Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Proximal policy optimization with an integral compensator for quadrotor control*

Huan HU, Qing-ling WANG†‡

School of Automation, Southeast University, Nanjing 210096, China

†E-mail: qlwang@seu.edu.cn

Received Nov. 22, 2019; Revision accepted Feb. 24, 2020; Crosschecked Apr. 27, 2020

Abstract: We use the advanced proximal policy optimization (PPO) reinforcement learning algorithm to optimize the stochastic control strategy to achieve speed control of the "model-free" quadrotor. The model is controlled by four learned neural networks, which directly map the system states to control commands in an end-to-end style. By introducing an integral compensator into the actor-critic framework, the speed tracking accuracy and robustness have been greatly enhanced. In addition, a two-phase learning scheme which includes both offline- and online-learning is developed for practical use. A model with strong generalization ability is learned in the offline phase. Then, the flight policy of the model is continuously optimized in the online learning phase. Finally, the performances of our proposed algorithm are compared with those of the traditional PID algorithm.

Key words: Reinforcement learning; Proximal policy optimization; Quadrotor control; Neural network
https://doi.org/10.1631/FITEE.1900641 **CLC number:** TP183; TP273

1 Introduction

In the past few decades, unmanned aerial vehicles (UAVs) have received much attention, and have been applied in many fields, such as agricultural services (Valente et al., 2013), aerial photography (Valenti et al., 2016), industrial inspection (Fumagalli et al., 2012), and search and rescue (Tomic et al., 2012). This has directly led to a great deal of research on the quadrotor. Because of its simple structure, it has been put into use in many practical applications.

Although quadrotor research has made significant progress, it still faces some problems. First, real-time control of the aircraft means real-time data acquisition and computation, which requires high time sensitivity and advanced equipment. Second, the aircraft should be able to adapt to various complex and harsh environments during the flight, and possible malfunctions have to be considered. Taking these factors into consideration, flight control is still an open research question.

Flight control systems are critical elements in a variety of missions and applications for unmanned aerial vehicles. They usually have two application levels: one is an advanced mission-planning control system, such as path planning, navigation, and obstacle avoidance; the other is a low-level stable flight system that performs simple motion control. In this study, we are concerned mainly with the second level. We update the control strategy by tracking the speed state to achieve stability and accuracy of the flight control system. However, the quadrotor is a highly nonlinear, multi-input multi-output underactuated coupling system, which makes the controller design very difficult and complicated. More seriously, usually a large number of unmodeled dynamic and

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2018AAA0101400), the National Natural Science Foundation of China (Nos. 61973074, U1713209, 61520106009, and 61533008), the Science and Technology on Information System Engineering Laboratory (No. 05201902), and the Fundamental Research Funds for the Central Universities, China

ORCID: Huan HU, https://orcid.org/0000-0001-5022-0771; Qing-ling WANG, https://orcid.org/0000-0003-2045-2920

nonlinear external disturbances are contained. Therefore, designing a quadrotor controller capable of anti-interference has gradually become a research issue. At present, many control strategies have emerged for anti-interference control of the quadrotor, such as proportional–integral–derivative (PID) control (Salih et al., 2010), adaptive control (Antonelli et al., 2018), and active disturbance rejection attitude control (Yang et al., 2018).

The PID control method is a common control method in practical applications of the quadrotor, and PID control acts as a baseline controller in many studies (Bouabdallah et al., 2004). However, since the anti-interference relies on its integral term, the control accuracy becomes very poor when the interference is not constant, and the interference is suppressed only after it is affected. It has become more and more difficult to achieve high-precision control of the quadruple drone. Moreover, the PID gain is selected through trial and error, and it is difficult to meet the dynamic performance requirements. Therefore, model-based control is more practical. Researchers have proposed many advanced control strategies, such as model predictive control (MPC) (Alexis et al., 2012), robust control (Lee, 2013), and sliding mode control (SMC) (Xu R and Ozguner, 2006; Xu B, 2018), to deal with the nonlinearity and uncertainty of the model. These methods have their strengths and weaknesses. For example, adaptive control estimates the unknown parameters in the model of a quadrotor; adaptive control relies on an accurate model of the controlled system, and the control performance will decrease when the system has large external interferences. For these model-based control strategies, the control performance is dependent on the accuracy and comprehensiveness of the preset dynamic model. Even if the uncertainty and disturbance are considered, the accuracy and comprehensiveness of the model cannot be achieved. If the model is too complicated, it will lead to more complex control strategies. In the past decade, the rapid development of artificial intelligence technology has affected the traditional control field. The intelligent flight control system based on machine learning is a hot research topic (Santoso et al., 2018), and it can be known from Yechiel and Guterman (2017) that good control performance is achieved even if the model is nonlinear and uncertain. We can also use deep artificial neural networks as approximators to reduce the noise (Miglino et al., 1995). The advantage of online learning methods is the ability to continuously learn aircraft dynamics in real time (Dierks and Jagannathan, 2010). Since online learning is based on experience to learn aircraft dynamics models, this directly leads to limited system performance when the flight system is operating in a new environment. It is also not advisable to train the aircraft control strategy based on the supervised learning method, mainly due to the inaccuracy of the sample data used for supervised training. The data is obtained using a PID control method similar to the aircraft model which is not accurate. The sample data is inaccurate and the cost of collecting data is high, which make the model based on supervised learning training not the optimal (Williams-Hayes, 2005; Bobtsov et al., 2016). To build a high-performance flight control system, it is necessary to find more suitable solutions.

An alternative to machine learning and optimization is reinforcement learning (RL) applied to continuous tasks, and the RL algorithm has proven to be a successful machine learning method for practical applications. The application of the RL algorithm to the quadrotor model flight control problem was first proposed by Waslander et al. (2005). They used the local weighted linear regression method to model the quadrotor as a Markov decision process. The final learned linear strategy has similar performance to the integral sliding mode controller. In recent years, due to the rise of deep learning, deep neural networks have been introduced into RL as value function approximators. This deep neural network based RL algorithm has outperformed human experts in complex scenes such as video games (Mnih et al., 2015) and board games (Silver et al., 2016). Some well-known deep RL algorithms, such as deterministic policy gradient (DPG) (Silver et al., 2014), deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016), trust region policy optimization (TRPO) (Schulman et al., 2015), and proximal policy optimization (PPO) (Schulman et al., 2017), show significant control performance in the continuous state action space (Duan et al., 2016). Hwangbo et al. (2017) used the RL algorithm to train the aircraft model, and successfully learned a deterministic strategy by mapping a given state to a set of aircraft executable action vectors through a neural network,

using a natural gradient descent method to optimize deterministic strategies (Amari, 1998). However, there are many differences between the real environment and the simulation environment. Direct application of the aircraft model learned from the simulation environment to the actual one will cause many problems, such as decreased accuracy and stability. It is necessary to increase the authenticity of the simulation, and even to place the quadrotor model in the real environment to train the learning control strategy.

In this work, we use an advanced RL algorithm, i.e., PPO, to train the controller. We introduce a state integrator in the actor critic framework to reduce the steady-state error by integrating the error state. This improves the accuracy of tracking control and the robustness of the controller. We compare the performance of the improved controller with that of the traditional PID controller. Experimental results show that the controller trained by the PPO algorithm with state integration is superior to the PID controller. We then adopt a two-stage learning mode to train the aircraft model. In the offline phase, we train a simplified aircraft model in the simulation to learn a controller with robustness. Then, we train a real quadrotor in the actual scene, and optimize the control strategies to build a high-performance flight controller in the online phase. In summary, the main contributions of this paper are as follows:

1. We propose a PPO with integral compensator (PPO-IC) algorithm by introducing a state integrator. The algorithm significantly improves the tracking accuracy in motion control.

2. We develop an offline and online learning scheme. We train a model with strong generalization ability in the offline phase, and optimize the control model in the online learning phase. The online learning phase is aimed to ensure safe and stable flight in practical applications of the quadrotor. The network weight in the online learning phase is not randomly initialized; the network parameters of the model that has been learned in the offline phase are used as the initial parameters, to ensure stable flight of the quadrotor at the beginning of the online learning phase. On this basis the networks can be learned and updated. An offline strategy is adopted so that if the quadrotor takes dangerous actions or some state components are outside the safe range during the online learning phase, the quadrotor can switch to the offline learning mode to ensure a safe and stable flight.

2 Background

In this section we introduce the dynamic model and basic principle of RL.

2.1 Dynamic model of the quadrotor

We set up two coordinate systems to describe the position and attitude of the quadrotor. The first is the inertial coordinate system, fixed on the earth, where Newton's laws of motion are applicable. The second is the body-fixed coordinate system, fixed on the quadrotor, with the origin being the center of mass of the quadrotor. The two coordinate systems are coincident initially, and then become different during the flight of the quadrotor. The inertial coordinate system remains unchanged during the flight; in contrast, the body-fixed coordinate frame will rotate and move in this process. Three Euler angles ϕ, φ, and ψ are used to describe the process and degree of rotation, which are around the X, Y, and Z axes, respectively. The center of mass of the quadrotor in the inertial coordinate system is defined as $\boldsymbol{P}=[x, y, z]^{\mathrm{T}}$. The velocity and acceleration of the quadrotor can be defined as $\dot{\boldsymbol{P}}$ and $\ddot{\boldsymbol{P}}$, respectively. The model of the quadrotor is shown in Fig. 1 (Rozi et al., 2017).

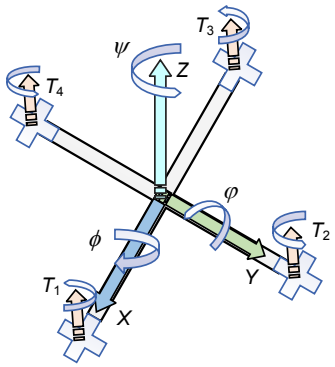

Fig. 1 The quadrotor model and body-fixed frame (Rozi et al., 2017)

The quadrotor has four rotors. Each rotor is distributed at the end of the cruciform frame. The distance between each rotor and the center of mass is L. Viewed from the forward Z axis, rotors 1 and 3 turn clockwise, while rotors 2 and 4 rotate counterclockwise. The speed of rotation is controlled by the pulse-width modulation (PWM) signals generated by the electronic speed controllers. Advanced electronic speed controllers ensure that the thrusts generated by

the four rotors are almost proportional to the PWM signals sent to the four rotors, that is,

$$T_i = Ku_i, \quad i = 1,\ 2,\ 3,\ 4. \tag{1}$$

Herein T_i (i=1, 2, 3, 4) represent the thrusts generated by the four rotors. u_i (i=1, 2, 3, 4) represent the PWM signals transmitted to the four rotors; u_i are normalized to [0, 1], with u_i=0 standing for zero thrust produced by the rotor and u_i=1 meaning that the rotor generates the maximum thrust. K represents the thrust gain. Then, we analyze the dynamic characteristics of the translational motion driven by force and the rotational motion driven by torque, and establish the dynamic model.

