sensors

MDPI

Article

Research on the Multiagent Joint Proximal Policy Optimization Algorithm Controlling Cooperative Fixed-Wing UAV Obstacle Avoidance

Weiwei Zhao [1,2], Hairong Chu [1,*], Xikui Miao [3], Lihong Guo [1], Honghai Shen [4], Chenhao Zhu [1,2], Feng Zhang [5] and Dongxin Liang [6]

1 Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, No. 3888, Dongnanhu Rd., Changchun 130033, China; zhaoweiwei215@mails.ucas.ac.cn (W.Z.); guolh@ciomp.ac.cn (L.G.); zhuchenhao17@mails.ucas.ac.cn (C.Z.)
2 University of Chinese Academy of Sciences, No. 19, Yuquan Rd., Beijing 100049, China
3 School of Information Engineering, Henan University of Science and Technology, Luoyang 471000, China; miaoxikui@gmail.com
4 Key Laboratory of Airborne Optical Imaging and Measurement, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, No. 3888, Dong Nanhu Road, Changchun 130033, China; shenhh@ciomp.ac.cn
5 School of Aviation Operations and Services, Aviation University of the Air Force, No. 2222, Dongnanhu Rd., Changchun 130022, China; zf00401@outlook.com
6 Xi'an Jiaotong University Health Science Center, No. 76, Yanta West Road, Xi'an 710061, China; abutfdevil@outlook.com
* Correspondence: chuhr@ciomp.ac.cn

Received: 10 June 2020; Accepted: 10 August 2020; Published: 13 August 2020

Abstract: Multiple unmanned aerial vehicle (UAV) collaboration has great potential. To increase the intelligence and environmental adaptability of multi-UAV control, we study the application of deep reinforcement learning algorithms in the field of multi-UAV cooperative control. Aiming at the problem of a non-stationary environment caused by the change of learning agent strategy in reinforcement learning in a multi-agent environment, the paper presents an improved multiagent reinforcement learning algorithm—the multiagent joint proximal policy optimization (MAJPPO) algorithm with the centralized learning and decentralized execution. This algorithm uses the moving window averaging method to make each agent obtain a centralized state value function, so that the agents can achieve better collaboration. The improved algorithm enhances the collaboration and increases the sum of reward values obtained by the multiagent system. To evaluate the performance of the algorithm, we use the MAJPPO algorithm to complete the task of multi-UAV formation and the crossing of multiple-obstacle environments. To simplify the control complexity of the UAV, we use the six-degree of freedom and 12-state equations of the dynamics model of the UAV with an attitude control loop. The experimental results show that the MAJPPO algorithm has better performance and better environmental adaptability.

Keywords: reinforcement learning; proximal policy optimization (PPO); the joint state-value function; multiagent cooperative; multiple unmanned aerial vehicles (multi-UAV) formation; obstacle avoidance

1. Introduction

The autonomy and intelligent development of the coordinated control of multi-agent systems such as multi-unmanned aerial vehicle (UAV) and multi-robot have received more and more attention. To solve the problem of coordinated control and obstacle avoidance of multiagent systems, researchers

have proposed various solutions, including rule-based methods, field methods, geometric methods, numerical optimization methods, and so on [1,2].

In recent years, the application and development of reinforcement learning (RL) in the field of robotics has attracted more and more attention. La et al. [3] integrated high-level behaviors by RL and low-level behaviors by flocking control to allow robots to learn to avoid predators/enemies collaboratively. Hung et al. [4] proposed a method of using the RL algorithm to fixed-wing UAV flocking, but the velocity and height of the UAV were set as constant to reduce the complexity of UAV control. Pham et al. [5] provided a framework for integrating Q-learning and Proportion Integration Differentiation (PID) to allow the UAV to navigate successfully. Koch et al. [6] investigated the performance and accuracy of the inner control loop, providing attitude control. Their investigations encompassed flight control systems trained with RL algorithms, including Deep Deterministic Gradient Policy (DDGP), Trust Region Policy Optimization (TRPO), and Proximal Policy Optimization (PPO) compared with PID controller. Fang Bin et al. studied the application of RL in the field of multi-UAV collaborative obstacle avoidance [2].

The success of reinforcement learning (RL) in the field of multiagent has been witnessed. However, it must also be pointed out that the development of multiagent reinforcement learning (MARL) is influenced by single-agent RL, which is the cornerstone of MARL development. There are three task environments in the multiagent field: cooperative tasks, competitive tasks, and mixed tasks [7]. We mainly consider the cooperative task environment. RL in the topic of multiagent cooperation is mainly used to optimize a common reward signal. The non-stationary environment due to the change of the policies of learning agents in a multiagent environment leads to the failure or difficulty of convergence of most single-agent RL algorithms in a multiagent environment [8,9]. Therefore, it is inevitable that we must find RL algorithms suitable for a multiagent environment.

Tan [10] explored the multiagent setting using independent Q-learning to complete some cooperative tasks in a 2D grid world. Tampuu et al. [11] manipulated the classical rewarding scheme of Pong to demonstrate how competitive and collaborative behaviors emerge by independent Q-learning (IQL). Foerster et al. [12] proposed a counterfactual multiagent (COMA) policy gradients algorithm by using a centralized critic to estimate the Q-function and decentralized actors to optimize the agents' policies and evaluated on StarCraft. Lowe et al. [13] extended Deep Deterministic Policy Gradient (DDPG) to a multiagent setting with a centralized Q-function and evaluated 2D games. Li et al. [14] proposed a novel minimax learning objective based on the multiagent deep deterministic policy gradient algorithm for robust policy learning. Yang et al. [15] proposed the Mean Field Reinforcement Learning (MFRL) algorithm to address MARL on a very large population of agents. Value-Decomposition Networks (VDN) [16] learned a centralized action–value function as a sum of individual action–value functions and a decentralized policy. QMIX [17] is an effective improvement of the VDN algorithm but unlike VDN. QMIX can learn a complex centralized action–value function with a factored representation that scales well in the number of agents and allows decentralized policies to be easily extracted via linear-time individual argmax operations. Hong et al. [18] introduced a deep policy inference Q-network (DPIQN) and its enhanced version deep recurrent policy inference Q-network (DRPIQN) to employ "policy features" learned from observations of collaborators and opponents by inferring their policies. Bansal et al. [19] explored that a competitive multiagent environment trained with self-play can produce complex behaviors than the environment itself using Proximal Policy Optimization (PPO). There are also many review articles [7–9] that explore the research and application of RL in the field of multiagent.

In order to make RL algorithms useful in the real world, researchers have exerted a lot of effort [3,5,6]. The unmanned aerial vehicle (UAV) is a popular tool in the military and civilian fields, and it is also the representation of agents in the real world. The UAV, as the research object of RL algorithms, has attracted the attention of many researchers [5,6,20]. Multi-UAV formation can accomplish many tasks that cannot be completed by a single UAV [21,22].

