applied sciences

MDPI

Article

Automated Enemy Avoidance of Unmanned Aerial Vehicles Based on Reinforcement Learning

Qiao Cheng *, Xiangke Wang *, Jian Yang and Lincheng Shen

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China; yj_ntx@163.com (J.Y.); lcshen@nudt.edu.cn (L.S.)

* Correspondence: qiao.cheng@nudt.edu.cn (Q.C.); xkwang@nudt.edu.cn (X.W.)

Received: 23 November 2018; Accepted: 8 February 2019; Published: 15 February 2019

check for updates

Abstract: This paper focuses on one of the collision avoidance scenarios for unmanned aerial vehicles (UAVs), where the UAV needs to avoid collision with the enemy UAV during its flying path to the goal point. Such a type of problem is defined as the enemy avoidance problem in this paper. To deal with this problem, a learning based framework is proposed. Under this framework, the enemy avoidance problem is formulated as a Markov Decision Process (MDP), and the maneuver policies for the UAV are learned based on a temporal-difference reinforcement learning method called Sarsa. To handle the enemy avoidance problem in continuous state space, the Cerebellar Model Arithmetic Computer (CMAC) function approximation technique is embodied in the proposed framework. Furthermore, a hardware-in-the-loop (HITL) simulation environment is established. Simulation results show that the UAV agent can learn a satisfying policy under the proposed framework. Comparing with the random policy and the fixed-rule policy, the learned policy can achieve a far higher possibility in reaching the goal point without colliding with the enemy UAV.

Keywords: enemy avoidance; reinforcement learning; decision making; hardware-in-the-loop simulation; unmanned aerial vehicles

1. Introduction

Unmanned Aerial Vehicles (UAVs) have received considerable attention in many areas [1], such as commercial, search and rescue, military, and so on. In the military area, there are applications such as the surveillance [2,3], target tracking [4,5], target following [6,7], and so on. Among these applications, collision avoidance is one of the most important concerns [8], especially in unsafe environment. In such cases, a UAV should keep safe separation with various kinds of objects, such as static obstacles [9,10], teammates [11], and moving enemies. The strategies toward different approaching objects are different due to specific requirements in dealing with those objects. This paper focuses on avoiding the collision with moving enemies. There are many researches on collision avoidance problems. However, relatively fewer works are on the avoidance of moving enemies, comparing with those on the avoidance of static obstacles and flying teammates. Furthermore, the uncertain motion of enemies and the necessity to attack enemies create more challenges on the avoidance of moving enemies than avoiding other objects. Besides, the mission of the UAV, such as reaching a specific goal destination, should also be considered. Therefore, the avoidance of moving enemies is a challenge problem, and such a problem is defined as the enemy avoidance problem in this paper.

There are many approaches to handle the collision avoidance problem in different stages [12]. Many of those approaches rely on models for the dynamic of the environment and UAVs. However, the accuracy of these models can sometimes greatly affect the performance of those methods. Moreover, building these models is not easy work, and is even impractical. On the other hand, a complex model means heavy computation load when making decisions. Therefore, learning methods are increasingly used in collision avoidance problems, which are based on collected data. However, most of these learning methods are used to predict the effect of the decision, not directly used for the decision making. Different from other learning methods, reinforcement learning is a very popular method for sequential decision making problems [13]. It can learn to make decisions incrementally based on feedback from the environment. Therefore, it can generate a good policy even if the models of the environment are unknown. Since a sequence of appropriate actions are required to avoid the enemy UAV, the enemy avoidance problem can be regarded as a sequential decision making problem. Therefore, this paper proposes a framework which incorporates the reinforcement learning to deal with the enemy avoidance problem.

Many methods for the collision avoidance problem discretize the state space to make decisions [14,15]. However, this paper studies the enemy avoidance problem in continuous state space. Therefore, the function approximation technique, which can handle continuous space, is also embodied in the proposed framework.

As for the UAVs, most researches [16–18] focus on quadrotors rather than fixed-wing UAVs in collision avoidance problems. However, the dynamics of quadrotors and fixed-wing UAVs are different. In addition, the UAV in the enemy avoidance problem has the mission to reach the goal point, and thereby needs to keep away from the enemy UAV and even attack enemy UAVs. In practical application, the fixed-wing UAVs are more suitable for such a problem scenario for their better mission fulfillment properties, such as higher endurance and greater speeds. Therefore, this paper focuses on the enemy avoidance problem of fixed-wing UAVs.

Since learning the policy on the real UAV platform would bring about great consumption, a hardware-in-the-loop (HITL) simulation system is constructed. With hardware-in-the-loop, the simulation system can provide very consistent properties to that of the real environment, which highly respects the kinecmatic and maneuver constraints of the UAVs. Furthermore, it saves the energy to build a model for the related hardware, which is usually very hard to build accurately. Comparing with the real UAV platform, the HITL simulation system can repeat the experiments as many times as needed without worrying about UAV costs.

The contributions of this paper are summarized as follows.

(i) An interesting new problem called the enemy avoidance problem is defined, which can be a good adding up scenario to the collision avoidance problem. The newly defined problem is different from most of the existing collision avoidance problems, for it is to avoid the collision with the enemy UAV rather than static obstacles or moving teammates.

(ii) A novel framework is proposed to learn the policy for the decision making UAV. The proposed framework formulates the enemy avoidance problem as a Markov Decision Process (MDP) problem, and solves the MDP problem by a temporal-difference reinforcement learning method called Sarsa. The Cerebellar Model Arithmetic Computer (CMAC, [19]) technique is also embodied in the proposed framework for the generalization of the continuous state space. With this framework, such a decision making problem is transformed from the usual computational problem to a learning problem. Besides, it can learn the policy with an unknown environment model, and can make decisions based on continuous state space rather than discrete ones like most existing works do.

(iii) A hardware-in-the-loop (HITL) simulation environment for the enemy avoidance problem is constructed, which is used for the policy learning and policy testing experiments. Different from the simulation environment in most of the existing works, this HITL simulation system saves a lot of model designing trouble, and has better consistency to the real environment, such as the environment noise. When comparing with real environment platforms, the HITL simulation system has the advantage of saving experimental cost.

The remainder of this paper is outlined as follows. Section 2 gives some reviews on the related literature. The enemy avoidance problems are presented in Section 3. The proposed framework for the enemy avoidance problem is elaborated in Section 4. Section 5 details the construction of the hardware-in-the-loop simulation environment. Simulation experiments and results are illustrated in Section 6. Finally, Section 7 concludes the whole work and discusses future works.

2. Literature Review

There are many researches on collision avoidance problems, and different methods are used to solve different collision avoidance problems. Therefore, this section will give a summary about several widely used methods in collision avoidance problems, as well as a comparison between our work and these existing works.

One of the most widely used methods is to formulate the collision avoidance problem as an optimization problem, while considering all kinds of constrains. Therefore, to avoid collision is to solve the optimization problem with appropriate methods under different constrains. The work in [20] formulates the collision avoidance problem as a convex optimization problem, and seeks for a suitable control constraint set for participating UAV based on reachable sets and tubes for UAVs. This method is limited to linear systems. The collision avoidance in work [21] is formulated as a set of linear quadratic optimization problems, which are solved with an original geometric based formulation. To handle flocking control with obstacle avoidance, work [22] proposes a UAV distributed flocking control algorithm based on the modified multi-objective pigeon-inspired optimization (MPIO), which considers both the hard constraints and the soft ones. Our previous works [23,24] formulate the conflict avoidance problem as a nonlinear optimization problem, and then use different methods to solve such an optimization problem. The work in [23] proposes a two-layered mechanism to guarantee safe separation, which finds the optimal heading change solutions with the vectorized stochastic parallel gradient descent-based method, and finds the optimal speed change solutions with a mixed integer linear programming model. The work in [24] uses the stochastic parallel gradient descent (SPGD) method to find the feasible initial solutions, and then uses the Sequential quadratic programming (SQP) algorithm to compute the local optimal solution. Even for the obstacle avoidance problem in other areas, the optimization methods are also used. For example, two swarm based optimization techniques are used in work [25] to offer obstacle-avoidance path planning for mobility-assisted localization in wireless sensor networks (WSN), which are grey wolf optimizer and whale optimization algorithm. The main difference between the collision avoidance for UAVs and the obstacle-avoidance path planning in WSN lies in the constrains and objective of the optimization model. Usually, solving the optimization problem requires a lot of computation. Therefore, our work does not formulate the enemy avoidance problem as an optimization problem, but formulates it as an MDP problem and solves the MDP incrementally by interaction with the environment.

