Learning Continuous 3-DoF Air-to-Air Close-in Combat Strategy using Proximal Policy Optimization

Luntong Li[*†], Zhiming Zhou[* §], Jiajun Chai[†], Zhen Liu[§], Yuanheng Zhu[†], Jianqiang Yi[§]
[†]The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
Email: {luntong.li, chaijiajun2020, yuanheng.zhu }@ia.ac.cn
[§]Integrated Information System Research Center, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
Email: {zhiming.zhou, liuzhen, jianqiang.yi}@ia.ac.cn

***Abstract*—Air-to-air close-in combat is based on many basic fighter maneuvers and can be largely modeled as an algorithmic function of inputs. This paper studies autonomous close-in combat, to learn new strategy that can adapt to different circumstances to fight against an opponent. Current methods for learning close-in combat strategy are largely limited to discrete action sets whether in the form of rules, actions or sub-polices. In contrast, we consider one-on-one air combat game with continuous action space and present a deep reinforcement learning method based on proximal policy optimization (PPO) that learns close-in combat strategy from observations in an end-to-end manner. The state space is designed to promote the learning efficiency of PPO. We also design a minimax strategy for the game. Simulation results show that the learned PPO agent is able to defeat the minimax opponent with about 97% win rate.**

Index terms—air-combat, reinforcement learning, proximal policy optimization, flight simulation**.**

I. Introduction

ARTIFICAL intelligence (AI) technology has been introduced to autonomous air-combat, and is considered to be one of the critical components in future aircrafts [1]. However, air-combat autonomy is not easy, because of the challenging and dynamic nature lying in it.

Current air-combat autonomy solutions vary from rule-based methods to end-to-end deep reinforcement learning (DRL) methods. The rule-based system is reasonable and trustable. However, the rule-based systems, which depend on the expert knowledge, suffer from the curse of dimensionality and cannot learn any new strategies. Recently, many researches focus on the end-to-end solutions based on DRL. DRL has been applied in many fields like games [2], simulated missile guidance [3], neural architecture search [4] and autonomous driving [5]. Recently, Pope *et al.* [6] proposed a hierarchical reinforcement learning method to dynamically select pretrained sub-policies, and won 2nd place in the final AlphaDogfight Trials event. Combining what with game theory [7], [8], a multi-agent hierarchical policy gradient algorithm was proposed in [9]. Other explorations include using approximate dynamic programming [10], fuzzy logic [11], and model predictive control [12] to address the air-combat problem.

Most of the aforementioned approaches successfully solve air-combat problem with the discrete actions space whether in the form of rules, actions or sub-polices. However, three-degree-of-freedom (3-DoF) air-to-air close-in combat problem with continuous action space is less explored in the literature. To address the problem with continuous action space, this paper proposes an end-to-end DRL solution based on proximal policy optimization (PPO) [13] with an efficient state space design.

The rest of this paper is organized as follows. In Section II, the background of air-combat and Markov decision process (MDP) is presented. The opponent's minimax strategy is designed in Section III. We present our solution in Section IV. Simulation results are shown in Section V. Section VI provides the conclusions and future directions.

II. Problem Formulation

A 3-DoF one-on-one air-combat environment consists of two air-crafts. Each aircraft has the following six variables: x, y, z, v, φ, γ and ϕ, where x, y and z are the position variables of the aircraft in the global coordinate system, with respect to northward displacement, eastward displacement, and altitude, respectively; v, φ, γ and ϕ are the velocity, aircraft heading angle, flight-path angle and bank angle, respectively. We denote one of the aircrafts, which is controlled by our DRL agent, as red aircraft, and the opponent as blue aircraft. The control input of red aircraft consists of tangential load n_x, normal overload n_z and ϕ. Furthermore, the first-order inertial links are considered in the control input. We define n_{xcmd}, n_{zcmd} and ϕ_{cmd} as the control commands.

Under the global coordinate system, the dynamics is adapted from [14]. In one-on-one air-combat game, the goal of any aircraft is to attain and maintain advantage over the opponent, and to escape the weapon engagement zone (WEZ)

[*]The two authors contribute equally.

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101005, the National Natural Science Foundation of China under Grant 62136008, the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27030400, and Youth Innovation Promotion Association CAS under Grant 2021132.

of the opponent. The positions of advantage are quantized by three terms: antenna train angle α, aspect angle β and the distance D between the two aircrafts. Fig. 1 gives examples of WEZs. If one of the aircrafts attains the region that satisfies conditions of $|\alpha| \le 30°$, $|\beta| < 150°$ and $D \le 3000$m, then we say the aircraft wins the game. If both aircrafts fall into each other's WEZ, then we call it a draw.