This paper proposes the Multiagent Joint Proximal Policy Optimization (MAJPPO) algorithm, which uses the moving window average of the state–value functions of different agents to get the centralized state–value function to solve the problem of multi-UAV cooperative control. The algorithm can effectively improve the collaboration among agents in the multiagent system than the multiagent independent PPO (MAIPPO) algorithm. Since the PPO algorithm uses a state–value function as the evaluation function, it is different from Deep Q-Network (DQN) [23], which uses the action–value function as the evaluation function. Therefore, the centralization value function of the MAJPPO algorithm does not require the policies of collaborative agents during training, thereby reducing the complexity of the algorithm. Finally, we use algorithms to train multi-UAV formation through a multi-obstacle environment to evaluate the performance of the algorithm. In the process of reinforcement learning of UAV, there are two choices of the controlled object. One is the use of the UAV dynamics model with an attitude control loop, and the other is the use of the UAV dynamics model without an attitude control loop. We use the UAV dynamics model with the attitude control loop as the control object of multi-UAV cooperative control. This is mainly because the UAV dynamic model with the attitude control loop has less freedom and fewer optimization targets.

Therefore, the main contributions of the paper are as follows:

1 The development of the MAJPPO algorithm; and,
2 The MARL algorithm is applied to the multi-UAV formation and obstacle avoidance field.

In regards to the rest of the paper, we will firstly introduce the background related to this paper in Section 2. Section 3 describes the PPO algorithm. Section 4 presents the independent PPO algorithm for the multiagent environment. Section 5 describes the novel MAJPPO algorithm, and it brings in some discussion. Section 6 describes the dynamics model of small UAV with the attitude control loop, RL of a single UAV, and the basic settings of the formation. Section 7 introduces experiments and analysis. The conclusions appear in Section 8.

2. Background and Preliminary

In the field of RL, the Markov decision process (MDP) is a key concept. RL enables the agent to learn a policy with good profits through interaction with the environment in an unknown environment. Such environments are often formalized as Markov Decision Processes (MDPs), described by a five-tuple $(S; A; P; R; \gamma)$. At each time step t, an agent interacting with the environment observes a state $s_t \in S$, and chooses an action $a_t \in A$, which determines the reward $r_t \sim R(s_t; a_t)$ and next state $s_{t+1} \sim P(s_t; a_t)$. The purpose of RL is to maximize the cumulative discount rewards $G_t = \sum_{\tau=t,\ T} \gamma^{\tau-t} r_\tau$, where T is the time step when an episode ends, t denotes the current time step, $\gamma \in [0, 1]$ is the discount factor, and r_τ is the reward received at the time step τ. The action–value function (abbreviated as Q-function) of a given policy π is defined as the expected return starting from a state–action pair (s, a), expressed as $Q^\pi(s, a) = E[G_t | s_t = s, a_t = a, \pi]$. Q-learning is a widely used RL algorithm. Q-learning mainly uses the action–value function $Q^\pi(s, a) = E[G_t | s_t = s, a_t = a, \pi]$ to learn the policy [24]. DQN [23] is a kind of RL algorithm combining Q-learning and neural network, which learns the action–value function Q * corresponding to the optimal policy by minimizing the loss: $L(\theta) = E_\pi\left[(Q_\theta(s, a) - y)^2\right]$, where $y = r + \gamma max Q_{\theta'}(s', a')$, and y represents the Q-learning target value.

In the real world, the agent often cannot obtain all the information of the environment, or the environment information obtained by the agent is incomplete and noisy, that is, only part of the environment can be observed. In this case, we can use the partially observable Markov decision process (POMDP) to model such problems. A POMDP can be described as a six-tuple $\langle S; A; P; R; \gamma; O \rangle$, where O is the observation perceived by the agent. The deep recurrent Q-network (DRQN) [25] is proposed to deal with partially observable problems and POMDP problems, which extends the architecture of DQN with Long Short-Term Memory (LSTM).

In a multiagent learning domain, the POMDP generalizes to a stochastic game or a Markov game.

A multiagent learning environment can be modeled as a decentralized POMDP (Dec-POMDP) framework [26]. The Dec-POMDP model extends single-agent POMDP models by considering joint actions and observations.

Solving the Dec-POMDP problem is at the core of the MARL algorithms. The mainstream solution is to optimize the decentralized policy by centralized learning, such as MADDPG, VDN, and QMIX.

3. PPO Algorithm

Policy gradient (PG) methods are the same as Q-learning in the sense that they explicitly learn a stochastic policy distribution π_θ parametrized by θ. The objective of PG is to maximize the expected return over the trajectories induced by the policy π_θ. If we denote the reward of a trajectory τ generated by policy $\pi_\theta(\tau)$ as $r(\tau)$, the policy gradient estimator has the form $gE[r(\tau)\nabla_\theta \log \pi_\theta(\tau)]$. This is the REINFORCE algorithm [27]. However, the REINFORCE algorithm has a high variance. A baseline, such as a value function baseline, can be used to improve the shortcomings of this type of algorithm. A generalized advantage estimate [28] is to use this method at the expense of some bias to reduce variance.

Schulman et al. [29] proposed that the TRPO algorithm can solve the shortcomings of the PG method that needs to be carefully adjusted for the step size. The PPO algorithm [30,31] is a simplification of the TRPO algorithm, which has a simpler execution method and sampling method.

The PPO algorithm optimizes the surrogate objective (1):

$$L^{clip}(\theta) = \mathbb{E}\Big[\min\Big(l_t(\theta)\hat{A}_t, clip(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big] \quad (1)$$

where $l_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ denotes the likelihood ratio and $\hat{A}_t$ is the generalized advantage estimate.

Similar to the DRQN algorithm, the combination of PPO and LSTM has a good effect on solving the POMDP problem [19,31].

4. Multiagent Independent PPO Algorithm

Tampuu et al. [11] demonstrate how competitive and collaborative behaviors emerge by independent Q-learning. Bansal et al. [19] explored that a multiagent environment produces complex behaviors by independent Proximal Policy Optimization (PPO) algorithm (MAIPPO).

Of course, the MAIPPO algorithm may also cause policies to fail to converge due to environmental non-stationary caused by the changing policies of the learning agents. The network structure of MAIPPO is shown in Figure 1.

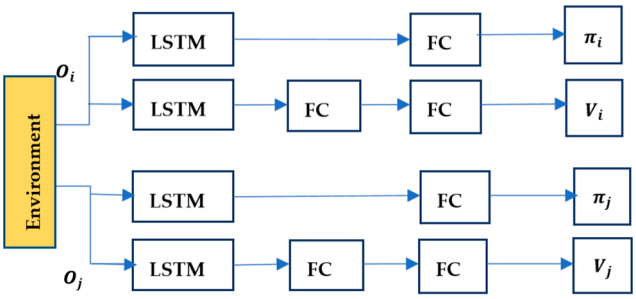


Figure 1. The network structure of the Multiagent Independent Proximal Policy Optimization (MAIPPO) algorithm.

