Research on autonomous maneuvering decision of UCAV based on approximate dynamic programming

Zhencai Hu, Peng Gao, Fei Wang

Zhencai Hu, Peng Gao, Fei Wang, "Research on autonomous maneuvering decision of UCAV based on approximate dynamic programming," Proc. SPIE 11321, 2019 International Conference on Image and Video Processing, and Artificial Intelligence, 113212P (27 November 2019); doi: 10.1117/12.2547893

SPIE.

Event: The Second International Conference on Image, Video Processing and Artifical Intelligence, 2019, Shanghai, China

Research on Autonomous Maneuvering Decision of UCAV Based on Approximate Dynamic Programming

Zhencai Hu, Peng Gao, Fei Wang*
Harbin Institute of Technology, Shenzhen, China

ABSTRACT

Unmanned aircraft systems can perform some more dangerous and difficult missions which manned aircraft systems cannot perform. For tasks with high complexity, such as air combat, maneuvering decision mechanism is required to sense the combat environment and make the optimal strategy in real time. This paper formulates one-to-one air combat maneuvering problem in 3D environment, and proposes an approximate dynamic programming approach to make optimal maneuvering decisions automatically. The aircraft searches for combat strategies based-on Reinforcement Leaning, while sensing the environment, taking available maneuvering actions and receiving feedback reward signals. To solve the problem of dimensional explosion in the air combat, the proposed method is implemented through feature selection, trajectory sampling, function approximation and Bellman backup operation in the air combat simulation environment. This approximate dynamic programming approach provides a fast response to rapid changing tactical situations, and learns effective strategies to fight against the opponent aircraft.

Keywords: Air Combat, Maneuvering Decision, Approximate Dynamic Programming, Reinforcement Learning

1. INTRODUCTION

Unmanned combat aerial vehicle (UCAVs) carry kinds of weapons and perform missions automatically, UCAVs have been successfully employed to replace manned aircrafts in a variety of military aerial missions for their high mobility and large overloaded maneuvers. It is foreseeable that UCAVs will become primary choice in future air combat. Autonomous maneuvering decision mechanism determines how UCAVs choose maneuvering actions during fight. Taking fast and accurate tactical maneuvers plays a crucial role in complex and constant changing air-to-air combat situations. The goal of autonomous maneuvering in air combat simulation is to develop a method that can make effective maneuvering decisions in real time, and compute desirable sequentially maneuvers without expert pilot involvement directly.

UAV air combat has been explored by several researchers in past. Autonomous maneuvering decision of UAV roughly fall into two categories, i.e., non-learning method and self-learning method. Non-learning method may include influence diagram[1,2], differential game[3] and expert system[4] and so on. There is no training and optimizing process in non-learning strategies. Moreover, these maneuvering strategies are fixed and limited for situational awareness, because it is almost impossible to build a knowledge base that can completely cover all air combat situations. Self-learning methods, may include artificial immune system[5] and reinforcement learning[6], and so forth. These methods adopt their own experiences, and the models are optimized to cope with complex and changeable environments. Reinforcement learning (RL) methods have been successfully applied to many difficult decision making tasks[7,8].

Unfortunately, in large-scale air combat simulation, RL-based dynamic programming (DP) method are intractable because of dimensional explosion, i.e., it is impractical to store all the continuous air combat states or learn the values of each state individually. Function approximation[9] is capable of estimating the objective state value functions[10] and can be generalized to invisible states by using characterized state representation. Inspired by Mcgrew and Williams[6], by combining DP and function approximation, the approximate dynamic programming (ADP) can be used to make the air combat UCAV maneuvering decisions in 2D environment.

In this paper, we create a simplified 3D one-to-one air combat adversarial simulation environment and learn an automatic online maneuvering decision making model for UCAV based on ADP.

* Contact Author: wangfeiz@hit.edu.cn (Fei Wang).

2019 International Conference on Image and Video Processing, and Artificial Intelligence,
edited by Ruidan Su, Proc. of SPIE Vol. 11321, 113212P · © 2019 SPIE
CCC code: 0277-786X/19/$21 · doi: 10.1117/12.2547893

2. APPROXIMATE DYNAMIC PROGRAMMING TO AIR COMBAT

Given UCAV dynamics formulations and an approximate objective function, dynamic programming provides an efficient method to extract an optimal maneuvering policy for air combat simulation. The optimal policy provides the best sequential actions at any state, while dramatically reduces massive online computation. This section gives a brief review on dynamic programming, and then describes the proposed approximate solution in detail.

