is the variance of the output actions, and a smooth curve shows that multi-UAVs have learned a stable policy after a sufficient exploration, because exploration is indispensable in reinforcement learning.

Moreover, we found that adding a LSTM layer greatly improved the performance of the algorithm. After adding an LSTM, the average reward is significantly increased, which proved that the agent could make more correct decisions, and the policy entropy and critic loss converge faster, which shows that our method for adding an LSTM to the network effectively utilized historical information. The parameters of the model were continuously updated during the training phase.

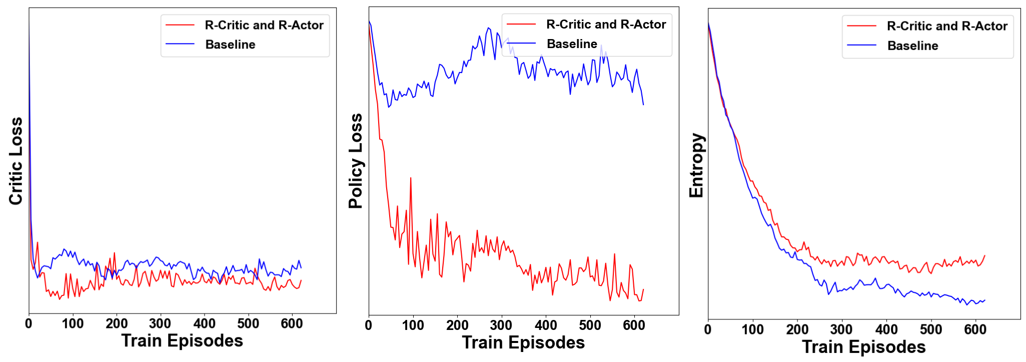

Figure 15. Critic loss, policy loss, and entropy.

Reinforcement learning is much faster than traditional swarm intelligence algorithms in the execution phase, and it is suitable for real-time decision-making tasks, as shown in Figure 16. Once the training stage was completed, the weight parameters of the networks were frozen during the execution phase. When we performed the navigation task with the trained model, the total average reward of our method was higher and stable.

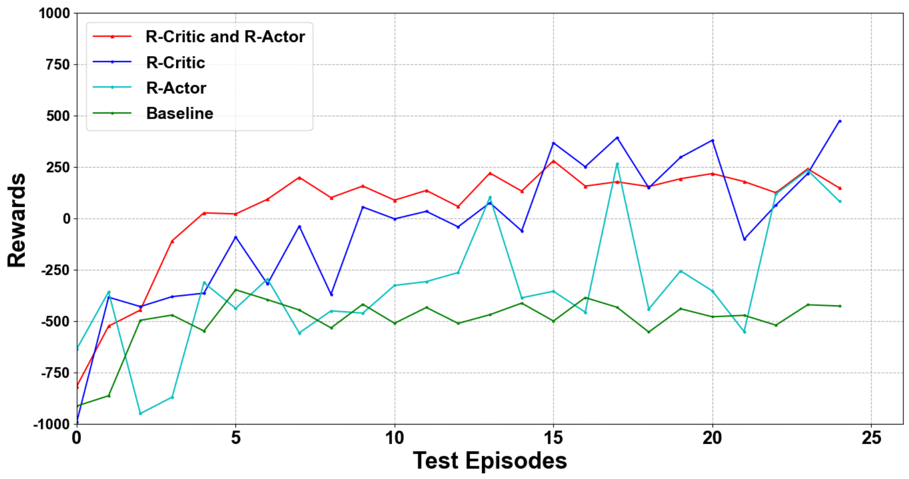

Figure 16. Our improved algorithm performs better in test.

In the simulation platform, multi-UAV realized path planning online by the pre-model in the inference stage as shown in Figure 17. We found that our method performs well under the constraints of security, time, and coverage. As shown in Table 2, compared with the state-of-the-art particle swarms optimization algorithms, our method has a better performance in many aspects especially the speed of reference. Reinforcement learning shows the powerful ability in real-time path planning task.

