Hindawi
Complexity
Volume 2021, Article ID 8818013, 12 pages
https://doi.org/10.1155/2021/8818013

Research Article

Reinforcement Learning-Based Collision Avoidance Guidance Algorithm for Fixed-Wing UAVs

Yu Zhao, Jifeng Guo, Chengchao Bai, and Hongxing Zheng

School of Astronautics, Harbin Institute of Technology, Harbin 150001, China

Correspondence should be addressed to Jifeng Guo; guojifeng@hit.edu.cn

Received 6 August 2020; Revised 5 November 2020; Accepted 4 January 2021; Published 18 January 2021

Academic Editor: Zhile Yang

A deep reinforcement learning-based computational guidance method is presented, which is used to identify and resolve the problem of collision avoidance for a variable number of fixed-wing UAVs in limited airspace. The cooperative guidance process is first analyzed for multiple aircraft by formulating flight scenarios using multiagent Markov game theory and solving it by machine learning algorithm. Furthermore, a self-learning framework is established by using the actor-critic model, which is proposed to train collision avoidance decision-making neural networks. To achieve higher scalability, the neural network is customized to incorporate long short-term memory networks, and a coordination strategy is given. Additionally, a simulator suitable for multiagent high-density route scene is designed for validation, in which all UAVs run the proposed algorithm onboard. Simulated experiment results from several case studies show that the real-time guidance algorithm can reduce the collision probability of multiple UAVs in flight effectively even with a large number of aircraft.

1. Introduction

With the rapid development of unmanned aerial vehicle (UAV) technology, fixed-wing UAVs have been playing an increasingly important role in both modern life and military affairs [1, 2]. Due to the technical characteristics of fixed-wing UAVs such as unable to hover and limited speed control range, it is easy to cause safety accidents which can lead to property losses and even personal injuries in environments with high route density [3]. Therefore, a large number of fixed-wing UAV flight conflicts in the given airspace have become a prominent problem that needs to be solved in related fields [4]. The pilot of a conventional aircraft can judge the distance to other aircraft by visual inspection and make collision avoidance operations timely [5]. However, UAV operators cannot obtain direct vision in time, and there is always a delay or lag in data transmission processes [6]. Therefore, it is necessary to study the intelligent autonomous guidance method of UAVs to prevent collision.

Many recent research studies for limited airspace operations, such as the unmanned aircraft system traffic management (UTM) [7] and air traffic management (ATM) [8], require an autonomous collision avoidance guidance system to maintain safety and efficiency. There are many contributions to the topic of real-time collision avoidance methods, and most of the studies are focused on ground robots [9]. One of the most well-known works in air traffic control is the Autoresolver program developed by NASA scientists [10]. Kuchar and his colleagues presented a comprehensive summary of more than 30 different methods for flight conflict resolution, in which the key methods such as artificial potential field approach, biological evolution method, and optimization algorithm are given in detail [11]. Zhou et al. proposed a model predictive control method integrated with trajectory prediction of obstacles and targets, which considers the overall design of multi-UAV cooperative anticollision [12]. The overall decision making of UAVs' collision-free flight was optimized in their research. Although the above algorithms perform well in specific scenarios, they cannot adapt to stochastic dynamic models. Yang and Wei presented a message-based decentralized computational guidance algorithm which is implemented by the Monte Carlo tree search technique [13]. Their work

requires a lot of computing resources onboard, and complete conflict-free flight is not achieved in some cases. This paper focuses on designing an onboard guidance system that is able to provide real-time flight commands to UAVs to ensure safe distance along air routes.

Multiagent approaches have been used in airspace conflict resolution for a long time. In an earlier article [14], Tomlin considered the evasive maneuver as a hybrid control system so that separation is assured in the actions of the other aircraft. Kravris et al. recently used the multiagent method to solve the congestion problem in the field of air traffic management [15]. They formalized the problem as a multiagent Markov game in which the aircraft needs to reach an equilibrium. In this paper, a similar idea is used to abstract every fixed-wing UAV as an agent that can interact with other agents. Each agent has the capability of autonomous decision-making guidance and makes decisions by obtaining the current airspace information. Therefore, the problem of UAV collision avoidance guidance can be transformed into a multiagent sequence decision process with clear targets. Reinforcement learning is a prevailing way to solve such problems.

Artificial intelligence (AI) techniques have achieved superior performance in a lot of engineering applications nowadays. Reinforcement learning, as the mainstream algorithm of AI, has been widely studied in many fields. Crespo presented a comparative study of air traffic flow management measures obtained by a program based on reinforcement learning [16]. The computational agent is used to establish delays upon schedules of departing from restricted airspace so as to avoid congestion or saturation in the air traffic control work. Keong et al. used the deep Q-value network (DQN) algorithm to replace the traditional control method of aerial collision resolution [17]. The DQN was trained in a customized scenario to get a stable policy which provides actions for collision avoidance. This paper studies the guidance application that involves the interaction between multiple UAVs, where emergent behavior and complexity arise from multiple agents together. The traditional reinforcement learning algorithms such as DQN and other methods mentioned above are poorly suited to multiagent environments [18]. We present an extension of the actor-critic (AC) model in reinforcement learning where the critics have access to global information, while the actors only receive the current overall states of the environment. The optimal policy is formed by training decision-making neural network in this AC framework. The actors are used at the computational guidance process after training is completed, acting in a decentralized operation.

In this work, a deep multiagent reinforcement learning (MARL) framework is proposed to solve the real-time flight collision avoidance guidance problem of multiple fixed-wing UAVs in the same altitude and common airspace, where each UAV is abstracted by an agent. Each agent receives the situations from the simulator environment and performs online sequential decision making to select turning actions in real time to avoid conflicts along routes. In order to realize the function of the guidance decision-making neural network, a multiagent reinforcement learning-based self-training system with actor-critic framework consisting of centralized training with decentralized execution is proposed. The proposed system can handle a variable number of UAVs in the limited airspace. To achieve this, a custom decision-making neural network with long short-term memory (LSTM) networks is used to encode and control the information about the whole environment into a fixed-length vector [19]. We also design the Q-value estimation network of evaluating joint action in the critic which takes all other agents' actions as input and give the relevant training process description based on logit coordination mechanism. This research provides a novel potential solution to enable multiple UAVs' coordinated collision avoidance flight in a given airspace. Due to the advantages of simulation calculation, the training of decision-making neural network can be realized with the development of "digital twin" and other technologies. The trained guidance system can be transplanted to the physical system, and the application requirements can be met after a small amount of training.

