Research on Autonomous Maneuvering Decision of UCAV Based on Deep Reinforcement Learning

Yesheng Zhang[1], Wei Zu[2], Yang Gao[3], Hongxing Chang[4]

1. Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
University of Chinese Academy of Sciences, Beijing 101408, China
E-mail: zhangyesheng2015@ia.ac.cn

2. Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
E-mail: wei.zu@ia.ac.cn

3. Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
E-mail: yang.gao@ia.ac.cn

4. Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
E-mail: hongxing.chang@ia.ac.cn

Abstract: In order to improve the intelligent level of UCAV in one-to-one air combat, an autonomous maneuvering decision algorithm based on deep reinforcement learning is proposed. UCAV learns strategies by sensing the environment, performing maneuvering actions, and getting feedback. In this way, we can avoid the limitations of existing theories and human operations. Firstly an environment is modeled to simulate the real-time situation of air combat. Then a situation assessment method based on Energy-Maneuverability theory is utilized to design the reward functions. Finally model based on deep reinforcement learning is created for UCAV to learn strategies to gain the advantage for the opponent.

Key Words: Air Combat, Autonomous Maneuvering Decision, Deep Reinforcement Learning

1 INTRODUCTION

Unmanned combat aerial vehicle(UCAV) is a kind of unmanned aerial vehicle(UAV) that usually carries kinds of weapons and performs combat missions. Normal UAV has a lot of good characteristics such as high mobility, great flying height, excellent stealth ability so that it can carry out battlefield investigation, monitoring tasks with less casualties and minimal cost. With the rapid development of control engineering technology, electronic technology and aviation technology, UAV has got better endurance, improved performance and more powerful combat capability. UCAV paricipates in more and more combat missions and becomes an important force in air combat.

A lot of autonomous maneuvering decision algorithms have been proposed and can be divided into two categories. One category includes traditional algorithms such as differential game algorithm[1–3], rule-based algorithm[4, 5] and matrix counterrmeasure algorithm[6], while another category is intelligent algorithms including neural network[7, 8], genetic algorithm[9], particle swarm optimization algorithm[10] and reinforcement learning[11]. The traditional algorithms consider the air combat problem as a particular mathematical model and can not entirety simulate the air combat environment. The intelligent algorithms either need lots of training data and time, or cant meet real-time requirements.

In this paper, we create an air combat simulation environment and train an end-to-end neural network based on deep reinforcement learning to realize real-time autonomous maneuvering decision.

2 DEEP REINFORCEMENT LEARNING

2.1 Q Learning

As reinforcement learning methods have been successfully used to solve many difficult tasks, and work very well, we consider to apply deep reinforcement learning in air combat field. Different from normal reinforcement learning, deep reinforcement learning use deep neural networks to approximate the value function, policy, and model in normal reinforcement learning. Deep neural network can approximate non-linear function very well to achieve reasonable model, which is the autonomous maneuvering decision we need.

There are some symbols will be used in this paper, such as follows:

1. State s: At any time, State s is the representation of the environment in which the agent is located, for example, the entire game screen, or information that has been abstracted as a location, direction, obstacle location, and so on. In one-to-one air combat, State s is combined by the state of red UCAV s_r and the state of blue UCAV s_b, while the red UCAV represents our side and the blue UCAV is the opponent.

2. Action a: In each state, the agent can take Action a_i from Action set A. For example, accelerating, slowing down, turning left or right, etc. After taking an action, agent will go forward to the next state. To the UCAV, the transition of states follows flight dynamics equation according to the chosen action a_i.