For translational motion characteristics, in the inertial coordinate system, according to Newton's second law, we have

$$\boldsymbol{F}_{\mathrm{e}} = \boldsymbol{R}\boldsymbol{F}_{\mathrm{l}} + \boldsymbol{F}_{\mathrm{d}} + \boldsymbol{G}, \tag{2}$$

$$\boldsymbol{F}_{\mathrm{e}} = m\ddot{\boldsymbol{p}}. \tag{3}$$

Here, we take the positive direction of the Z axis as the positive direction of the Z-axis force, and $\boldsymbol{F}_{\mathrm{e}}$ represents the combined external force. $\boldsymbol{F}_{\mathrm{l}}$ is the lift force vector, $\boldsymbol{F}_{\mathrm{l}}$=[0, 0, T_z]$^{\mathrm{T}}$, where T_z is the sum of upward lift forces generated by the four rotors relative to the quadrotor frame, $T_z = \sum_{i=1}^{4} T_i$, with T_i (i=1, 2, 3, 4) being the lift force generated by each rotor. $\boldsymbol{F}_{\mathrm{d}} = [-f_x\dot{x}, -f_y\dot{y}, -f_z\dot{z}]^{\mathrm{T}}$ is the air resistance, where f_x, f_y, and f_z are air drag coefficients in three directions. $\boldsymbol{G}$=[0, 0, $-mg$]$^{\mathrm{T}}$ is the gravity, where g is the acceleration of gravity. $\boldsymbol{R}$ is a transformation matrix, used to transform the lift force defined in the body-fixed coordinate system to the inertial coordinate system. As the position and velocity are defined in the inertial coordinate system, unity of the coordinate system is achieved with $\boldsymbol{R}$:

$$\boldsymbol{R} = \begin{bmatrix} \cos\phi\cos\varphi & \cos\phi\sin\varphi\sin\phi - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\varphi \\ \sin\psi\cos\varphi & \sin\phi\sin\varphi\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\varphi\sin\psi - \cos\psi\sin\phi \\ -\sin\varphi & \cos\varphi\sin\phi & \cos\phi\cos\varphi \end{bmatrix}.$$

For translational motion characteristics, the Euler equation of rotation is applied to a fixed quadrotor frame:

$$\boldsymbol{M} = \boldsymbol{M}_{\tau} + \boldsymbol{M}_{\mathrm{c}} + \boldsymbol{M}_{\mathrm{f}} = \boldsymbol{I}\dot{\boldsymbol{w}} + \boldsymbol{w} + \boldsymbol{I}\boldsymbol{w}, \tag{4}$$

where $\boldsymbol{M}$ is the sum of torques applied to the quadrotor, $\boldsymbol{w}$ is the angular velocity of the three axes of the quadrotor, $\boldsymbol{w} = [\dot{\phi}, \dot{\varphi}, \dot{\psi}]^{\mathrm{T}}$, $\boldsymbol{I}$ is the diagonal inertia matrix of the quadrotor, $\boldsymbol{I}$=diag(I_x, I_y, I_z), and $\boldsymbol{M}_{\tau}$=[τ_{ϕ}, τ_{φ}, τ_{ψ}]$^{\mathrm{T}}$ is the control torque of the quadrotor, which is the result of differential lift thrusts:

$$\boldsymbol{M}_{\tau} = \begin{bmatrix} \tau_{\phi} \\ \tau_{\varphi} \\ \tau_{\psi} \end{bmatrix} = \begin{bmatrix} L(T_2 - T_4) \\ L(T_1 - T_3) \\ K_{\psi}(T_1 - T_2 + T_3 - T_4) \end{bmatrix}. \tag{5}$$

Herein the control torques τ_{ϕ} and τ_{φ} are around the X and Y axes, respectively. The control torque τ_{ψ} is the result of lift thrusts of all the four rotors around the Z axis, with K_{ψ} being a coefficient. Because of the existence of the four rotating rotors, a gyroscope effect $\boldsymbol{M}_{\mathrm{c}}$=[$-I_{\mathrm{p}}\dot{\varphi}\Omega, I_{\mathrm{p}}\dot{\phi}\Omega, 0$]$^{\mathrm{T}}$ is generated, where I_{p} is the inertia moment of each rotor and Ω is the disturbance effect generated by each rotor. $\boldsymbol{M}_{\mathrm{f}}$ is the resistance moment suffered by the quadrotor during the flight, $\boldsymbol{M}_{\mathrm{f}} = [-d_{\phi}\dot{\phi}, -d_{\varphi}\dot{\varphi}, -d_{\psi}\dot{\psi}]^{\mathrm{T}}$, where d_{ϕ}, d_{φ}, and d_{ψ} are the three axial drag coefficients.

Finally, we obtain the nonlinear dynamic model in the following form:

$$\begin{cases} \ddot{x} = [T_z(\cos\phi\sin\varphi\cos\psi + \sin\phi\sin\psi) - d_x\dot{x}]/m, \\ \ddot{y} = [T_z(\cos\phi\sin\varphi\sin\psi - \sin\phi\cos\psi) - d_y\dot{y}]/m, \\ \ddot{z} = (T_z\cos\phi\cos\varphi - d_z\dot{z} - mg)/m, \\ \ddot{\phi} = [\tau_{\phi} - I_{\mathrm{p}}\dot{\varphi}\Omega - d_{\phi}\dot{\phi} + \dot{\varphi}\dot{\psi}(I_y - I_z)]/I_x, \\ \ddot{\varphi} = [\tau_{\varphi} - I_{\mathrm{p}}\dot{\phi}\Omega - d_{\varphi}\dot{\varphi} + \dot{\phi}\dot{\psi}(I_z - I_x)]/I_y, \\ \ddot{\psi} = [\tau_{\psi} - d_{\psi}\dot{\psi} + \dot{\phi}\dot{\varphi}(I_x - I_y)]/I_z. \end{cases} \tag{6}$$

2.2 Reinforcement learning and policy gradient

RL is learning how to map situations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, but must discover which actions yield the most reward. It uses the formal framework of Markov decision processes

(MDPs) to define the interaction between a learning agent and the environment concerning states, actions, and rewards (Sutton and Barto, 1998). The core idea of RL actually comes from psychology, where agents interact with the environment to obtain information about rewards and punishments and then improve their behavior to maximize the rewards. The environment is often modeled as an MDP, described by a four-tuple (S, A, $\boldsymbol{P}$, R), where S is the set of states, A the set of actions, $\boldsymbol{P}$ the state transition matrix, and R the reward function. In recent years, with the rapid development of deep learning, a new learning agent different from the traditional RL agents has emerged, which is usually called a deep RL agent. The cyclic process is shown in Fig. 2.

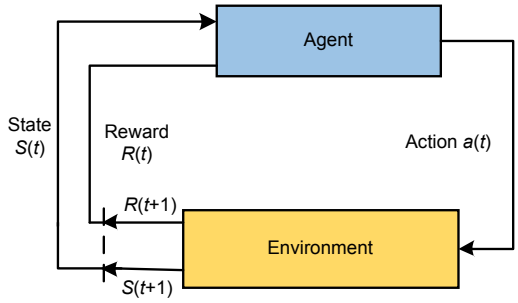

Fig. 2 The agent-environment interaction in a Markov decision process

It is well known that describing a task as MDP can better guarantee the resolution of RL tasks, which means that the future state of a Markov process depends only on the current state. That is, for any trajectory s_1, a_1, …, s_N, a_N in the state-action space, we have

$$P(s_{t+1}\,|\,s_1,a_1,...,s_t,a_t)=P(s_{t+1}\,|\,s_t,a_t). \quad (7)$$

The goal of RL is to train a strategy π which can map states to executable actions, and to continuously improve π by maximizing a goal reward function, which is generally set as $J(\pi)=E\left(\sum_{t=1}^{N}\gamma^{t-1}r_t\,|\,\pi\right)$ and is called the expected cumulative discount reward function, where the discount factor $\gamma\in[0, 1)$, r_t is the reward that the agent obtains after taking action a_t at state s_t. Policy π can be either stochastic $\pi(a|s)$ or deterministic $\pi(s)$, where $\pi(a|s)$ means that each action $a\in A(s)$ that an agent can perform at state s satisfies a probability distribution, and $\pi(s)$ directly maps a state s to a determined action a. In the literature, the steps of the RL algorithm can be summarized as follows: (1) generating samples by executing policy π; (2) evaluating the reward to generate a model; (3) improving policy π; (4) repeating this process until optimal parameter values are obtained.

There are many types of RL algorithms. The main approaches are value-function-based algorithms, such as SARSA (Sutton, 1995) and Q-learning (Watkins and Dayan, 1992). We consider the best action by estimating the return, and then repeatedly improve policy $\pi(s)$. Another classical RL algorithm is based on policy gradient. In policy gradient approaches, there are usually two networks, actor network and critic network. The actor network is used to select an appropriate action under the current state, unlike value-function-based methods which select the action with the largest value. Approaches based on policy gradient use the actor network to select the actions. The parameters of the network are a set of vectors that can be trained repeatedly. The critic network can be used to evaluate the value of the current action produced by the actor network. In actor-critic algorithms the parameters of the critic neural network can also be trained repeatedly.

In general, we use a gradient descent algorithm to optimize the policy weights. Policy $\pi_{\boldsymbol{\theta}}$ is commonly a deep or shallow neural network. From Schulman et al. (2017), the most commonly used form of the gradient approximator is

$$\hat{g}=\hat{E}\left[\nabla_{\boldsymbol{\theta}}\log_{\pi_{\boldsymbol{\theta}}}(a_t\,|\,s_t)\hat{A}_t\right], \quad (8)$$

where $\hat{A}_t$ is called the advantage function, $\hat{A}_t(s_t,a_t)=Q(s_t,a_t)-\hat{V}(s_t)$, representing the empirical average of a limited number of samples generated during the interaction between agents and the environment at each time step t. The estimator is obtained by differentiating the objective function:

$$L^{\mathrm{PG}}(\boldsymbol{\theta})=\hat{E}\left[\log_{\pi_{\boldsymbol{\theta}}}(a_t\,|\,s_t)\hat{A}_t\right]. \quad (9)$$

This approach can be applied to deep learning, and makes RL tasks more like general optimization problems (Schulman, 2016). In the RL problem, what we are trying to solve is an unknown dynamic model

that needs continuous learning and may even be non-differentiable. Such a property makes the variance of the gradient estimate increase when calculating the gradient. In most deep learning problems, we can know the target loss function and the parameters of the neural network, and the input data is not relevant to the current neural network. In contrast, in RL, the input data is the states of the agent, which are produced by the previous policy neural network and have strong correlation. This is harmful to the stability and convergence of the policy neural network. Taking these factors into account, we use PPO to optimize the policy in a more stable and efficient way and improve the performance of the RL agent. The main reasons are as follows: (1) PPO is considered to be a suitable RL algorithm for quadrotor control tasks; (2) PPO is known to outperform state-of-the-art methods in challenging environments. PPO is a policy gradient method and is similar to TRPO, while it is easier to implement and tune. It is therefore the default choice of algorithm in OpenAI's projects.