2.1 Dynamic Programming

Decisions have to be made sequentially during UCAV air combat, which can be abstracted into a Markov Decision Process[11] (MDP). State vector s_i represents the combat situation at each state $i = 1, \ldots, n$, and the whole state space is consisted of all possible single discrete states as $S = \begin{bmatrix} s_1 & s_2 & \cdots & s_N \end{bmatrix}$, with N the total number of states. The UCAV agent is allowed to take an available action a_i from action set $A = \begin{bmatrix} a_1 & a_2 & \cdots & a_M \end{bmatrix}$ at each time step, with M the total number of actions. And a scalar feedback reward signal r is given by the environment situation. Given current state and action input, we use the state transition function $f(s, a_r, a_b)$ to compute the next state based on the dynamics of UCAV movement.

A state value function $V(s)$ is defined at each state, representing the long-term future reward from current state s, which is used to evaluate the goodness or badness of the state. The optimal policy is determined by the optimal future reward value function of each state $V^*(s)$, which can be computed by performing the Bellman backup[12] operation on each state iteratively, as Eq.(1).

$$V^{k+1}(s_i) = \max_{a \in A} \left\{ r_{s_i}^a + \gamma V^k [f(s_i, a, a_b)] \right\} \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor applied at each one-step look forward prediction, k is the index of iteration, and r_s^a is the immediate reward of agent by taking action a at state s_i. The Bellman backup optimization can be implemented on many states simultaneously. After adequate iteration, $V(s)$ will converge to the optimal value $V^*(s)$. Then an optimal policy π^* can be determined from $V^*(s)$. Playing greedy[13,14], the optimal action at each time step is defined as

$$a_i = \pi^*(s_i) = \arg\max_{a \in A} \left\{ r_{s_i}^a + \gamma V^* [f(s_i, a, a_b)] \right\} \tag{2}$$

2.2 Function approximation

A continuous function approximator eliminates the need to represent and compute the future reward for every individual state s_i [15]. Some correlated features are chosen to approximate the value function of a continuous state space, where the components of state vector s_i can take any continuous value within the boundary. The original discrete look-up table of $V(s)$ is now replaced by approximated function $V_{approx}(s)$ which is initialized to be zero at all state. $V_{approx}(s)$ is used in a greedy way to choose actions. State space S can be determined by some manageable state sampling approach. The Bellman backup operation is applied to each state sample, and the values are stored in target vector $\hat{V}^{k+1}(S)$.

$$\hat{V}^{k+1}(S) = \max_{a_r \in A} \left\{ r_S^a + \gamma V_{approx}^k [f(S, a_r, a_b)] \right\} \tag{3}$$

The linear estimator uses a set of descriptive features $\phi(s)$ and the target vector to estimate the future reward function $V_{approx}^{k+1}(s)$ by a standard least-square estimation.

$$V_{approx}^{k+1}(s) = \phi(s)(\Phi^T \Phi)^{-1} \Phi^T \hat{V}^{k+1}(s) \tag{4}$$

Feature vector set $\Phi = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_N \end{bmatrix}^T$ stores the feature vectors computed from all available states $s_i \in S$. The ADP architecture relieves some of the difficulties associated with dimensional curse in classical DP techniques.

3. AIR COMBAT ENVIRONMENT

In this section, system states, control actions, aircraft dynamics, goal and reward function are described for one-to-one air combat simulation.

3.1 States, Actions and Dynamics

Assuming the aircraft as a particle, the combat state is defined by its speed, position and the flight attitude angles of a red UCAV (denoted by the r subscript) and a blue UCAV (denoted by the b subscript) at any time step. The state vector is

$$s = [v_r, x_r, y_r, z_r, \theta_r, \psi_r, \varphi_r, v_b, x_b, y_b, z_b, \theta_b, \psi_b, \varphi_b] \tag{5}$$

where x, y and z indicate the coordinates of the craft in the three-dimensional spatial inertial system, with x consisting with the east, y the north and z the height. v represents its flying velocity, θ, ψ and φ refer to the pitch, yaw and roll angle, respectively.