Another kind of method for solving collision avoidance problems is to predict the potential collision with certain techniques. The work in [26] proposes an approach based on radio signal strengths (RSS) measurements to obtain position estimation of the UAV, and to detect the potential collisions based on the position estimations, and then to distribute the UAVs at different altitudes to avoid collision. The work in [27] proposes a model-based learning algorithm that enables the agent to learn an uncertainty-aware collision prediction model through deep neural networks, so as to avoid the collision with unknown static obstacles. The work in [28] proposes a data-driven end-to-end motion planing approach which helps the robot navigate to a desired target point while avoiding collisions with static obstacles without the need of a global map. This approach is based on convolutional neural networks (CNNs), and the robot is provided with expert demonstrations about navigation in a given virtual training environment. One of the problems for such kind of methods is that high capacity learning algorithms like deep learning tend to overfit when little training data is available. Therefore, the work in [29] collects a lot of crash samples to build a dataset by crashing their drone 11,500 times. The used data driven approach demonstrates such negative data is also crucial for learning how to navigate without collision. However, to collect both positive data and negative data for prediction is very costly. Different from these works, our work aims to obtain a policy with the proposed framework.

The policy is a mapping from the state directly to the action, therefore, no prediction of the collision is needed.

As reinforcement learning gains its popularity in decision making problems, there are works that use different reinforcement learning to solve the collision avoidance problem. The work in [30] proposes to combine Model Predictive Control (MPC) with reinforcement learning to learn obstacle avoidance policies for the UAV in a simulation environment. In this method, the MPC is used to generate data at the training time, and the deep neural network policies are trained with an off-policy reinforcement learning method called guided policy search based on the generated data. The work in [31] proposes a geometric reinforcement learning algorithm for UAV path planning, which constructs a specific reward matrix to include the geometric distance and risk information. This algorithm considers the obstacles as risk and builds a risk model for the obstacles, which is used in constructing the reward matrix. The work in [32] models the UAV collision avoidance problem as a Partially Observable Markov Decision Process (POMDP) and uses Monte Carlo Value Iteration (MCVI) to solve the POMDPs, which can cope with high-dimensional continuous-state space in a collision avoidance problem. The work in [33] formulates the problem of collision avoidance as an MDP and a POMDP, and uses generic MDP/POMDP solvers to generate avoidance strategies. Though the framework proposed in our work is also based on reinforcement learning, many details are different from the these works. For example, this paper uses neither special data generating process, nor complex reward function designing. Besides, the environment transition model is unknown in this paper, and a different reinforcement learning method is adopted.

Another big difference between the existing works and our work is that the collision avoidance problem in this paper is not exactly the same as those in previous works. First, the UAV in this paper needs to avoid collision with a moving enemy UAV, not static obstacles [25] or teammates [11]. Besides, the actions the UAV uses to avoid collision include both heading angle change and velocity change. The work in [34] investigated strategies for multiple UAVs to avoid collision with moving obstacles, which is a little similar with collision with enemy UAV. However, their work assumes all UAVs and all obstacles have constant ground speeds, and the direction of the velocity vector of an obstacle is constant. Therefore, they only consider change in direction of the UAV for collision avoidance, and do not consider change in velocity of the UAV. Furthermore, this paper does not attempt to build an environment model, but approximates it by continuous interaction with the environment. Similarly, work [35] also approximates the unmodeled dynamics of the environment, but it uses back propagation neural networks and proposes a tree search algorithm to find the near optimal conflict avoidance solutions. In addition, this paper considers continuous state space in the decision making process, not the discrete one like many other related works do.

3. Problem Definition

We call the problem posed in this paper the enemy avoidance problem, which is different from the usual collision avoidance problems or path planning problems. Before proposing methods to solve this problem, we first give a detailed description for the enemy avoidance problem, as well as the related assumptions and definitions.

3.1. Problem Description

For the convenience of the research, we define the enemy avoidance problem in a fixed region. There are two UAVs flying toward each other in the region, namely the decision making UAV f and the enemy UAV e. The fixed region is where the two UAVs may collide with each other. The decision making UAV f enters the region from the left side, while the enemy UAV e enters the region from the right side. Both UAVs are flying toward their own goal points. Let G_f and G_e denote the goal point of the decision making UAV f and that of the enemy UAV e, respectively. The goal point G_f for the decision making UAV f is located near the right edge of the region, while the goal point G_e for the enemy UAV e is located near the left edge of the region. Both goal points are on the middle line of the region, as shown in Figure 1.

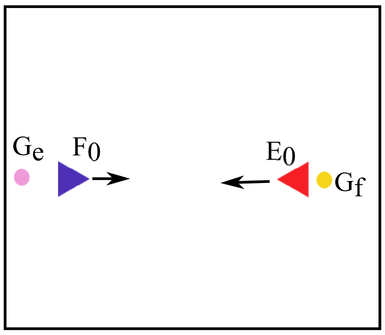

Figure 1. The enemy avoidance scenarios.

In such a region, there is no hovering requirement or taking off and landing requirements, therefore the fixed-wing UAV can be easily applied in such problem settings.

The mission of the decision making UAV f is to reach the goal point G_f safely and with as little cost as possible. However, the decision making UAV f and the enemy UAV e are flying in opposite directions along the middle line of the region toward their own goal points, which poses the decision making UAV f the danger of collision with the enemy UAV e. Therefore, the decision making UAV f needs to avoid the enemy UAV e during its flight towards the goal point G_f. Ways to avoid collision with the enemy UAV include changing the heading angle or the velocity, and attacking the enemy UAV. Though changing the heading angle can let the decision making UAV avoid flying directly into the enemy UAV if the the enemy UAV happens to be in the heading direction of the decision making UAV, inappropriate heading angle change may make the decision making UAV fly too far away from the goal point G_f. Similarly, changing the velocity at an appropriate time can also avoid collision with the enemy UAV, such as accelerating to pass the enemy UAV before the collision or decelerating to wait for the enemy UAV to pass. Successfully attacking the enemy UAV can also provide good insurance for the decision making UAV to fulfill its mission. However, the attacking action may fail in destroying the enemy UAV, and the decision making UAV suffers certain losses when using attacking action. Therefore, the decision making UAV should not use the attacking action too often. To achieve the mission requirements, the decision making UAV cannot use just one single avoiding action, but should arrange all the actions in an appropriate sequence.

On the other hand, the enemy UAV simply flies toward the goal point G_e with constant velocity and heading angle if the decision making UAV does not collide with it or attack it successfully. The scenario is supposed to end as soon as the decision making UAV has collided with or successfully attacked the enemy UAV, or the decision making UAV has reached the goal point G_f successfully or has been out of the region.

To arrange an appropriate action sequence is the process of decision making, which is also the main focus of this work. In decision making, such an action sequence is called the policy. Based on the action chosen by the on-board agent at each decision making step, the decision making UAV can adjust its flying attitude or attack the enemy UAV. The agent makes decisions based on both its own information from its on-board sensor system and the enemy UAV's information from the ground station. The action executed by the decision making UAV makes the UAV changes its state in the environment. On the other hand, the enemy UAV also updates its states in the environment and senses its own states from the environment with its on-board sensor system. The ground station captures all UAVs' information, and then transmits the information to the decision making UAV. Figure 2 presents the overall decision making process of the enemy avoidance problem. Therefore, how the decision making agent uses the gathered information to make decisions for avoiding collision with the enemy UAV is what this paper is going to solve. Furthermore, the time and the location at which the enemy

UAV enters the region are not fixed each time. Therefore, the decision making ability of the decision making UAV should be able to generalize to different enemy avoidance cases.