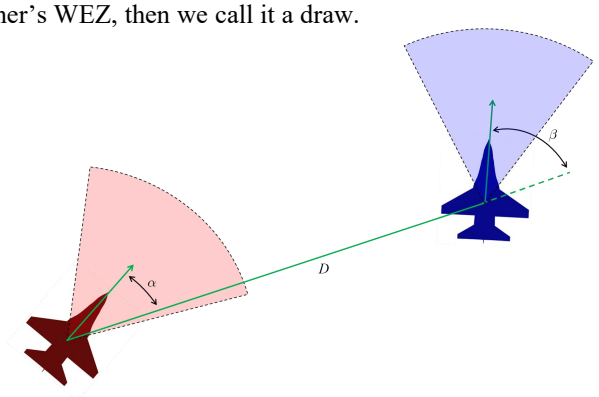


Fig. 1. Aircraft relative geometry.

III. Minimax Strategy

A. Decision Space Design

The decision space of minimax strategy is defined as the variations in altitude, aircraft heading angle and velocity: $\Pi = (\Delta z, \Delta\varphi, \Delta v)$, where $\Delta z \in$ [-200m, 200m], $\Delta\varphi \in [-10°, 10°]$ and $\Delta v \in$ [-30m/s, 30m/s]. Furthermore, all candidate decisions of each variables correspond to a list of five discrete values, i.e. Δz = [-200m, -100m, 0m, 100m, 200m], $\Delta\varphi = [-10°, -5°, 0°, 5°, 10°]$, and Δv = [-30m/s, -15m/s, 0m/s, 15m/s, 30m/s].

B. Minimax Strategy

Minimax strategy first uses the dynamics model and controllers to roll forward aircraft states with many steps, given one of the candidate decisions. Then minimax strategy assesses the future state according to a situation assessment function (SAF). After obtaining all the scores of all candidate decisions, minimax strategy searches for the optimal decision d^* by solving

$$d^* = \max_i \min_j \Omega_{i,i}, \tag{1}$$

where Ω denotes the score matrix whose element $e_{i,j}$ at row i and column j stands for the score when own aircraft takes decision i and the opponent takes decision j. Finally, the output decision of the minimax strategy can be set as the tracking signals of controllers to compute control commands, to take position of advantage in the future situation.

C. Situation Assessment Function

To assessing all the possible tactical situations of different maneuvers, an SAF is defined to calculate the score matrix Ω. We build SAF according to [14], with appropriate modifications to close-in air-combat case. The main difference is that the maximum detection range of radar $D_{R\max}$=100000m, the maximum range of the missile's radar $D_{M\max}$=60000m, and the maximum range of the missile $D_{Mk\max}$=30000m.

D. Controller Design

Controllers are designed to track the decision of minimax strategy, i.e., the desired values of altitude $z_d \triangleq z_t + \Delta z$, aircraft heading angle $\varphi_d \triangleq \varphi_t + \Delta\varphi$ and velocity $v_d \triangleq v_t + \Delta v$.

To track the desired velocity, we first denote the tracking error as $e_v = v_d - v$. A simple controller is designed as

$$u_1 = \sin(\gamma) + (\dot{v}_d - k_v e_v)/g, \tag{2}$$

where $k_v > 0$ is the controller gain.

To track the desired altitude, we design a cascade control system, which comprises an outer-loop altitude control system and an inner-loop flight-path angle control system. In the inner loop, the controller tracks the desired flight-path angle γ_d, which is commanded by the outer loop. The tracking errors for inner-loop and outer-loop are defined as $e_\gamma = \gamma_d - \gamma$ and $e_z = z_d - z$, respectively. The flight-path angle controller and altitude controller can be designed as

$$\begin{cases} u_2 = -v(\dot{\gamma}_d - k_\gamma e_\gamma)/g - \cos(\gamma) \\ \gamma_d = -(\dot{z}_c - k_z e_z)/v \end{cases}, \tag{3}$$

where k_γ>0 and k_z>0 are the controller gains of inner-loop and outer-loop, respectively.

Similar to the velocity controller, the aircraft heading angle controller signal can be designed as

$$u_3 = -v\cos(\gamma)(\dot{\varphi}_d - k_\varphi e_\varphi)/g, \tag{4}$$

where $e_\varphi = \varphi_d - \varphi$ is the tracking error, and $k_\varphi > 0$ is the controller gain.