This research uses the common full connection network and LSTM network, but the number of neural nodes and hidden states in the network is specially designed according to the route problems. The difference between this work and the previous research is that the multiagent system is used to simulate the flight environment of UAV. Aiming at the dynamic change of the number of entities in the environment, a solution based on LSTM network is proposed. The structure of this paper is as follows. In Section 2, the kinematics of fixed-wing UAVs, dynamics constraints, and multiagent Markov game theory model are given. In Section 3, the description of the guidance problem and its mathematical formulation of multiagent AC framework are presented. Section 4 proposes the designed self-training system to solve this problem. The simulation experiments and results are shown in Section 5. Section 6 gives the conclusion.

2. Preliminaries

2.1. Kinematics Model and Dynamics Constraints. Many scholars have carried out detailed research on the structure and related dynamics of fixed-wing UAVs [20, 21]. In order to simplify the complexity of the guidance problem, this paper only considers horizontal actions in the process of controlling the UAVs. This assumption makes it possible to deal with the high-density air traffic of UAVs by distinguishing the multiple flight altitudes. This assumption can also effectively limit the range of state space, which is beneficial to reinforcement learning system training. State transition for every UAV in the simulator environment can use the kinematics model as follows:

$$\begin{aligned} \dot{x} &= V\cos\psi, \\ \dot{y} &= V\sin\psi, \\ \dot{\psi} &= a_c, \end{aligned} \tag{1}$$

where (x, y) denotes the position of an UAV, V is the cruising flight velocity, ψ stands for the flight path angle, and a_c is the controlled variable which represents the selected

action describing the changing rate of flight path angle for an UAV.

At the beginning of the simulation, the flight path angle of each aircraft is set to point to its goal position and keep flying for a certain amount of time. At each time step, for 1 second, the UAV can choose to turn its flight path angle at a certain rate. The changing rate of this angle is less than 5 degrees per second. The proposed algorithm will run onboard to determine a precise action, which corresponds to the changing rate of the angle, for the UAV based on the current state. After running the algorithm, the UAV will maintain the chosen action during this time step.

In each episode, the simulator will generate 200 UAVs randomly. The time interval for two flights to be generated is uniformly distributed between 50 seconds and 200 seconds. The specific number of UAVs in the simulation and related conditions will be described in detail in Section 5.

2.2. Multiagent Markov Games. If there is only one UAV in the environment, its guidance problem en route can be formulated by a single agent sequence decision model. However, there are multiple interactive entities in the specified airspace in this study. The traditional Markov decision process (MDP) method is no longer suitable for this scenario. A multiagent extension of MDP called Markov games is considered in this work.

A Markov game can be defined as a tuple $\{n, \mathbf{S}, \mathbf{A}_{1,\ldots,n}, R, T\}$, in which n is a scalar to represent the total number of agents in the system. $\mathbf{S}$ is a finite set of system possible states. $\mathbf{A}_i, i \in 1, \ldots, n$ denotes the actions set of agent i. Each agent i will obtain an instant reward as a function R, when the system state changes under the influence of the joint actions $a_1 \times \cdots \times a_n, a_i \in \mathbf{A}_i$, where a_i is the action chosen by agent i using a stochastic policy π_i. The joint actions $\mathbf{a}$ will produce the next state according to the transition function T.

The agents are trying to maximize their future average reward under the joint policy π over time, which can be defined as

$$J^{\pi}(s_t) = \lim_{T \longrightarrow \infty} \frac{1}{T_m} E^{\pi} \left\{ \sum_{t=0}^{T_m} \gamma^t R(t+1) \right\}, \quad s_t \in \mathbf{S}, \tag{2}$$

where T_m is the total time, γ is a discount factor, and t denotes the index of time step in a simulation.

The goal of Markov games is to find an optimal joint policy π^* that maximizes the expected cumulative rewards followed from an initial state.

The nonstationarity of multiagent Markov game is due to the fact that the best policy of a single agent changes as the other agents' policies change. Most studies of this problem focus on Nash equilibrium or long-term stable behaviors. However, it is difficult to get the Nash equilibrium when the calculation time is limited for a system with only one equilibrium [22]. All the agents in this research follow the logit strategy, which is one of our methods to solve the above problems.

More specifically, in the logit model, a high-level agent assumes that all the others adopt original actions. There is only one UAV at a high level and all the others are still at a low level until the high-level agent has made its decision. After receiving the action information from the high-level agent, the next UAV begins making decision onboard assuming all the others are following the most recent joint action. This process will iterate over all the UAVs until the joint action is completely updated. In this way, the actions of other agents become fixed except the selected actor; thus, the algorithm can avoid the action explosion issue.

3. MARL-Based Computational Guidance Approach

3.1. Problem Statement. The purpose of this study is to develop a distributed autonomous decision-making algorithm that runs on each UAV to provide guidance commands in the route. The goals of these commands are twofold: the first is to avoid collision among all the UAVs in the scene and the second is to guide the UAVs to their respective destinations during the flight. In order to make the algorithm use a scenario closer to the actual application, it is assumed that there are a variable number of UAVs flying at the same altitude in the limited airspace. To test the performance of the proposed algorithm, we only consider the case that the restricted airspace is a regular shape in our research. Each flight is generated at the vertex of a regular polygon, and any vertex, except the neighborhoods, can be randomly selected as the destination of this UAV. Figure 1 is a schematic diagram of the training case in this paper. A more detailed explanation of the scenario will be given in Section 5.

We assume that all the UAVs are cooperative in the limited airspace. They can exchange their own information such as position, speed, and selected actions through wireless communication without delay. The onboard algorithm of each UAV may choose an action that is available based on current global states at each time step.