3. Reward r: Every time reaching a new state, the agent is likely to receive a feedback called reward.For example, being attacked will receive a negative reward, and reaching advantageous situation will receive a positive reward.
4. Policy P: P is the strategy about how to choose the action. We hope to learn a strategy that can make the agent get the maximum cumulative reward. In deep reinforcement learning, P is represented by an end-to-end neural network.
A reinforcement learning mission used to be defined as a Markov Decision Process (MDP), which repeatedly transforms between state, action, reward, state... until the end of the mission.[12] So we get the trace

$$< s_0, a_0, r_1, s_1, a_1, r_2 ... s_n, a_n, r_{n+1} >.$$

The goal is to learn a good strategy that maximize the discounted future reward R. At the state s_0, total reward R_0 can be caculated as formula (1).

$$R_0 = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^n r_{n+1} = r_1 + \gamma R_1 \quad (1)$$

In a kind of reinforcement learning called Q learning, Q function $Q(s, a)$ is defined to represent the maximum R when action a is taken at the state s. With the help of Bellman Equation, we can iteratively update the Q value constantly. If the Q function is accurate enough and environment is definitely set, we only need to take the strategy to choose the action that have the maximum Q value.
In the traditional Q Learning, the Q value is stored in a Q table, the column of the table represents all possible states while the row represents all possible actions. This method can be a good solution for some problems, especially the states are few and can be represented by a short tensor. But in reality, the number of states is always too large and sometime the states are continuous and can not be represented discretely.

2.2 Deep Q Learning

As we know, neural network can approximate the nonlinear function very well. We can replace Q table with neural network, which is also known as Q network. Researchers from DeepMind Technologies have trained a convolution neural network based on Q-learning to play Atari games and make a great success.

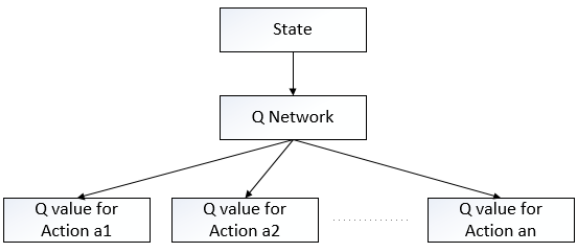


Figure 1: Basic Structure of Deep Q Learning

We take all kinds of actions, receive corresponding reward, then train the Q Network constantly. After the Q Network is trained enough to approximate the Q function, we can take some strategies to choose action through the Q value, such as ε-greedy algorithm and softmax algorithm.

3 AIR COMBAT ENVIRONMENT

One-to-one air combat is only considered in this paper and to each UCAV, the state of either itself or the opponent is available.[13]

3.1 State of UCAV

The state of UCAV is represented by a tensor $[v\ \psi\ \theta\ \phi\ x\ y\ z]$. $[xyz]$ is the position of UCAV, x represents the North coordinate, y represents the East coordinate, and z represents the height of UCAV. Except the position, there are some more elements: v represents the velocity, ψ represents yaw angle, θ represents pitch angle, and ϕ represents roll angle.

3.2 Flight Dynamics Equation

The state of UCAV changes according to its vertical overload N_x, tangential overload N_z and roll angle ϕ.
The UCAV dynamic characteristics follow the differential equations shown as formula (2).

$$\begin{aligned}
\dot{v} &= g(N_x - \sin\theta) \\
\dot{\psi} &= \frac{gN_z \sin\phi}{v\cos\theta} \\
\dot{\theta} &= \frac{g}{v}(N_z\cos\phi - \cos\theta) \\
\dot{x} &= v\cos\theta\cos\psi \\
\dot{y} &= v\cos\theta\sin\psi \\
\dot{z} &= v\sin\theta
\end{aligned} \quad (2)$$

Given the input $[N_x\ N_z\ \phi]$ and the time step Δt, we can easily calculate the next state of UCAV using the Runge-Kutta method, commomly RK4 method.

3.3 Actions

There are seven basic kinds of maneuvering actions in NASA standard[14], which are accelerating, slowing down, uniform flight, turning left, turning right, climbing and diving. Each maneuvering action consists of different combination of the three maneuvering inputs: N_x, N_z and ϕ.