Then the control commands can be calculated by

$$n_{xcmd} = u_1, \tag{5}$$

$$n_{zcmd} = \begin{cases} \sqrt{u_2^2 + u_3^2}, \text{if } u_2 \ge 0 \\ -\sqrt{u_2^2 + u_3^2}, \text{otherwise} \end{cases}, \tag{6}$$

and

$$\phi_{cmd} = \begin{cases} \arctan(u_3/u_2), \text{if } u_2 > 0 \\ 0, \text{otherwise} \end{cases}. \tag{7}$$

IV. Policy Optimization for Close-in Air-Combat

A. Markov Decision Processes and Reinforcement Learning

We assume that the state transitions in air-combat game satisfies Markov property. A Markov decision process is formalized as a quintuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, r, ρ), where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $\mathcal{P}(s'|s,a)$: $\mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ is the probability associated with transition from the state s to next state s' given the action a of the RL agent, $r: \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ is the reward function, and $\rho \in [0,1)$ is the discount factor. A stationary policy of a RL agent is a mapping from state space to action

space: $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. Given a stationary policy π and a starting state-action pair (s_0, a_0), the action value function is defined as

$$Q^{\pi}(s,a) \triangleq \mathbb{E}[\sum_{t=1}^{\infty} \rho^{t-1} r_t \,|\, s_0 = s, a_0 = a, \pi], \tag{8}$$

where r_t is the reward at time-step *t*. The discounted sum of future rewards is also known as return. Similarly, the state value function is defined as follows:

$$V^{\pi}(s) \triangleq \mathbb{E}[\sum_{t=1}^{\infty} \rho^{t-1} r_t \,|\, s_0 = s, \pi]. \tag{9}$$

The control commands $[n_{xcmd}, n_{zcmd}, \phi_{cmd}]$ is defined as the action of the agent.

B. Reward Function

According to the aircraft relative geometry in Fig. 1, we can define the reward function of the game. When *D* is less than firepower range, i.e. 3000*m*, the reward function of the agent at time-step *t* is defined as follows:

$$r_t = \begin{cases} -10, \text{if } |\alpha_t| > 30^\circ \textit{ and } |\beta_t| \geq 150^\circ \\ 0, \text{if } |\alpha_t| \leq 30^\circ \textit{ and } |\beta_t| \geq 150^\circ \\ 10, \text{if } |\alpha_t| \leq 30^\circ \textit{ and } |\beta_t| < 150^\circ \end{cases} \tag{10}$$

where both α and β are normalized to $[-180^\circ, -180^\circ]$.

C. State Space Design

We design the state space of DRL agent based on the fight-path coordinate system. The relative position vector in the view of red aircraft is defined as follows

$$p_{re} = T\left[x_b - x_r, y_b - y_r, z_b - z_r\right]^{\mathrm{T}}, \tag{11}$$

where the subscript *r* and *b* respectively stand for red and blue aircraft, and T represents the transformation matrix from the global coordinate system into fight-path coordinate system:

$$T = \begin{bmatrix} \cos\gamma\cos\varphi & \cos\gamma\sin\varphi & \sin\gamma \\ -\sin\varphi & \cos\varphi & 0 \\ \sin\gamma\cos\varphi & \cos\gamma\sin\varphi & \cos\gamma \end{bmatrix}.$$

Similarly, the relative velocity vector in the view of red aircraft is

$$v_{re} = T\left[\dot{x}_b - \dot{x}_r, \dot{y}_b - \dot{y}_r, \dot{z}_b - \dot{z}_r\right]^{\mathrm{T}}. \tag{12}$$

We add velocity, γ , n_x , n_z and ϕ of own-ship into state with the following form

$$O_{add} = \left[\nu, \cos(\gamma), \sin(\gamma), n_x, n_z, \cos(\phi), \sin(\phi)\right]^{\mathrm{T}}, \tag{13}$$

where n_x , n_z and ϕ are the actions of red aircraft at time-step *t*. Then the overall state *s* are the concatenation of p_{re} , v_{re} and O_{add} , resulting in a vector of 13 dimensions:

$$s = \left[p_{re}^{\mathrm{T}}, v_{re}^{\mathrm{T}}, O_{add}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{14}$$

D. Proximal Policy Optimization

In each iteration, the actor of PPO attempts to maximize the following surrogate objective:

$$\begin{aligned} L^{actor}(\theta) = \tilde{\mathbb{E}}\Big[& \min\left(r_t(\theta)\tilde{A}_t, \mathrm{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\tilde{A}_t\right) \\ & -\zeta \mathrm{KL}\left(\pi_{\theta}(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t)\right)\Big] \end{aligned} \tag{15}$$

where $\tilde{\mathbb{E}}[\cdot]$ denotes the empirical average over a batch of samples generated by current policy, and $\tilde{A}_t$ is an estimation of $A^{\pi_\theta}(s,a)$, $r_t(\theta) = \pi_\theta(a_t|s_t) / \pi_{\theta_{old}}(a_t|s_t)$, ε and ζ are hyperparameters to control the changes of policy. Furthermore, ζ is adaptive updated to achieve the target value of the KL divergence d_{targ}.