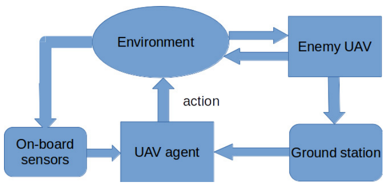

Figure 2. The decision making framework.

3.2. Assumptions and Definitions

At first, we need to make some assumptions about the posed enemy avoidance problem.

Assumption 1. *The enemy UAV has constant desired velocity and desired heading angle, while the actual velocity and heading angle of the enemy UAV oscillate a little around the desired ones.*

Assumption 2. *The decision making UAV can change its attitude and attack the enemy UAV during its flight according to the action decided by the UAV agent.*

Assumption 3. *The heights of the UAVs are not considered. Therefore, all the distances in the problem are simply computed by two dimensions.*

Assumption 4. *Each UAV obtains its own position and attitude (velocity and heading angle) with its on-board sensor system. The decision making UAV can obtain information about all the UAVs through the ground station.*

Assumption 5. *There are no other UAVs in the region except the decision making UAV and the enemy UAV, as well as no obstacles in the region.*

In these assumptions, Assumptions 1 and 3 are used to simplify the enemy avoidance problem, so that we can focus more on other more important factors in the enemy avoidance problem. The researched results then can be used as the basis of more practical problems. With Assumption 5, this paper can focus on the collision avoidance of the enemy UAV, and does not need to consider collision with other UAVs and obstacles.

Furthermore, we give some definitions that will be used in solving the enemy avoidance problem.

Definition 1. *(Distance). The distance between two points $a = (x_a, y_a)$ and $b = (x_b, y_b)$ is calculated with the following equation:*

$$d_{ab} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{1}$$

Definition 2. *(Reaching Goal). A UAV f is regarded as having reached a goal G when the following condition is satisfied:*

$$d_{fg} < r_g \tag{2}$$

where d_{fg} denotes the distance between the UAV f and the goal point G, and r_g is the specified goal radius.

Definition 3. *(Collision). A UAV f is regarded as having collided with the enemy UAV e when the following condition is satisfied:*

$$d_{fe} < r_c \tag{3}$$

where d_{fe} denotes the distance between the UAV f and the enemy UAV e, and r_c is the specified collision radius.

Definition 4. *(Attacking Probability). The success of an attacking action is defined by the attacking probability P, which is specified by the following equation:*

$$P = e^{1-\frac{d_{fe}}{30}} \tag{4}$$

4. Problem Sovling

This paper proposes a new framework to solve the enemy avoidance problem, which formulates the enemy avoidance problem as the Markov Decision Process (MDP) and learns the decision making policy for the enemy avoidance problem based on reinforcement learning. Firstly, the detail of formulating the enemy avoidance problem as the Markov Decision Process (MDP) is presented, which is the basis of the reinforcement learning. Secondly, how to learn the policy based on a temporal-difference reinforcement learning method called Sarsa is elaborated, as well as the embodied function approximator called CMAC for the generalization of the continous state.

4.1. Formulate the Problem as the MDP

Reinforcement learning has been widely used in sequential decision making problems which are formulated as the Markov Decision Process (MDP). Typically, an MDP comprises of four elements: the state set $\mathcal{S}$, the action set $\mathcal{A}$, the transition function $\mathcal{T}$, and the reward function $\mathcal{R}$. When an agent is in a state $s \in \mathcal{S}$, it can choose an action $a \in \mathcal{A}$ according to a policy π. After executing the action a, the agent will enter into the next state $s' \in \mathcal{S}$ according to the transition function $\mathcal{T}$, and will receive an immediate reward r according to the reward function $\mathcal{R}$. In this enemy avoidance problem, the environment transition function $\mathcal{T}$ is unknown, and will be learned by interaction with the environment. To formulate the enemy avoidance problem as the MDP, the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the reward function $\mathcal{R}$ are defined as follows.

4.1.1. State Space

In this enemy avoidance problem, the design of the state space mainly considers the positions and attitudes of the UAVs, as well as the position of the goal point G_f. However, these raw data are not used directly as the state variables. Instead, higher-level variables based on these data are defined. To be specific, the state space contains three sets of variables:

(i) Variables about the status of the decision making UAV f:

- v_f: The velocity of the decision making UAV f.
- ψ_f: The heading angle of the decision making UAV f.

(ii) Variables about the goal point G_f:

- d_{fg}: The distance from the decision making UAV f to the goal point G_f.
- ω_g: The angle between the north direction and the line from the decision making UAV f to the goal point G_f.

(iii) Variables about the status of the enemy UAV e:

- v_e: The velocity of the enemy UAV e.
- ψ_e: The heading angle of the enemy UAV e.
- d_{fe}: The distance from the decision making UAV f to the enemy UAV e.
- ω_e: The angle between the north direction and the line from the decision making UAV f to the enemy UAV e.

As we can see, there are 8 state variables in total. Figure 3 illustrates these state variables. The value ranges for all the velocity variables are $[10, 20]$, while the value ranges for all the angle variables are $[0, 360)$. Suppose the length and width of the region are l and w, respectively. Thus, the value ranges of all the distance variables are $(0, d)$, where $d = \sqrt{l^2 + w^2}$. The reference frame is set in this way: the X axis points to the North, and the Y axis points to the East. Besides, the system neglects the rotation and acceleration of the earth, and the earth is assumed to be flat.

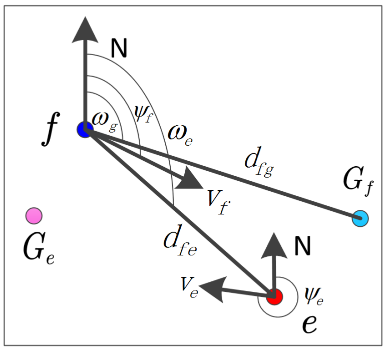

Figure 3. The denotations for the state variables.

4.1.2. Action Space

At each decision making step, the agent needs to decide an action for execution, so as to change the altitude of the decision making UAV. Six actions are defined in this paper: flying toward the goal, accelerating, decelerating, increasing the heading angle, decreasing the heading angle, and attacking the enemy. Denote these actions with $A = \{a_0, a_1, ..., a_5\}$. Each action corresponds to a way to change the desired attitude. The details are listed as follows.

- a_0: Fly toward the goal straightly, while keeping the desired velocity as the same in the previous step.
- a_1: Increase the desired velocity with ΔV and fly with the same desired heading angle as the previous step.
- a_2: Decrease the desired velocity with ΔV and fly with the same desired heading angle as the previous step.
- a_3: Increase the desired heading angle with $\Delta\phi$ and fly with the same desired velocity as the previous step.
- a_4: Decrease the desired heading angle with $\Delta\phi$ and fly with the same desired velocity as the previous step.
- a_5: Attack the enemy UAV, while the desired attitude changes as that of a_0.

The desired height is set with a fixed value h in all cases, for the flying height is not considered in this problem.

4.1.3. Reward Function

There are two types of rewards in this enemy avoidance problem, denoted by r_1 and r_2, respectively. The first type of reward r_1 is the usual reward given at each decision making step, which aims to encourage the agent to reach the goal point in as few steps as possible. The other reward r_2 is the reward given at the end of the episode for different ending reasons. There are four situations that will end an episode. Denote these four situations by $S_T = \{s_{T1}, s_{T2}, s_{T3}, s_{T4}\}$, which are defined as follows.

- s_{T1}: The decision making UAV has collided with the enemy UAV.
- s_{T2}: The decision making UAV has reached the goal point.
- s_{T3}: The decision making UAV has attacked the enemy UAV successfully.
- s_{T4}: The decision making UAV has been out of the region.