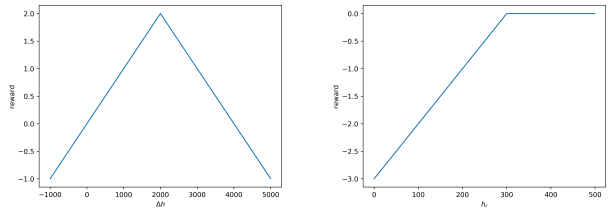
Table 1: Maneuvering Inputs of Actions

Actions	Maneuvering Inputs		
	N_x	N_z	ϕ
Uniform	$\sin\theta$	$\cos\theta$	0
Accelerating	$\sin\theta + 3$	$\cos\theta$	0
Slowing down	$\sin\theta - 3$	$\cos\theta$	0
Climbing	$\sin\theta$	$\cos\theta + 5$	0
Diving	$\sin\theta$	$\cos\theta - 5$	0
Turning left	$\sin\theta$	$\cos\theta + 3$	$-\pi/3$
Turning right	$\sin\theta$	$\cos\theta + 3$	$\pi/3$

3.4 Rewards

Rewards play an important role in deep reinforcement learning. Reasonable rewards lead the Q Network to choose better actions while training iteratively. More directly, the loss function is calculated from the rewards.

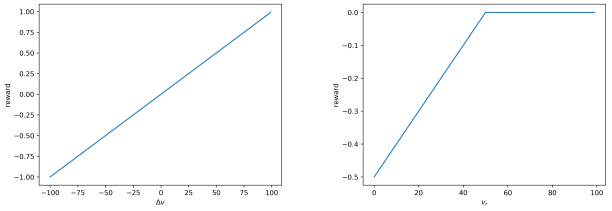
Energy-Maneuverability theory is commonly utilized in air combat. Energy consists of kinetic energy and potential energy of the aircraft. UCAV takes maneuvering actions to achieve the transformation of its kinetic and potential energy, that is, the speed and height. Of course in the process of turning energy will get lost, part of which can be obtained from the engine power supplement. Energy-Maneuverability theory, can be understood as a series of tactical maneuver to manage its own energy, reasonable to adjust the speed and the relationship between the enemy and the position to gradually obtain the advantages of the state of energy, and ultimately into the advantages of the location relationship and obtain the appropriate angle of fire. So we design the rewards mainly considering the speed, height and the relative positional relationships.



(a) Rewards of Δh (b) Rewards of Own Height

Figure 2: Rewards of Height

Normally, UCAV of our side is marked as red while the opponent is blue, so the difference of height $\Delta h = h_r - h_b$. If Δh is between 0 to 2000m, we can gain corresponding positive rewards, otherwise negative reward. In particular, if the height of the red UCAV is below 300m, we gain negative rewards. That is how we can prevent the UCAV flying too low to stall and crash.



(a) Rewards of Δv (b) Rewards of Own Velocity

Figure 3: Rewards of Velocity

Similar to the height, $\Delta v = v_r - v_b$. If Δv is greater than 0, we can gain reward +1 per 100m/s, vice versa. If the velocity of the red UCAV is lower than 50m/s, we gain reward -1 per 100m/s. Thus we can warn the UCAV to care about the velocity to keep safe.

Figure 4 shows the relative positional relationships between red and blue UCAV. q_r is deviation angle, represents the angle between the velocity direction of the red UCAV and the direction of the connection of centroids of the red and blue UCAV; and q_b is disjunctive angle, represents the angle between the velocity direction of the blue UCAV and the direction of the connection of centroids of the red and

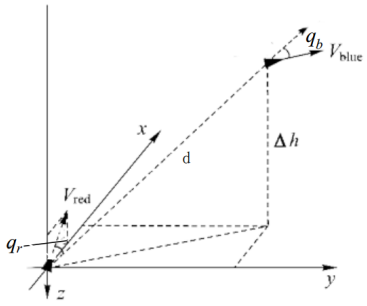
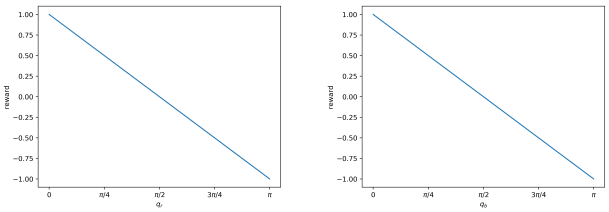


Figure 4: Relative Positional Relationship

blue UCAV. Both q_r and q_b can be calculated using the state elements of the red and blue UCAV as formula (3).