To estimate the advantage function and value function, we use the generalized advantage estimator (GAE) [15]. Given a batch of samples, the value network aims to minimize following objective:

$$L^{critic}(\omega) = \tilde{\mathbb{E}}\left[\left\| V_{\omega}(s_t) - R_t \right\|^2\right]. \tag{16}$$

In addition, we also use an entropy bonus to promote exploration. Then we have the following overall objective function that PPO maximizes at each iteration:

$$L(\theta, \omega) = L^{actor}(\theta) - c_1 L^{critic}(\omega) + c_2 E(\pi_\theta) \tag{17}$$

where c_1, c_2 are coefficients, and the definition of the entropy bonus is as follows:

$$E(\pi_\theta) \triangleq \mathbb{E}_{s_t \sim \mathcal{D}, a \sim \pi_\theta}\left[-\ln \pi_\theta(\cdot|s_t)\right]. \tag{18}$$

V. Simulation and Analysis

A. Experimental Settings

In each episode of the training process, the initial positions, velocities, and heading angles of PPO agent and opponent are randomized. Specifically, the initial *x*, *y*, *z*, *v* and φ are uniformly sampled from the following intervals: $x \in$ [-10000*m*, 10000*m*], $y \in$ [-10000*m*, 10000*m*], $z \in$ [-5000*m*, 0], $v \in$ [0.4Mach, 1.5Mach] and $\varphi \in [-180^\circ, 180^\circ]$. Whereas both the initial values of γ and ϕ are set to 0.

We use neural network as our function approximator to represent policy and value. The network is simply a fully-connected multilayer perceptron with two hidden layers of 128 units and ReLU nonlinear activation function, followed by two separate heads for policy and value. Here, the outputs of policy head are the means and standard deviations of three Gaussian distributions.

In each environment time-step, the minimax strategy of the opponent rolls forward with 2s for *z*, 2s for φ , and 4s for *v*, under each candidate decision.

B. Simulation Results

We train PPO agent for 1000 iterations. Fig. 2 shows the average return of the PPO agent. The solid curves correspond to the average return and the shaded region to the standard deviation, averaging over three independent runs. After 400 iterations, PPO learns a policy with average return of 9.3. When the training process is finished, the learned policies are tested 100 times to estimate the probability of win, lose and draw combating with the minimax opponent, averaging over three independent runs. In Table I, the final testing average probabilities of success of PPO and minimax strategy (combating with minimax strategy) are 97.3% and 1.7%, respectively.

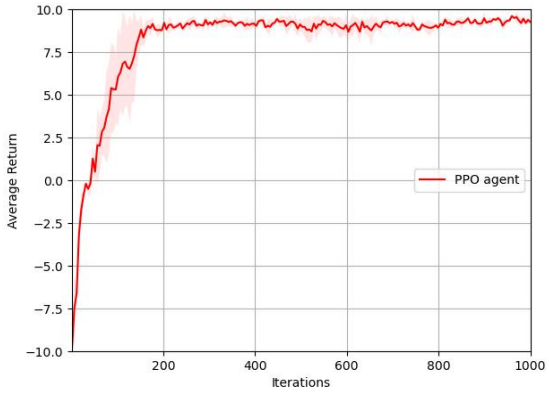
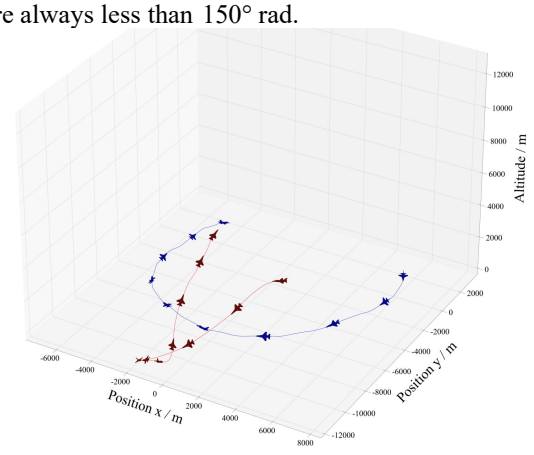


Fig. 2. Average return of PPO agent.