Since the mission of the decision making UAV is to reach the goal point successfully, the agent will be rewarded with a very big value when the mission is fulfilled. Cases that the decision making

UAV needs to avoid, such as colliding with the enemy UAV or out of the region, should be punished. Successfully attacking the enemy UAV can guarantee the fulfillment of the mission for the decision making UAV, therefore it should be rewarded. However, since the attacking action is costly and may fail, it will be better to limit the use of the attacking action. Considering this, a small punishment is given when attacking action is used. Therefore, we define the reward $r = r_1 + r_2$, where $r_1 = -1$, and r_2 is defined as below:

$$r_2 = \begin{cases} -20 & a = a_5; \\ -100 & s = s_{T1}; \\ -100 & s = s_{T4}; \\ 2 & s = s_{T3}; \\ 500 & s = s_{T2}. \end{cases} \tag{5}$$

4.2. Learning Policy with the Sarsa Method

In the first part of the new framework, the enemy avoidance problem has been formulated as an MDP. For the second part of the new framework, the agent of the decision making UAV is allowed to learn the policy based on the Sarsa reinforcement learning method.

There are mainly three classes of methods for solving the reinforcement learning problem [13]: dynamic programming, Monte Carlo methods, and temporal-difference (TD) learning. Dynamic programming methods require a complete and accurate model of the environment (the transition function $\mathcal{T}$), while the Monte Carlo methods are not suitable for step-by-step incremental computation. Only the temporal-difference methods require no model and are fully incremental. In the enemy avoidance problem, the environment transition function $\mathcal{T}$ is unknown, and the policies need to be learned by continuous interaction with the environment. Therefore, the temporal-difference methods are more suitable for the enemy avoidance problem. As one of the temporal-difference reinforcement learning methods, the Sarsa method is chosen to be used in the proposed framework to learn the policy for the enemy avoidance problem.

Since the state space in the enemy avoidance problem is continuous, it is impractical to visit each state with each action infinitely often. Therefore, certain function approximations are needed to generalize the state space from relatively sparse interaction samples and with fewer variables than there are states. In this paper, the CMAC (cerebellar model arithmetic computer, [19]) technique is also embodied in the proposed framework to approximate the Q-value function when learning with the Sarsa method.

The details of how the proposed framework embodies the Sarsa method and the CMAC function approximator to learn the policy for the enemy avoidance problem are elaborated as follows.

4.2.1. Sarsa Method

The Sarsa method is an on-policy temporal-difference learning method, where the agent attempts to update the policy that is used to make decisions for the decision making UAV f at the same time. Different from most reinforcement learning methods where the main goal is to estimate the optimal value function, the Sarsa agent learns an action-value function $Q(s,a)$ rather than a state-value function $V(s)$. For the enemy avoidance problem, the Sarsa agent updates its action-value function $Q(s,a)$ after every transition from a state $s \in S$, where s is not an episode ending situation. That is to say, $s \notin S_T$. If the state $s' \in S_T$, then $Q(s',a') = 0$.

The updating rule is given in Equation (6), where γ is the discount rate, and α is a step-size parameter. Every element of the quintuple of the enemy avoidance events, (s,a,r,s',a'), are used in this updating rule. Such a quintuple makes up a transition from one state-action pair of the enemy avoidance problem to the next, and therefore gives rise to the name Sarsa for the algorithm.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s', a' - Q(s,a)) \tag{6}$$

In the proposed framework, the Sarsa algorithm [13] is adapted into the enemy avoidance problem for the agent to learn the policy, which is presented in Algorithm 1.

Algorithm 1 Sarsa Algorithm for the Enemy Avoidance Problem

```
1:  Initialize $Q(s,a)$ arbitrarily
2:  for each episode do
3:      Initialize $s$ as all the UAVs having entered the problem region
4:      Choose $a \in A$ for $s$ using policy derived from $Q$ with $\epsilon-$greedy
5:      for each step of episode do
6:          Take action $a$, observe $r$, $s'$
7:          Choose $a' \in A$ for $s'$ using policy derived from $Q$ with $\epsilon-$greedy
8:          $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$
9:          $s \leftarrow s'; a \leftarrow a'$
10:     end for
11:     until $s \in S_T$
12: end for
```

4.2.2. CMAC Function Approximation

In the proposed framework for the enemy avoidance problem, the state space is continuous. Therefore, the learning agent for the decision making UAV needs to use function approximation to generalize from limited experienced states. With function approximation, the action-value function $Q(s,a)$ of the enemy avoidance problem is maintained in a parameterized functional form with parameter vector $\overrightarrow{\theta}$. In this framework, the linear function form is used, as presented by Equation (7), where $\overrightarrow{\phi}_{(s,a)}$ is the feature vector of the function approximation.

$$Q(s,a) = \overrightarrow{\theta}^T \overrightarrow{\phi}_{(s,a)} \tag{7}$$

The CMAC (cerebellar model arithmetic computer, [19]) is one of those linear function approximators, and thus is used to construct the feature vector $\overrightarrow{\phi}_{(s,a)}$ in the proposed framework. To update the parameter vector $\overrightarrow{\theta}$, the gradient-descent method is adopted in the proposed framework as well.

The CMAC discretizes the continuous state space of the enemy avoidance problem by laying infinite axis-parallel tilings over all the eight state variables and then generalizes them via multiple overlapping tilings with some offset. Each element of a tiling is called a tile, which is a binary feature, as shown by Equation (8).

$$\phi_{(s,a)}(i) = \begin{cases} 1 & \text{tile } i \text{ is activated.} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Therefore, the CMAC maintains $Q(s,a)$ of the enemy avoidance problem in the following form:

$$Q(s,a) = \sum_{i=1}^{n} \theta(i)\phi_{(s,a)}(i) = \sum_{i \in I(\overrightarrow{\phi}_{(s,a)})} \theta(i) \tag{9}$$

where $I(\overrightarrow{\phi}_{(s,a)})$ is the set of tiles that are activated by the pair (s,a) in the enemy avoidance problem, whose tile values are 1.

The parameter vector $\overrightarrow{\theta}$ is adjusted by the gradient-descent method, whose updating rule is as follows:

$$\overrightarrow{\theta}_{t+1} = \overrightarrow{\theta}_t + \alpha\delta_t \tag{10}$$

where δ_t is the usual TD error,

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{11}$$

5. Simulation Environment

Since it is hard to build an accurate environment model for the enemy avoidance problem, and the RL agent needs to approximate the environment model through continuous interaction with the environment, this paper builds a hardware-in-the-loop (HITL) simulation system for the enemy avoidance problem. In this HITL simulation system, the kinecmatic and dynamic of UAVs are modeled by the X-plane simulators, while the maneuver and control properties of the system are confined by the hardware controller called Pixhawk. With such a HITL simulation system, the simulation can be more consistent to real flying, and can reduce the energy of building a complex environment model and save the cost of a real flying test.

In this section, how the HITL simulation system is constructed will be detailed. After this, the simulation process for an episode will be given.

5.1. System Construction

The X-plane flight simulator is used in this paper to simulate the flying dynamics of both the decision making UAV and the enemy UAV, and each X-plane is controled by a Pixhawk (PX4) flight controller. The Pixhawk is a hardware which is also used in controlling the real UAVs. In the simulation system, there is a ground station which can broadcast the information of all the UAVs to every UAV. The PX4 can control the flying of the UAVs according to the desired attitude. The desired attitude is composed of the desired velocity, the desired heading angle, and the desired height. In each UAV, there is an on-board sensor system which is used to sense the position and the attitude information of the UAV. Besides, each UAV has an agent for the communication and the decision control. To be specific, the agent for the decision making UAV has three modules: communication module, decision making module, and translator module. The on-board sensor system sends the sensed information to the agent through the communication module, while the communication module also sends the received information to the ground station and receives the enemy UAV's information from the ground station. Based on all the information received by the communication module, the decision making module then decides which action the decision making UAV should take, while the translator module interprets the action into desired attitude and sends it to the PX4 for the UAV flying control. Since the enemy UAV in this paper is assumed to fly towards the goal point G_e directly all along the process, there is no decision making module in the enemy agent. Therefore, the enemy agent is composed of two modules: communication and control module. The communication module of the enemy agent has the same function as that of the agent for the decision making UAV. In the enemy agent, the control module sends the desired attitude to the PX4, where the desired velocity and the desired height are fixed at the initialization of the simulation, and the desired heading angle is calculated based on the relationship between the goal point and the position of the enemy UAV.

The structure of the simulation system is illustrated by Figure 4.