The structure of the MAIPPO algorithm we construct is relatively simple. Actor and critic networks of the MAIPPO algorithm are composed of LSTM layers and a series of fully-connected layers (abbreviated as FC layers). The critic network obtains state–value function $V(O_t)$ and optimizes

the critic network by minimizing the loss. The generalized advantage estimate $A(O_t, a_t)$ is calculated from the value function $V(O_t)$ and then to optimize the actor network by the surrogate objective.

To update the critic network by minimizing the loss function (2):

$$L_t = E_t\Big[(r_t + \gamma V(O_{t+1}) - V(O_t))^2\Big], \tag{2}$$

To update the actor network by optimizing the surrogate objective (4):

$$L_t^{clip}(\theta) = E_t\Big[\min\Big(l_t(\theta)\hat{A}_t, clip(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big], \tag{3}$$

$$L_t^{clip+S}(\theta) = E_t\Big[L_t^{clip}(\theta) + cS[\pi_\theta](O_t)\Big], \tag{4}$$

where S denotes an entropy bonus, and c is coefficient. We can use a truncated version of generalized advantage estimation (5), so:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{5}$$

$$\delta_t = r_t + \gamma V(O_{t+1}) - V(O_t) \tag{6}$$

O_t in the above formulas has the following expressions for agent i and agent j:

$$O_t = \begin{cases} O_t^i = \left(s_t^i, s_t^{j,p}\right) \\ O_t^j = \left(s_t^j, s_t^{i,p}\right) \end{cases}, \tag{7}$$

where O_t^i represents the observation of agent i, which is the union of the state s_t^i of agent i and the partial state $s_t^{j,p}$ of the state s_t^j of agent j. O_t^j is the same. According to the theory of RL, we know that agent i selects policy π_i from O_t^i, and gets the reward and the next state s_{t+1}^i. That is to say, the policy π_i of agent i has no direct influence on $s_{t+1}^{j,p}$.

In other words, V^i obtained by the critic network of agent i evaluates of the partial state $s^{j,p}$ of agent j. However, the policy π_i obtained by the actor network only affects the state s^i of agent i. This creates a division between critic and actor, which is one of the directions in which various MARL algorithms strive to improve.

5. Multiagent Joint PPO Algorithm

In a multiagent learning environment, the environment becomes the non-stationary due to the changing policies of the learning agents. For the independent Q-learning algorithm, agents optimize policies through the local action–value function, which obstructs convergence.

There are a series of improved algorithms whose main purpose is to learn the centralized critic. The counterfactual multiagent (COMA) policy gradients algorithm and the multiagent Deep Deterministic Policy Gradient (MADDPG) use a centralized critic to estimate the Q-function and decentralized actors to optimize the agents' policies. VDN is to optimize the decentralized policies by using the sum of Q-value functions of each agent as a centralized evaluation function. QMIX is an improved algorithm of VDN, which learns a more complex joint action–value function by constructing a mixed network. DPIQN and DRPIQN propose to employ policy features of collaborators and opponents to infer and predict their policies.

The MAJPPO algorithm is proposed based on the MAIPPO algorithm. Different from the Q-learning algorithm, which uses the action–value function to evaluate and optimize policy, the PPO algorithm mainly uses the state–value function and the generalized advantage estimate to evaluate and optimize a policy. The MAJPPO algorithm learns mostly to obtain the joint state–value function and the generalized advantage estimate to evaluate and optimize the distribution policies.

To enhance the stability of training and the cooperative between agents, we use the moving window average of the state–value functions of different agents to obtain joint state–value functions V^i_{joint} (8) and V^j_{joint} (9):

$$V^i_{joint} = \xi V^i\left(O^i_t\right) + (1-\xi)V^j\left(O^j_t\right), \tag{8}$$

$$V^j_{joint} = \xi V^j\left(O^j_t\right) + (1-\xi)V^i\left(O^i_t\right), \tag{9}$$

where ξ is constant.

Agent i and agent j simultaneously obtain their respective observations O^i and O^j, which include both observations of their own and partial observations of other agents. The state–value functions V^i and V^j are obtained through the processing of their respective critic networks. Then, to obtain the joint state–value functions V^i_{joint} and V^j_{joint} through the weighted average of the state–value functions. The joint state–value function V^i_{joint} includes both the evaluation of the state of agent i and the evaluation of the state of other agents. The small $(1-\xi)$ in V^i_{joint} is mainly to reduce the effect of the evaluation of the remaining state of the state s^j of agent j except for $s^{j,p}$ on the joint state–value function. The state–value function V^j_{joint} obtained by the agent j includes both the evaluation of the state of agent j and the evaluation of the partial state of other agents. The surrogate objective obtained by V^i_{joint} and V^j_{joint} optimize the actor networks to get the cooperative policy.

The value function that the MAJPPO algorithm learns through critic networks is a combination of state features with its states and other agents. The VDN's paper pointers out that lazy agents arise due to the partial observability of state. The critic networks of the MAJPPO algorithm use global information to learn the value function. The advantage functions deriving from the value function are used to update actor networks. This can solve the lazy agent problem to some extent. The MAJPPO algorithm and VDN algorithm or QMIX algorithm have similarities. MAJPPO uses the weighted average of the state–value function of each agent to replace the local state–value function of each agent to achieve the goal of centralized learning.

6. Multi-UAV Formation

6.1. Dynamics Model of Small UAV and Attitude Control

We use the six-degree-of-freedom, 12-state equations of motion with the quasi-linear aerodynamic and propulsion models [32]. The model is provided in Appendix A. It is a fairly complicated set of 12 nonlinear, coupled, first-order, ordinary differential equations. Among the variables of these equations, in addition to 12 state variables $[p_n; p_e; h; u; v; w; \varphi; \theta; \psi; p; q; r]$, there are four input variables: the aileron deflection is denoted by δ_a, the elevator deflection is denoted by δ_e, and the rudder deflection is denoted by δ_r and the throttle command δ_t.

We could use the attitude control method of Appendix B to control the attitude of the above-mentioned UAV dynamics model.

6.2. RL of Single UAV

We use the above-mentioned UAV model as the control body of RL. The basis of multiagent RL for multi-UAV collaborative control is that RL can control the stable flight of a UAV.