3 The proposed approach

In this section, the PPO-IC algorithm is proposed, and the control structure and implementation process are given for the quadrotor. A two-stage learning and training technique is proposed for the actual flight.

3.1 Network structure

The actor-critic network structure of the PPO-IC algorithm is shown in Fig. 3. Two types of neural networks are used for training. One is the actor neural network $\boldsymbol{\theta}$, which is composed of four policy sub-networks θ_i (i=1, 2, 3, 4), whose weights can be optimized by training. The other is the critic neural network. These two neural networks have the same network input; that is, a batch of new state vectors $[\dot{\phi},\dot{\varphi},\dot{\psi},\phi,\varphi,\psi,\dot{x}_c,\dot{y}_c,\dot{z}_c]$ are used as the network input, where $\dot{x}_c$, $\dot{y}_c$, $\dot{z}_c$ are the differences between the expected velocity and the current velocity after integration. The nine-dimensional state vector is fed to

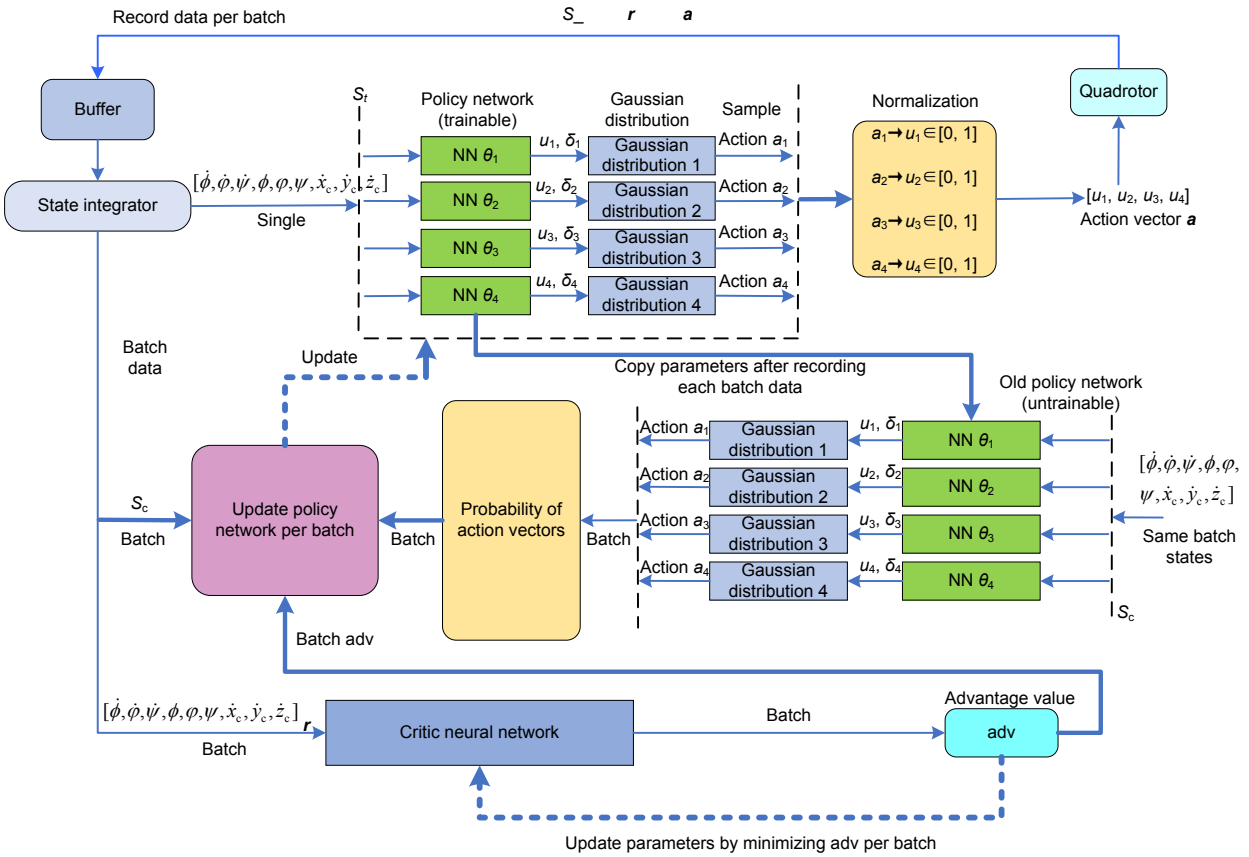

Fig. 3 System network framework structure

the four policy sub-networks at the same time. The outputs of the four sub-networks are u_i and δ_i (i=1, 2, 3, 4), which are the mean and variance of the four Gaussian distributions, respectively. For each Gaussian we sample a random action a_i (i=1, 2, 3, 4). Then each collected action is normalized to [0, 1]. The four actions constitute a group of actions a_i (i=1, 2, 3, 4), which would be used as the input to the four rotors, and the quadrotor changes to a new state. Since we use batch training, after the batch group action vectors are collected by the current policy network, we copy the network parameters of the current four policy sub-networks to another four networks, which we call the old policy networks. The parameters of the old network remain fixed, and then the four policy networks used to sample new actions are trained in the next batch training. We need to record the new states into a buffer. After the integration and compensation process, a batch of state vectors are fed to the critic neural network. A batch of advantage values can thus be generated to evaluate the quality of the actions taken to achieve these states. The critic neural network updates its parameters by minimizing these values through the gradient descent method. After the state vectors of the batch group are extracted from buffer and compensated for by integration, we feed the advantage values to the policy neural network together with the action vectors and the state vectors after integral compensation. Thus, the updating process of the policy neural network is completed.

The four sub-networks of the policy network have the same structure (Fig. 4). It takes the nine-dimensional state vector after integral compensation as the input. Such a network structure can maintain balance between the control performance and the training speed of the quadrotor. Each sub-policy network has two fully connected hidden layers, and each hidden layer has 128 hidden nodes with the sigmoid activation function. Finally, there are two outputs: one is the mean μ of the Gaussian distribution, and the tanh activation function is adopted in this output layer; the other is the standard deviation δ of the Gaussian distribution, and the softplus activation function is adopted in this output layer.

The structure of the critic neural network is similar to that of the policy sub-network (Fig. 5). The critic neural network also has two fully connected hidden layers, and each hidden layer has 128 hidden nodes with the sigmoid activation function. The final output is an advantage value used to evaluate how well a given action is taken in a given state. In fact, we do not try different numbers of nodes and layers. From our experience, both neural networks are versatile and can cope with a variety of problems with a single structure. We use the PPO-IC algorithm, which is helpful in reducing static errors. For details, refer to Fig. 3.

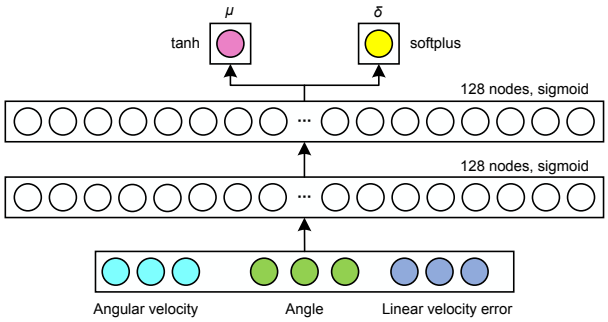

Fig. 4 The structure of four policy sub-networks used in this work

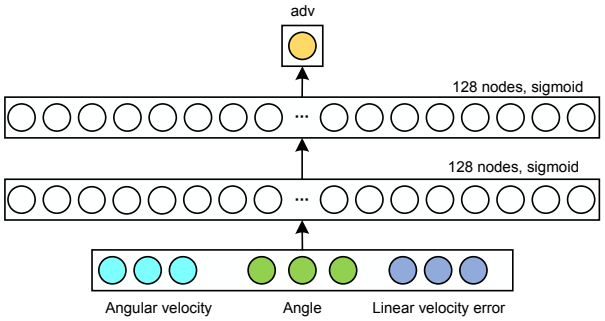

Fig. 5 The structure of the critic neural network used in this work

3.2 PPO-IC algorithm

The PPO-IC algorithm is developed based on the traditional PPO algorithm, adding an integral part to reduce the steady-state error, so as to achieve higher precision in quadrotor control and make the tracking error close to zero. PPO is model-free, on-policy, actor-critic, and is based on the policy gradient learning algorithm. It has reliable performance similar to the TRPO algorithm and is much easier to implement. When only first-order optimization is used, the improvement of monotonicity can be guaranteed by introducing the Kullback-Leibler (KL) divergence related to policy updates. In TRPO and PPO methods, the objective function is the following surrogate objective function L^{CPI} (conservative policy iteration)

proposed in Kakade and Langford (2002), and we want to maximize this objective function.

$$L^{\mathrm{CPI}}(\boldsymbol{\theta})=\hat{E}_t\left[\frac{\pi_{\boldsymbol{\theta}}(a_t\,|\,s_t)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(a_t\,|\,s_t)}\hat{A}_t\right]=\hat{E}_t\left[r_t(\boldsymbol{\theta})\hat{A}_t\right]. \quad (10)$$

Here, $\boldsymbol{\theta}_{\mathrm{old}}$ is the vector of policy parameters before the update. $r_t(\boldsymbol{\theta})$ is the probability ratio, in the form of $r_t(\boldsymbol{\theta})=\pi_{\boldsymbol{\theta}}(a_t\,|\,s_t)\big/\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(a_t\,|\,s_t)$, so $r_t(\boldsymbol{\theta}_{\mathrm{old}})=1$. In the absence of constraint conditions, maximizing the objective function of L^{CPI} will lead to excessive updating of the policy network, which is not conducive to network convergence and stability. Therefore, we need to punish the updating policy that keeps $r_t(\boldsymbol{\theta})$ away from 1. Although both TRPO and PPO algorithms use this surrogate objective function, they use different methods to constrain the update step size of the policy network. The TRPO algorithm optimizes the surrogate objective function by introducing KL divergence; that is, it is subject to a constraint $\hat{E}_t\left\{\mathrm{KL}\left[\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(\cdot\,|\,s_t),\pi_{\boldsymbol{\theta}}(\cdot\,|\,s_t)\right]\right\}\le\delta$ in the process of maximizing the objective function. By linear approximation of the objective function and quadratic approximation of the constraint condition, a conjugate gradient algorithm can be used to solve the problem. An alternative approach is to add penalty terms to the surrogate objective function to limit excessive updating of the policy network, which can also solve the unconstrained optimization problem.

