Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs

1st Yan ZHEN
Science and technology on complex system control and intelligent agent cooperation laboratory
Beijing, China
zhenyan_hit@163.com

2nd Mingrui HAO*
Science and technology on complex system control and intelligent agent cooperation laboratory
Beijing, China
hmrhit@163.com

3rd Wendi SUN
Science and technology on complex system control and intelligent agent cooperation laboratory
Beijing, China
judysun07@163.com

***Abstract*—The fixed-wing UAV is a non-linear and strongly coupled system. Controlling UAV attitude stability is the basis for ensuring flight safety and performing tasks successfully. The non-linear characteristic of the UAV is the main reason for the difficulty of attitude stabilization. Deep reinforcement learning for the UAV attitude control is a new method to design controller. The algorithm learns the nonlinear characteristics of the system from the training data. Due to the good performance, the PPO algorithm is the mainly algorithm of reinforcement learning. The PPO algorithm interacts with the reinforcement learning training environment by gazebo, and improve attitude controller, different from the traditional PID control method, the attitude controller based on deep reinforcement learning uses the neural network to generate control signals and controls the rotation of rudder directly.**

Keywords—aircraft, reinforcement learning, controller, attitude, policy

I. INTRODUCTION

The fixed-wing UAVs have been widely used in various fields, because of its simple structure, low cost and convenient. The large outdoor UAVs are in use for military and commercial task, which save a lot of material cost and improve efficiency[1]. In recent years, despite the fixed-wing UAVs control technology has greatly developed, flight control is still a hot research area. Due to the performance of autopilot systems, the application of UAVs is greatly limited. In the actual flight process, various uncertain disturbances brought by the external complex environment affect the stability of the aircraft's flight, and the flight control algorithm of the UAVs must have good robustness[2]. Generally, the control system of UAV is divided into inner control loop and outer control layers. The inner control loop provides low-level stabilization of the UAVs' attitude, and the outer control layers provide path planning and guidance[3].

The flight control system uses the onboard computer to generate actuator control signals based on the status data returned by the sensors. In addition, we also want the controller to control UAVs stably, when there are lots of external disturbances. Aiming at the control difficulties of UAVs, many researchers have made great efforts in the research of control algorithms to solve tracking problems[4]. Autopilots for fixed-wing UAVs are typically designed using cascaded single-variable loops under assumptions of decoupled longitudinal and lateral motion, using classical linear control theory[5]. However, there are strongly coupled and nonlinear in the dynamics of the fixed-wing UAVs. Considering the nonlinear characteristics of the systems, nonlinear control method is applied to UAVs. The method includes linear parameter varying, sliding model control, nonlinear model predictive control and so on. These methods require accurate aerodynamic model of the UAVs. We propose a method of UAV attitude control using deep reinforcement learning, which is a model-free control method.

One of the primary goals of the field of artificial intelligence is to solve complex tasks from unprocessed, high-dimensional, sensory input. While DQN[6] algorithm has solved problems with high-dimensional observation spaces, it can only handle discrete and low-dimensional action spaces. Flight control is a continuous control problem, which is time-sensitive. In this paper, we use PPO algorithm to train RL controller, the PPO[7] algorithm is a model-free, off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces. According to OpenAI, the PPO algorithm has been their main algorithm using continuous control.

We can divide the intelligent controller into two categories according to application of deep neural network. One is that the neural network is used as the controller directly; the other is that based on the existing control structure, the controller parameters are generated by the deep neural network. We designed a control network using reinforcement learning, which is used to generate rudder deflection commands for fixed-wing UAVs using the flight data as input.

In this paper, we design an intelligent flight controller for the fixed-wing UAVs and train it using reinforcement learning. The methods can be extended to a wide range of aircraft. We create a training environment based on Gazebo software[8], which can interact with reinforcement learning algorithms by ROS.