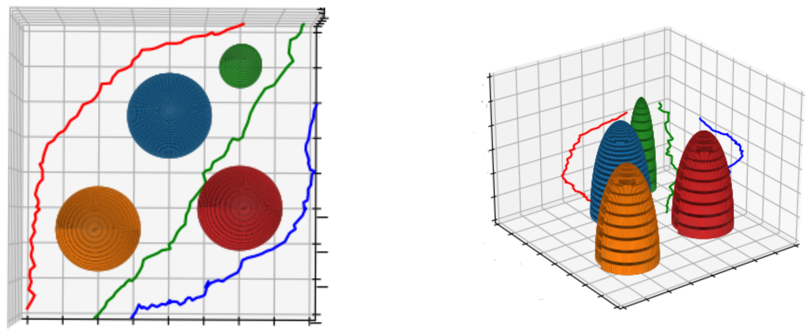

Figure 17. Multi-UAV path planning in three-dimensional environment through reinforcement learning.

Table 2. Performance comparison between our method and other state-of-the-art algorithms.

	Time Cost	Coverage	Success Rate	Reference Speed
Our method	92	57.9%	92.7%	0.126 s
RL baseline	93	54.5%	90.1%	0.115 s
DPSO	97	41.2%	65.2%	1.35 s
GAPSO	95	43.7%	62.1%	1.16 s

5. Conclusions

In this study, we proposed a multi-UAV autonomous path planning algorithm based on model-free reinforcement learning, which is able to adapt to dynamic environments. It was shown that the algorithm coordinates all of the UAVs through centralized training, which effectively lessens the difficulty of training distributed systems. When the training stage is completed, each UAV can make optimal decisions based on its own observations. We also introduced an RNN to remember historical information and prevent the model from falling into the local optimum due to incomplete information caused by partial observations. Finally, we designed a joint reward function to cooperatively guide the UAVs. Our experiments performs well in this type of task. Considering its communication capabilities in the real world, we plan to constrain the communication range and communication frequency between UAVs in follow-up research. The authors of [30,31] have contributed new ideas regarding the security of UAV communication. We believe this algorithm can be deployed to real drone swarms.

Author Contributions: Conceptualization, Y.C. and Q.D.; methodology, Y.C.; software, X.S.; validation, Y.C., Z.W. and J.W.; resources, Z.W.; data curation, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: The project was supported by Open Fund of Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation under Grant CSSAE-2021-003 and Grant CSSAE-2021-004.

Data Availability Statement: Not applicable.

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

References

1. Mohiuddin, A.; Tarek, T.; Zweiri, Y.; Gan, D. A survey of single and multi-UAV aerial manipulation. *Unmanned Syst.* **2020**, *8*, 119–147. [CrossRef]
2. Stern, R. Multi-agent path finding–An overview. In *Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 96–115.
3. Ma, H.; Wagner, G.; Felner, A.; Li, J.; Kumar, T.; Koenig, S. Multi-agent path finding with deadlines. *arXiv* **2018**, arXiv:1806.04216.
4. Ngatchou, P.; Zarei, A.; El-Sharkawi, A. Pareto multi objective optimization. In Proceedings of the 13th international Conference on, Intelligent Systems Application to Power Systems, Arlington, VA, USA, 6–10 November 2005; pp. 84–91.
5. Konak, A.; Coit, D.W.; Smith, A.E. Multi-objective optimization using genetic algorithms: A tutorial. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 992–1007. [CrossRef]
6. Tang, J.; Duan, H.; Lao, S. Swarm intelligence algorithms for multiple unmanned aerial vehicles collaboration: A comprehensive review. In *Artificial Intelligence Review*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–33.
7. Wang, Y.; Bai, P.; Liang, X.; Wang, W.; Zhang, J.; Fu, Q. Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms. *IEEE Access* **2019**, *7*, 105086–105099. [CrossRef]
8. Wan, Y.; Zhong, Y.; Ma, A.; Zhang, L. An Accurate UAV 3-D Path Planning Method for Disaster Emergency Response Based on an Improved Multiobjective Swarm Intelligence Algorithm. *IEEE Trans. Cybern.* **2022**. [CrossRef] [PubMed]
9. Shao, S.; Peng, Y.; He, C.; Du, Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization. *ISA Trans.* **2020**, *97*, 415–430. [CrossRef] [PubMed]
10. Krell, E.; Sheta, A.; Balasubramanian, A.P.R.; King, S.A. Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning. *J. Artif. Intell. Soft Comput. Res.* **2019**, *9*, 267–282 [CrossRef]
11. Ajeil, F.H.; Ibraheem, I.K.; Sahib, M.A.; Humaidi, A.J. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. *Appl. Soft Comput.* **2020**, *89*, 106076. [CrossRef]
12. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.