The multiagent computational guidance (MACG) problem will be mathematically formulated in the next section based on the aforementioned description.

3.2. MACG Algorithm Formulation. Here we formulate the MACG case studies as a multiagent reinforcement learning process by treating each UAV as an agent. The objective in case study is to avoid collision by choosing actions and to maintain a safe separation between aircraft. The UAVs enter the limited airspace stochastically, so the scenarios studied in this paper are dynamic. For this reason, agents are required to provide a policy instead of memorizing the reactive actions. Although the speed of an UAV has upper and lower limits and there is noise, each UAV can only achieve collision avoidance by changing the flight path angle. The changing rate of the angle is less than 5 deg/s as mentioned above, which means that the optional action set is bounded.

In the following, we will define the state space, action space, reward function, and some other parameters of reinforcement learning algorithm which are used in this paper.

(1) *State Space*. The state information of an UAV includes the position (x, y), the speed V, the flight path angle ψ, and the goal position (g_x, g_y). We will get the current environment complete state $\mathbf{s}_t$ by stacking all the agent states at one time step t. $\mathbf{s}_t$ is an $n \times 6$ matrix, where n is the number of UAVs. We define $\mathbf{s}_t^i$ as the state of the ith UAV at the time t; thus:

$$\mathbf{s}_t = \left\{\mathbf{s}_t^1, \mathbf{s}_t^2, \ldots, \mathbf{s}_t^n\right\}, \quad \mathbf{s}_t \in \mathbf{S}. \tag{3}$$

We assume that $\mathbf{s}_t$ is available to all the agents in the simulation environment.

(2) *Action Space*. The UAV can choose to turn its flight path angle at a certain rate at each time step. The turning rate of the angle for each aircraft constitutes the action set $\mathbf{A}_i \in [-5, 5]$ deg/s, where positive corresponds to right turn and negative corresponds to left turn. However, continuous action space will take too much time to train the neural networks of MACG algorithm. We discretize the action space to be $\mathbf{A}_i = \{-5, -3, 0, 3, 5\}$ deg/s, in which 0 means no turning occurs.

(3) *Reward Function*. The goal of our MACG algorithm is to maintain safety while guiding the UAVs to their destinations during the flight. Cooperation is encouraged by defining a same reward function for all UAVs, and it composes of a sum of the reward for each individual aircraft.

We define a minimum separation distance $r_{\text{sep}} = 0.5$ km to warn aircraft that they are about to collide. If the distance between UAVs is less than 0.1 km, collisions will take place. If there is a collision between two UAVs, they will both receive a penalty and be removed from the simulator. If a UAV losses its separation or reaches the boundary, it will receive a penalty.

Based on the above settings, the reward function for the i^{th} UAV can be defined as follows:

$$r_i(s_t) = \begin{cases} -2, & \text{collision}, \\ -1, & \text{loss separation/boundary}, \\ 2, & \text{reach the goal}, \\ -\dfrac{\left(d_g / T_{\max}^i\right)}{d_{g\max}}, & \text{otherwise}, \end{cases} \tag{4}$$

where d_g is the distance from the UAV to its goal position, $d_{g\max}$ is set to be the diagonal distance of the map, and $T_{\max}^i$ denotes the max time step of agent i which is set before the simulation runs. Adding the constraint with losing separation event can effectively improve the training efficiency. The reward function of the environment state can be defined like this:

$$R(s_t) = \sum_{i=1}^{n} r_i(s_t). \tag{5}$$

With this reward setting, maximizing the cumulative reward will correspondingly improve the performance of the MACG algorithm.

(4) *Other Parameters*. Since the reward function and the dynamics function (transition function) have been determined, the goal of reinforcement learning is to find the optimal policy for the multiagent problem in this paper. To solve this stochastic game, we need to find a policy π_i^* that can maximize UAV i's discounted future reward with a discount factor γ. Obviously, this is a fully cooperative game, since all the agents have the same reward function. The state-value function at the time step t can be represented as $V(s_t, \pi_1, \ldots, \pi_n)$. We also define the action-value function $Q(s_t, a_1, \ldots, a_n)$ as the expected sum of discounted rewards given the current state and the joint action of all the agents.

In each episode of a case studied simulation, a certain number of UAVs will be generated. If an UAV reaches its destination, collides with other aircraft, or touches the boundary of the map, it will be removed from the simulation environment. In order to avoid UAVs to fly in an infinite loop in the early training period, the maximum action duration $T_{\max}$ is set for aircraft. Each UAV with a flight time of $T_{\max}$ will also be removed. The episode will terminate when all the UAVs have been removed from the limited airspace.

4. Design of Self-Training System

In this paper, the most popular method, the actor-critic model, will be used in the reinforcement learning field to solve the MACG problem formulated previously. A new deep MARL framework is designed and developed in this paper, and we call it MACG framework for convenience.

As discussed previously, the MACG framework is a centralized learning with decentralized execution framework with four neural networks corresponding to two actors and two critics. All agents (UAVs) will share one team of neural networks during the training period, which encourages cooperation. By this way, we can train an actor that improves the cumulative reward of all agents in the environment. In the decentralized execution scenario, each UAV has a private actor neural network, which allows it to make decisions independently without being affected by other aircraft.

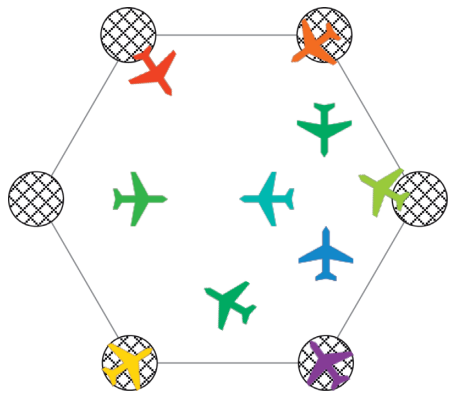


Figure 1: A simplified scenario of the training case.

The next part of this section will introduce the design details of neural networks structure and simulation system structure explicitly.