II. BACKGROUND

A. Fixed-wing UAV model

978-1-7281-8025-0/20/$31.00 ©2020 IEEE

The UAV is modeled as a rigid body, which has 6 degrees of freedom in space[9]. The elevator and aileron generate pitching moment and rolling moment, respectively. Besides, the yaw direction is controlled by rudder and aileron. To describe the fixed-wing UAV model, we establish inertial coordinate system, body coordinate system and airflow coordinate system. The attitude dynamics equation of UAV can be established according to the aerodynamic torque. The coordinate system is shown in Figure 1:

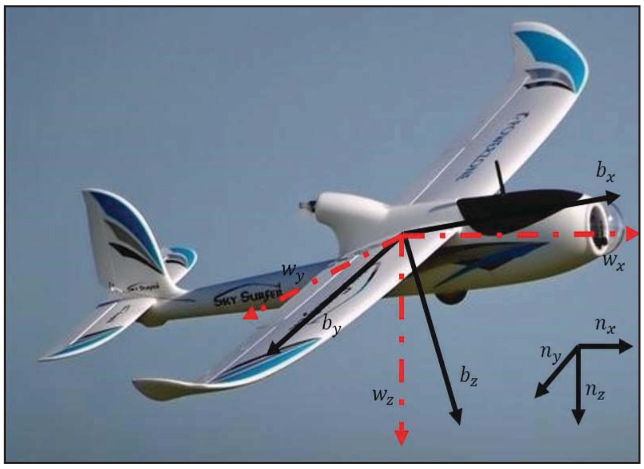

Figure 1. Coordinate system of the UAV

We use n, b and w to represent the inertial coordinate system, body coordinate system and airflow coordinate system. The α is angle of attack and β is sideslip angle. We can get the dynamic equation of the angle of attack and sideslip angle as:

$$\begin{cases}\dot{\alpha} = q + \alpha(\rho V_T S C_{Z\alpha})/2m \\ \dot{\beta} = -r + \beta(\rho V_T S C_{Y1})/2m\end{cases} \tag{1}$$

where m is the mass, V_T is the airspeed of the UAV, ρ is air density, and S is wing area.

The attitude can be represented using Euler angels $\Theta = [\varphi, \theta, \psi]^T$, where φ, θ, ψ are the roll, pitch and yaw angels respectively. The attitude angular velocity relative to the body coordinate system is p, q and r, The transformation between them is following:

$$\begin{bmatrix} p & q & r \end{bmatrix}^T = C_n^b \begin{bmatrix} \dot{\varphi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{2}$$

where C_n^b is the transformation matrix from the inertial frame to the body coordinate :

$$C_n^b = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\varphi & \sin\varphi\cos\theta \\ 0 & -\sin\varphi & \cos\varphi\cos\theta \end{bmatrix} \tag{3}$$

To establish the dynamic equation of the attitude system, we must define some variables to describe the fixed-wing UAV. The dynamic pressure of air during the flight of UAV is $\bar{q} = \rho V_T^2 / 2$. We use b and $\bar{c}$ represent wingspan and mean chord length. I_{xx}, I_{xz}, I_{yy}, I_{zx} and I_{zz} are moment of inertia. The torque coefficient directly affects the attitude of the UAV, we define that E_L is the coefficient of rolling moment, E_M is the coefficient of pitching moment, E_N is the coefficient of yawing moment.

$$\begin{cases} E_L = C_{La1}\delta_{a1} + C_{La2}\delta_{a2} + C_{Le1}\delta_{e1} + C_{Le2}\delta_{e2} + \\ \qquad C_{L\beta}\beta + C_{L\tilde{p}}\tilde{p} + C_{L\tilde{r}}\tilde{r} \\ E_M = C_{M1} + C_{Me1}\delta_{e1} + C_{Me2}\delta_{e2} + C_{Ma1}\delta_{a1} + \\ \qquad C_{Ma2}\delta_{a2} + C_{M\tilde{q}}\tilde{q} + C_{Ma}\alpha \\ E_N = C_{N\delta_r}\delta_r + C_{N\tilde{r}}\tilde{r} + C_{N\beta}\beta \end{cases} \tag{4}$$

where δ_a is the angel of aileron, δ_e is the angel of elevator, δ_r is the angel of rudder. $\tilde{p} = \frac{bp}{2V_T}$, $\tilde{q} = \frac{\bar{c}q}{2V_T}$ and $\tilde{r} = \frac{br}{2V_T}$.