The optimization method of the TRPO algorithm is described above. The PPO algorithm is similar to the TRPO algorithm, inheriting some advantages of the TRPO algorithm, and is much simpler to implement. The PPO algorithm is more universal and can take multiple optimization steps. From experience, it has a better sample utilization rate than TRPO. The new objective function proposed is the following:

$$L^{\mathrm{CPI}}(\boldsymbol{\theta})=\hat{E}_t\left[\min\left(r_t(\boldsymbol{\theta})\hat{A}_t,\mathrm{clip}\left(r_t(\boldsymbol{\theta}),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right)\right], \quad (11)$$

where $r_t(\boldsymbol{\theta})=\pi_{\boldsymbol{\theta}}(a_t\,|\,s_t)\big/\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(a_t\,|\,s_t)$ is the probability ratio, $\hat{A}_t$ represents the advantage of taking action a_t at state s_t, and ε is a hyperparameter (generally ε=0.2). There are two main terms. The second term in min(), $\mathrm{clip}\left(r_t(\boldsymbol{\theta}),1-\varepsilon,1+\varepsilon\right)\hat{A}_t$, is adjusted based on the surrogate objective function by clipping the probability ratio. This adjustment could limit the size of $r_t(\boldsymbol{\theta})$ to the set interval $[1-\varepsilon, 1+\varepsilon]$, that is, setting the values of $r_t(\boldsymbol{\theta})$ greater than $1+\varepsilon$ to $1+\varepsilon$, setting the values of $r_t(\boldsymbol{\theta})$ less than $1-\varepsilon$ to $1-\varepsilon$, and taking the value of $r_t(\boldsymbol{\theta})$ within $[1-\varepsilon, 1+\varepsilon]$ without adjustment. Finally, we take the minimum of these two terms (clipped and unclipped objective). Note that the probability ratio $r_t(\boldsymbol{\theta})$ is clipped at $1-\varepsilon$ or $1+\varepsilon$ depending on whether the advantage $\hat{A}_t$ is positive or negative. If $\hat{A}_t$ is positive, then $r_t(\boldsymbol{\theta})$ is clipped at $1+\varepsilon$; otherwise, $r_t(\boldsymbol{\theta})$ is clipped at $1-\varepsilon$.

Through the above adjustment, the PPO algorithm can avoid excessive updating of the policy network, and it is easier to train the policy neural network. Previous work showed that the PPO algorithm is very suitable for the control of continuous action space missions, and the learning strategy is very fast and stable. However, through the many experiments that we have conducted, it is found that the PPO algorithm will lead to obvious steady-state errors when dealing with tracking control tasks; no matter increasing the number of training rounds or using other return functions, the control strategy finally learned cannot eliminate this tracking error. An important reason is that the estimation of the action-value function is not accurate. Even in very simple control tasks, the exact action value cannot be obtained using the same action value function approximation strategy (Lillicrap et al., 2016), and for more complex control tasks, the estimation accuracy of the action-value function decreases greatly. The cause of the tracking error is the result of off-policy learning, TD-learning, and nonlinear function approximators. These factors will lead to a large deviation and variance in the estimation of the action-value function (van Hasselt, 2010; van Hasselt et al., 2016). For discrete action space control tasks, the Q-learning method can avoid this problem to some extent (Mnih et al., 2015). However, in the case of the continuous action control task, the update and optimization of the policy network function in the PPO algorithm is based on the advantage value generated by the critic neural network. The inaccurate estimation of the advantage value will lead to the updating and iteration of the policy network based on gradient updating in the

wrong direction, and thus cannot achieve accurate and effective tracking.

To solve the above problems, we improve the policy neural network and the critic neural network. The integrator is used to reduce the steady-state error. We know that the proportional-integral controller will generate a control deviation based on the given value and the actual output value, and form a control amount through the combination of the proportional deviation and integral deviation, so that the tracking error of the controlled object approaches zero. The state vector is extended by the linear quadratic integral control method, and the tracking error can be greatly reduced by calculating the optimal state feedback controller. However, increasing the dimension of the state vector will make it more difficult to train the policy network and the critic network. Therefore, considering various factors, we propose the PPO-IC algorithm to reduce the tracking error. The core idea is to introduce the error integral under the condition that the dimension of the state vector remains unchanged (Wang et al., 2019). By integrating the error state vectors of the previous time steps and combined with the current state vector, a new compensation state vector is formed:

$$\boldsymbol{s}_{\mathrm{c}}^{t}=\boldsymbol{s}_{\mathrm{e}}^{t}+\beta\sum_{i=1}^{t}\boldsymbol{s}_{\mathrm{e}}^{i}. \tag{12}$$

Here, β is the integral gain coefficient of the error state vector, $\boldsymbol{s}_{\mathrm{c}}^{t}$ is the state vector after compensation, and $\boldsymbol{s}_{\mathrm{e}}^{t}$ is the state vector for each time step. Superscripts represent time steps. The state vector after integral compensation is used as the input of the network. When there is a steady-state error, the input of the network can be adjusted to update the network towards the direction of error reduction, and finally the tracking error can be reduced.

To use the PPO-IC algorithm to deal with the quadrotor control task, we first need to choose the state observation quantity satisfying the Markov property. We choose three axial Euler angles and their corresponding angular velocities to describe the rotation motion, and the angular limit and failure conditions are set to avoid approaching a dangerous attitude during the flight. Then the translational motion is described using three axial linear velocity states. Although our final control object is position control, we still choose its speed state here, considering the following: First, the position value depends on the flight mission. For an indoor flight mission, a narrow operational space inevitably leads to a smaller position. However, for the outdoor wide sky space, the position value changes greatly due to the large operable space. After comparison, it is found that the position value may be different by several orders of magnitude. This makes it difficult to set the sampling range. Second, because of the different sampling ranges in different control tasks, the state space may be greatly increased, which is not only not conducive to strategy training and learning, but also more time consuming. Thus, we choose to track and control the speed along the three axes. Finally, the state vector is represented as $\boldsymbol{s}_t=(\dot{\phi},\dot{\varphi},\dot{\psi},\phi,\varphi,\psi,\dot{x}_{\mathrm{e}},\dot{y}_{\mathrm{e}},\dot{z}_{\mathrm{e}})$. Here $\dot{x}_{\mathrm{e}}=\dot{x}-\dot{x}_{\mathrm{d}}$, $\dot{y}_{\mathrm{e}}=\dot{y}-\dot{y}_{\mathrm{d}}$, and $\dot{z}_{\mathrm{e}}=\dot{z}-\dot{z}_{\mathrm{d}}$ represent the difference between the current velocity and the desired velocity. We use the speed state represented by the speed error to form part of the state vector, and design an integral compensator to compensate for the speed error. After error compensation, the state vector is sent to the neural network for training. This can better complete the control task and reduce the tracking error.

Considering the stability of flight and real-time error tracking, we should make the return signal as simple as possible. For the speed tracking task, to minimize the accumulated speed error in the shortest time, we design the following return function:

$$r_t=-\sqrt{\dot{x}_{\mathrm{e}}^2+\dot{y}_{\mathrm{e}}^2+\dot{z}_{\mathrm{e}}^2+\psi^2}, \tag{13}$$

where $\dot{x}_{\mathrm{e}}$, $\dot{y}_{\mathrm{e}}$, and $\dot{z}_{\mathrm{e}}$ are velocity errors between the expected velocity and the current velocity in the global velocity frame, and the sum of the three terms constitutes the velocity tracking error. In the above equation, ψ represents the heading angle. Since the quadrotor structure adopts the drive mode of four symmetrical rotors, it can fly at a constant speed or hover at a given heading angle. Here we need only to consider speed control, so we can simply set ψ to zero. It can be seen that the maximum value of the return function is zero. As shown in Fig. 3, the velocity error state is changed from $\dot{x}_{\mathrm{e}}$, $\dot{y}_{\mathrm{e}}$, and $\dot{z}_{\mathrm{e}}$ to $\dot{x}_{\mathrm{c}}$, $\dot{y}_{\mathrm{c}}$, and $\dot{z}_{\mathrm{c}}$ respectively through the integral compensator.

Then the new nine-dimensional state vector is sent to the neural network.

3.3 Two-stage learning and training scheme

Because the RL algorithm we use is model-free (that is, there is no prior information of any dynamic model of the given agent), we learn from experience through the interaction between the agent and the environment. This idea limits the application of quadrotor control. Because the quadrotor is inherently an unstable system, the parameters of the initial policy θ_i (i=1, 2, 3, 4) are randomly initialized and are unknown. We train the model in batches by sampling the state data, and finally the model learns the policy that meets the requirements. This process is learned by the velocity error. Since there is an error, the quadrotor cannot fly stably as expected. This will cause the aircraft to lose control or even fall. Therefore, to transfer the flight strategy learned by the simulation model to the actual flight, we propose the two-stage learning and training scheme.

This technique consists of two learning phases, offline learning phase and online learning phase. In the offline learning phase, we use a simplified model to learn a flight control strategy. It can also be understood as the simulation training phase of the model, which prepares for the control of the quadrotor during the online flight. In the online learning phase, we use the flight strategy learned in the offline phase as the initial one, and the quadrotor continues to optimize the strategy, while it also runs the offline flight strategy. Once the aircraft in the online phase reaches the limit condition, it immediately switches to the offline strategy to ensure that the quadrotor can always fly safely and steadily. Through this technique, we can simulate the learning process of the actual flight strategy as much as possible to ensure the safe and stable operation of the aircraft.