4.1. Neural Network Structure Design. In the collision avoidance scenario studied in this paper, the number of UAVs in each time step is not fixed. For the variable number of agents in the environment, agents need to process many input states. Recall that one of the key limitations of many reinforcement learning approaches is that the feedforward neural networks typically used require a fixed number of neurons of input. Two kinds of network structures are designed to deal with the above problems. The first is to fix the maximum number of input states, which we call it the FixN method, and the other is to encode the varying size of input states into a fixed-length vector by using LSTM networks. The schematic diagram of the two structures is shown in Figure 2.

Figure 2 is illustrated by the example of the guidance decision actor network of agent i, where $\mathbf{s}_t^i$ is the state information of agent i at time t. $(\mathbf{s}_t^1, \ldots, \mathbf{s}_t^n)$ represent the current states of all agents except agent i in the environment. They are combined to form the information of complete environment states. In the FixN structure, $\mathbf{s}_t^c$ represents the united states after sorting and integrating by the filter. The output of the LSTM structure is a fixed-length, encoded hidden state $\mathbf{h}_n$. $\mathbf{s}_t^c$ and $\mathbf{h}_n$ have the same data structure in theory, and they will be used as an input data $(\mathbf{s}_t^e)$ to feed back-end network.

For FixN neural network structures, we take the states of N UAVs, $(\mathbf{s}_t^j, \ldots, \mathbf{s}_t^k)$, which are closest to the current decision-making agent in the environment, combined with the agent's state $\mathbf{s}_t^i$ as the input states. However, a considerable amount of experiments are needed to select the hyperparameter N, which limits the transferability of the neural network structure to migrate to new environments. From the simulation results in the following section, we can also see that although FixN method can accelerate the convergence speed of training curves, it is difficult to find the optimal policy even in simple scenarios because agents do not take all the environmental states into account in the decision-making process.

Although LSTM networks are often used to handle time sequence data, here we only use its ability to store the relevant information between sequence input states which is not time dependent. Every state except the current decision agent's own state in the environment will be fed into the LSTM cell sequentially at each time step, as shown in Figure 3. After importing all the agents' states, the LSTM network stores the relevant information in its hidden states, $(h_1, \ldots, h_n)$. We can regard the last hidden state of LSTM network as a fixed-length, encoded global state for an agent to make a decision. In the case of a variable number of agent states, the states are fed from far to near according to the distance from the agent, meaning that the closest agent will have the biggest effect on the last hidden state of the LSTM network.

The neural networks of actor and critic designed in this paper have similar front-end structures. They all have a LSTM network, an input layer, and two fully connected hidden layers. But the difference between the actor and the critic is the case that the LSTM network input of the critic contains not only the current states of all agents in the environment but also the current selected actions of all agents. The function of actor neural network is to approximate the policy function so as to select a suitable action for agent. Since the action space in this paper is discrete, a Gumbel-Softmax estimator should be added at the end of the actor network as the output layer [23]. The function of the critic network is to approximate the action-state value (also known as the Q-value) function, so its output layer can be selected as a single neuron.

4.2. Simulation System Structure Design. In order to solve the cooperative collision avoidance guidance problem above, we design and develop a MACG algorithm. Lowe et al. [18] proposed a similar idea of using the actor-critic method to solve the guidance case study. Our approach differs from theirs in the following ways: (1) they designed a critic for each agent, whereas agents share a single actor and a single critic during the learning process in our work, (2) in principle, the actions for different agents are updated in an asynchronous way in this paper, and (3) they learned cases with fixed number of agents in the environment, whereas we learned variable number of agents in scenarios.

There are four neural networks involved in the learning process of MACG algorithm in this paper. They are named as decision actor (or ActorD for short), estimation actor (or ActorE for short), decision critic (or CriticD for short), and estimation critic (or CriticE for short). ActorD is used to approximate the guidance policy, which can be parameterized by θ. It is the only network that will be used in both agents' training and execution. CriticD network is used to approach Q-value, and it is parameterized by ω. Taking agent i as an example, the training framework of self-learning system is shown in Figure 3.

During the training period, ActorD may generate immediate decisions, that is, it selects actions for the agent according to the current global states in the environment. In each time step t, all agents customize their actions $\pi(s_t^{-i}, \theta) = a_t^i, i \in 1, \ldots, n$ through the same ActorD, and all selected actions integrate into

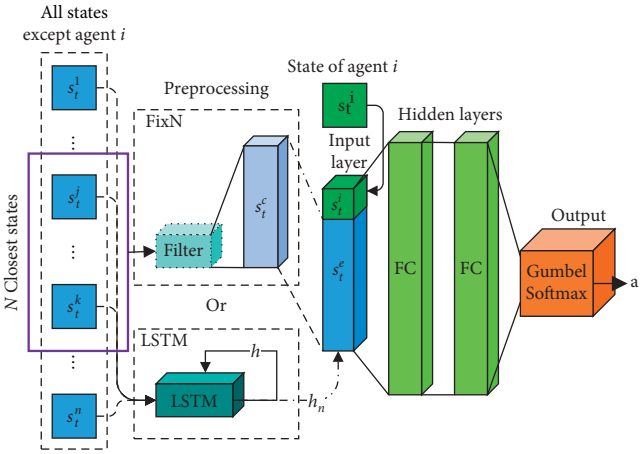


Figure 2: Illustration of the actor neural network architecture.