According to the dynamics of the system, we can establish the dynamic equation of the attitude system of the fixed wing UAV as following:

$$\dot{\Omega}^b = (I^b)^{-1}(M^b - \Omega^b) \times (I^b \cdot \Omega^b) \tag{5}$$

where $\Omega^b = \begin{bmatrix} p & q & r \end{bmatrix}^T$, $M^b = \begin{bmatrix} \bar{q}SbE_L & \bar{q}S\bar{c}E_M & \bar{q}SbE_N \end{bmatrix}^T$, $I^b = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{zx} & 0 & I_{zz} \end{bmatrix}$.

B. Reinforcement Learning Algorithm

PPO algorithm is proposed to attain the data efficiency and reliable performance of TRPO, which guarantee monotonic improvements by penalizing changes to the policy, while using only first-order optimization[10]. PPO is an Actor-Critic method, Actor-Critic framework utilizes two neural networks to compute the control signal u and the state value function Q respectively. In general, the policy network is represented by π, its parameters are θ. The policy network takes the states as input, outputs an action as control signal.

PPO is one of the policy gradient algorithms, it estimates the policy gradient and apply a gradient ascent algorithm to update its parameters[11]. We run the policy in an environment. and obtain samples of the policy, then we can calculate the loss of the policy $J(\theta)$ and its gradient $\nabla_\theta J(\theta)$. They are the following:

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t R(s_t, a_t)\right] = E_{\tau\sim\pi_\theta(\tau)}\left[R(\tau)\right]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\right)R(\tau)\right] \tag{6}$$

PPO algorithm uses advantage function to measures how good an action is compared to the other actions available in the state, such that good actions have positive rewards, and bad actions have negative rewards. Critic network estimates the average reward of the state as value function $V(s)$.The advantage function is following[12]:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$
$$Q^{\pi}(s,a) = R + \gamma V^{\pi}(s')$$
$$A^{\pi}(s) = R + \gamma(s') - V^{\pi}(s) \tag{7}$$

PPO maximizes the surrogate objective function. Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \dfrac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$, so $r(\theta_{old}) = 1$.TRPO maximizes a "surrogate" object

$$L^{CPI}(\theta) = \hat{E}_t\left[\frac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{E}_t\left[r_t(\theta)\hat{A}_t\right] \tag{8}$$

The superscript CPI refers to conservative policy iteration, where this objective was proposed. Without a constrain, maximization of L^{CPI} would lead to an excessively large policy update; PPO modifies the objective, to penalize change to the policy that move $r_t(\theta)$ away from 1. PPO's main objective is the following:

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{9}$$

Where epsilon is a hyperparameter, say, $\varepsilon = 0.2$. The motivation for this objective is as follows. The first term inside the min is L^{CPI} . The second term, $clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval $[1-\varepsilon, 1+\varepsilon]$. Finally, we take the minimum of the clipped and unclipped objective, so the final objective is a lower bound on the unclipped objective. With this scheme, PPO only ignore the change in probability ratio when it would make the objective improve, and include it when it makes the objective worse. PPO is outlined in Algorithm1.

```
Algorithm 1: PPO
  for iteration=1, 2, … do
      for actor=1, 2, …, N do
          Run policy $\pi_{\theta_{old}}$ in environment for T time steps

          Compute advantage estimates $\hat{A}_t$ for
              t = 1, 2, …, T
      end
      Optimize surrogate L wrt. $\theta$ .
      $\theta_{old} \leftarrow \theta$
  end
```

III. ENVIRONMENT

The common feature of learning algorithms is to learn from data, so data is the most important component of learning algorithms[13]. The reinforcement learning algorithm needs to interact with the training environment to generate training data. In addition, during the training process of the reinforcement learning algorithm, there is a explore process in the external environment, that is, the control signal generated by the controller is perturbed and try to find a control action with higher reward. This is the reason why the application of RL to UAV platforms is limited compared to other robotics applications. Such actions may cause serious consequences. For example, the UAV may crash during the control of the RL controller. Therefore, we use the Gazebo simulation platform to generate UAVs' flight data in a simulated environment, and train intelligent controller using reinforcement learning.

