Neurocomputing 415 (2020) 135–145

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Event-triggered reinforcement learning control for the quadrotor UAV with actuator saturation ☆

Xiaobo Lin [a], Jian Liu [b], Yao Yu [a,*], Changyin Sun [b]

[a] Key Laboratory of Knowledge Automation for Industrial Processes of Ministry of Education, School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, PR China
[b] School of Automation, Southeast University, Nanjing 210096, PR China

ARTICLE INFO

Article history:
Received 1 September 2019
Revised 22 April 2020
Accepted 15 July 2020
Available online 28 July 2020
Communicated by Lei Zou

2010 MSC:
00-01
99-00

Keywords:
Quadrotor
UAV
Reinforcement learning
Flight control
Event-triggered control
Actuator saturation

ABSTRACT

This paper proposes an event-triggered reinforcement learning (RL) control strategy to stabilize the quadrotor unmanned aerial vehicle (UAV) with actuator saturation. As the quadrotor UAV equips with a complex dynamic is difficult to be model accurately, a model free reinforcement learning scheme is designed. Due to the practical limitation of actuators, the end of controller is constrained with a bounded function. In order to reduce the calculation consumption for the onboard computer, an event-triggered mechanism is developed, which only update the controller when the triggered condition is satisfied. The proposed controller is implemented with two neural networks which are called critic and actor. Some advanced RL technologies are utilized for speeding up the train process, e.g. off-policy training, experience replay, etc. The stability of closed-loop system is proved by the Lyapunov analysis. The simulation results including a stability task and a tracking task verify the theoretical analysis, in which we find the updating frequency of controller is decreased greatly.

1. Introduction

In recent years, the quadrotor unmanned aerial vehicle (UAV) has received huge interests from many researchers and has acquired abundant application [1–4]. The quadrotor UAV is becoming one of the most popular Unmanned Aerial Vehicles (UAVs) due to its attractive features: stable hovering, Vertical Take-Off Landing (VTOL), high maneuverability, and simple mechanical structure. However, high performance control for the quadrotor UAV is difficult because its dynamic is nonlinear, strong coupling, underactuated, and Multi-Input Multi-Output. Then parameters uncertainty, different tasks, and capricious work environment increase the control difficulty. Meanwhile the requirement of optimizing the compound performance index (e.g, minimizing the state error and control signal simultaneously) is also a challenge. Thus, the quadrotor UAV requires an adaptive controller which is able to reject model uncertainty and optimize the performance index.

As the quadrotor UAV arouses interests of many researchers, various control methods were proposed to achieve required performance. At beginning, the linear controllers like PID or PD were applied on the control for quadrotor UAVs due to easy-implementation and respectable performance [5,6]. The linear approach was able to stabilize the quadrotor UAV well in common tasks, but it might lose performance when the vehicle moved away from the equilibrium. Then, some nonlinear approaches were shown to overcome this problem, e.g. the back-stepping controllers [7], the nonlinear controller with the hierarchical dynamic inversion and recursive control [8], and the second order sliding mode controller [9]. The conventional nonlinear controller achieved high performance in the general domain when the high accurate system model was reachable, which was hard in practical. Meanwhile, the quadrotor UAV often suffered from many types of disturbance like wind when it worked outdoor. Considering these issues, some disturbance reject controllers were developed, e.g. robust signal compensation controller [10], disturbance observer based controller [11,12], radical basis function neural networks (RBFNN)

☆ This work is supported by the National Natural Science Foundation of China [Grant No. 61921004, 61703037] , the National Postdoctoral Program for Innovative Talents [Grant No. BX20200081] and the Jiangsu Planned Projects for Postdoctoral Research Funds [Grant No. 2020Z081].

* Corresponding author.
E-mail addresses: yuyao@ustb.edu.cn (Y. Yu), cysun@seu.edu.cn (C. Sun).

https://doi.org/10.1016/j.neucom.2020.07.042
0925-2312/

compensation based controller [13,14]. The disturbance-rejection technology enhanced the performance while the quadrotor UAV worked outdoor. However, there were still some features they could not satisfy, e.g. optimizing the compound performance index, rejecting more general disturbance with unknown bounds.