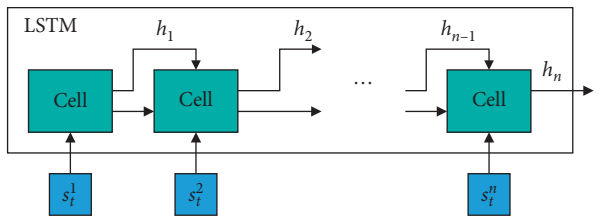


Figure 3: LSTM unrolled diagram.

the joint action set $\mathbf{a}_t = (a_t^1, \ldots, a_t^n)$. At the time step t, agents receive representation of the environment's state "st", and on that basis select actions "at". As a consequence of their actions, agents receive a reward "Rt" and update the environment to a new state "st, ". The set $(\mathbf{s}_t, \mathbf{a}_t, R_t, \mathbf{s}_t')$ of each step will be stored in the experience pool **D**. When the number of cases in the experience pool reaches a certain number, we randomly sample M groups of sets from **D** to train all networks. CriticD first generates Q-value (Q_t) according to the state $\mathbf{s}_t$ and the joint action $\mathbf{a}_t$. Then, ActorE generates joint action $\mathbf{a}_t'$ according to the state $\mathbf{s}_t'$. Then, CriticE generates Q_t' based on $\mathbf{s}_t'$ and a'. Thus, the error (loss) value of CriticD network can be calculated by the following formula:

$$L(\omega) = E\left[\left(Q_t(\mathbf{s}_t, \mathbf{a}_t) - R_t - \gamma Q_t'(\mathbf{s}_t', \mathbf{a}_t')\right)^2\right]. \quad (6)$$

The network parameters of CriticD are updated by minimizing the loss.

The policy gradient of ActorD network is calculated by the following equation:

$$\nabla_\theta J(\pi) = E\left[\nabla_a Q(\mathbf{s}_t, \mathbf{a}_t) \cdot \nabla_\theta \pi(\mathbf{s}_t)\right], \quad (7)$$

where $J(\pi) = E[R]$ as shown in (2). We use Adam optimizer to update the network parameters of ActorD with $\nabla_\theta J(\pi)$. It should be noted that all LSTM networks are uniformly updated using the AdaGrad optimizer with $\nabla_\theta J(\pi)$.

We provide the multiagent computational guidance algorithm in Algorithm 1.

During the execution period, only an independent ActorD network on each UAV will be used to make decisions and select actions onboard.

5. Simulation and Result Analysis

5.1. Simulator and Interface. To train the agents and test the performance of the MACG algorithm, we built a simulator in an OpenAI Gym environment where a large number of UAVs can fly in the 2D airspace. The limited airspace has 34 km width and 34 km length. As shown in Figure 4, the UAVs will be initialized at the vertices in a regular polygon. In each episode, the simulator will generate a fixed number of UAVs stochastically.

In this study, the cruising speed of an UAV is set to 60 m/s to verify the effectiveness of the algorithm in the high-speed scene [24]. Because there is uncertainty in flight velocity, the initial velocity is restricted to be between Vmax = 80 m/s and Vmin = 40 m/s, and the cruising speed will be with a standard deviation of 5 m/s for the duration of the flight. The noises here aim to account for the uncertainties because fixed-wing UAVs are flying at a higher speed than quadrotors.

In the simulator, a collision is defined when the distance between two UAVs is less than 0.1 km. There will be a warning if the shortest distance between UAVs is less than 0.5 km, which means that they have lost the separation. During the running of this simulator, the number of collisions and warnings will be recorded and returned at the end of a simulation.

Then, the simulator will keep running until max_num of UAVs have been generated or the episode time runs out. The max time step in an episode is usually set as

$$T_{\max} = 500 * \text{max\_num}. \quad (8)$$

The output of this simulator is a current reward and a state's sequence of all UAVs in the environment. The input is the joint action sequence of all UAVs.

Two kinds of network structures, FixN and LSTM, are used to carry out simulation experiments in this paper. According to Figure 2, we choose $N = 3$ for the FixN-MACG algorithm, so the input layer will be 24 neurons. For the LSTM-MACG algorithm, we let the LSTM network have a hidden layer with 32 nodes and limit its output to 18 parameters. Softsign function is selected as LSTM network activation function, and the input data need to be regularized. The policy of the actor is parameterized by a two-layer ReLU MLP with 128 units per layer.

The hardware environment used in this paper includes Intel i5-9600k, GTX1060, DDR4 16 GB, and 240 GB SSD. The operation system is Windows10 x64, and the toolkit version is TensorFlow 2.1.0 based on Python 3.7.

5.2. Training Case and Results. In this training case, we design a scenario that generates UAVs randomly. There are six birthplaces distributed at the vertexes of a regular hexagon, and the distance between them is 16 km. At each birthplace, after the previous UAV flies away, the time interval for the next UAV to be generated is uniformly distributed between 1 and 3 minutes, and the new flight will select a nonneighbor vertex as its destination stochastically.

(1) Initialize ActorD $\pi(\mathbf{s})$ and CriticD $Q(\mathbf{s}, \mathbf{a})$ with the weights θ and ω randomly chosen.
(2) Initialize ActorE $\pi'(\mathbf{s})$ and CriticE $Q'(\mathbf{s}, \mathbf{a})$ with the weights $\theta' = \theta$ and $\omega' = \omega$.
(3) Initialize the experience pool **D** and a counter $C_d = 0$.
(4) **for** episode = 1 to Max-Episode **do**
(5) Initialize the joint action set $\mathbf{a} = (0, \ldots, 0) \in R^n$.
(6) Reset the environment **S**.
(7) Set the max time steps in an episode, $T_{\max}$.
(8) **for** $t = 1$ to $T_{\max}$ **do**
(9) Generate UAVs randomly.
(10) For each agent i, select action a_i by ActorD and combine the actions, $\mathbf{a}_t = (a_t^1, \ldots, a_t^n)$.
(11) Execute $\mathbf{a}_t$ and obtain $\mathbf{s}_t, \mathbf{R}_t, \mathbf{s}_t'$.
(12) Store $(\mathbf{s}_t, \mathbf{a}_t, R_t, \mathbf{s}_t')$ in **D**, C_d + +.
(13) Replace $\mathbf{s}_t$ with $\mathbf{s}_t'$.
(14) **if** ($C_d >$ batch_size) and (t % sample_time = 0) **do**
(15) Sample a random minibatch of M from **D**.
(16) Use CriticD to get Q_t, use ActorE to get $\mathbf{a}_t'$, and use CriticE to get Q_t'.
(17) Update CriticD by minimizing the loss $L(\omega)$ in equation (6).
(18) Update ActorD using the sampled policy gradient $\nabla_\theta J(\pi)$ in equation (7).
(19) **if** t% replace_time = 0 **do**
(20) Update the ActorE and CriticE networks: $\theta' = \tau\theta + (1 - \tau)\theta'$, $\omega' = \tau\omega + (1 - \tau)\omega'$
(21) (19) T **if** number of UAVs > max_num **do**
(22) (20) break
(23) **end for** (t)
(24) **end for** (episode)

Algorithm 1: MACG algorithm.