The Gazebo is a 3D dynamic simulator that can accurately and efficiently simulate robot swarms in complex indoor and outdoor environments. Unlike game engines that provide high-fidelity visual simulation classes, Gazebo provides high-fidelity physical simulations, which provide a complete set of sensor models, as well as a user and program friendly way of interacting. In order to realize the interaction between gazebo and reinforcement learning algorithms, it is important to realize the communication between the various modules. The intelligent controller obtains the attitude information of the UAV from Gazebo, and Gazebo obtains the control actions from the intelligent controller. The system framework is shown in Figure 2.

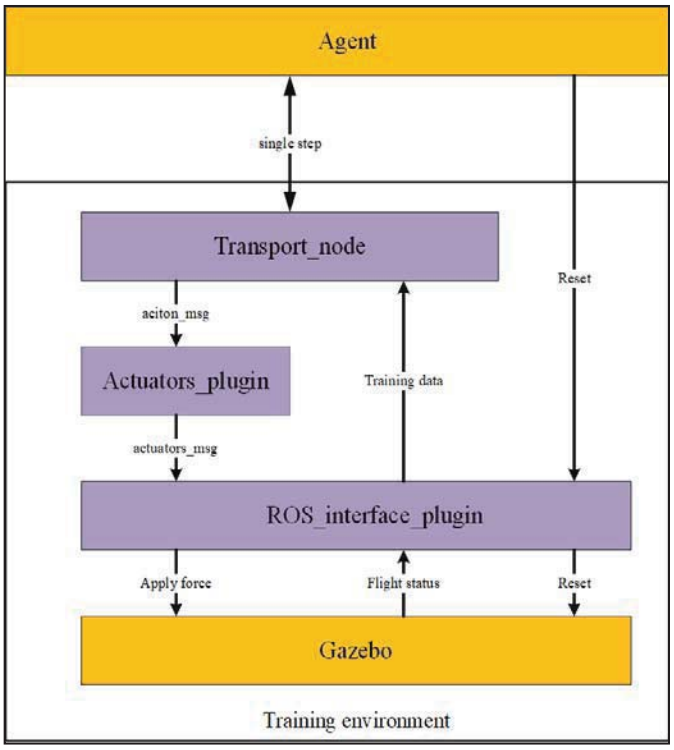

Figure 2. RL training system framework

A. Communication

We design the RL training environment to allow agent to learn attitude controller depending on the flight data. The first problem we must solve is the communication between different modules in this frame. Gazebo is a 3D dynamic simulator and we can simulate flight while training the reinforcement learning controller. The fixed-wing UAV in Gazebo is shown in Figure 3:

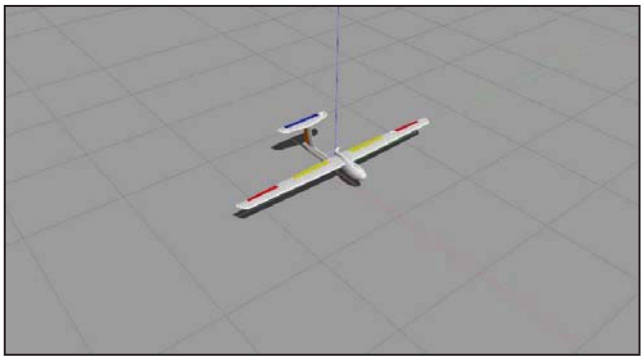

Figure 3. The fixed-wing in Gazebo

The transport node is a data switching center, which reads flight data from gazebo and send it to the agent. The RL controller uses the observed state to calculate actions, then sends the actions to the transport node. As the transport node gets the actions, it packs the actions and publishes to actuators plugin. Actuators plugin connect to gazebo by ROS interface plugin[14]. Once actuators plugin gets actuators message, it will set the angel of the actuators. The fixed-wing UAV will respond to the control signal in gazebo.