We have two ways to control the UAV's stable flight by using reinforcement learning. One is to reinforce learning to control the dynamic model of the UAV directly, and the other is to reinforce learning to control the dynamic model of the UAV through the attitude loop. For the first method, the details are as follows. We use the 12-state of UAV as the input of the neural network of PPO. The network output of PPO is $[\delta_a; \delta_e; \delta_r; \delta_t]$, where $[\delta_a; \delta_e; \delta_r] \in [-1, 1]$ and $\delta_t \in [0, 1]$. The output $[\delta_a; \delta_e; \delta_r; \delta_t]$ is applied to the UAV motion model to obtain the next states of UAV after 0.1 s, and this cycle, as shown in Figure 2. We opt to use 10 s, which is 100 steps as an episode. Additionally, every 10 episodes update the network. A reasonable reward function structure is necessary to learn a stable

control model. To accomplish such a task, the UAV starts from the appropriate position and reaches another position $\left[p_n^{target}; p_e^{target}; h^{target}\right]$ in a stable attitude and a certain velocity V^{target}. Then, this reward function (11) can be constructed like this:

$$r_{navig} = \left|p_n - p_n^{target}\right| + \left|p_e - p_e^{target}\right|, \text{ with} \tag{10}$$

$$R_{single} = -r_{navig} - \left|h - h^{target}\right| - \eta_v\left|V - V^{target}\right| - \left|\varphi\right| - \left|\theta\right| - \left|\psi\right| - \left|p\right| - \left|q\right| - \left|r\right| \tag{11}$$

where η_v is constant, and $V = norm([u, v, w])$. r_{navig} can be transformed according to specific tasks. In this way, the UAV can get a stable policy model to complete the task.

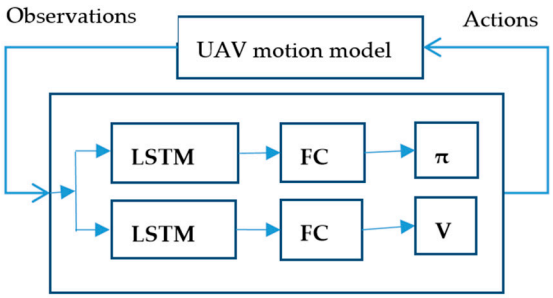


Figure 2. The single-agent PPO algorithm and the unmanned aerial vehicle (UAV) motion model for the feedback loop.

For the second method, the details are as follows. We use the 6-state of UAV (position and velocity) as the input of the neural network of PPO. The network output of PPO is $[\theta; \phi; \delta_t]$, where $[\theta; \phi] \in [-0.5,\ 0.5]$ and $\delta_t \in [0, 1]$. The output $[\theta; \phi; \delta_t]$ is applied to the UAV dynamics model with attitude control loop to obtain the next states of UAV after 0.5 s, and this cycle, as shown in Figure 3. In order to learn a stable control model, a reasonable reward function structure is necessary. To accomplish such a task: the UAV starts from the appropriate position and reaches another position $\left[p_n^{target}; p_e^{target}; h^{target}\right]$ in a stable attitude and a certain velocity V^{target}. Then, this reward function (13) can be constructed like this:

$$r_{navig} = \left|p_n - p_n^{target}\right| + \left|p_e - p_e^{target}\right|, \tag{12}$$

$$R_{single} = -r_{navig} - \eta_h\left|h - h^{target}\right| - \eta_v\left|V - V^{target}\right|, \tag{13}$$

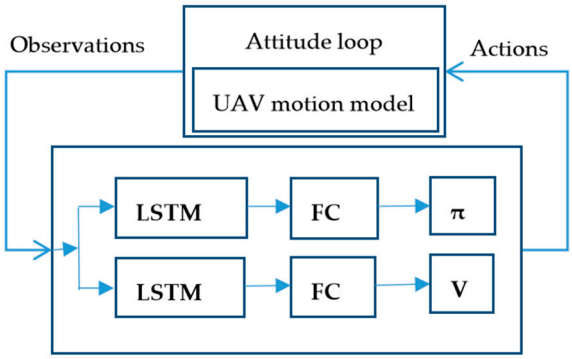


Figure 3. The single-agent PPO algorithm and the UAV motion model controlled by attitude loop for the feedback loop.

In this way, the UAV can obtain a stable policy model to complete the task.

Comparing the two methods, we can find that controlling the UAV with the attitude loop has fewer optimization targets, which can reduce the complexity of the UAV control and facilitate multi-UAV coordinated control.

6.3. Multi-UAV Formation

We use the MAIPPO and MAJPPO algorithms to solve multi-UAV collaborative control tasks. The control tasks we studied mainly constitute three UAV formations and obstacle avoidance.

The inputs to the MAIPPO and MAJPPO algorithms include their states, distance from obstacles and partial states of other UAVs. We found that using positions of other UAVs in experiments can result in more stable training results. For example, three UAVs are represented by UAV1, UAV2, and UAV3, respectively. Moreover, two obstacles are represented as obstacle1 and obstacle2. Then, the network input of UAV1 is $\left[p_n^1;\ p_e^1;\ h^1;\ u^1;\ v^1;\ w^1;\ p_n^2;\ p_e^2;\ h^2;\ p_n^3;\ p_e^3;\ h^3;\ dist^{11}; dist^{12}\right]$, where $\left[p_n^2; p_e^2; h^2\right]$ is the position of UAV2, $\left[p_n^3; p_e^3; h^3\right]$ is the position of UAV3, and so on. $dist^{11}$ and $dist^{12}$ are the distance between UAV1 and obstacle1 and the distance between UAV1 and obstacle2. UAV has a detection distance $d^{detection}$ for obstacles, when $dist^{11} > d^{detection}$, $dist^{11} = d^{detection}$. The reward function (16) consists of three parts: (I) one to fly UAVs with a stable attitude and velocity, denoted by R_{single}, (ii) another part to coordinate UAVs' flight while maintaining a certain formation distance, denoted by R_{form}, (iii) the third part to implement UAVs' obstacle avoidance, as follows.

$$\begin{aligned} R_{form} = &\left|norm\left(p_n^{12}, p_e^{12}, h^{12}\right) - d_{form}\right| - \left|norm\left(p_n^{13}, p_e^{13}, h^{13}\right) - d_{form}\right| - \\ &\left|norm\left(p_n^{23}, p_e^{23}, h^{23}\right) - d_{form}\right| - \left|norm\left(p_n^{21}, p_e^{21}, h^{21}\right) - d_{form}\right| - \\ &\left|norm\left(p_n^{31}, p_e^{31}, h^{31}\right) - d_{form}\right| - \left|norm\left(p_n^{32}, p_e^{32}, h^{32}\right) - d_{form}\right|, \end{aligned} \tag{14}$$

$$R_{obstacle}^1 = \begin{cases} 0, & dist^{11} > d^{detection} \\ -\frac{d_{form}}{dist^{11}}, & 0 < dist^{11} \le d^{detection} \\ -1000, & 0 \ge dist^{11} \end{cases} \tag{15}$$

$$R = \frac{1}{\mathrm{K}}\left[\alpha_s R_{single} + \alpha_f R_{form} + \alpha_o\left(R_{obstacle}^1 + R_{obstacle}^2\right)\right], \tag{16}$$

where K, α_s, α_f, *and* α_o are constants, and $\alpha_s + \alpha_f + \alpha_o = 1$, d_{form} represent the safe distance of the formation, $norm\left(p_n^{12}, p_e^{12}, h^{12}\right)$ represents the distance of between UAV1 and UAV2 while $norm\left(p_n^{13}, p_e^{13}, h^{13}\right)$ represents the distance of between UAV1 and UAV3. $R_{obstacle}^1$ and $R_{obstacle}^2$ are the reward function of UAV for obstacle1 and obstacle2.