13. Luo, W.; Tang, Q.; Fu, C.; Eberhard, P. Deep-sarsa based multi-UAV path planning and obstacle avoidance in a dynamic environment. In *Proceedings of the International Conference on Swarm Intelligence*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 102–111.
14. Liu, C.H.; Ma, X.; Gao, X.; Tang, J. Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning. *IEEE Trans. Mob. Comput.* **2019**, *19*, 1274–1285. [CrossRef]
15. Bayerlein, H.; Theile, M.; Caccamo, M.; Gesbert, D. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open J. Commun. Soc.* **2021**, *2*, 1171–1187. [CrossRef]
16. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
17. Vera, J.M.; Abad, A.G. Deep reinforcement learning for routing a heterogeneous fleet of vehicles. In Proceedings of the 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guayaquil, Ecuador, 11–15 November 2019; pp. 1–6.
18. Brittain, M.; Wei, P. Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, NZ, USA, 27–30 October 2019; pp. 3256–3262.
19. Wang, Z.; Yao, H.; Mai, T.; Xiong, Z.; Yu, F.R. Cooperative Reinforcement Learning Aided Dynamic Routing in UAV Swarm Networks. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; pp. 1–6.
20. Ibarz, B.; Leike, J.; Pohlen, T.; Irving, G.; Legg, S.; Amodei, D. Reward learning from human preferences and demonstrations in atari. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, Canada, 3–8 December 2018; Volume 31.
21. Li, K.; Zhang, T.; Wang, R. Deep reinforcement learning for multiobjective optimization. *IEEE Trans. Cybern.* **2020**, *51*, 3103–3114. [CrossRef] [PubMed]
22. Xu, J.; Tian, Y.; Ma, P.; Rus, D.; Sueda, S.; Matusik, W. Prediction-guided multi-objective reinforcement learning for continuous robot control. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 10607–10616.
23. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
24. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
25. Amato, C.; Konidaris, G.; Cruz, G.; Maynor, C.A.; How, J.P.; Kaelbling, L.P. Planning for decentralized control of multiple robots under uncertainty. In Proceedings of the 2015 IEEE international conference on robotics and automation (ICRA), Seattle, WA, USA, 25–30 May 2015; pp. 1241–1248.
26. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840. [CrossRef]
27. Konda, V.; Tsitsiklis, J. Actor-critic algorithms. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999; Volume 12.
28. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
29. Hausknecht, M.; Stone, P. Deep recurrent q-learning for partially observable mdps. In Proceedings of the 2015 AAAI Fall Symposium Series, Arlington, VA, USA, 12–14 November 2015.
30. Krichen, M.; Adoni, W.Y.H.; Mihoub, A.; Alzahrani, M.Y.; Nahhal, T. Security Challenges for Drone Communications: Possible Threats, Attacks and Countermeasures. In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 9–11 May 2022; pp. 184–189.
31. Alrayes, F.S.; Alotaibi, S.S.; Alissa, K.A.; Maashi, M.; Alhogail, A.; Alotaibi, N.; Mohsen, H.; Motwakel, A. Artificial Intelligence-Based Secure Communication and Classification for Drone-Enabled Emergency Monitoring Systems. *Drones* **2022**, *6*, 222. [CrossRef]