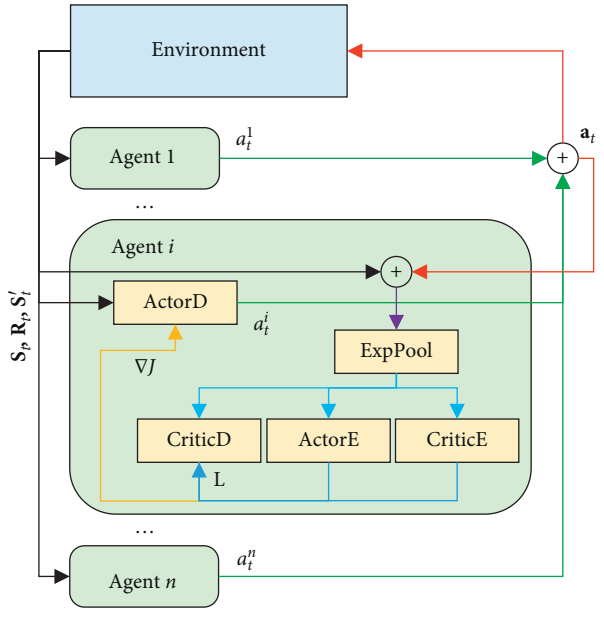
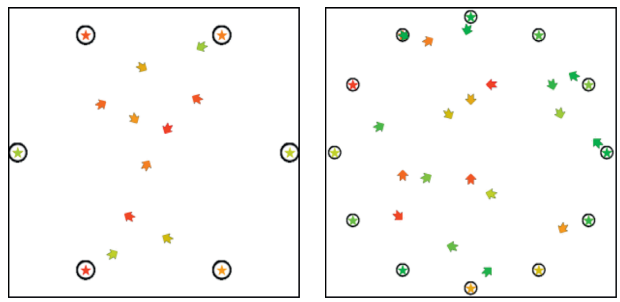


Figure 4: Training framework diagram.

A screenshot of the simulation running scenario is shown in Figure 5(a).

In the case study, the max number of episodes is set to be 4×10^4. In each episode, there will be 200 UAVs generated, and the max time step is $T_{\max} = 10^5$. The reinforcement learning system includes these parameters: $\tau = 0.01$, $\gamma = 0.98$, batch_size = 10^4, and $M = 10^3$, and the optimizers use default parameters. The neural networks will be stored every 100 episodes during the training process.

Figure 6 plots the learning curve of the relationship between cumulative reward and episodes of the two MACG algorithms. From this figure, we can see that the FixN-MACG has a faster convergence rate, but the final average cumulative reward is lower than that of LSTM-MACG. This



(a) (b)

Figure 5: Screenshots in simulation running scenarios. (a) Training case. (b) Stress test case.

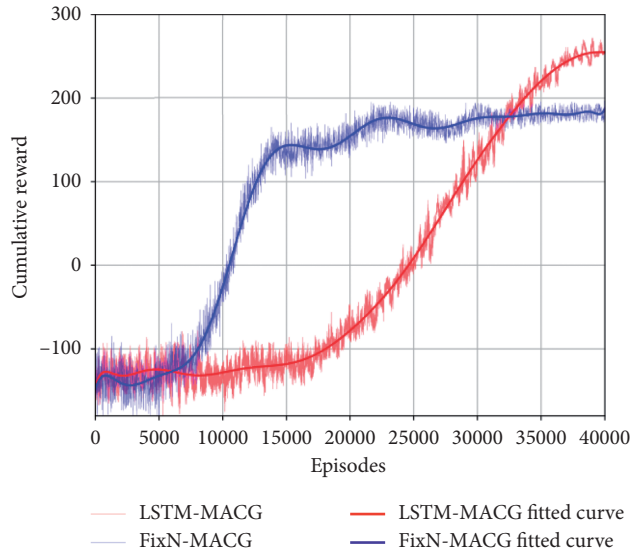


Figure 6: Cumulative reward after the training episodes.

is because the FixN-MACG algorithm does not consider the global states, so the training speed is better than the other variant. We can see the process starts to converge around 37000 episodes for LSTM-MACG algorithm. These curves show that LSTM-MACG algorithm has good performance in guidance and collision avoidance, and it can effectively reduce the conflict risk for UAVs. The FixN-MACG method cannot guarantee that there is no collision at all.

After the training, to provide a fair comparison, we analyze the final policy of each variant for 1000 independent episodes. We also change the number of UAVs in the airspace from 4 to 200 to evaluate the performance of different frameworks. It should be noted that each UAV uses an independent guidance policy to select actions onboard in these execution simulation processes.

Figure 7 shows the total computation time required for all the UAVs to run the onboard algorithms in a simulation time step. Figure 8 shows the relationship between the number of collisions and the number of UAVs in the airspace when two MACG methods are used to guide aircraft. By discussing these two figures, we can know that LSTM-MACG can achieve UAV collision avoidance guidance in high-density route airspace. FixN-MACG can only achieve complete conflict avoidance in case of a small number of UAVs. However, the operation speed of the former is much lower because the LSTM network structure needs to process the status of all UAVs in the environment. The computation time for both methods is growing with the increase of the number of UAVs, and the calculation time of LSTM-MACG increases exponentially.

We recorded the number of UAVs at each time step in Figure 9. From this figure, we can see that there are more than 20 UAVs flying in the limited airspace most of the time, which further proved the effectiveness of LSTM-MACG algorithm to avoid collision of multiple UAVs in high-density route airspace.