7. Experiments

7.1. Experimental Condition

7.1.1. Network Settings

Critic network architectures first process the input using an LSTM layer with 128 hidden units, and then a fully connected linear layer with 128 hidden units followed by a TanH layer, and then a fully connected linear layer with 128 hidden units followed by a TanH layer.

The actor consists of two parts: a neural network and a normal distribution. The actor network has an LSTM layer with 128 hidden units, and then a fully connected linear layer with 128 hidden units followed by a TanH layer. The output of the network is the mean value of the normal distribution with covariance matrix $C = 0.05\ I$, where I is the identity matrix [33]. The distribution generates actions. The output range of the angles $[\theta; \phi]$ in the actor output is limited to [−0.5, 0.5], and the range of the throttle δ_t is limited to [0, 1]. Therefore, the mean value of the angle uses TanH as the activation function, and the mean of the throttle uses sigmoid as the activation function.

Due to the computational complexity of the UAV motion model, to shorten the training time, the use of multiple processes is inevitable.

7.1.2. Parameter Settings

The learning rate of Adam is 0.0001. The clipping parameter $\epsilon = 0.2$, discounting factor $\gamma = 0.995$ and generalized advantage estimate parameter $\lambda = 0.95$. We use large batch sizes, which can improve the variance problem to some extent and help to explore. In each iteration, we collect 1000 samples or 20 episodes, and 50 steps as one episode, and perform 20 episodes of training in mini-batches consisting of 512 samples. We found l_2 regularization with parameter 0.01 of the policy and value network parameters to be useful. The coefficient of the entropy is $c = 0.001$. the parameters in the reward function are set to K $= 100$, $\alpha_s = 0.5$, $\alpha_f = 0.4$, $\alpha_o = 0.1$, $\eta_h = 5$, $\eta_v = 2$. The sampling time is set to $\Delta t = 0.5$.

7.2. Mission Environment

Assume that the three UAVs fly from the initial area to the target area at a certain speed and a stable attitude as required by the formation, and pass through the area with six obstacles. The following initial values are assumed to simplify the task environment:

(1) p_e^1 and p_n^1 of UAV1 are uniformly distributed in the interval [0, 100] and [0, 100] respectively,
(2) The initial values of p_e^2 and p_n^2 of UAV2 are uniformly distributed in the interval [−100, 0] and [0, 100] correspondingly, and
(3) The initial values of p_e^3 and p_n^3 of UAV3 are uniformly distributed in the interval [−50, 50] and [−100, 0].

The initial values of h^1, h^2, and h^3 of UAV1, UAV2, and UAV3 are uniformly distributed in the interval [160, 240]. The initial values of velocity u, v and w of UAV1, UAV2 and UAV3 are uniformly distributed in the interval [10, 40], [1, 5] and [1, 5], the initial values of angle and angular velocity φ, θ, ψ, p, q, and r are uniformly distributed in the interval [−0.5, 0.5]. The target area is set to $p_e^{target} = 0$, $p_n^{target} = 1000$ and $h^{target} = 200$. We assume that the safety distance between UAVs is $d_{form} = 50$. The formation flight velocity is $V^{target} = 40$. Six spherical obstacle centers with a radius of 20 are uniformly distributed in a range of obstacles [400, 700] × [−200, 200] × [150, 250] in a uniformly distributed manner. The maximum detection distance of UAV to obstacles is $d^{detection} = 200$.

7.3. Experimental Comparison and Analysis

In order to compare the performance of the algorithm intuitively, we use the above parameters and task environment to perform experiments on the MAIPPO and MAJPPO algorithms until convergence, where the parameter in the MAJPPO algorithm is $\xi = 0.9$. The learning curves of the algorithms are revealed in Figure 4. It should be noted that the reward in Figure 4 is the sum of the rewards of three UAVs. We performed 10,000 iterations for the MAIPPO algorithm and the MAJPPO algorithm. It can be clearly seen from Figure 4 that the MAJPPO algorithm demonstrates better performance than the MAIPPO algorithm in dealing with multi-UAV collaboration and obstacle avoidance problems. It can also be seen from Figure 4 that the learning curve of the MAIPPO algorithm is not stable after convergence, and the MAJPPO algorithm can get higher reward value and convergence more stable. Therefore, the MAJPPO algorithm can get better results than the MAIPPO algorithm when dealing with this Dec-POMDP environment. The training learning curve of the MAIPPO algorithm cannot converge well because of the instability of the environment.

Figure 5 shows trajectory curves, distance curves, altitude curves, velocity curves, and distance curves between UAVs and obstacles of UAVs after training using MAIPPO algorithm and MAJPPO algorithm. As can be seen from Figure 5, these three UAVs can fulfill the mission requirements well.

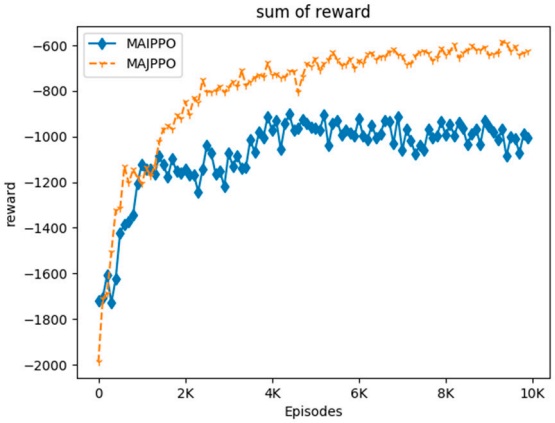
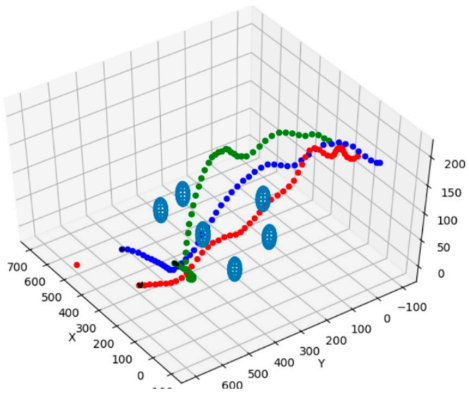
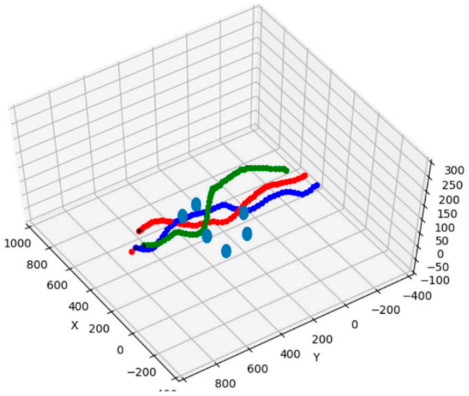


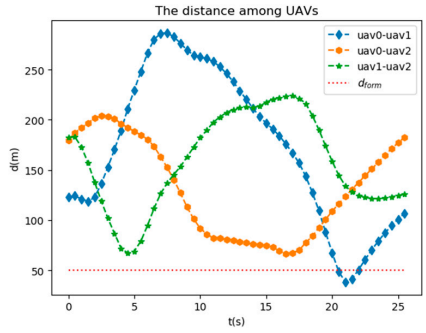
Figure 4. The curves of reward for multi-UAV formation uses the MAIPPO and the Multiagent Joint Proximal Policy Optimization (MAJPPO) for training.