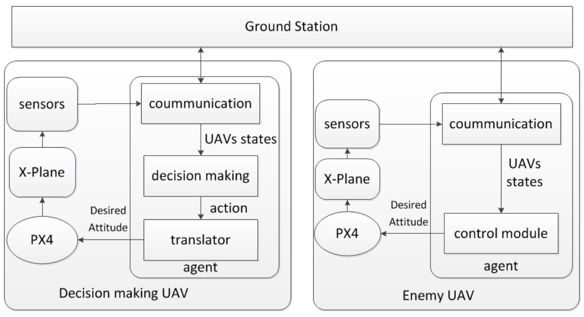

Figure 4. The structure of the simulation system.

Since another purpose of this paper is to explore the collision avoidance solution for the fixed-wing UAV, the X-plane simulator is set to use the fixed-wing simulation model. The simulation environment is shown as in Figure 5.

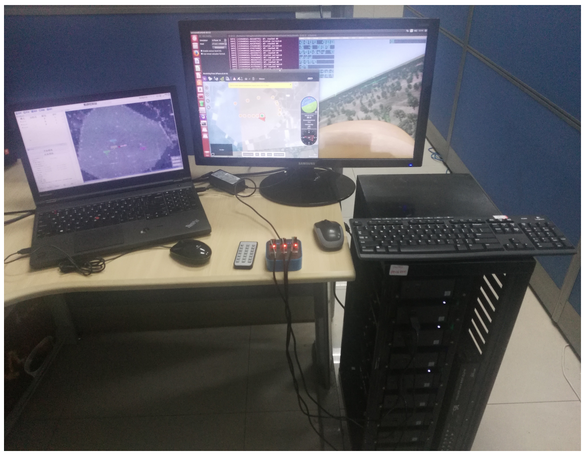

Figure 5. The simulation environment.

Beside receiving and sending the flying information of the UAVs, the ground control station also needs to send commands to all the UAVs. With these commands, the simulation process can be well controlled by the ground station.

5.2. Simulation Process

In order to collect as many samples as possible for the policy learning, the simulation should be carried for many episodes. Each episode is run with the same process, as shown by Figure 6. There are five steps for an episode, listed as follows.

(i) Both PX4s are set on the mission mode, so as to control the two UAVs to loiter around their own loiter points outside of the region. The loiter points are send to the PX4 from the ground station before the simulation begins.

(ii) The ground station sends a command asking all the UAVs to fly to their goal points. Both PX4s are set to the offboard mode, so that the UAV agents can guide their own UAVs to fly towards their goal points. In this stage, the agents use the simple-fly pattern to guide the flying of the UAVs. In the simple-fly pattern, the desired heading angle of the UAV is the angle toward the goal point directly, while the desired velocity and the desired height stay unchanged.

(iii) When both UAVs have entered the region for the enemy avoidance problem, the ground station sends a command to inform both UAV agents that the new episode begins. Upon receiving this command, the decision making agent changes into its decision making pattern from the simple-fly pattern, during which the agent can either learn the policy with reinforcement learning or make decisions with certain policy. On the other hand, the enemy agent still guide the enemy UAV fly straightly toward its goal point G_e with simple-fly pattern during this stage.

(iv) When the simulation meets one of those ending conditions, the ground station sends a command to both UAVs to inform the ending of the episode. Both PX4s are set back to the mission mode after receiving this command from the ground station.

(v) Under the mission mode, both PX4s control their own UAVs fly back to their loiter points again, since the loiter points remain as their next point in the mission mode. When both UAVs have returned to their loiter points, the ground station sends the command to make both UAVs fly into the region again for the start of the next episode.

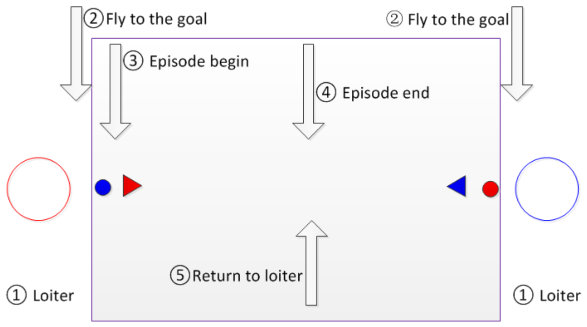

Figure 6. The simulation process.

6. Implementation and Results

Based on the HITL simulation environment built in Section 5, the decision policy for the enemy avoidance problem will be learned with the new framework constructed in Section 4. Furthermore, a fixed-rule policy and a random policy are designed to compare with the policy learned by the Sarsa based framework.

The experiment settings are as follows. The region for the enemy avoidance problem has a length of $l = 600$ m and a width of $w = 450$ m. The outside loiter center of each UAV is 100 meters away from the nearest region edge, and the loiter radium is 100 m. The goal point G_f is inside the region, which is 150 m away from the right edge of the region. The decision making UAV loiters around the goal point G_f with a loiter radium of 50 m when it has arrived the goal point G_f, and the goal radius is $r_g = 100$ m. The collision radius is $r_c = 40$ m. Set $\Delta V = 1.0$ m/s and $\Delta\phi = 5.0$ degree.

6.1. Learning with the Sarsa Based Framework

First, we use the newly constructed framework based on the Sarsa method and the CMAC function approximator to learn the policy for the enemy avoidance problem in the HITL simulation environment.

As in reinforcement learning, the goal is to maximize the expected accumulate rewards, thus the action-value function Q is an effective metric to measure the performance of the policy learning.

Considering the mission of the decision making UAV, it has to reach the goal point G_f successfully as soon as possible. On the other hand, the number of steps that an episode lasts can partially indicates how long it takes the decision making UAV to reach the goal point G_f. Therefore, the number of steps for an episode can be used as a metric to measure the performance to some extend.

However, there are several ending reasons. It can take very few steps to end the episode if the decision making UAV collides with or attacks the enemy UAV at a very early stage of an episode. Therefore, the number of steps alone is not enough to measure the performance of the learning in the enemy avoidance problem. To this end, the numbers of episodes that ending for different reasons are calculated, so as to measure more accurately how the learning changes the episode ending reasons.

The experiment for learning the policy with the proposed framework has been run for 14,362 episodes. The related parameter settings are: $\epsilon = 0.01$, $\alpha = 0.1$, $\gamma = 0$, and the number of tiles laid for each variable is $n = 32$. The results of the above three metrics are presented as follows.

Figure 7 illustrates how the Q -value changes with the increase of the episodes during the policy learning process. The figure is drawn with data filtered by a window of 1000 episodes. From Figure 7, it can be seen that the Q-value increases as the number of episodes increases, and becomes stable after around 10,000 episodes.

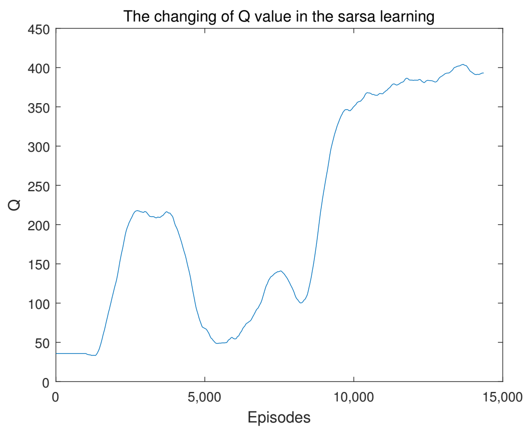

Figure 7. The *Q*-value for each episode in the policy learning.

Figure 8 shows how the number of steps for each episode changes as the number of policy learning episodes increases. The figure is also drawn with data filtered by a window of 1000 episodes. In Figure 8, the number of steps fluctuates very much at the beginning, but converges to a relatively small number as the number of episodes increases, and it comes to a plateau after around 10,000 episodes.