5.3. Stress Test Case and Results. To further test the performance of the proposed LSTM-MACG algorithm, we design a multithreat stress test scenario of randomly generated UAVs. Compared with the previous case, the total number of UAVs in the environment ranged from 100 to

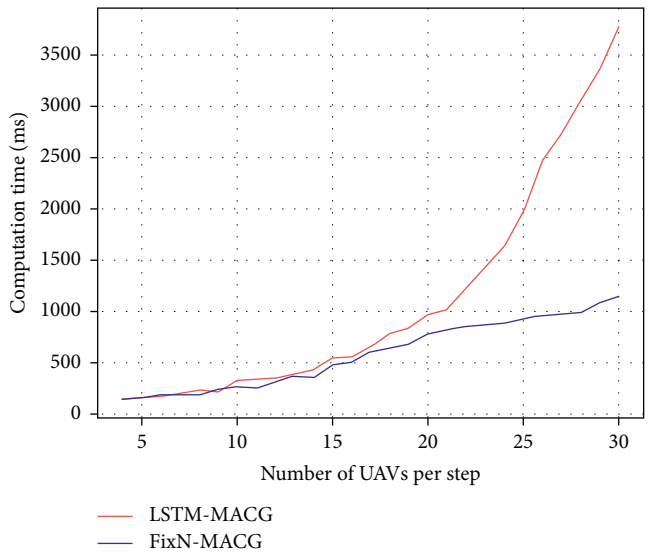


Figure 7: Computation time with the number of UAVs.

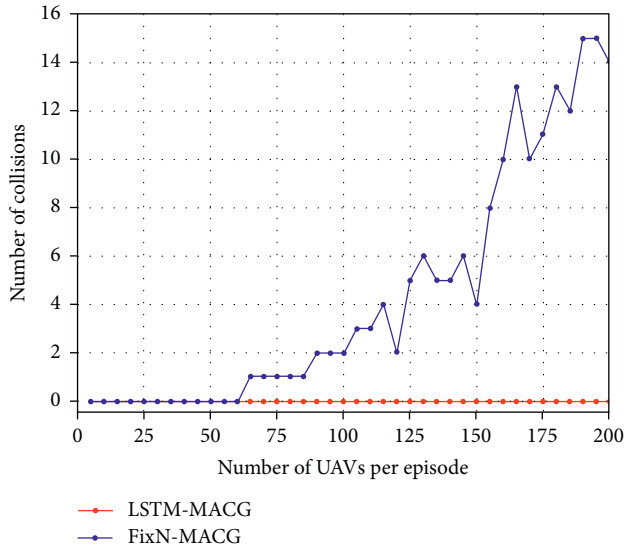


Figure 8: Collision counting of two algorithms.

400 in this case study. The number of birth points has increased from 6 to 12, and the time interval of newly generated aircraft has been expanded to 50 s to 200 s. UAVs are still randomly generated at each vertex. Figure 5(b) shows an example of the stress test scenario.

Here we also recorded the number of UAVs per step and plotted the histogram in Figure 10. We can see that the maximum UAV density in the airspace is more than 50 per time step. Figure 11 presents the performance of the proposed LSTM-MACG algorithm in the stress test case study, where the curves denotes the average of result data in 1000 episodes. From the figure, we can see that the collision probability is less than 0.1% for UAVs by using the LSTM-MACG onboard. We also give the result curve of the baseline, where all the UAVs take no actions and they just fly to their destinations straightly. The comparison of two curves in this figure shows that the proposed method has good performance even in the case of

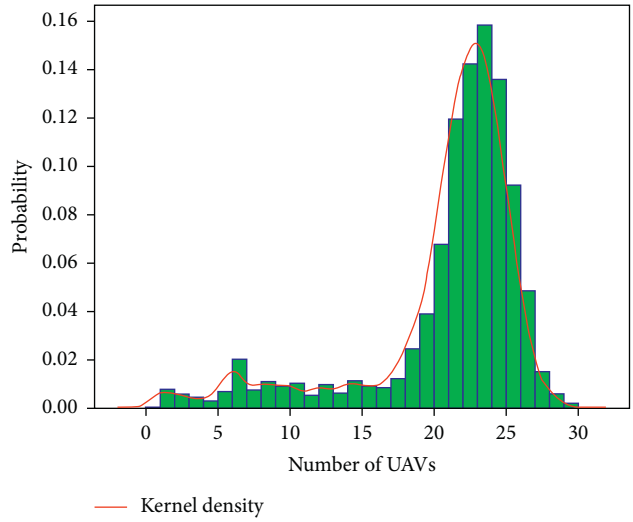


Figure 9: Statistical histogram of UAVs in a single step.

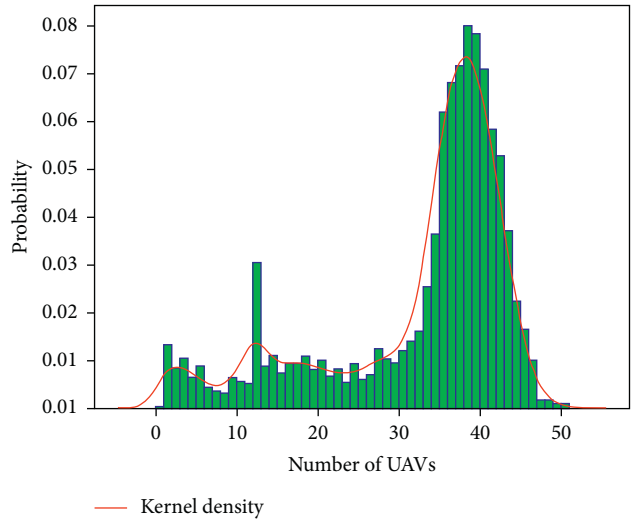


Figure 10: Histogram of UAVs in a time step of test case.