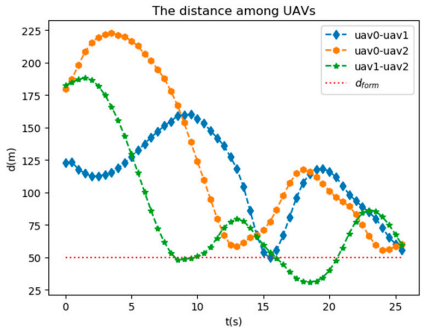
(**a**1) Trajectory curves



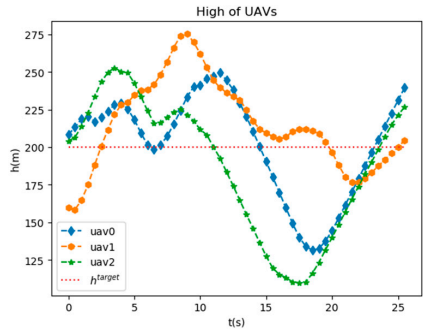
(**b**1) Trajectory curves



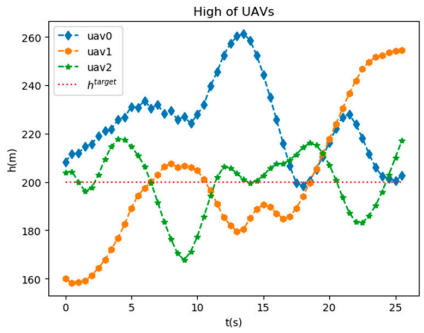
(**a**2) Distance curves



(**b**2) Distance curves

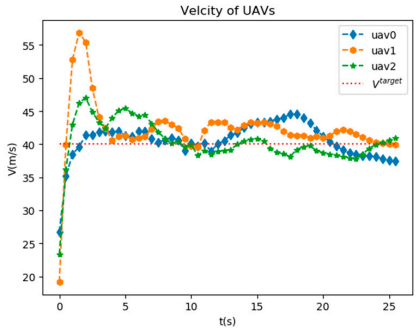


(**a**3) Altitude curves

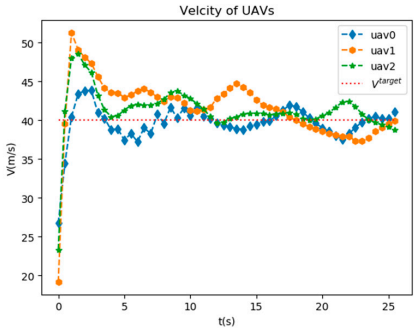


(**b**3) Altitude curves

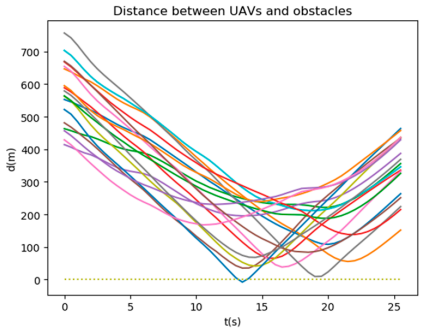
Figure 5. *Cont.*



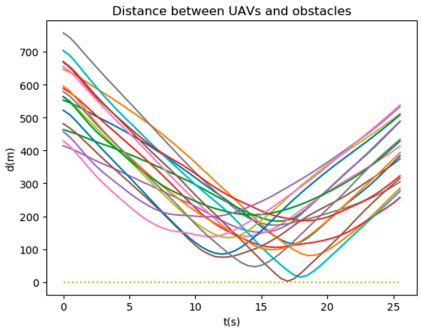
(**a**4) Velocity curves



(**b**4) Velocity curves



(**a**5) Distance curves between UAVs and obstacles



(**b**5) Distance curves between UAVs and obstacles

Figure 5. The trajectory curves, distance curves, altitude curves, velocity curves, and distance curves between UAVs and obstacles of UAVs after training using MAIPPO algorithm and MAJPPO algorithm. (**a**1–**a**5) is the result of the MAIPPO algorithm formation obstacle avoidance test, and (**b**1–**b**5) is the result of the MAJPPO algorithm formation obstacle avoidance test.

It can be seen from Figure 5 that the network trained by the MAJPPO algorithm performs better in the multi-obstacle environment for multi-UAV obstacle avoidance control. To specifically evaluate the performance of distance, altitude, and velocity of UAVs, we calculate the sum of first-order absolute center moment separately as (17), (18), and (19).

For d:

$$\begin{cases} \bar{e}_d(\mathrm{t}) = \frac{1}{N}\sum_{i=1}^{N}\left|d_i(t) - d_{form}\right| \\ E_d = \sum_{t=0}^{T}\bar{e}_d(t) \end{cases} \tag{17}$$

For h:

$$\begin{cases} \bar{e}_h(\mathrm{t}) = \frac{1}{N}\sum_{i=1}^{N}\left|h_i(t) - h^{target}\right| \\ E_h = \sum_{t=0}^{T}\bar{e}_h(t) \end{cases} \tag{18}$$

For v:

$$\begin{cases} \bar{e}_v(\mathrm{t}) = \frac{1}{N}\sum_{i=1}^{N}\left|v_i(t) - V^{target}\right| \\ E_v = \sum_{t=0}^{T}\bar{e}_v(t) \end{cases} \tag{19}$$

As can be seen from Table 1, E_d, E_h, and E_v of the MAJPPO algorithm improve by 42.72% ((5408.17–3097.77)/5408.17 × 100%), 39.66%, and 8.23% compared with the MAIPPO algorithm. This shows that MAJPPO algorithm has better performance.

Table 1. The sum of first-order absolute center moment.

	Multiagent Joint Proximal Policy Optimization (MAIPPO)	Multiagent Independent Proximal Policy Optimization (MAJPPO)	Percentage
E_d	5408.17	3097.77	42.72%
E_h	1609.09	970.94	39.66%
E_v	126.46	116.05	8.23%

7.4. Parameter Evaluation

Since the weighted average parameter ξ in the MAJPPO algorithm has a great influence on the performance of the algorithm, we discuss and analyze it. The learning curves of MAIPPO and MAJPPO for ξ = 0.8, 0.9, 0.99, and 0.999 are shown in Figure 6.