In order to get real-time flight data information, we define 'odometry function', which contains attitude data, position data, linear velocity and angular velocity data. Transport node subscribes flight data from ROS interface plugin. During RL controller training, we need reset the training environment, which set the UAV's initial attitude and velocity. The ROS interface plugin provides a convenient interface for users to set the model state. In this frame, all data is exchanged by topic in ROS, the most important thing is to define the structure of the packet.

B. Set up

The topmost layer interfacing with the agent is the environment interface layer which implements the OpenAI Gym environment API[15]. Each OpenAI Gym environment defines an observation space and an action space. The essence of reinforcement learning controller is to establish a nonlinear mapping between observation state and action, and express this mapping with neural network. The observation state is of size $m \times N$, where m is the number of past moments, N is the number of states at each moment. According to the observation state, we choose CnnMlpPolicy as control network. The CnnMlpPolicy has two convolution layers, followed by two fully connected layers. The output dimension of the control network is depending on the dimension of action space.

The observation state is to extract features, so we need expand the state spaces as much as possible[16]. We describe UAV's attitude using degrees, and limit the observation state value to a range. In contrast, we need to squeeze the action value as much as possible. Reinforcement learning algorithm needs to explore the action space and search for the appropriate action value. When the search space of action is too large, the network training time is greatly increased. The output of the control network is normalized between $[-1, 1]$ using the tanh function to squash[17].

To ensure the UAV to fly randomly in the gazebo training environment without crushing, we set the initial height of the UAV to 1000 meters. The wind disturbance term is added to the training environment, and the training environment reset is different from controller reset with a 2-millisecond delay. Thus, we can set the UAV attitude at a fixed value every time, but RL algorithm gets a random initial value. In this paper, we control the pitch angel, roll angel and the airspeed of the UAV. We randomly select a value from target setpoints in a training episode. The constrains and ranges for initial conditions and target setpoints are illustrated in table1, as following:

TABLE I. CONSTRAINTS AND RANGES

Variable	Constrains	Target ranges
φ	$\pm 180^\circ$	$\pm 30^\circ$
θ	$\pm 85^\circ$	$\pm 30^\circ$
ψ	$\pm 180^\circ$	$-$
p	$\pm 720^\circ / s$	$-$
q	$\pm 720^\circ / s$	$-$
r	$\pm 720^\circ / s$	$-$
V	$-$	$15-25 m/s$

IV. EVALUATION

In this section, we train an attitude controller for UAV with reinforcement learning. We control the fixed-wing UAV to track target roll angel, pitch angel and airspeed. Thus, the tracking precision is used to evaluate the performance of the RL controller.

A. Training of Controller

We use the RL algorithms PPO to train the attitude controller using the implementation in the OpenAI Stable-Baselines project. The goal of this project is to solve the stability problem of reinforcement learning algorithm. There are two implementations for PPO algorithm in this project, PPO1 and PPO2. PPO1 is a single-threaded reinforcement learning algorithm. Besides, PPO2 is a multi-threaded reinforcement learning algorithm, and it is more powerful than PPO1. However, in this paper, we can't communicate with the training environment in multithreading method[18]. The PPO1 is our default algorithm.

In order to improve the control stability, the convolutional neural network (CNN) was used as the control network. We saved the state of the system current moment and previous moments. We used the saved data build a state matrix. The outputs were angel of actuators.

Reinforcement learning takes control as a Markov decision-making process, and it needs to come to a natural end. However, the control of UAV is a continuous process and we can not find the natural end. In this case, we train the attitude controller in an episodic method, and the controller trained in this way can be adapted to actual flight verification. We can set new target value and initial state in every episode. The reference setpoints for the fixed-wing UAV are randomized in the ranges shown in Table I. The training framework we built is a real-time simulation system, the transmission frequency of the control signal is 100hz. There are 2000 time steps in one episode, corresponding to 20 seconds of flight time. In the experiment, we find that the length of episode can't be too short, it may lead to the algorithm training not to get good results.