According to the kinematic principles, the flight dynamics of both the Red and Blue side follows the differential equations shown as formula (6).

$$\begin{aligned}
&\dot{v} = g(N_x - \sin\theta) \\
&\dot{\psi} = gN_z \sin\varphi / (v\cos\theta) \\
&\dot{\theta} = (N_z \cos\varphi - \cos\theta) g / v \\
&\dot{x} = v\cos\theta\sin\psi \\
&\dot{y} = v\cos\theta\cos\psi \\
&\dot{z} = v\sin\theta
\end{aligned} \tag{6}$$

N_x is tangential overload, N_z is normal overload and φ is the roll angle of the aircraft. Relevantly, the list of $[N_x, N_z, \varphi]$ is regarded as the control commands input to the system. The dynamics explains the state transition mechanism. With the dynamics formula (6), the state transition in combat space can be represented as function $s' = f(s, a_r, a_b)$. Given current state s, successively UCAV maneuvering control actions a_r and a_b, a suitable simulation time interval Δt, the following state s' can be solved with $f(s, a_r, a_b)$ using the fourth order Runge-Kutta[16] method. To ensure that the spatial areas most likely to be seen during combat are sampled sufficiently, the state samples for ADP training process are sampled by the trajectory sampling. Simulation is initialized at a randomly generated state, and the sampling process continues until all needed points are generated.

The establishment of the UCAV maneuver library can refer to the pilot operations. NASA have designed seven typical flight maneuver actions that can be used in air combat, which are continued flying, maximum acceleration, maximum deceleration, turn left, turn right, pull up and push down. The corresponding control inputs $[N_x, N_z, \varphi]$ of these maneuvers are shown in Table 1.

Table 1. Control inputs for basic UCAV maneuver Commands

Maneuvering inputs	Air Combat Maneuvers Library						
	Continued	Acceleration	Deceleration	Turn left	Turn right	Pull up	Push down
N_x	0	2	0	0	0	0	0
N_z	1	1	1	5	5	5	-5
φ	0	0	0	$-\pi/3$	$\pi/3$	0	0

3.2 Goal and Reward function

The goal of a UCAV is to attain and maintain a position advantage over the opponent aircraft, and to shoot the enemy, while minimizing its own risks. The value function $V(s)$ is actually the discounted cumulative reward along state trajectory. Thus, a proper defined reward can better guide the aircraft to its goal areas from any stating state. The reward function at every time step is computed through the real-time combat situation. Figure 1 shows the spatial relative relationships between two aircrafts.

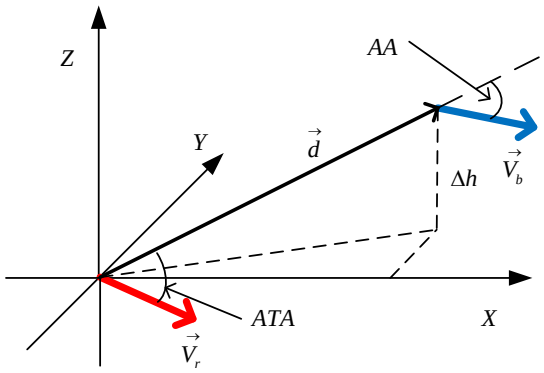


Figure1 Spatial relative relationship between the red and blue aircraft

The angle factors in air combat are mainly Aspect Angle (AA) and Antenna Train Angle (ATA). From the perspective of the red side, AA represents the angle between the distance vector and the tail of blue speed vector. ATA is the angle between red speed vector and its radar guideline. Both AA and ATA can be calculated using the components in state vector as formula (7).

$$\begin{aligned}
&\vec{v_r} = [\cos\psi_r \cos\theta_r \quad \sin\psi_r \cos\theta_r \quad \sin\theta_r] \\
&\vec{v_b} = [\cos\psi_b \cos\theta_b \quad \sin\psi_b \cos\theta_b \quad \sin\theta_b] \\
&\vec{d} = [x_r - x_b \quad y_r - y_b \quad z_r - z_b] \\
&AA = \arccos(\vec{v_b}.\vec{d} / |\vec{d}|) \\
&ATA = \arccos(\vec{v_r}.\vec{d} / |\vec{d}|)
\end{aligned} \tag{7}$$

The extracted features include velocity advantage reward $\Delta v = v_r - v_b$, height advantage reward $\Delta h = z_r - z_b$, and the distance-angle reward R_3, as Eq.(8).

$$R_3 = \left[\frac{(1-|AA|/\pi)+(1-|ATA|/\pi)}{2}\right] \bullet e^{-(\frac{|d|-R_d}{k\pi})} \tag{8}$$

R_d is the expected attacking range of weapons in aircraft, $|d|$ is the relative distance between aircrafts, and k is an coefficient adjusting the influence of R in reward. All the three rewards are scaled to the interval $[-1,1]$, and the total reward function at every time step is the weighted sum of them, as Eq.(9).

$$R = (\Delta v + \Delta h + R_3)/3 \tag{9}$$

Feature vector chosen to approximate the value function for every state s is $\phi(s) = [AA, ATA, \Delta z, \Delta v, |\vec{d}|]$. The reward function for blue side UCAV can be computed in the same way.