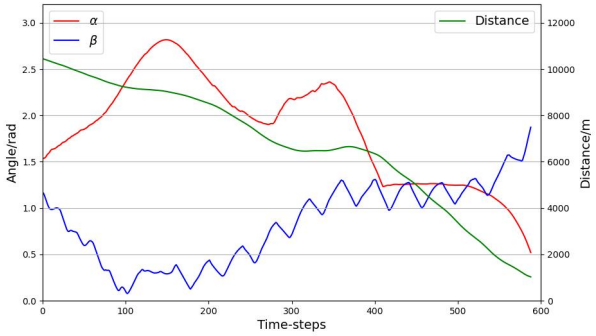
TABLE I
THE PROBABIITY OF SUCCESS, LOSE AND DRAW

Method	Probability of success	Probability of lose	Probability of draw
PPO agent	**97.3%**	2.7%	0.0%
Minimax strategy	1.7%	12.3%	86%

To visualize what strategy and maneuvering PPO agent has learned, we show some trajectories in the test of PPO combating with minimax strategy. In Fig. 3(a), PPO uses combinations of many primary maneuvers, e.g., accelerations, climbs and turns, to stay close with the opponent and finally attain the position of advantage. In Fig. 3(b), the values of α are decreasing to the range of PPO's WEZ. And the values of β are always less than 150° rad.



(a) Trajectories of PPO agent (red) versus minimax strategy (blue).



(b) α, β and D values of PPO in the close-in combat.

Fig. 3. Testing scenario in close-in air-combat.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we develop a DRL agent for one-on-one close-in 3-DoF air-combat with continuous action space. We first build a 3-DoF air-combat environment, develop a minimax strategy and design controllers based on feedback control strategy. Then we apply PPO algorithm to close-in air-combat game and propose an efficient state space design method. PPO performs strategies and flight maneuvers that can adapt to different circumstances to defeat an opponent in combat with minimax strategy. Future works include extensions to more challenging scenarios like continuous 6-DoF air-combat and multi-aircraft combat.

REFERENCES

[1] M. Byrnes, "Nightfall: Machine autonomy in air-to-air combat," *Air Space Power J.* vol.28, no. 5, pp. 48–75, May, 2014.

[2] S. Hao, L. Li, M. Liu, Y. Zhu, and D. Zhao, "Learning representation with Q-irrelevance abstraction for reinforcement learning," in *Proceeding of* 11th *International Conference on Intelligent Control and Information Processing*, 2021.

[3] W. Li, Y. Zhu, and D. Zhao, "Missile guidance with assisted deep reinforcement learning for head-on interception of maneuvering target," *Complex Intell. Syst*., 2021.

[4] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. P. Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst*., early access, 2021.

[5] J. Wang, Q. Zhang, and D. Zhao, "Highway lane change decision-making via attention-based deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 3, pp. 567-569, 2022.

[6] A. P. Pope *et al*., "Hierarchical reinforcement learning for air-to-air combat," *arXiv preprint*, arXiv: 2105.00990, 2021.

[7] Y. Zhu and D. Zhao, "Online minimax Q network learning for two-player zero-sum Markov games," *IEEE Trans. Neural Netw. Learn. Syst*., vol. 33, no. 3, pp. 1228–1241, 2022.

[8] Z. Tang, Y. Zhu, and D. Zhao. "Enhanced rolling horizon evolution algorithm with opponent model learning," *IEEE Transactions on Games*, 2020.

[9] Z. Sun *et al*., "Multi-Agent hierarchical policy gradient for air combat Tactics emergence via self-play," *Engineering Applications of Artificial Intelligence*, vol. 98, no. 1, article 104112, 2021.

[10] J.S. McGrew, J.P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *J. Guid. Control Dyn*, vol. 33, no. 5, pp. 1641–1654, 2010.

[11] N. Ernest, *et al.*, "Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions," *J. Def. Manag.* vol. 6, no. 1, pp. 2167–0374, 2016.

[12] M. Ramirez, *et al.*, "Integrated hybrid planning and programmed control for real time UAV maneuvering," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Stockholm, Sweden, pp. 1318–1326, 2018.

[13] J. Schulman, *et al*., "Proximal policy optimization algorithms," *arXiv preprint* arXiv:1707.06347, 2017.

[14] Y. Kang, et al., "Beyond-Visual-Range tactical game strategy for multiple UAVs," in *Proceeding of Chinese Automation Congress (CAC)*, Nov, 2019.

[15] J. Schulman, *et al*., "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint* arXiv:1707.06347, 2017.