The reinforcement learning is to maximize the value of the reward, so the control goals should be reflected in rewards function. Thus, we set the immediate reward returns to the RL controller are all negative rewards in the normalized range of -1 to 0:

$$\begin{aligned}
&R_{\varphi} = clip(\frac{|\varphi_c - \varphi|}{\mu_1}, 0, \lambda_1) \\
&R_{\theta} = clip(\frac{|\theta_c - \theta|}{\mu_2}, 0, \lambda_2) \\
&R_{V} = clip(\frac{|V_c - V|}{\mu_3}, 0, \lambda_3) \\
&R_t = -(R_{\varphi} + R_{\theta} + R_V) \\
&\mu_1 = 50, \mu_2 = 25, \mu_3 = 40 \\
&\lambda_1 = 0.50, \lambda_2 = 0.25, \lambda_3 = 0.25
\end{aligned} \tag{9}$$

The clip function makes the result between the $[-1,\ 0]$, because researchers have found sparse rewards to give poor performance[19]. In the early stage of training, μ must be set to be larger, so that the reward value will fluctuate greatly according to the control effect, and the algorithm can quickly find a good control action. Different from traditional controller design methods, we need to reduce the μ values gradually to improve control accuracy. Besides, when the controller can't control precisely for certain variable, we should increase λ. It represents that the algorithm pays more attention to the variable, and will improve the control precision of the variable.

B. Result

The controller was trained on a desktop computer with an i9-9900k CPU and an RTX 2080 GPU. The model on this hardware takes about four hours to converge. Firstly, we verify the tracking ability of the controller to typical signals, and then compare the control accuracy with the PID controller. For the PID controller, we first turned using classical method, and then manually adjust to improve the performance.

When we design the PID controller's parameters, we need to continuously adjust the parameter value to improve the control performance. However, due to the coupling nature of UAVs, there is a great blindness in the design of controller parameters. Compared with PID controller parameter design method, it is more convenient to design the controller by reinforcement learning. In this paper, the step signal and the sinusoidal signal are taken as typical instruction signals. The tracking of typical signals by the reinforcement learning controller is shown in the Figure 4,5,6:

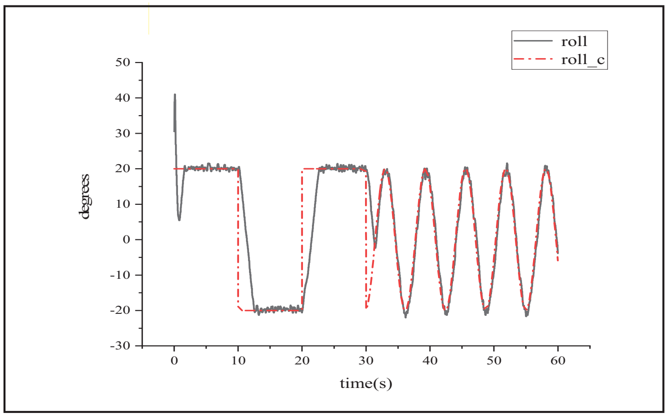

Figure 4. Roll Angle tracking curve

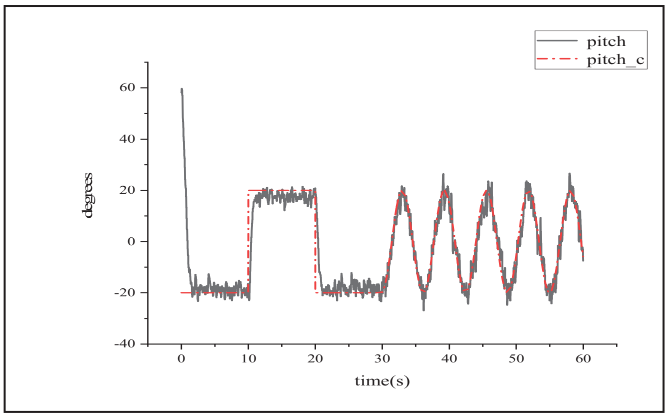

Figure 5. Pitch Angle tracking curve

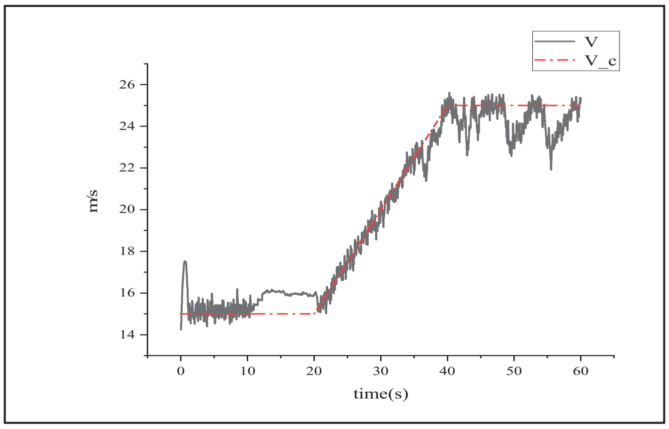

Figure 6.Airspeed tracking curve

We compare the performance of the reinforcement learning controller with the PID controller, and add wind disturbance in the training environment. The reinforcement learning controller and the PID controller both can control the UAV, the stability control of UAV attitude directly reflects the robustness of the controller in the case of external disturbance. The following figures show the pitch angel and roll angel response curves with external disturbance.

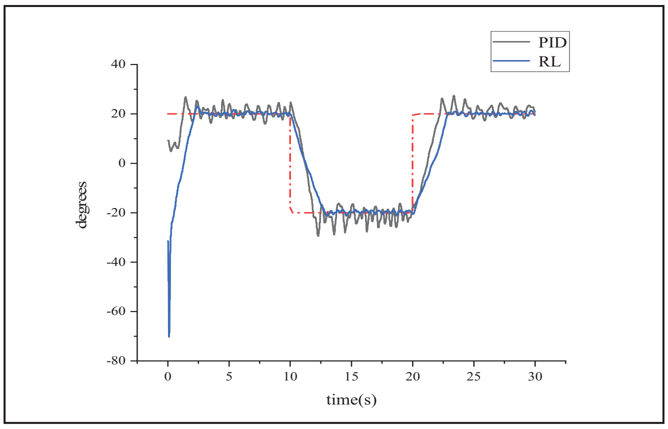

Figure 7. Roll Angle contrast tracking curve

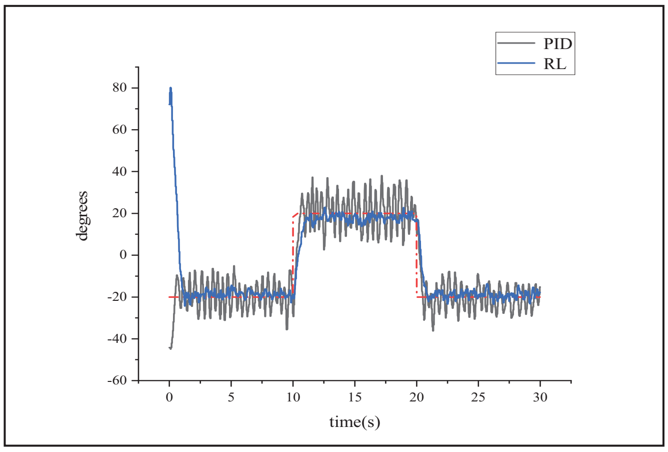

Figure 8. Pitch Angle contrast tracking curve

We compare the result of RL controller with PID, when the model changes, RL controller controlled more stably. The control network of RL controller generated control signals according to the state of several system moments. We find that the robustness of RL controller is stronger.