4. EXPERIMENTS

As we have designed the simplified 3D one-to-one air combat environment and expounded ADP application in air combat. In this section, simulations are conducted to demonstrate the capability of a UCAV to make intelligent successively maneuvering decisions.

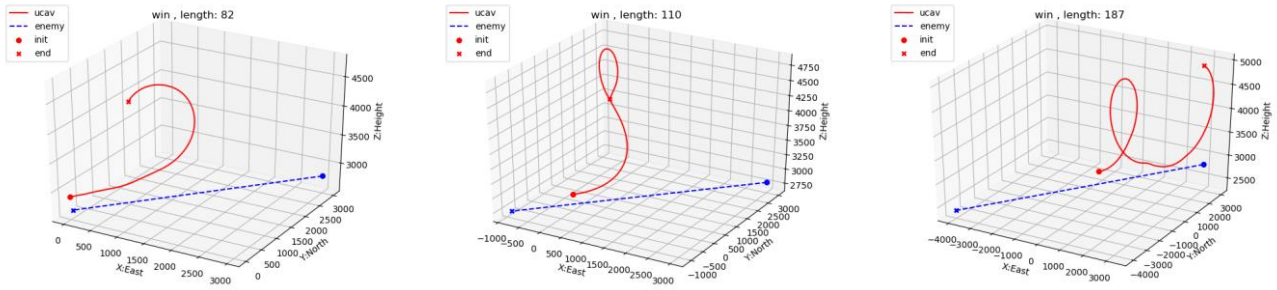
We set the red and the blue UCAV flying oppositely. The original positional values are sampled from a Gaussian distribution with a standard deviation $\sigma = 10$ m. The pitch and yaw angle are also sampled from a narrow range uniform distribution. The mean values used in Gaussian distribution and uniform distribution are listed in Table 2. The state space in the ADP model includes 100,000 discrete samples obtained by trajectory sampling, and the number of dynamic programming learning iterations is set as 40.

Table 2. Initial state mean values used for simulation

S_init	v_r (m/s)	x_r (m)	y_r (m)	z_r (m)	θ_r (°)	ψ_r (°)	v_b (m/s)	x_b (m)	y_b (m)	z_b (m)	θ_b (°)	ψ_b (°)
values	250.	0.	0.	2900.	0.	45.	204.	3000.	3000.	2800.	0.	-135.

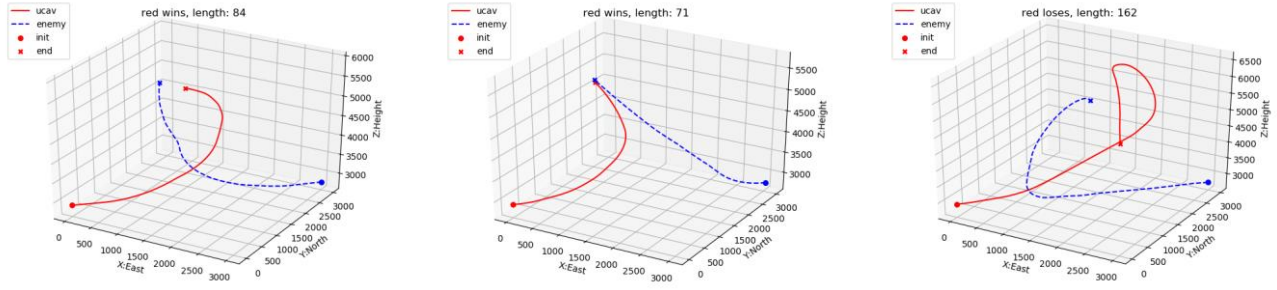
The time interval in the whole simulation process is $\Delta t = 0.25$s , that is, every maneuvering decision is made after 0.25s. Once a UCAV enters the dominated area of the opponent UCAV, or the total number of steps in current episode reaches a predefined maximum number, the combat episode ends immediately. The dominated area is defined as $|ATA| < 1.1$ and $|AA| < 0.6$, and the maximum number of steps of each combat episode is defined as 200 to avoid long time standoff in the simulation.