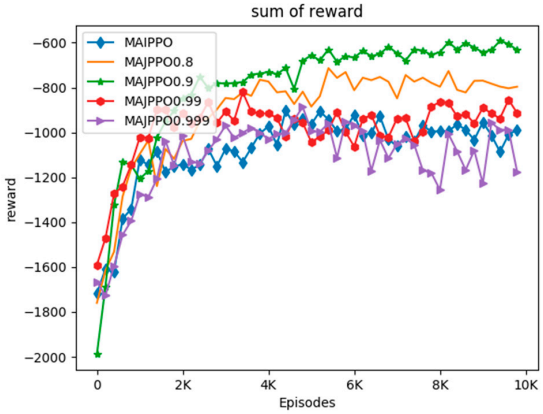


Figure 6. The learning curves of MAIPPO and MAJPPO for ξ = 0.8, 0.9, 0.99, and 0.999.

In the MAJPPO algorithm, when ξ = 1, it is actually the independent PPO algorithm. This can be seen from Figure 6, when the value of ξ is closer to 1. The performance of the algorithm will also show similar performance to the independent PPO algorithm, such as ξ = 0.999. However, the performance of the algorithm does not become better as the value of ξ becomes smaller. For example, when ξ = 0.8, the performance of the algorithm is not as good as ξ = 0.9.

8. Conclusions and Future Work

Based on the MAIPPO algorithm, we propose the MAJPPO algorithm that uses the moving window averaging of state-valued function to obtain a centralized state value function to deal with multiagent coordination problems. The MAJPPO algorithm is also a kind of centralized training and distributed execution algorithm. We also presented a new cooperative multi-UAV simulation environment, where multi-UAV work together to accomplish formation and obstacle avoidance. In order to accomplish this task, we use the dynamic model of the UAV with attitude control capability as the control object. It can be seen from the experimental comparison that the MAJPPO algorithm can better deal with the partial observability of the state in the multiagent system and obtain better experimental results.

The comparison of the MAJPPO algorithm with other multi-agent reinforcement learning algorithms, such as MADDPG, VDN, and QMIX, is left for future work.

Author Contributions: Conceptualization, W.Z., H.C., X.M., L.G. and H.S.; Data curation, W.Z.; Formal analysis, W.Z., X.M., H.S., C.Z., F.Z. and D.L.; Investigation, W.Z.; Methodology, W.Z.; Resources, W.Z.; Software, W.Z.; Supervision, H.C., X.M., L.G. and D.L.; Validation, W.Z.; Visualization, W.Z.; Writing – original draft, W.Z.; Writing – review & editing, W.Z., C.Z., F.Z. and D.Z.. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by The National Key Research and Development Program of China, 2017YFC0822403.

Conflicts of Interest: Authors declare no conflict of interests.

Appendix A

"SMALL UNMANNED AIRCRAFT Theory and Practice" presents the six-degree-of-freedom, 12-state equations of motion with the quasi-linear aerodynamic and propulsion models for the small UAV in chapter 5 as follows.

$$\dot{p}_n = (\cos\theta\ \cos\psi)u + (\sin\phi\ \sin\theta\ \cos\psi - \cos\phi\ \sin\psi)v + (\cos\phi\ \sin\theta\ \cos\psi + \sin\phi\ \sin\psi)w$$

$$\dot{p}_e = (\cos\theta\ \sin\psi)u + (\sin\phi\ \sin\theta\ \sin\psi + \cos\phi\ \cos\psi)v + (\cos\phi\ \sin\theta\ \sin\psi - \sin\phi\ \cos\psi)w$$

$$\dot{h} = u\ \sin\theta - v\ \sin\phi\ \cos\theta - w\ \cos\phi\ \cos\theta$$

$$\dot{u} = r\,v - q\,w - g\ \sin\theta + \frac{\rho V_a^2 S}{2\mathrm{m}}\left[\mathrm{C_X}(\alpha) + \mathrm{C_{X_q}}(\alpha)\frac{cq}{2\mathrm{V_a}} + \mathrm{C_{X_{\delta_e}}}(\alpha)\delta_e\right] + \frac{\rho \mathrm{S_{prop}} \mathrm{C_{prop}}}{2\mathrm{m}}\left[(\mathrm{k_{motor}}\delta_t)^2 - V_a^2\right]$$

$$\dot{v} = p\,w - r\,u + g\ \cos\ \theta\ \sin\ \phi + \frac{\rho V_a^2 S}{2\mathrm{m}}\left[C_{Y_0} + C_{Y_\beta}\,\beta + C_{Y_p}\frac{bp}{2V_a} + C_{Y_r}\frac{br}{2V_a} + C_{Y_{\delta_a}}\,\delta_a + C_{Y_{\delta_r}}\,\delta_r\right]$$

$$\dot{w} = q\,u - p\,v + g\ \cos\ \theta\ \cos\ \phi + \frac{\rho V_a^2 S}{2\mathrm{m}}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{cq}{2V_a} + C_{Z_{\delta_e}}(\alpha)\delta_e\right]$$

$$\dot{\phi} = p + q\ \sin\ \phi\ \tan\ \theta + r\ \cos\ \phi\ \tan\ \theta$$

$$\dot{\theta} = q\ \cos\ \phi - r\ \sin\ \phi$$

$$\dot{\psi} = q\ \sin\ \phi\ \sec\ \theta + r\ \cos\ \phi\ \sec\ \theta$$

$$\dot{p} = \Gamma_1\,pq - \Gamma_2\,qr + \frac{1}{2}\rho V_a^2 Sb\left[C_{p_0} + C_{p_\beta}\,\beta + C_{p_p}\frac{bp}{2V_a} + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta_a}}\,\delta_a + C_{p_{\delta_r}}\,\delta_r\right]$$

$$\dot{q} = \Gamma_5\,pr - \Gamma_6\left(p^2 - r^2\right) + \frac{\rho V_a^2 Sc}{2J_y}\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta_e}}\,\delta_e\right]$$

$$\dot{r} = \Gamma_7\,pq - \Gamma_1 qr + \frac{1}{2}\rho V_a^2\,Sb\left[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta_a}}\,\delta_a + C_{r_{\delta_r}}\,\delta_r\right],$$

The variables and constants in the above formulas are specifically explained in [32]. The data we used in the experiment are from Aerosonde UAV of APPENDIX E in [32].

Appendix B

We use the following three control methods as the attitude control algorithm for the fixed-wing UAV dynamic model introduced by Appendix A.