The following algorithm is used in the offline learning and training phase. First, the network parameters of the four policy sub-networks and the critic network are randomly initialized. In addition, a mini-batch-sized buffer E is defined to store the experience data. Then the model is loaded, and an episodic style algorithm is used to train the network parameters with PPO-IC. The simplified model has a maximum number of training rounds. In each episode, the initial state is randomly initialized. The random initialization states include three axial angular velocities, three angular attitudes, three linear velocities, and the expected linear velocities. To train the quadrotor safely and effectively and meet the actual needs, we set a safe range for each state component. These randomly initialized state components are generated within the safe range, and the maximum number of training steps is set for each episode. For each training step, we run the strategy network and record the experience data generated in each step into buffer E. When the empirical data in the buffer reaches the set minimum batch, all will be collected. Then, this batch of data will be used to train and update the policy network and the critic network using the gradient descent method, to make the network gradually converge and stabilize. At any given step, as long as there is a state component out of the safe range, the episode is terminated. This episode is regarded as a failed round. Note that in an episode of sampling, if the sampled data reaches the set minimum batch, the minimum batch of data collected in the previous steps will be used to repeatedly train the policy network and the critic network to update the network parameters, and then empty the buffer and resample the data; if the state has exceeded the safe range when the sampled data has not reached the set minimum batch size, the experience data collected in the previous steps will be cleared and the next episode will be entered, which leads to the waste of data collected and increases the training time. Selection of the minimum batch size is very important. If the size is too large, the network learning update is too slow; if the size is too small, the networks will easily oscillate and do not converge. We have conducted a large number of experiments, and find that the effect of learning every four batches is good. Of course, if the number of training steps in the episode reaches the maximum, this episode training will be suspended to the next episode. The speed tracking in the offline learning phase of the quadrotor is summarized in Algorithm 1.

This multi-round training algorithm has several advantages. First, randomly initializing the starting point of exploration is an effective way of learning because it can improve the ability to explore the state space (Sutton and Barto, 1998). Second, by setting a safe range for each state component, the model can learn and train more effectively and safely, which is of great significance for practical applications, and also facilitates the normalization of state components.

Algorithm 1 Offline learning with PPO-IC

Initialize:
Randomly initialize the weights of the four policy sub-networks θ_i (i=1, 2, 3, 4)
Randomly initialize the weights of the critic network
Initialize the mini-batch-sized buffer E
Load the simplified quadrotor dynamic model
for episode=1 to max_episode **do**
  Initialize the state integrator (clear all integrators)
  Randomly initialize the quadrotor states
  Randomly initialize the desired velocity
  Observe the initial quadrotor states S_1
  for time step t=1 to max_step **do**
    Run policy $\boldsymbol{\theta}$ with state S_t to generate action a_t
    Run a dynamic model with control signal a_t
    Observe reward r_t and the next state S_{t+1}
    Process velocity errors with the state integrator
    Store transition (S_t, a_t, r_t, S_{t+1}) into mini-batch-sized buffer E
    if the data in buffer reaches mini-batch **then**
      Sample mini-batch group experience from the buffer
      Estimate the batch group advantages from the above experience: $\hat{A}_t=\sum_{t'>t}\gamma^{t'-t}r_{t'}-V_{\phi}(s_t)$
      $\pi_{\text{old}}\leftarrow\pi_{\boldsymbol{\theta}}$
      for update step i=1 to max_update_step **do**
        $J_{\text{ppo}}(\boldsymbol{\theta})=\hat{E}_t[\min(r_t(\boldsymbol{\theta})\hat{A}_t,\text{clip}(r_t(\boldsymbol{\theta}),1-\varepsilon,1+\varepsilon)\hat{A}_t)]$ and $r_t(\boldsymbol{\theta})=\dfrac{\pi_{\boldsymbol{\theta}}(a_t\mid s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t\mid s_t)}$
        Update $\boldsymbol{\theta}$ w.r.t. $J_{\text{ppo}}(\boldsymbol{\theta})$
      end for
      for update step j=1 to max_update_step **do**
        $L(\phi)=\sum_{t=1}^{T}\left(\sum_{t'>t}\gamma^{t'-t}r_{t'}-V_{\phi}(s_t)\right)^2$
        Update ϕ w.r.t. $L(\phi)$
      end for
      Empty buffer E
    end if
    if S_{t+1} exceeds the safe range **then**
      Empty buffer E
      break
    end if
  end for
end for

In the offline learning phase, i.e., the simulation training phase, we use a highly simplified four-rotor model to learn and train the flight strategy. All the aerodynamic drag, external disturbances, and gyroscope effects are ignored, but the most basic four-rotor dynamics model is used. The advantage is that the generalization ability and robustness of the simplified model are strong enough to ensure the most basic normal flight control of the model in the online learning stage. The simplified dynamic model equations are as follows:

$$\begin{cases}\ddot{x}=T_z(\cos\phi\sin\varphi\cos\psi+\sin\phi\sin\psi)/m,\\ \ddot{y}=T_z(\cos\phi\sin\varphi\sin\psi-\sin\phi\cos\psi)/m,\\ \ddot{z}=(T_z\cos\phi\cos\varphi-mg)/m,\\ \ddot{\phi}=L(T_2-T_4)/I_x,\\ \ddot{\varphi}=L(T_1-T_3)/I_y,\\ \ddot{\psi}=K_{\psi}(T_1-T_2+T_3-T_4)/I_z.\end{cases}\tag{14}$$

After the offline learning phase, we obtain a robust quadrotor model which can ensure the stable and safe flight of the aircraft. However, in actual flight control, to improve the flight performance, it is necessary to go through the online learning phase to continue the learning and training.

The purpose of the online learning phase is to ensure safe and stable flight in practical applications. The network structure and network parameters in this phase are the same as in the offline learning phase. The difference lies in the training method. First, the network weight of the online learning phase is not randomly initialized. The network parameters of the model that has been learned in the offline phase are used as the initial parameters, to ensure a stable flight at the beginning of the online learning phase. The networks can be learned and updated on this basis. Second, the offline learning phase takes multiple rounds of exploration training to obtain a stable model. In the online learning phase we do not use multiple rounds of repeated training, but a continuous flight process. While learning new strategies, we also run an offline strategy so that if a dangerous action is taken or some state components are outside the safe range during the online learning phase, the quadrotor can switch to the offline learning mode to ensure safe and stable fly. We need only to randomly initialize a state, let the quadrotor learn in the above way, and finally achieve a stable and safe flight strategy.

4 Experiments and results

In this section, we evaluate the speed tracking accuracy of a neural network based attitude flight controller using RL training. The execution details of

the above mentioned methods, the experimental parameters used, and the final results will be presented. For episodic tasks, they are comparable to those of the baseline (PID).

4.1 Setup of simulation

The dynamics model used for simulation is as shown in Section 2.1, which is a Qball-2 quadrotor designed by Quanser Inc. Some basic parameters such as size, mass, and moment of inertia are known from the official manual. Other parameters such as aerodynamic coefficients refer mainly to the multirotor aircraft performance evaluation article (Shi et al., 2017; http://www.flyeval.com/). The parameters of the simplified quadrotor model are shown in Table 1. In each episode of training, the maximum number of sampling steps is 500, and the sampling time per step is 0.01 s. Due to the small batch training method, the data sampled every two steps is sent to the network for learning and updating once. In each episode the largest step size is 250. It allows the aircraft to have sufficient time to respond to commands and explore as many unknown states as possible to learn more strategies. The range of the target angular velocity is set to $\Omega_{\min}=-2.27$ rad/s, $\Omega_{\max}=2.27$ rad/s (±130 deg/s), the range of the target sampling attitude angle is set from −45° to 45°, and the range of the target sampling linear velocity is set to $V_{\min}=-5$ m/s, $V_{\max}=5$ m/s. These limits are set by examining the performance of the PID to ensure physically feasible constraints while meeting the safety requirements of the actual flight of the quadrotor. In this study four policy sub-networks and one evaluation network are run at the same time, which is equivalent to paralleling four PPO-IC algorithms. Four policy sub-networks are used to generate the actions required by the four rotors respectively, which is better than using only one network, because this can avoid mutual interference between actions, and it is more conducive to stable generation of actions and network convergence. Training and evaluations are run on the Ubuntu 16.04 operating system with an Inter eight-core i7-8550U CPU and an NVIDIA GeForce MX150 graphics card. The simulated aircraft is developed with Python, the networks for learning and training are built with TensorFlow (Abadi et al., 2016) using Python, and the algorithm for optimizing the networks is the Adam optimizer (Kingma and Ba, 2014).

4.2 Offline learning phase evaluation and test

In the offline learning phase, the PPO-IC algorithm is used. The training parameters of the algorithm are shown in Table 2. The learning rate of the critic network is larger than that of the four policy sub-networks, which is reasonable. First, from the experimental point of view, it is very effective. Second, because the critic network is used to judge the current action and then generate the advantage value to update the policy network, the current critic network is required to judge the action as accurately as possible. Hence, having a higher learning rate for the critic network can improve the evaluation ability of the critic network.

Table 1 Parameters used in the simulation

Parameter	Meaning	Value
m	Mass	1.79 kg
L	Rotor to center distance	0.2 m
g	Gravity acceleration	9.81 m/s^2
K	Thrust gain	8.78
K_ψ	Reaction torque gain	0.4
J_p	Propeller moment of inertia	0.002 kg·m^2
I_x	X-axis moment of inertia	0.04 kg·m^2
I_y	Y-axis moment of inertia	0.04 kg·m^2
I_z	Z-axis moment of inertia	0.03 kg·m^2
d_x	X-axis air resistance coefficient	0.01
d_y	Y-axis air resistance coefficient	0.01
d_z	Z-axis air resistance coefficient	0.02

Table 2 PPO-IC algorithm parameters

Parameter	Value
Reward discount factor γ	0.9
Maximum number of episodes	6000
Learning rate for actor	0.0001
Learning rate for critic	0.0002
Minimum batch size for updating	2
Loop update operation	10
ε for the clipping surrogate objective	0.2
Sampling time per step	0.01 s
Integral gain factor β	0.01
Maximum number of steps in an episode	500
PPO-IC parallel number	4

4.2.1 Training performance evaluation

There are two indicators for evaluating the effect of learning and training at this phase: (1) average

accumulated reward and (2) average steady-state error. There is a negative correlation between these two indicators. In each step, the larger the return value returned, the smaller the error of the current state from the expected state. The process of quadrotor learning is toward the direction of the error becoming smaller and smaller. A maximum return value of zero means that the current state is exactly the desired one; however, because an error inevitably exists during the learning and training process, the ideal state will never be reached. It is only possible to approach the ideal state as much as possible. With the error in an acceptable range, it can still fly safely and stably. Hence, a larger accumulated reward value means a faster, more stable, and more accurate policy network. In this study, we train the model 50 times, and in each experiment 6000 episodes are performed to record the error and return the value generated by each step of the training process. The errors are based on the difference between the current velocities and the expected velocities and then the absolute values of the errors are summed. When the 50 experiments are completed, the average accumulated reward and the average steady-state error are calculated based on the 50 sets of data recorded previously. The algorithm used in this work is PPO-IC. One of its significant advantages is to reduce the steady-state error. We make an experimental comparison to highlight this advantage. The results of the PPO-IC algorithm are compared with those of the PPO algorithm (baseline). The results are shown in Fig. 6.