$$\begin{aligned} d &= \sqrt{(x_r - x_b)^2 + (y_r - y_b)^2 + (z_r - z_b)^2} \\ q_r &= \arccos\{[(x_b - x_r)\cos\psi_r \cos\theta_r + \\ &\quad (y_b - y_r)\sin\psi_r \cos\theta_r + (z_b - z_r)\sin\theta_r]/d\} \\ q_b &= \arccos\{[(x_r - x_b)\cos\psi_b \cos\theta_b + \\ &\quad (y_r - y_b)\sin\psi_b \cos\theta_b + (z_r - z_b)\sin\theta_b]/d\} \end{aligned} \tag{3}$$



(a) Rewards of q_r (b) Rewards of q_b

Figure 5: Rewards of Relative Positional Relationship

Both q_r and q_b vary between 0 to π. When q_r equals to 0, red UCAV is in a good position to attack. While q_r varies from 0 to π, position gets worse and worse. When q_b equals to 0, blue UCAV is in a bad position and easy to be attacked. While q_b varies from 0 to π, position gets better and more threatening to our red UCAV. So we convert both of them from angle 0 to π to rewards 1 to -1.

All in all, the total reward is the sum of all the rewards described above.

4 EXPERIMENTS

4.1 Structure of Deep Q Learning

As we have talked of all the four elements about air combat:

a) state of the UCAV

b) seven different actions

c) rewards

d) Q Network as policy,

we can apply the Deep Q Learning method to our air combat mission.[15–18]

Figure 6 shows the structure of deep Q learning method applied in air combat. The replay memory is utilized to store the experience tuples and break the correlation between consecutive samples.

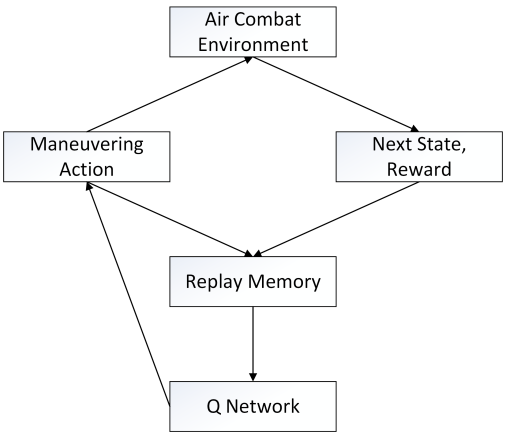


Figure 6: Structure of DQL in Air Combat

Algorithm 1 Deep Q Learning in Air Combat

Initialize replay memory D to capacity N;
Initialize Q Network with random weights θ;
for $episode = 1 \rightarrow M$ **do**
  Initialize the states of UCAVs on both sides;
  for $t = 1 \rightarrow T$ **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Red UCAV executes action a_t while blue UCAV taking some other strategies
    Calculate the next states $s_{t+1} = [s_{r,t+1}, s_{b,t+1}]$, the reward r_t, and the sign: done
    if $done$ **then**
      break
    else
      Store experience (s_t, a_t, r_t, s_{t+1}) in D
      Sample random minibatch from D
      Update the Q Network with the sample
    end if
  end for
  if the end condition is fulfilled **then**
    return
  end if
end for

When it comes to the detail of our training update rule, we use the temporal difference error δ based on the Bellman eqution:

$$\delta = Q(s, a) - (r + \gamma \max_a Q(s', a)) \tag{4}$$

To minimise this error, we will use the Huber loss. The Huber loss acts like the mean squared error when the error is small, but like the mean absolute error when the error is large.

$$L(\delta) = \begin{cases} \frac{1}{2}\delta^2 & for|\delta| \leq 1 \\ |\delta| - \frac{1}{2} & otherwise. \end{cases} \tag{5}$$

In order to prevent overfitting, we define the final loss function l as formula (6).

$$l = L(\delta) + c||\theta||^2 \tag{6}$$

where c is a parameter controlling the level of L2 weight regularisation.