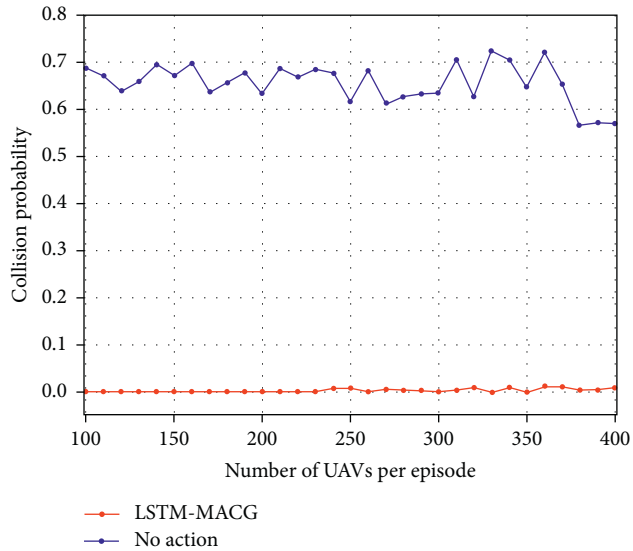


Figure 11: Conflict probability in the test case.

high-density route airspace. This result is helpful to increase the total number of fixed-wing UAVs in a limited airspace.

6. Conclusions

In this paper, a novel approach has been proposed to provide a solution to the problem of collision avoidance guidance of UAVs in a limited airspace. For this purpose, two kinds of decision-making networks have been designed, specifically trained for the decentralized computational guidance scenarios. The algorithm can be scaled to multiple cooperative UAVs, where we formulate the problem as a Markov game. A flight airspace simulator is built to validate the performance of the proposed algorithm. A stress test is carried out to evaluate the algorithm. The simulation results show that this algorithm has good performance to guide the UAVs to reach their destinations and avoid conflicts even for the high-density route airspace. The contributions of this paper include the following: (1) a self-learning system is designed for training guidance decision networks onboard; (2) LSTM networks and deep neural networks are incorporated to handle a variable number of UAVs in a limited airspace to enable an automated and safe environment; and (3) the concept of computational guidance is explored for fixed-wing UAVs. However, the designed framework is still in the exploratory phase. Further research should focus on making the algorithm more practical in real-world applications.

Data Availability

The data used to analyze the results of this study are included within the article. Besides, all data included in this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China under grant no. 61973101 and the Aeronautical Science Foundation of China under grant no. 20180577005.

References

[1] Y. Liu, X. Zhang, Y. Zhang, and X. Guan, "Collision free 4D path planning for multiple UAVs based on spatial refined voting mechanism and PSO approach," *Chinese Journal of Aeronautics*, vol. 32, no. 6, pp. 1504–1519, 2019.

[2] Z. Chen and H. Jia, "Design of flight control system for a novel tilt-rotor UAV," *Complexity*, vol. 2020, no. 4, 14 pages, Article ID 4757381, 2020.

[3] J. Zhang, J. Yan, and P. Zhang, "Fixed-wing UAV formation control design with collision avoidance based on an improved artificial potential field," *IEEE Access*, vol. 6, pp. 78342–78351, 2018.

[4] C. Huang, Y. Lan, Y. Liu et al., "A new dynamic path planning approach for unmanned aerial vehicles," *Complexity*, vol. 2018, Article ID 8420294, 17 pages, 2018.

[5] I. Alzugaray and A. Sanfeliu, "Learning the hidden human knowledge of UAV pilots when navigating in a cluttered environment for improving path planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1589–1594, Deajeon, South Korea, October 2016.

[6] M. Campbell, S. Sukkarieh, and A. Goktogan, "Operator decision modeling in cooperative UAV systems," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–13, Keystone, CO, USA, August 2006.

[7] P. Kopardekar, J. Rios, T. Prevot et al., "Unmanned aircraft system traffic management (UTM) concept of operations," in *Proceedings of the16th AIAA Aviation Technology, Integration, and Operations Conference*, pp. 1–16, Washington, DC, USA, June 2016.

[8] H. A. P. Blom and G. J. Bakker, "Safety evaluation of advanced self-separation under very high en route traffic demand," *Journal of Aerospace Information Systems*, vol. 12, no. 6, pp. 413–427, 2015.

[9] M. Everett, Y. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3052–3059, Madrid, Spain, October 2018.

[10] H. Erzberger and K. Heere, "Algorithm and operational concept for resolving short-range conflicts," *Journal of Aerospace Engineering*, vol. 224, no. 6, pp. 225–243, 2010.

[11] J. K. Kuchar, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[12] H. Zhou, R. Wei, J. Cui et al., "Multi-UAV cooperative collision avoidance against uncertain environment," *Electronics Optics and Control*, vol. 21, no. 1, pp. 91–96, 2014.

[13] X. Yang and P. Wei, "Scalable multiagent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1–14, 2020.

[14] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, 1998.

[15] T. Kravaris, C. Spatharis, A. Bastas et al., "Resolving congestions in the air traffic management domain via multiagent reinforcement learning methods," 2019, https://arxiv.org/abs/1912.06860.

[16] A. M. F. Crespo, W. Li, and A. G. Barros, "Reinforcement learning agents to tactical air traffic flow management," *International Journal of Aviation Management*, vol. 1, no. 3, pp. 145–161, 2017.

[17] C. W. Keong, H. Shin, and A. Tsourdos, "Reinforcement learning for autonomous aircraft avoidance," in *The 2019 International Workshop on Research, Education and Development on Unmanned Aerial Systems*, pp. 126–131, Cranfield, UK, 2019.

[18] R. Lowe, Y. Wu, A. Tamar et al., "Multiagent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1–16, 2017.

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

[20] P. Champasak, N. Panagant, N. Pholdee et al., "Self-adaptive many-objective meta-heuristic based on decomposition for many-objective conceptual design of a fixed-wing unmanned aerial vehicle," *Aerospace Science and Technology*, vol. 100, pp. 1–11, Article ID 105783, 2020.

[21] S. He, H.-S. Shin, and A. Tsourdos, "Computational guidance using sparse gauss-hermite quadrature differential dynamic programming," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 13–18, 2019.

[22] M. L. Littman, "Markov games as a framework for multiagent reinforcement learning," in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163, Boca Raton, FL, USA, July 1994.

[23] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pp. 1–13, Toulon, France, April 2017.

[24] P. Pradeep and P. Wei, "Energy optimal speed profile for arrival of tandem tilt-wing eVTOL aircraft with RTA constraint," in *Proceedings of the 2018 IEEE CSAA Guidance, Navigation and Control Conference*, Beijing, China, August 2018.