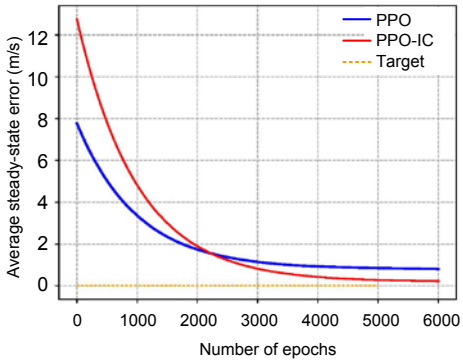

Fig. 6 Averaged steady-state error in the evaluation of polices learned by PPO-IC and PPO

The network structure and algorithm and simulation model parameters are the same. The initial network parameters are randomly initialized, and the state generated is uncertain. Therefore, the two errors at the beginning of the training are different. As the number of training rounds increases, for both algorithms the error gradually decreases to zero, and both are stable in the many experiments. However, the PPO-IC algorithm approaches zero faster, and the error is smaller. Moreover, the PPO algorithm always has an error after training a certain number of rounds, and the error does not decrease with the increase of the number of training rounds. This does not appear for the PPO-IC algorithm.

The comparison of the average cumulative reward is shown in Fig. 7. The network structure and the model and algorithm parameters are the same. The initial weights of the networks are randomly initialized. Because of the large deviation between the current state and the target state at the beginning of training, the reward values returned are different and very small. With the increase in the number of training rounds, the reward value is getting larger and larger. The PPO-IC algorithm has a higher convergence speed, and the network is faster and more accurate and more stable than the network trained with the PPO algorithm. The training times of the two algorithms are both around 136 s. The PPO-IC algorithm does not show an advantage in reducing the training time. However, this training time is much less than those of other deep RL algorithms in controlling the quadrotor. As mentioned in Hwangbo et al. (2017), using a high-performance GPU graphics card, it still takes at least 10 min to learn a control strategy that meets the requirements.

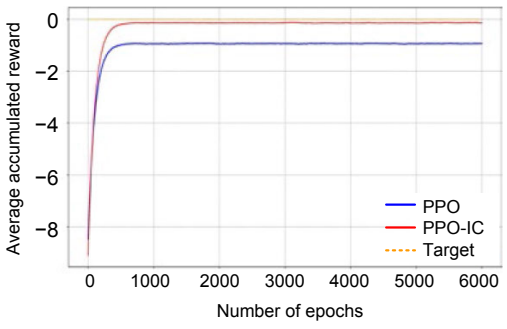

Fig. 7 Averaged accumulated reward in the evaluation of polices learned by PPO-IC and PPO

We use the PPO-IC and PPO algorithms to train the simplified model 50 times, training 6000 rounds per experiment, and learning the final control strategy by tracking the velocities.

In the 50 experiments, we set the target speed to be [1, −1, 1] m/s, and record the velocity tracking during the 6000 iterations. The results are shown in Fig. 8. We calculate the mean absolute steady-state errors of the two algorithms in the steady state that meet the accuracy requirements. In this experiment, we believe that the steady state is reached after 5000 rounds of training, after which 1000 rounds of training data are used to calculate the mean absolute steady-state error. The comparison diagram is shown in Fig. 9.

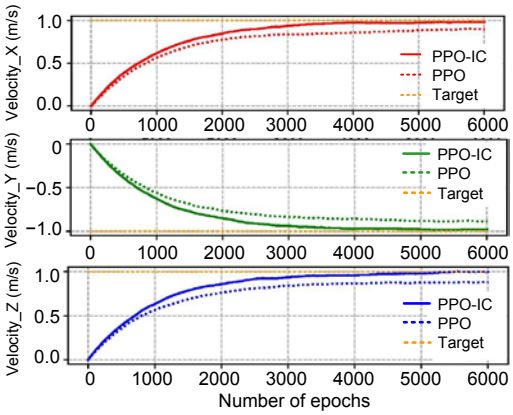

Fig. 8 Velocity tracking curves during the learning and training of PPO-IC and PPO

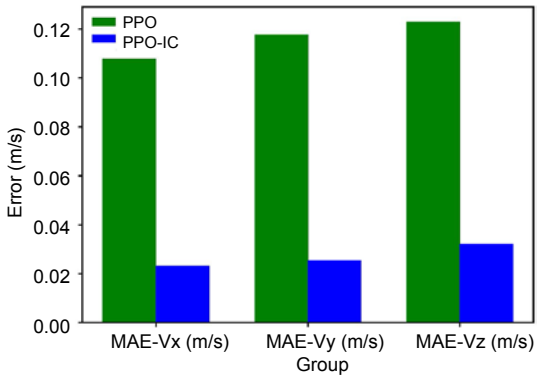

Fig. 9 Velocity errors at the steady state using the control polices learned by PPO-IC and PPO

The models of these two algorithms have zero initialization velocities, and the models start training from zero velocity. The dynamic performances of the two algorithms are shown in Fig. 8. Both algorithms can eventually learn a stable strategy, but the PPO-IC algorithm achieves higher tracking accuracy and is more conducive to reducing the steady-state error than the PPO algorithm for the quadrotor velocity tracking task. In addition, we compare the attitude angle changes learned by the two algorithms. The attitude angles are not initialized to zero, but are randomly initialized within the set safety range, so at the beginning of the training, the attitude angles are different. As the number of training rounds increases, the three attitude angles learned by the two algorithms eventually approach zero. The results of the two algorithms are shown in Fig. 10.

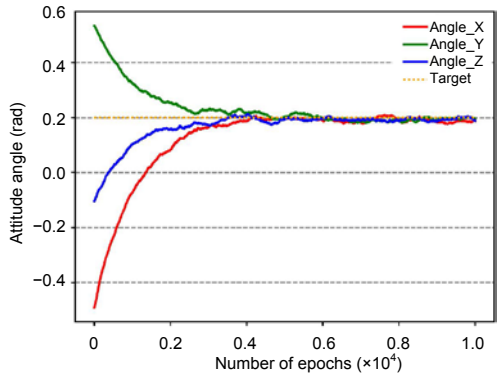

Fig. 10 Attitude angle curves by PPO and PPO-IC

To highlight the advantage of the PPO-IC algorithm in reducing the error, we compare the average absolute errors between the attitude angles learned by the two algorithms and the ideal zero state at the steady state. As shown in Fig. 11, the control performance of the PPO-IC algorithm is better than that of the PPO algorithm.

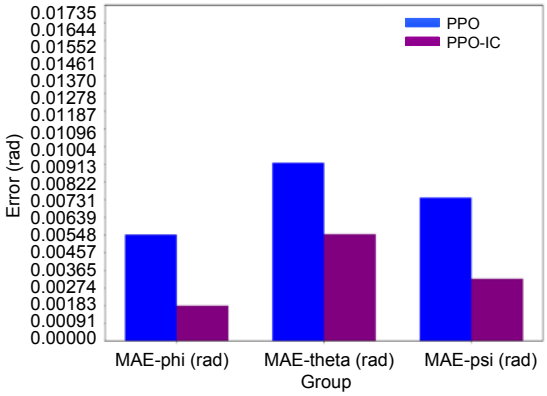

Fig. 11 Angle errors at the steady state using the control policies learned by PPO-IC and PPO

4.2.2 Simulated quadrotor model test

In the offline learning phase, the simplified model finally learns a stable control strategy through

a large number of rounds of trial-and-error training. We have conducted a test on the final model that has been learned. In the experiment, we show a two-dimensional plane flight path diagram (Fig. 12). The slope represents the flight speed of each axis. The slopes of the three axes are 1, −1, and 1, respectively, coinciding with the preset expected speed [1, −1, 1] m/s, which verifies that the model we have trained is effective. Because the quadrotor uses the PPO-IC algorithm to track the target velocity, we set the target velocity to [1, −1, 1] m/s. The model learns a steady flight speed which is very close to the set target speed. With this model, the quadrotor flies from the original position [0, 0, 0] for 10 s, and the trajectory is recorded. Fig. 13 shows the flight path of the aircraft from three dimensions. Taking the flight distance along the *X* axis as an example, since the expected speed along the *X* axis is 1 m/s, the distance is 10 m after 10 s of flight. The *Y*- and *Z*-axis flight analysis is similar.

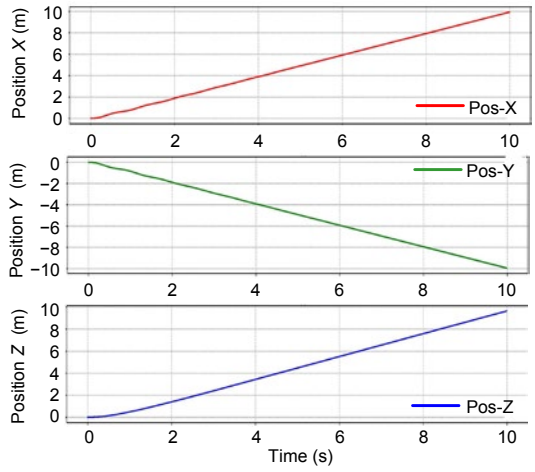

Fig. 12 Ten-second two-dimensional flight trajectory

4.3 Generalization ability test and comparison with PID

The quadrotor offline learning phase is to learn a stable, robust control strategy that allows the real four-rotor model to cope with differences in the actual flight. Through experimental demonstration, we believe that a powerful model is learned. In this subsection we test the generalized ability of the trained model. To do a comprehensive generalization test, we set up three different types of robustness testing for comparison with PID control.

4.3.1 Model generalization test of different sizes

We change the load and distance from the four rotors to the centroid to test the robustness and generalization ability of the control strategy learned in the offline phase of different sizes. The robust comparison is done by letting the model track the preset desired velocity. The specific implementation process of the control task is as follows: The initial flight velocity vector is set to [0, 0, 0]. Then the simulated aircraft flies for 10 s with the flight strategy learned during the offline phase and the speed curve is recorded. It is found that the quadrotor of different sizes converges to the desired speed [1, −1, 1] m/s set in offline learning according to the control strategy that has been learned in the offline phase. We use a traditional PID controller for comparison. The PID parameters have been set using the Qball-2 model (Quanser, 2015). Under these parameters, the aircraft can fly quickly and stably in the actual flight. We use the PID controller instead of advanced control methods for comparison, due mainly to the following reasons: First, the PID controller repeatedly adjusts the output by the error between the desired input and the actual output of the quadrotor state to achieve a given desired input value. This is similar to the PPO-IC algorithm; both use the trial-and-error strategy to achieve the control target. Second, the PPO-IC method is a model-free learning algorithm, but the design of other control methods is based on the analysis of the dynamics of the model, so it is not comparable to other control methods. In the robustness test, we select the sum of the absolute errors generated in each step during the flight to evaluate the performance. Obviously, the smaller the sum of errors, the more precise and faster the control strategy.