4.2 Preprocessing and Network Architecture

The different state values have different dimensions, so we need to preprocess them to a uniform format. Velocity is divided by 100m/s, the angles are divided by , and the coordinates are divided by 1000m. Now the states are all converted to non-dimensional values. So the final input of the Q Network is the new tensor combined by converted states of red and blue UCAV.

Table 2 shows the details of Q network. The hidden layers are all fully-connected linear layers and consist of 32, 64, 64, 32 rectifier units respectively. The output layer is also fully-connected linear layer with 7 units which represent the Q values of different actions.

Table 2: Details of Q Network

layers	units number	activation func
input	14	-
hidden1	32	ReLU
hidden2	64	ReLU
hidden3	64	ReLU
hidden4	32	ReLU
output	7	-

4.3 Experiment platform

TensorFlow is an open-source deep learning platform developed by Google. Its easy to use and well performed. It can also handle complex missions by running on GPUs. Table3 shows the specific platform we use.

Table 3: Experiment Platform

OS	Ubuntu 16.04
CPU	i7-6850K
GPU	NVIDIA GeForce GTX 1080Ti
Memory	32GB
Platform	TensorFlow 1.3.0

4.4 Training Results

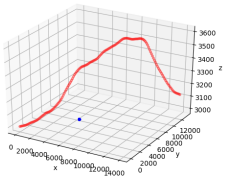
a) At first, we set the blue UCAV stable and the red UCAV attacking from different directions: front, behind, left side and right side. After each 5 rounds of training, the results are shown as Figure 7.

In this way, we hope the red UCAV can learn different rewards it can get in different positions.

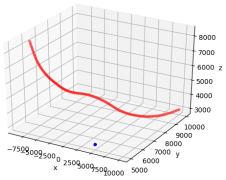
b) Then we set the red UCAV and the blue UCAV flying oppositely. The red UCAV is under training while the blue UCAV is flying by constant speed.

As we can see from Figure8, the red UCAV firstly learns to climb up to gain an advantage of height. But height is not always the bigger the better as we set in chapter 3.4. The red UCAV gradually learns to control its height in a good position. Not only the height, but also the angles are our goals to learn and important for the UCAV to gain advantages. Finally, the red UCAV learn to turn around to fly against to the back of blue UCAV, which is a definitely advantageous situation.

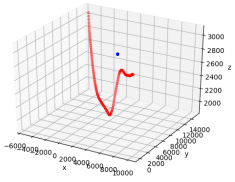
c) After finishing the designed training above, we let both the red and blue UCAV take actions through the Q Network, which is also known as self-play. After each 10



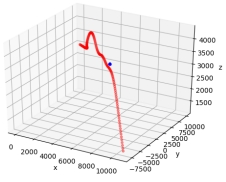
(a) Attack from Front



(b) Attack from Behind

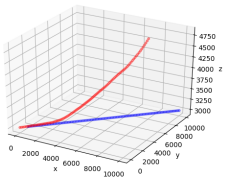


(c) Attack from Left Side

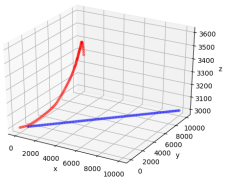


(d) Attack from Right Side

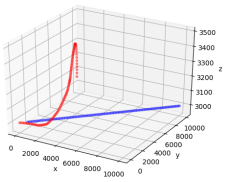
Figure 7: Traning Results Against Stable Blue UCAV



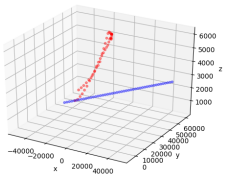
(a) After 50 Episodes



(b) After 100 Episodes



(c) After 150 Episodes



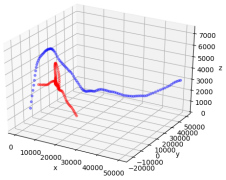
(d) After 200 Episodes

Figure 8: Training Results Against Uniform Flying Blue UCAV

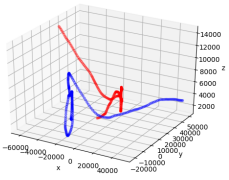
episodes of training, we can save the training result. Figure 9 shows the training results after 100 and 200 episodes.