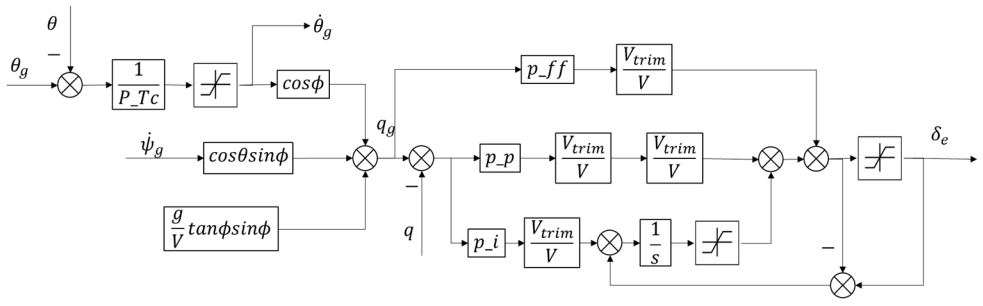


Figure A1. Pitch angle control structure.

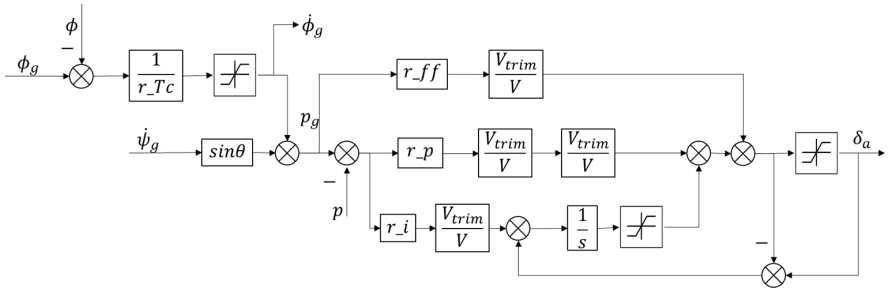


Figure A2. Roll angle control structure.

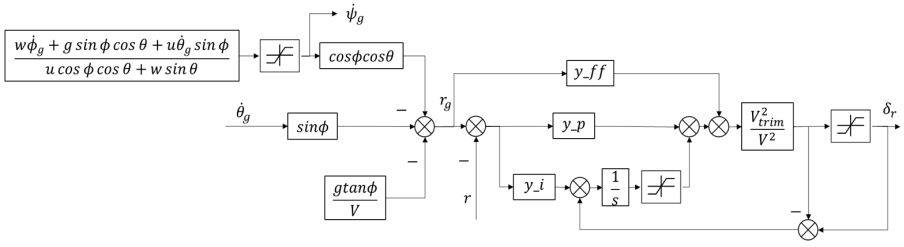


Figure A3. Yaw angle control structure.

The three tables below contain some of the variables and values in the control block diagram above, which are also used in our simulations.

Table A1. Pitch angle control system parameters.

Parameter	Notation and Value
p_Tc	0.4
p_p	0.08
p_i	0.02
p_ff	0.5
p_integrator max	0.4
p_rmax_pos	$+60\ deg/s$
p_rmax_neg	$-60\ deg/s$
Output_max	$1\ rad$

Table A2. Roll angle control system parameters.

Parameter	Notation and Value
r_Tc	0.4
r_p	0.05
r_i	0.01
r_ff	0.5
r_integrator_max	0.2
r_rmax	$70\ deg/s$
output_max	$1\ rad$

Table A3. Yaw angle control system parameters.

Parameter	Notation and Value
y_p	0.05
y_i	0.0
y_ff	0.3
y_integrator max	0.2
y_rmax	0
y_coordinated_min_speed	1000
y_coordinated_method	0

References

1. Olfati-Saber, R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Autom. Control* **2006**, *51*, 401–420. [CrossRef]
2. Bin, F.; Xiaofeng, F.; Shuo, X. Research on Cooperative Collision Avoidance Problem of Multiple UAV Based on Reinforcement Learning. In Proceedings of the International Conference on Intelligent Computation Technology & Automation 2017, Changsha, China, 9–10 October 2017.
3. La, H.M.; Lim, R.; Sheng, W. Multirobot cooperative learning for predator avoidance. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 52–63. [CrossRef]
4. Hung, S.-M.; Givigi, S.N. A Q-learning approach to flocking with UAVs in a stochastic environment. *IEEE Trans. Cybern.* **2017**, *47*, 186–197. [CrossRef] [PubMed]
5. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Nguyen, L.V. Autonomous uav navigation using reinforcement learning. *arXiv* **2018**, arXiv:1801.05086.
6. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans. Cyber Phys. Syst.* **2019**, *3*, 22. [CrossRef]
7. Busoniu, L.; Babuška, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2008**, *38*, 156–172. [CrossRef]
8. Kapoor, S. Multi-agent reinforcement learning: A report on challenges and approaches. *arXiv* **2018**, arXiv:1807.09427.
9. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. *arXiv* **2018**, arXiv:1812.11794.
10. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 June 1993; pp. 330–337.
11. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [CrossRef]
12. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
13. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6379–6390.
14. Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; Russell, S. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI), Honolulu, HI, USA, 27 January–1 February 2019.
15. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean field multi-agent reinforcement learning. *arXiv* **2018**, arXiv:1802.05438.
16. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296.
17. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv* **2018**, arXiv:1803.11485.
18. Hong, Z.-W.; Su, S.-Y.; Shann, T.-Y.; Chang, Y.-H.; Lee, C.-Y. A deep policy inference q-network for multi-agent systems. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2018.
19. Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; Mordatch, I. Emergent complexity via multi-agent competition. *arXiv* **2017**, arXiv:1710.03748.
20. Wang, C.; Wang, J.; Zhang, X.; Zhang, X. Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 14–16 November 2017.
21. Zhao, W.; Chu, H.; Zhang, M.; Sun, T.; Guo, L. Flocking Control of Fixed-Wing UAVs with Cooperative Obstacle Avoidance Capability. *IEEE Access* **2019**, *7*, 17798–17808. [CrossRef]

22. Guerrero-Castellanos, J.F.; Vega-Alonzo, A.; Durand, S.; Marchand, N.; Gonzalez-Diaz, V.R.; Castañeda-Camacho, J.; Guerrero-Sánchez, W.F. Leader-Following Consensus and Formation Control of VTOL-UAVs with Event-Triggered Communications. *Sensors* **2019**, *19*, 5498. [CrossRef]
23. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
24. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
25. 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.
26. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; Springer International Publishing: New York, NY, USA, 2016; Volume 1.
27. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
28. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
29. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
30. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
31. Heess, N.; TB, D.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S.M.; et al. Emergence of locomotion behaviours in rich environments. *arXiv* **2017**, arXiv:1707.02286.
32. Beard, R.W.; McLain, T.W. *Small Unmanned Aircraft: Theory and Practice*; Princeton University Press: Princeton, NJ, USA, 2012.
33. Wawrzynski, P. Real-time reinforcement learning by sequential actor–critics and experience replay. *Neural Netw.* **2009**, *22*, 1484–1497. [CrossRef] [PubMed]