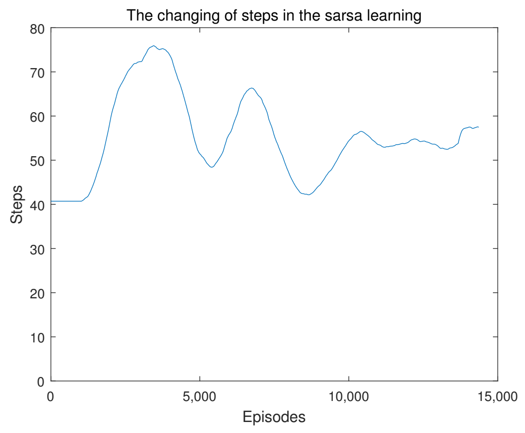

Figure 8. The number of steps for each episode in the policy learning.

Figure 9 presents how the accumulated numbers of different ending reasons change during the policy learning process. From Figure 9, it can be see that the episodes are mostly ended for collision with the enemy UAV at the initial period of the learning process. After about 1500 episodes, the number of episodes ending for reaching the goal begins to increase rapidly, which surpasses those of other ending reasons very soon. The number of episodes ending for successfully attacking the enemy UAV has a rise from about the 4000th episode to about the 8000th episode, and surpasses the number of episodes ending for collision with the enemy UAV during this period. Except for ending for reaching the goal point, the numbers of other ending reasons all stop to increase after about 9000 episodes. These results mean that the learning agent gradually learns to avoid the enemy UAV and attack the enemy UAV, and then it learns to limit the usage of the attacking action, and finally it has learned a very good policy to reach the goal point.

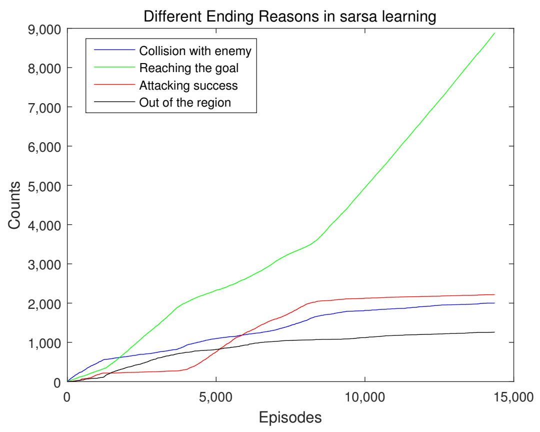

Figure 9. The numbers of different ending reasons in the policy learning.

From the above results for three metrics, it can be seen that the policy learning process under the proposed framework converges after about 10,000 episodes. Such episode number for converging is a reflection of the sample complexity of this learning based method. The proposed framework is to reduce the computation cost on the price of the sample complexity. However, such policy learning can be done before the real-time decision making, while decision making depending on some computation methods need to bear those computation costs at each real-time decision making step. Therefore, the proposed framework still has its advantages on these points.

6.2. Designing Comparison Policies

In order to show the effectiveness of the policy learned by the Sarsa based framework, we design a set of rules to guide the flying of the decision making UAV based on human experience, which is called the fixed-rule policy. As a more baseline comparison, a random policy is also used for the comparison. The detail of these two comparison policies are introduced in the following.

6.2.1. Fixed-Rule Policy

The fixed-rule policy is designed with human experience. It composes of several if-then rules. At each decision making step, the agent examines the current state and decides which rule can be used. The detail of the designed rules are listed as in Algorithm 2.

The designing of these rules aims at guiding the UAV flying toward the goal as straightforward as possible and trying to avoid the enemy UAV at the same time. However, these if-then rules are very simple ones due to the limitation of the human knowledge. Therefore, the hypothesis here is that the Sarsa based framework can help the agent discover more latent rules to guide the flying of the decision making UAV.

Algorithm 2 Fixed-rule Policy

```
1: if the distance to the nearest enemy<100 m then
2:     if decision steps from the last attacking action>10 decision steps then
3:         execute the attacking action
4:     else
5:         fly to the goal
6:     end if
7: else
8:     if the enemy UAV in the flying direction (within 2.5 degree) then
9:         if the enemy is on the right side then
10:            increase the flying angle
11:        else
12:            decrease the flying angle
13:        end if
14:    else
15:        fly to the goal
16:    end if
17: end if
```

6.2.2. Random Policy

The purpose of designing this random policy is to set a fundamental baseline policy for the effectiveness comparison of all other policies. The hypothesis is that those effective policies should all have better performance than this random policy.

In this random policy, the agent chooses action at each decision making step randomly. The probabilities of choosing each action are equal, so that no special favor is given to any action.

6.3. Policy Comparison Results

To test the effectiveness of the policy learned with the Sarsa based framework, another set of experiments are carried on the same HITL simulation platform. For comparison, the fixed-rule policy and the random policy are also tested with the same experimental settings. Each policy is run for 1200 episodes, and the results are summarized as follows.

Firstly, the numbers of steps taken in each episode for using different policies are compared in Figure 10. The figure is drawn with data filtered with a window of 100 episodes. It can be seen that the number of steps are stable in the testing process, no matter which policy is used. However, it takes different numbers of steps to end an episode when different policies are used. Overall, the random policy takes the least number of steps, while the policy learned with the Sarsa based framework takes the most. Though more number of steps means more cost, it is also a reflection of higher possibility to reach the goal point and better policy to avoid the enemy UAV. Such understanding is from the following two aspects. First, the episode ending for reaching the goal point takes more steps than the episode ending before reaching the goal point for other reasons, because of the longer distance. On the other hand, it needs more steps to fly away to avoid the collision with the enemy UAV than to fly straightly towards the goal. To more accurately show the performance of different policies, more results are presented below.

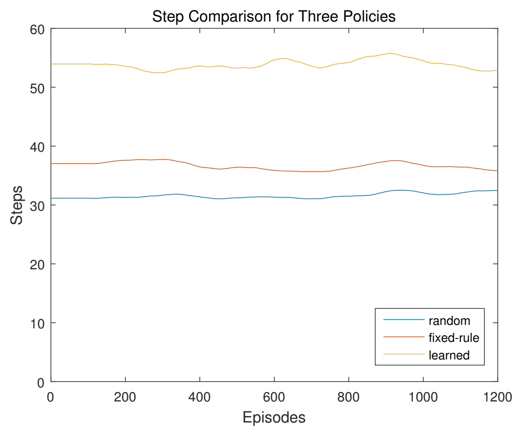

Figure 10. The steps of three policies.

6.3.1. Results of the Random Policy

Figure 11 illustrates how the numbers of episodes ending for different reasons change as the number of the total episode increases under the random policy. From Figure 11, it can be seen that there are mainly three ending reasons for these episodes, which are collision with the enemy UAV, reaching the goal point successfully, and attacking the enemy UAV successfully. Besides, there is nearly no case for flying out of the region. The number of the episodes ending for reaching the goal point increases slowest among the three main ending reasons, while the episode number for successfully attacking of the enemy UAV increases the quickest. This indicates that it is easier to end the episode by attacking the enemy UAV. The reason for this should be that ending the episode by successfully attacking takes only an attacking action as long as the attacking action is executed at a relatively near distance from the enemy UAV. On the contrast, to reach the goal point successfully is much harder, for it needs to avoid the moving enemy UAV by taking many decision steps in an appropriate sequence.

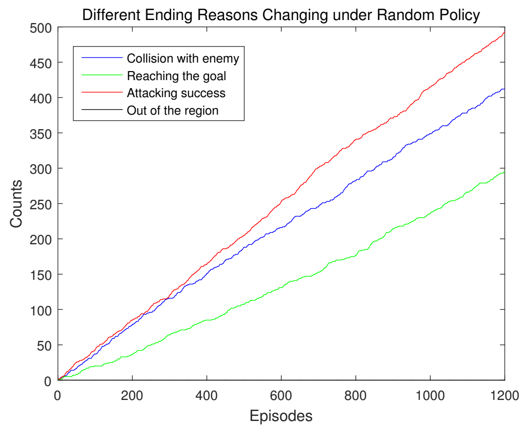

Figure 11. The numbers of different ending reasons when using the random policy.

Figure 12 presents the total episode numbers of different ending reasons when the random policy is used. It can be seen more clearly that the number of episodes ending for reaching the goal point is smaller than those for the other two ending reasons (collision with the enemy UAV and attacking the enemy UAV successfully). Therefore, the random policy is unable to guide the decision making UAV to fulfill its mission very successful, and the results of this policy only reflect different challenges for achieving different ending reasons.