We perform two sets of experiments to test the control performance of the proposed PPO-IC algorithm under different quadrotor specification models, and compare the results with those of PID control to verify the generalization ability of the model. One type of experiments is conducted under different payloads. The results are shown in Fig. 14. Considering that the load should have the same effect on the four rotors, the payload is added to the centroid of the four rotors. As shown in Fig. 14c, the weight of the payload added at the centroid of the quadrotor is gradually increased. The quadrotor is not loaded from the beginning; that is, it is flying at the standard

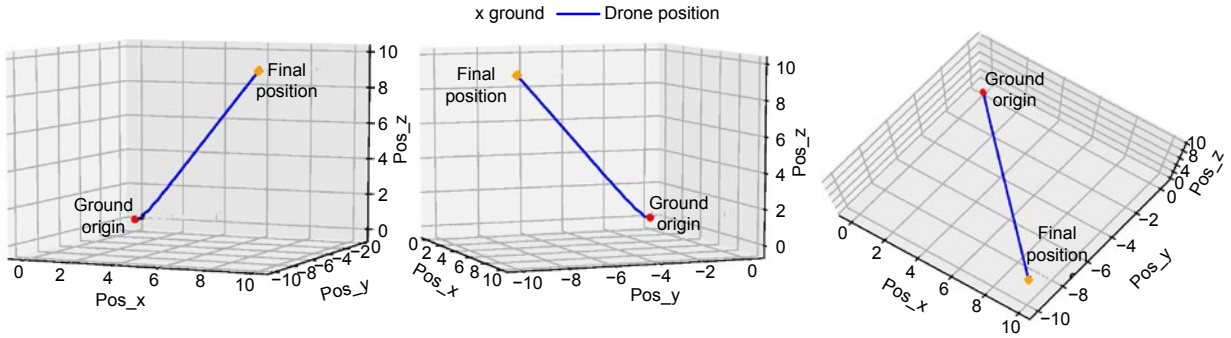

Fig. 13 Ten-second three-dimensional flight trajectory with a well-learned drone model

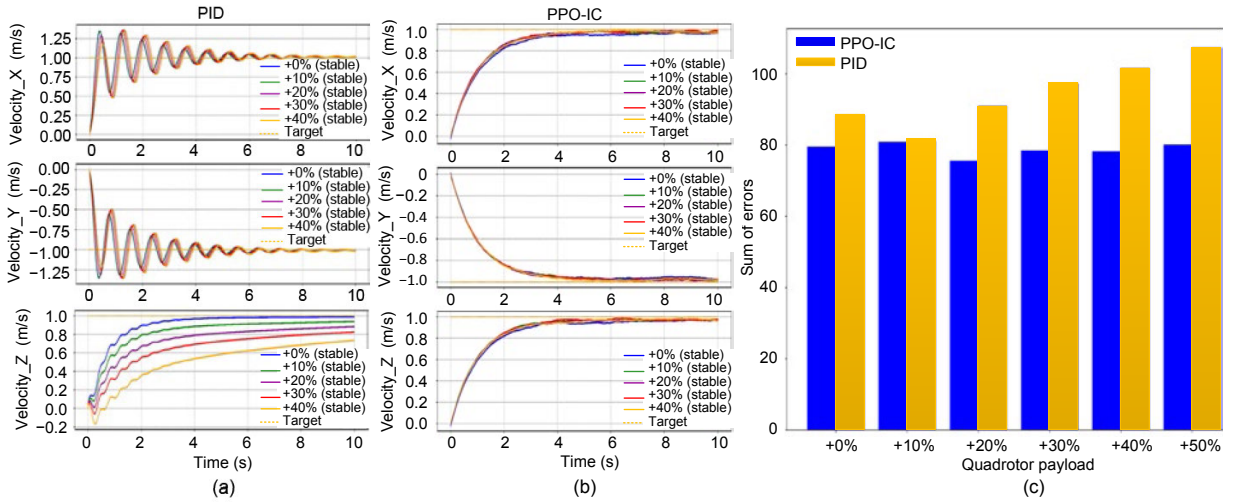

Fig. 14 Comparison of the control performance of the quadrotor model with the PPO-IC algorithm and the PID controller with different payloads: (a) three-dimensional velocity tracking response curve with the PID controller; (b) three-dimensional velocity tracking response curve with the PPO-IC algorithm; (c) sum of the errors under different payloads

weight of the offline learning phase. Then 10% of the standard weight is used for each load increase, and the payload increase eventually reaches 50% of the standard weight. The weight of the whole quadrotor is increased due to the load on the center, which has some influence on the moment of inertia of the model. New inertia moment parameters for different payload models can be estimated by the online evaluation package (Shi et al., 2017). Under the premise of keeping the air resistance coefficient of the model and the maximum trust of the four rotors unchanged, we conduct six sets of experiments. In each set of experiment only the quadrotor payload weight is changed. The aircraft starts flying at a zero initial velocity for 10 s in each experiment, and the speed tracking response curve is recorded (Figs. 13a and 13b). Because the expected speed we set is [1, −1, 1] m/s, the speed will eventually converge to the given target value in different experiments. Both the PPO-IC algorithm and the PID control method achieve the control requirements under the condition of increased load. The quadrotor controlled by the two algorithms achieves stable flight attitude and flight speed under different loads. However, Fig. 14a shows that although the quadrotor finally achieves a steady state by PID control under different loads, as the load increases, this final steady state deviates much from the desired one, and the quadrotor responds more slowly to the desired speed tracking. The increase of the payload has little effect on the speed tracking control of the X and Y axes, but the influence on the Z axis is obvious. The larger the payload on the quadrotor increases, the farther the final linear velocity of the Z axis deviates from the desired one. The main reason is that, as the weight of the payload increases, most of the upward thrust balances the increased gravity due to the increased payload, and only a small portion of the thrust is used to adjust the

attitude and speed of the aircraft, which directly leads to slower speed tracking response. This is verified by Fig. 14c. As the load increases, the speed error of the PID control method is larger and larger. In fact, the error increase is derived mainly from the speed deviation increment along the Z axis. In sharp contrast, the flight performance of the model under the control of the PPO-IC algorithm is very stable, and the speed tracking response curve is not subject to large fluctuations with the increase of the load. The quadrotor is still able to reach the desired speed quickly and accurately. The sum of the final speed errors of the models under different payloads is almost constant, even if the payload is increased by 50%. The speed tracking task has no large error, in sharp contrast with the large deviation in case of PID control. This proves the robustness of the model learned with PPO-IC.

In the offline learning phase, the radius of the quadrotor is 0.2 m. We take this as the standard radius. We select the other sets of models with different radii, from 0.1 m to 1.1 m (Fig. 15). That is, the radius is increased from 50% to 550% of the standard, which is a wide range, and a total of 12 experiments are conducted. In these experiments, only the radius is changed. The mass of the quadrotor model and the maximum thrust of the four rotors remain unchanged. The change in the radius directly results in a change in the inertia moment and the air resistance coefficient. These new parameters can be estimated by the online evaluation package (Shi et al., 2017). When the radius is relatively small, from 0.2 to 0.4 m, the control performances of the PPO-IC algorithm and the PID control method are similar. The quadrotor can complete the speed tracking control task quickly and stably. However, as the radius continues to increase, the PID-controlled quadrotor model becomes slower and slower in response to speed tracking, and the final state deviates more and more from the desired setting state. When the radius is increased to 0.8 m, although the steady state can be finally achieved, there are obviously some fluctuations compared with the steady state at small radii. When the radius is increased to 1 m and above, the final oscillating state of the model controlled by PID is unstable. This is verified by Fig. 15c. In contrast, the response of the model controlled by the PPO-IC algorithm to speed tracking is not much affected. During the change of the radius from 0.2 to 1.0 m, the model controlled by the PPO-IC algorithm is still fast, accurate, and stable. Note that with the increase of the radius, the control performances of the quadrotor models controlled by the two algorithms degrade; the performance degradation of PID control is more obvious.

To conclude, the quadrotor model learned by the PPO-IC algorithm in the offline phase shows robustness in the tests with different payloads and radii separately. Although the model based on PID control has good and even better generalization ability, overall PPO-IC control is superior to PID control.

4.3.2 Model generalization test in different initial states

We also test the control performance of the four-rotor model with the PPO-IC algorithm in different initial states. The control task is to make the

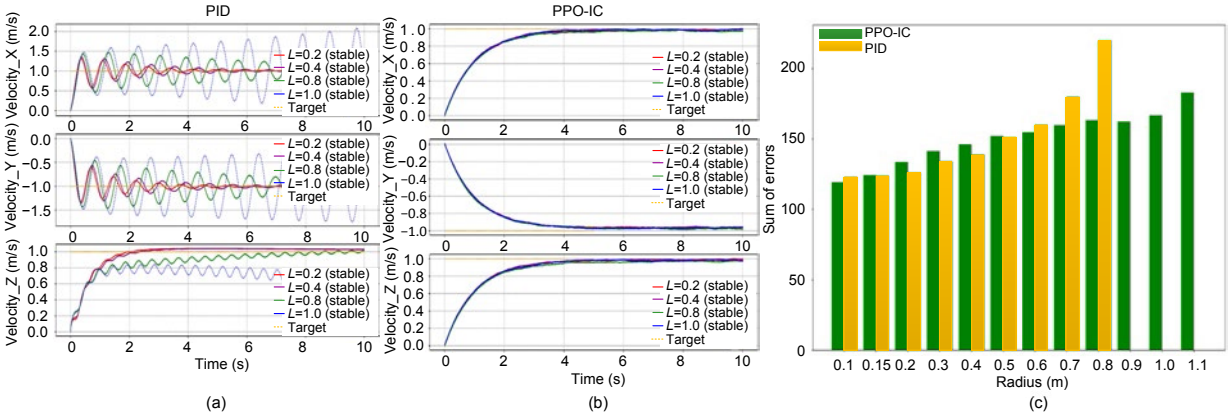

Fig. 15 Comparison of the control performance of the quadrotor model with the PPO-IC algorithm and the PID controller of different sizes: (a) three-dimensional velocity tracking response curve with the PID controller; (b) three-dimensional velocity tracking response curve with the PPO-IC algorithm; (c) sum of the errors of different sizes

If a model of a certain size fails to reach a steady state, the sum of errors is not shown in (c)

four rotors run in different initial states and finally reach the desired steady state. The attitude angles of the four rotors and the linear velocities along the three axes are randomly initialized from the respective safety ranges. We perform each experiment 20 times. Each experiment runs for 10 s. The three linear velocities start from different values and finally converge to the desired one (Fig. 16a). The three attitude angles start from different initial values and finally converge (Fig. 16b). The control strategy learned by the PPO-IC algorithm can drive the four-rotor model with different initial states to reach a stable state within 10 s, and the error is very small, which shows that the strategies learned in the offline phase have good generalization ability.