5 CONCLUSION

In this paper, we successfully apply the deep reinforcement learning method to one-to-one air combat. We create the air combat environment, design the rewards functions and train the neural network to get a reasonable model. After some man designed training missions, the UCAV can keep optimizing the neural network by self-play all the time. Due to the limitation of the machine performance, mainly the GPU, we limit the air combat in a relatively short distance. Actually, thanks to the epic approximation ability of neural network, it's easy to expand to bigger battlefield or add more factors like different weapons. Air combat is still



(a) After 100 Episodes



(b) After 200 Episodes

Figure 9: Training Results of Self-play

a very complex task, with better the development of deep learning technology, the UCAV will reach a higher level of intelligence.

REFERENCES

[1] FU L, WANG X G. Research on Close Air Combat Modeling of Differential Games for Unmanne Combat Air Vehicles [J]. Acta Armamentarii, 2012, 10:1210-1216..

[2] Austin F, Carbone G, Falco M. Automated maneuvering decisions for air-to-air combat [J]. AIAA, 1987: 87-2393.

[3] Guo H, Zhou D Y, Zhang K. Study on UCAV Autonomous Air Combat Maneuvering Decision-Making [J]. Electronics Optics and Control, 2010, 08:28-32.

[4] Virtanen K, Karelahti J, Raivio T. Modeling air combat by a moving horizon influence diagram game [J]. Journal of Guidance, Control, and Dynamics, 2006, 29(5):1080-1091.

[5] Burgin G H, Sidor L B. Rule-based air combat simulation[R]. Titan systems inc la jolla, 1988.

[6] Zhang Lipeng, Wei Ruixuan, LI Xia, Autonomous Tactical Decision Making of UCAVs in Air combat, in Electronics Optics & Control, 2012, 19(2):92-96.

[7] ZHU D F, JI F B, GUAN Y Y. Study On Air Combat Tactics Decision-making for the Fourth Generation Fighters [J]. Command Control and Simulation, 2012, 01:41-43.

[8] ZHONG L, TONG M A, ZHONG W Y. Cooperative Team Air Combat Decision based on Integration of Rough Sets and Neural Networks [J]. Fire Control and Command Control, 2006, 06:881-884.

[9] Lachner R, Breitner M H, Pesch H J. Three-dimensional air combat: Numerical solution of complex differential games[M]//New trends in dynamic games and applications. Birkh?user Boston, 1995: 165-190.

[10] GU J J, ZHAO J J, LIU W H. Air Combat Maneuvering Decision Framework Based on Game Theory and Memetic Algorithm [J]. Electronics, Optics and Control, 2015, 01:20-23.

[11] Li D, Jiang J, Xu H. Reinforcement learning methods for finding equilibria and tracking evolution paths in conflicts [C] //Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on. IEEE, 2008:3292-3297.

[12] Virtanen K, Raivio T, Hamalainen R P. Modeling pilot's sequential maneuvering decisions by a multistage influence diagram[J]. Journal of Guidance, Control, and Dynamics, 2004, 27(4): 665-677.

[13] Smith R E, Dike B A, Mehra R K, et al. Classifier systems in combat: two-sided learning of maneuvers for advanced fighter aircraft[J]. Computer Methods in Applied Mechanics and Engineering, 2000, 186(2): 421-437.

[14] Lewis M S, Aiken E W. Piloted Simulation of One-on-One Helicopter Air Combat at NOE (Nap-of-the-Earth) Flight Levels [P]. America:ADA160538, 1985-4.

[15] Moriarty D, Schultz A, Grefenstette J, Evlutionary algorithms for Reinforcement Learning [J]. Journal of Artificial Intelligence Research, 1999, 11(1):241-276.

[16] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. Cambridge: MIT press, 1998.

[17] G. A. Rummery. Problem Solving with Reinforcement Learning. Ph. D. dissertation. Cambridge University, Cambridge, U. K, 1995

[18] Abbeel P, Coates A, Quigley M, et al. An application of reinforcement learning to aerobatic helicopter flight[J]. Advances in neural information processing systems, 2007, 19: 1.