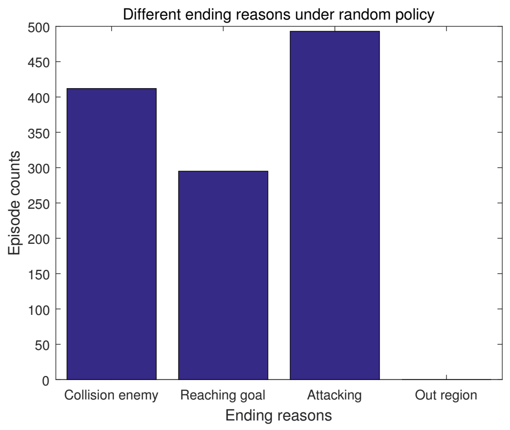

Figure 12. The total numbers of different ending reasons when using the random policy.

6.3.2. Results of the Fixed-Rule Policy

When the fixed-rule policy is used, the numbers of episodes ending with different reasons increase during the testing process, as shown in Figure 13. Still, there are three main ending reasons, which are collision with the enemy UAV, reaching the goal point successfully, and attacking the enemy UAV successfully. The number of episodes ending for attacking the enemy UAV is less than that of collision with the enemy UAV, while the number of the episodes ending for reaching the goal point successfully is nearly equal to that of collision with the enemy UAV. Compared with the random policy, these results show that the fixed-rule policy can reduce the unnecessary use of the attacking action, and can increase the use of some effective enemy avoidance actions so as to increase the probability of reaching the goal point successfully.

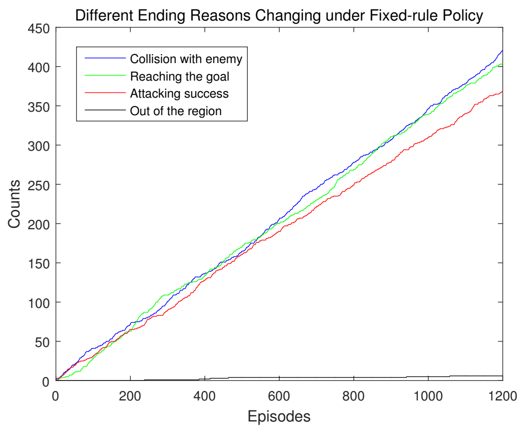

Figure 13. The numbers of different ending reasons when using the fixed-rule policy.

Figure 14 gives the total number of episodes for each ending reason in the fixed-rule policy experiment. Compared with that of the random policy, the fixed-rule policy can obviously reduce the number of episodes ending for attacking the enemy UAV, but is not effective in reducing the number of episodes for colliding with the enemy UAV.

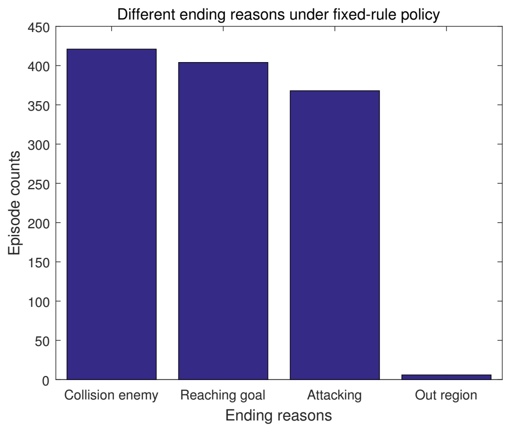

Figure 14. The total numbers of different ending reasons when using the fixed-rule policy.

6.3.3. Results of the Sarsa Learned Policy

Finally, the policy learned with the Sarsa based framework is also tested in the platform. In this experiment, the agent only takes action at each decision making step by following the learned policy, without any further learning.

Figure 15 presents how the numbers of episodes ending for different reasons change when the policy learned with the Sarsa based framework is adopted. Different from the other two policies, there is only one main ending reason: reaching the goal point. The numbers of episodes ending for both collision with the enemy UAV and attacking the enemy UAV are sharply reduced in the learned policy, and can be neglected when compared with that of the reaching goal case. Obviously, the policy learned with the Sarsa based framework can successfully guide the decision making UAV reaching the goal point.

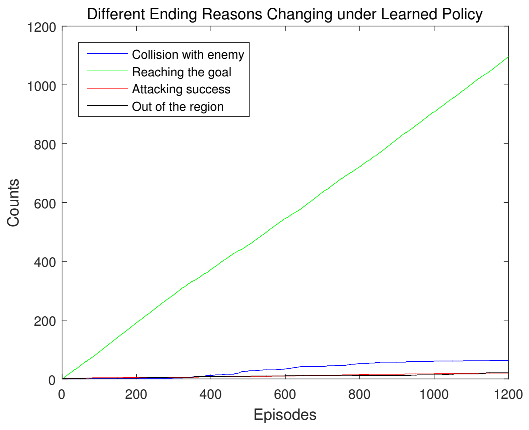

Figure 15. The numbers of different ending reasons when using the policy learned with the Sarsa based framework.

Figure 16 gives the total numbers of episodes ending for different reasons when the policy learned with the Sarsa based framework is used. It can be seen that the number of episodes for collision with the enemy UAV and the number of episodes for attacking the enemy UAV are both reduced to quite small numbers, which indicates the effectiveness of the policy learned with the Sarsa based framework.

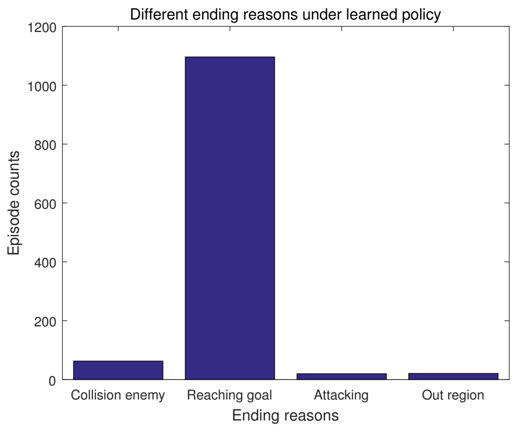

Figure 16. The total numbers of different ending reasons when using the learned policy.

7. Conclusions and Future Work

In this paper, we have defined the enemy avoidance problem, which is quite different from most collision avoidance problems. To solve the enemy avoidance problem, we have proposed a new framework, which formulates the enemy avoidance problem as the Markov Decision Process, and learns policy for the decision making UAV based on the Sarsa reinforcement learning method, and generalizes the continuous state space with the CMAC function approximation technique. Furthermore, we have constructed a HITL simulation system to learn the policy with the proposed framework and to verify the effectiveness of different policies for the enemy avoidance problem. The carried experiments show that the HITL simulation system is valid in presenting the enemy avoidance problem, and also suitable for the RL based framework. The Sarsa based framework can learn a satisfying policy for the enemy avoidance problem. Further experiments also show that the policy learned with the Sarsa based framework outperforms both the random policy and the fixed-rule policy in solving the enemy avoidance problem.

There are several advantages in our proposed method. Firstly, it does not require a mathematically built environment model to be computed when making decisions. Secondly, it learns the policy off-line, and the learned policy is a simple mapping from the state to the action. Therefore, little computation is needed for the on-line decision making. Thirdly, the policy is learned in the continuous space, and has good generalization ability. However, there are still some weaknesses in our proposed method. Firstly, samples are needed for the learning, which could also be costly. Secondly, there are a lot of simplifications in the researched problem, so inconsistency may occur if the method is applied directly into some more practical real problems.

Therefore, some future works can be explored. Firstly, a more practical problem setting could be studied. For example, equip the enemy UAV with more practical properties like changeable desired attitude and the attacking ability, or extend the two dimensional problem into three dimensions, or add noise like wind disturbance to the environment. Secondly, more efficient reinforcement learning methods and even some other techniques could be applied, so as to shorten the learning process. Thirdly, ways to combine the learning method with human experience could be explored, so as to make use of human experience to facilitate the learning and to discover latent knowledge by agent learning. Fourthly, methods which consider the collision with both obstacles and other UAVs would be more useful in improving the autonomy of the UAVs, and therefore could be studied in the future. Lastly, extending the problem to scenarios with more UAVs can be a very worth research area.