We present two combat situations. In the first case, the red UCAV makes online maneuvering decisions according to the policy learned from ADP method, while the blue UCAV keeps the continued flying maneuvering and flies in a straight line. As we can see in Figure 2, (a), (b) and (c) represent different three air combat episodes respectively. When the blue side flies straightly, the red UCAV learns an effective tactic strategy that pulls up the aircraft and gains the advantage of height firstly, and then turns the direction around and dive to the back side of the enemy aircraft immediately to obtain the relative dominating area.



(a) (b) (c)

Figure 2. Typical combat results against continued flying opponent



(a) (b) (c)

Figure 3. Typical combat results of self-confrontation fight

In the second case, we let both the red side and blue side make maneuvering decisions through the ADP decision model, which is also known as a self-confrontation process. In Figure 3, (a) and (b) indicate red side UCAV defeats the enemy, and (c) indicates the red side UCAV lose the air combat. The both side aircraft had learnt intelligent maneuvering decisions in changing and complex situations.

5. CONCLUSION

In this paper, we successfully apply the approximate dynamic programming method to air combat simulation, which avoids the problem of dimensional explosion problem by combining the function approximation and dynamic programming. Specifically, we create a simplified one-to-one adversarial air combat simulation in 3D environment, design the UCAV available action set, and specify the reward function of UCAV, which directs the optimization direction. Moreover, an ADP based air combat training and testing model is constructed. Without the involvement of experienced air combat pilot, the UCAV can intelligently select an effective tactic strategy after certain amount of training. Experimental evaluations demonstrate that our agent UCAV can learn the offensive and defensive tactics in highly flexible air combat game. Due to the limitation of the computing power, we constrain the air combat scene in a relatively simplified occasion and take action choice as discrete values. Continuous action values maneuvering model would more likely to characterize real pilot maneuvering, while at the cost of more complexity. In the future, more factors would be considered, such as expanding the spatial scales in air combat simulation, including weapon considerations, and even many-to-many UCAV battle strategy. Air combat simulation problem is still a highly complex issue. With the better development of reinforcement learning technology, both fighting performance and decision-making capabilities of UCAV will show higher intelligence.

REFERENCES

[1] Virtanen K, Karelahti J, Raivio T J J O G, Control,, et al. Modeling air combat by a moving horizon influence diagram game[J], 2006, 29(5): 1080-1091.

[2] Virtanen K, Raivio T, Hamalainen R P J J O G, Control,, et al. Modeling pilot's sequential maneuvering decisions by a multistage influence diagram[J], 2004, 27(4): 665-677.

[3] Park H, Lee B-Y, Tahk M-J, et al. Differential game based air combat maneuver generation using scoring function matrix[J], 2016, 17(2): 204-213.

[4] Mcmanus J W, Chappell A R, Arbuckle P D. Situation assessment in the Paladin tactical decision generation system[J], 2003.

[5] Kaneshige J, Krishnakumar K. Artificial immune system approach for air combat maneuvering[C]// Intelligent Computing: Theory and Applications V, International Society for Optics and Photonics, 2007: 656009.

[6] Mcgrew J S, How J P, Williams B, et al. Air-combat strategy using approximate dynamic programming[J], 2010, 33(5): 1641-1654.

[7] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J], 2016, 529(7587): 484-489.

[8] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J], 2013.

[9] Bhatnagar S, Precup D, Silver D, et al. Convergent temporal-difference learning with arbitrary smooth function approximation[C]// Advances in Neural Information Processing Systems, 2009: 1204-1212.

[10] Sutton R, Barto A. Reinforcement Learning:An Introduction[M]. MIT Press, 1998: 90-127.

[11] Puterman M L. Markov decision processes: discrete stochastic dynamic programming[M]. John Wiley & Sons, 2014.

[12] Bellman R J P O T N a O S O T U S O A. On the theory of dynamic programming[J], 1952, 38(8): 716.

[13] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J], 2015.

[14] Lange S, Riedmiller M. Deep auto-encoder neural networks in reinforcement learning[C]// The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010: 1-8.

[15] Powell W B J a O O R. Perspectives of approximate dynamic programming[J], 2016, 241(1-2): 319-356.

[16] Zhang M, Qin H, Lan M, et al. A high fidelity simulator for a quadrotor uav using ros and gazebo[C]// IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2015: 002846-002851.