5 Conclusions and future work

We have proposed a PPO-IC algorithm with state integration for the development of the UAV intelligent controller, which effectively reduces the steady-state error in speed tracking and significantly improves the tracking accuracy. This method, together with the proposed reward signal, provides good sample efficiency and reduces the convergence time. A two-stage learning program has been proposed to develop a high-performance flight controller.

In future work, we would explore new reward signals to reduce the observed steady-state errors. Various initialization strategies can be considered to improve the flight controller performance.

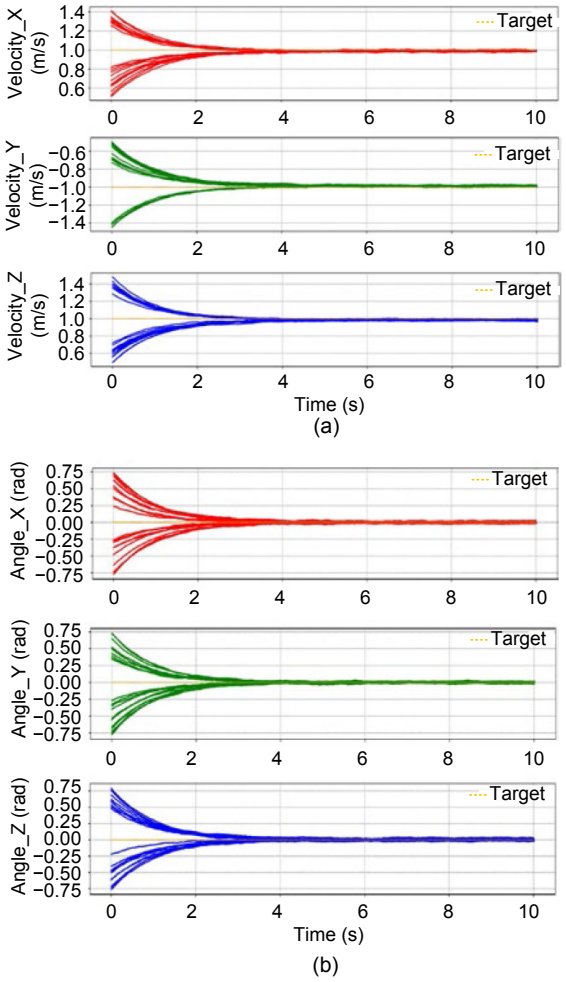

Fig. 16 PPO-IC control performance test in 20 different initial states: (a) linear velocity; (b) attitude angle

Contributors

Qing-ling WANG guided the research. Huan HU performed the experiments, drafted, revised, and finalized the paper.

Compliance with ethics guidelines

Huan HU and Qing-ling WANG declare that they have no conflict of interest.

References

Abadi M, Barham P, Chen JM, et al., 2016. TensorFlow: a system for large-scale machine learning. Proc 12th USENIX Conf on Operating Systems Design and Implementation, p.265-283.

Alexis K, Nikolakopoulos G, Tzes A, 2012. Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Contr Theory Appl*, 6(12):1812-1827. https://doi.org/10.1049/iet-cta.2011.0348

Amari SI, 1998. Natural gradient works efficiently in learning. *Neur Comput*, 10(2):251-276. https://doi.org/10.1162/089976698300017746

Antonelli G, Cataldi E, Arrichiello F, et al., 2018. Adaptive trajectory tracking for quadrotor MAVs in presence of parameter uncertainties and external disturbances. *IEEE Trans Contr Syst Technol*, 26(1):248-254. https://doi.org/10.1109/TCST.2017.2650679

Bobtsov A, Guirik A, Budko M, et al., 2016. Hybrid parallel neuro-controller for multirotor unmanned aerial vehicle. Proc 8th Int Congress on Ultra Modern Telecommunications and Control Systems and Workshops, p.1-4. https://doi.org/10.1109/ICUMT.2016.7765223

Bouabdallah S, Noth A, Siegwart R, 2004. PID vs LQ control techniques applied to an indoor micro quadrotor. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.2451-2456. https://doi.org/10.1109/IROS.2004.1389776

Dierks T, Jagannathan S, 2010. Output feedback control of a quadrotor UAV using neural networks. *IEEE Trans Neur Netw*, 21(1):50-66. https://doi.org/10.1109/TNN.2009.2034145

Duan Y, Chen X, Houthooft R, et al., 2016. Benchmarking deep reinforcement learning for continuous control. Proc 33rd Int Conf on Machine Learning, p.1329-1338.

Fumagalli M, Naldi R, Macchelli A, et al., 2012. Modeling and control of a flying robot for contact inspection. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.3532-3537. https://doi.org/10.1109/IROS.2012.6385917

Hwangbo J, Sa I, Siegwart R, et al., 2017. Control of a quadrotor with reinforcement learning. *IEEE Robot Autom Lett*, 2(4):2096-2103. https://doi.org/10.1109/LRA.2017.2720851

Kakade S, Langford J, 2002. Approximately optimal approximate reinforcement learning. Proc 19th Int Conf on Machine Learning, p.267-274.

Kingma DP, Ba J, 2014. ADAM: a method for stochastic optimization. https://arxiv.org/abs/1412.6980

Lee T, 2013. Robust adaptive attitude tracking on SO(3) with an application to a quadrotor UAV. *IEEE Trans Contr Syst Technol*, 21(5):1924-1930. https://doi.org/10.1109/TCST.2012.2209887

Lillicrap TP, Hunt JJ, Pritzel A, et al., 2016. Continuous control with deep reinforcement learning. https://arxiv.org/abs/1509.02971

Miglino O, Lund HH, Nolfi S, 1995. Evolving mobile robots in simulated and real environments. *Artif Life*, 2(4):417-434. https://doi.org/10.1162/artl.1995.2.4.417

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Quanser, 2015. User Manual Qball 2 for QUARC: Set Up and Configuration. Quanser, Inc., Markham, ON, Canada.

Rozi HA, Susanto E, Dwibawa IP, 2017. Quadrotor model with proportional derivative controller. Proc Int Conf on Control, Electronics, Renewable Energy and Communications, p.241-246. https://doi.org/10.1109/ICCEREC.2017.8226676

Salih AL, Moghavvemi M, Mohamed HAF, et al., 2010. Flight PID controller design for a UAV quadrotor. *Sci Res Essays*, 5(23):3660-3667.

Santoso F, Garratt MA, Anavatti SG, 2018. State-of-the-art intelligent flight control systems in unmanned aerial vehicles. *IEEE Trans Autom Sci Eng*, 15(2):613-627. https://doi.org/10.1109/TASE.2017.2651109

Schulman J, 2016. Optimizing Expectations: from Deep Reinforcement Learning to Stochastic Computation Graphs. PhD Thesis, University of California, Berkeley, USA.

Schulman J, Levine S, Moritz P, et al., 2015. Trust region policy optimization. Proc 31st Int Conf on Machine Learning, p.1889-1897.

Schulman J, Wolski F, Dhariwal P, et al., 2017. Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347

Shi DJ, Dai XH, Zhang XW, et al., 2017. A practical performance evaluation method for electric multicopters. *IEEE/ASME Trans Mechatr*, 22(3):1337-1348. https://doi.org/10.1109/TMECH.2017.2675913

Silver D, Lever G, Heess N, et al., 2014. Deterministic policy gradient algorithms. Proc 31st Int Conf on Machine Learning, p.1-9.

Silver D, Huang A, Maddison CJ, et al., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484-489. https://doi.org/10.1038/nature16961

Sutton RS, 1995. Generalization in reinforcement learning: successful examples using sparse coarse coding. Proc 8th Int Conf on Neural Information Processing Systems, p.1038-1044.

Sutton RS, Barto AG, 1998. Reinforcement Learning: an Introduction. MIT Press, Cambridge, USA.

Tomic T, Schmid K, Lutz P, et al., 2012. Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue. *IEEE Robot Autom Mag*, 19(3): 46-56. https://doi.org/10.1109/MRA.2012.2206473

Valente J, del Cerro J, Barrientos A, et al., 2013. Aerial coverage optimization in precision agriculture management: a musical harmony inspired approach. *Comput Electron Agric*, 99:153-159. https://doi.org/10.1016/j.compag.2013.09.008

Valenti RG, Jian YD, Ni K, et al., 2016. An autonomous flyer photographer. Proc IEEE Int Conf on Cyber Technology in Automation, Control, and Intelligent Systems, p.273-278. https://doi.org/10.1109/CYBER.2016.7574835

van Hasselt H, 2010. Double Q-learning. Proc 23rd Int Conf on Neural Information Processing Systems, p.2613-2621.

van Hasselt H, Guez A, Silver D, 2016. Deep reinforcement learning with double Q-learning. Proc 30th AAAI Conf on Artificial Intelligence, p.2094-2100.

Wang YD, Sun J, He HB, et al., 2019. Deterministic policy gradient with integral compensator for robust quadrotor control. *IEEE Trans Syst Man Cybern Syst*, p.1-13. https://doi.org/10.1109/TSMC.2018.2884725

Waslander SL, Hoffmann GM, Jang JS, et al., 2005. Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning. Proc IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.3712-3717. https://doi.org/10.1109/IROS.2005.1545025

Watkins CJCH, Dayan P, 1992. Q-learning. *Mach Learn*, 8(3-4):279-292. https://doi.org/10.1007/BF00992698

Williams-Hayes PS, 2005. Flight test implementation of a second generation intelligent flight control system. Proc Infotech@Aerospace, p.26-29. https://doi.org/10.2514/6.2005-6995

Xu B, 2018. Composite learning finite-time control with application to quadrotors. *IEEE Trans Syst Man Cybern Syst*, 48(10):1806-1815. https://doi.org/10.1109/TSMC.2017.2698473

Xu R, Ozguner U, 2006. Sliding mode control of a quadrotor helicopter. Proc 45th IEEE Conf on Decision and Control, p.4957-4962. https://doi.org/10.1109/CDC.2006.377588

Yang HJ, Cheng L, Xia YQ, et al., 2018. Active disturbance rejection attitude control for a dual closed-loop quadrotor under gust wind. *IEEE Trans Contr Syst Technol*, 26(4): 1400-1405. https://doi.org/10.1109/TCST.2017.2710951

Yechiel O, Guterman H, 2017. A survey of adaptive control. *Int Rob Autom J*, 3(2):290-292. https://doi.org/10.15406/iratj.2017.03.00053