V. Conclusion

In this paper, we train an attitude controller for fixed-wing UAV using reinforcement learning. It is a model-free controller designed method for aircraft, not only for fixed-wing UAV but for multi-rotors drone. Compared with the PID controller, the controller based on the deep reinforcement learning can still achieve precise control of the aircraft when there is disturbance in the training environment. This is not end, we will plant the controller to the hardware platform next.

References

[1] Zonghua Sun,et al.Research on Roll Attitude Control of UAV Based on Active Disturbance Rejection Control[J].IOP Conference Series: Earth and Environmental Science,2020,428(1):012083 (7pp).

[2] Yuqiong Shan,Yuqiong Shan,Song Wang, et al.Attitude Control of Flying Wing UAV Based On Advanced ADRC[J].IOP Conference Series: Materials Science and Engineering,2019,677(5):052075 (10pp).

[3] Yeonsik Kang,, and Hedrick, J.K. "Linear Tracking for a Fixed-Wing UAV Using Nonlinear Model Predictive Control." *IEEE Transactions on Control Systems Technology* 17.5:1202-1210.

[4] Wei Ren. "On Constrained Nonlinear Tracking Control of a Small Fixed-wing UAV." *Journal of Intelligent & Robotic Systems Theory & Applications* 48.4:525-537.

[5] Mingfeng Zhang, and Hugh H. Liu. "Tracking a Moving Target by a Fixed-wing UAV Based on Sliding Mode Control." *Aiaa Guidance, Navigation, & Control Conference* 2013.

[6] Kenny Young, Ryan Hayward, & Gautham Vasan. (2016). Neurohex: A Deep Q-learning Hex Agent. *Ijcai Computer Games Workshop*.

[7] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, & Klimov, Oleg. . Proximal policy optimization algorithms.

[8] Fadri Furrer, Michael Burri, Markus Achtelik, & Roland Siegwart. (2016). *RotorS – A Modular Gazebo MAV Simulator Framework*. Springer International Publishing.

[9] Jianbo Shao, Qing Fei, Qiong Hu, & Qingbo Geng. (2014). Multi-model control design for longitudinal dynamics of fixed-wing UAV. *2014 33rd Chinese Control Conference (CCC)*. IEEE.

[10] A. Hill, A. Raffin, M. Ernestus, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable baselines,” https://github.com/hill-a/stable-baselines, 2018.

[11] Schulman J , Wolski F , Dhariwal P , et al. Proximal Policy Optimization Algorithms[J]. 2017.

[12] Mnih V , Badia, Adrià Puigdomènech, Mirza M , et al. Asynchronous Methods for Deep Reinforcement Learning[J]. 2016.

[13] David W. Aha, Dennis F. Kibler, & Marc K. Albert. (1991). Instance-based learning algorithms. *Machine Learning, 6*(1), 37-66.

[14] Aditya Narayanamoorthy, Renjun Li, & Zhiyong Huang. (2015). Creating ROS launch files using a visual programming interface. *IEEE International Conference on Cybernetics & Intelligent Systems & IEEE Conference on Robotics*. IEEE.

[15] Brockman, G. , Cheung, V. , Pettersson, L. , Schneider, J. , Schulman, J. , & Tang, J. , et al. (2016). Openai gym.

[16] Yu Zheng, Si-wei Luo, & Zi-ang Lü. (2008). Algorithm of stable state spaces in reinforcement learning. *Journal of Computer Applications, 28*(5), 1328-1176.

[17] Engui Fan. . Extended tanh-function method and its applications to nonlinear equations. *Physics Letters A, 277*(4-5), 212-218.

[18] Hiram Ponce, Ricardo Padilla, Alan Davalos, Alvaro Herrasti, & Daniel Dovali. (2015). A Case Study in Hybrid Multi-threading and Hierarchical Reinforcement Learning Approach for Cooperative Multi-agent Systems. *2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI)*. IEEE Computer Society.

[19] Koch, W. , Mancuso, R. , West, R. , & Bestavros, A. . (2018). Reinforcement learning for UAV attitude control.