Author Contributions: Conceptualization, Q.C., X.W. and J.Y.; Funding acquisition, X.W. and L.S.; Investigation, Q.C. and J.Y.; Methodology, Q.C., X.W. and J.Y.; Project administration, L.S.; Supervision, X.W. and L.S.; Validation, Q.C.; Visualization, Q.C. and X.W.; Writing—original draft, Q.C.; Writing—review & editing, Q.C., X.W., J.Y. and L.S.

Funding: This work is supported by the National Natural Science Foundation of China under Grants 61403406.

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

References

1. Yu, X.; Zhang, Y. Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects. *Prog. Aerosp. Sci.* **2015**, *74*, 152–166. [CrossRef]
2. Motlagh, N.H.; Bagaa, M.; Taleb, T. UAV-based IoT platform: A crowd surveillance use case. *IEEE Commun. Mag.* **2017**, *55*, 128–134. [CrossRef]
3. Gu, J.; Su, T.; Wang, Q.; Du, X.; Guizani, M. Multiple moving targets surveillance based on a cooperative network for multi-UAV. *IEEE Commun. Mag.* **2018**, *56*, 82–89. [CrossRef]
4. Liu, Y.; Wang, Q.; Hu, H.; He, Y. A Novel Real-Time Moving Target Tracking and Path Planning System for a Quadrotor UAV in Unknown Unstructured Outdoor Scenes. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**. [CrossRef]
5. Yadav, I.; Eckenhoff, K.; Huang, G.; Tanner, H.G. Visual-Inertial Target Tracking and Motion Planning for UAV-based Radiation Detection. *arXiv* **2018**, arXiv:1805.09061.
6. Vanegas, F.; Campbell, D.; Roy, N.; Gaston, K.J.; Gonzalez, F. UAV tracking and following a ground target under motion and localisation uncertainty. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2017; pp. 1–10.
7. Li, S.; Liu, T.; Zhang, C.; Yeung, D.Y.; Shen, S. Learning Unmanned Aerial Vehicle Control for Autonomous Target Following. *arXiv* **2017**, arXiv:1709.08233.
8. Mahjri, I.; Dhraief, A.; Belghith, A. A review on collision avoidance systems for unmanned aerial vehicles. In *International Workshop on Communication Technologies for Vehicles*; Springer: Berlin, Germany, 2015; pp. 203–214.
9. Gottlieb, Y.; Shima, T. UAVs task and motion planning in the presence of obstacles and prioritized targets. *Sensors* **2015**, *15*, 29734–29764. [CrossRef]
10. Ramasamy, S.; Sabatini, R.; Gardi, A.; Liu, J. LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. *Aerosp. Sci. Technol.* **2016**, *55*, 344–358. [CrossRef]
11. Liu, Z.; Yu, X.; Yuan, C.; Zhang, Y. Leader-follower formation control of unmanned aerial vehicles with fault tolerant and collision avoidance capabilities. In Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, 9–12 June 2015; pp. 1025–1030.
12. Jenie, Y.I.; van Kampen, E.J.; Ellerbroek, J.; Hoekstra, J.M. Taxonomy of conflict detection and resolution approaches for unmanned aerial vehicle in an integrated airspace. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 558–567. [CrossRef]
13. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, UK, 1998.
14. Radmanesh, M.; Kumar, M.; Nemati, A.; Sarim, M. Dynamic optimal UAV trajectory planning in the national airspace system via mixed integer linear programming. *Proc. Inst. Mech. Eng. G J. Aerosp. Eng.* **2016**, *230*, 1668–1682. [CrossRef]
15. Ong, H.Y.; Kochenderfer, M.J. Short-term conflict resolution for unmanned aircraft traffic management. In Proceedings of the IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), Prague, Czech Republic, 13–17 September 2015.
16. Alonso-Mora, J.; Naegeli, T.; Siegwart, R.; Beardsley, P. Collision avoidance for aerial vehicles in multi-agent scenarios. *Auton. Robot.* **2015**, *39*, 101–121. [CrossRef]
17. Dentler, J.; Kannan, S.; Mendez, M.A.O.; Voos, H. A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In Proceedings of the IEEE Conference on Control Applications (CCA), Buenos Aires, Argentina, 19–22 September 2016; pp. 519–525.
18. Alvarez, H.; Paz, L.M.; Sturm, J.; Cremers, D. Collision avoidance for quadrotors with a monocular camera. In *Experimental Robotics*; Springer: Cham, Switzerland, 2016; pp. 195–209.
19. Albus, J.S. *Brains, Behaviour, and Robotics*; Byte Books: Perterborough, NH, USA, 1981.
20. Zhou, Y.; Baras, J.S. Reachable set approach to collision avoidance for UAVs. In Proceedings of the IEEE 54th Annual Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 5947–5952.
21. D'Amato, E.; Mattei, M.; Notaro, I. Bi-level Flight Path Planning of UAV Formations with Collision Avoidance. *J. Intell. Robot. Syst.* **2018**, *93*, 193–211. [CrossRef]

22. Qiu, H.; Duan, H. A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles. *Inf. Sci.* **2018**, in press. [CrossRef]
23. Yang, J.; Yin, D.; Cheng, Q.; Shen, L. Two-Layered Mechanism of Online Unmanned Aerial Vehicles Conflict Detection and Resolution. *IEEE Trans. Intell. Transp. Syst.* **2017**. [CrossRef]
24. Yang, J.; Yin, D.; Shen, L.; Cheng, Q.; Xie, X. Cooperative Deconflicting Heading Maneuvers Applied to Unmanned Aerial Vehicles in Non-Segregated Airspace. *J. Intell. Robot. Syst.* **2018**, *92*, 187–201. [CrossRef]
25. Alomari, A.; Phillips, W.; Aslam, N.; Comeau, F. Swarm Intelligence Optimization Techniques for Obstacle-Avoidance Mobility-Assisted Localization in Wireless Sensor Networks. *IEEE Access* **2017**, *6*, 22368–22385. [CrossRef]
26. Masiero, A.; Fissore, F.; Guarnieri, A.; Pirotti, F.; Vettore, A. UAV positioning and collision avoidance based on RSS measurements. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *40*, 219. [CrossRef]
27. Kahn, G.; Villaflor, A.; Pong, V.; Abbeel, P.; Levine, S. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv* **2017**, arXiv:1702.01182.
28. Pfeiffer, M.; Schaeuble, M.; Nieto, J.; Siegwart, R.; Cadena, C. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1527–1533.
29. Gandhi, D.; Pinto, L.; Gupta, A. Learning to fly by crashing. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3948–3955.
30. Zhang, T.; Kahn, G.; Levine, S.; Abbeel, P. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 528–535.
31. Zhang, B.; Mao, Z.; Liu, W.; Liu, J. Geometric reinforcement learning for path planning of UAVs. *J. Intell. Robot. Syst.* **2015**, *77*, 391–409. [CrossRef]
32. Bai, H.; Hsu, D.; Kochenderfer, M.J.; Lee, W.S. Unmanned aircraft collision avoidance using continuous-state POMDPs. *Robot. Sci. Syst. VII* **2012**, *1*, 1–8.
33. Temizer, S.; Kochenderfer, M.; Kaelbling, L.; Lozano-Pérez, T.; Kuchar, J. Collision avoidance for unmanned aircraft using Markov decision processes. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Toronto, ON, Canada, 2–5 August 2010; p. 8040.
34. Seo, J.; Kim, Y.; Kim, S.; Tsourdos, A. Collision avoidance strategies for unmanned aerial vehicles in formation flight. *IEEE Trans. Aerosp. Electron. Syst.* **2017**, *53*, 2718–2734. [CrossRef]
35. Yang, J.; Yin, D.; Cheng, Q.; Shen, L.; Tan, Z. Decentralized cooperative unmanned aerial vehicles conflict resolution by neural network-based tree search method. *Int. J. Adv. Robot. Syst.* **2016**, *13*. [CrossRef]