Optimizing the compound performance index for the quadrotor UAV was a challenge for a long time. The LQ control was a conventional method about optimizing performance index for the quadrotor UAV, which simplified the general dynamic to a linearized system and then acquired the optimal controller by solving the Riccati equation [15]. However, the linearization might cause performance loss when the controller worked in the original system. In fact the Riccati equation was a reduced format of Hamilton-Jacobi-Bellman (HJB) equation whose solution was the optimal controller for the original nonlinear system. An effective candidate for solving the HJB equation was the reinforcement learning (RL), which could be classified as value iteration (VI) [16], policy iteration (PI) [17] and actor-critic (AC) [18] according to the structure. Among them, the AC received a great development recent years in the filed of automatic control, e.g. the deep deterministic policy gradient algorithms (DDPG) [19], the goal representation dual heuristic dynamic programming (Gr-GDHP) [20], the iterative adaptive dynamic programming [21], and the twin delayed deep deterministic gradient algorithm (TD3) [22]. Moreover, some researches applied the AC based controller on the quadrotor UAV control. A reinforce learning based controller was proposed for the quadrotor UAV control utilizing the Monte-Carlo to approximate the critic, which was only able to be trained off-line [23]. Then some online AC based methods were developed for the quadrotor UAV, where the temporal difference (TD) was used to acquire the critic [24,25]. Although there were many RL based controller which were developed for quadrotor UAV, few of them considered the calculation consumption and actuators' saturation.

The calculation consumption and actuators' saturation were inevitable problems when the RL based controller was applied on the quadrotor UAV, because the calculation resource of the onboard computer was quite limited and the control input for the motor should be constrained in practical. In order to decrease calculation consumption, the event-triggered (ET) was a good selection, which only updated the controller when the condition was triggered. [26,10]. The ET technology was utilized to reduce the computation cost [27,28] and to relieve communication-load [29,30] for RL based controller. Meanwhile, the ET technology was also applied for the UAV control to decrease computational cost in [1]. But the controller in [31] is relied on the model and the model-free controller like RL controller is more practical when the accurate dynamics are difficult to obtain. However, the computation consumption of the RL controller is larger than the conventional methods, which means that it is necessary to develop ET algorithms to reduce computation. The actuator saturation was commonly regard as input constraint problem during controller design [32]. The input constraint was able to be solved through modifying the reward function and actor with constrained function tanh [33,34]. The problem of actuator saturation was studied for the UAVs' controllers in[35], but they could not optimize the compound performance index (control error and control consumption) or reject a general disturbance with unknown bound. In fact, the RL based controller which considers actuator saturation and ET technology together was rarely studied for the quadrotor UAV.

Motivated by all these significant works, in this paper, we develop an ET-RL controller for the quadrotor UAV with input constraint. The proposed control scheme is based on the framework of AC with two neural networks (NN) utilized to implement the critic and the actor, which is able to driver the quadrotor UAV complete the stabilized task and the tracking task. The main contributions of this paper are stated as follow: 1) The event-triggered is first applied on the RL controller for the quadrotor UAV, which is able to reduce the calculation consumption obviously for the onboard PC. 2) The actuators' saturation is rejected with the modified reward function and actor; 3) Many advance technologies are applied for speeding up the train process, e.g. off-policy training, replay buffer(from DDPG), smooth updating critic(from TD3). 4) The stability of closed-loop system and the convergence of NN are guaranteed by Lyapunov analysis. 5)A open-source simulation environment for quadrotor dynamic is provided.

This rest of this paper is organized as follows. The dynamic of quadrotor UAV and problem statement are presented in Section 2. The design of ET-RL based controller and the stability analysis are indicated in Section 3. Then the simulation results are provided to show the effectiveness of the proposed control scheme in Section 4. Finally, several concluding remarks are given in Section 5.

2. Quadrotor UAV system model and problem description

The structure type of the quadrotor UAV studied in this paper is the 'x'. For convenient notation, two frames are defined (see Fig. 1): the inertia frame E is fixed on a preset location and its positive directions of XYZ are east, north and sky; the body frame B is located on the center of mass (CoM) of the quadrotor UAV and its positive directions are head, right and top of the quadrotor UAV.

The dynamic model of quadrotor UAV used here comes from [36] and is given in (1).

$$\begin{aligned}
\ddot{p}_x &= [\cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi]\tfrac{\tau_4}{m} + d_1\\
\ddot{p}_y &= [\cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi]\tfrac{\tau_4}{m} + d_2\\
\ddot{p}_z &= \cos\theta cos\varphi\tfrac{\tau_4}{m} - g + d_3\\
\ddot{\varphi} &= \dot{\theta}\dot{\psi}\left(\tfrac{I_y - I_z}{I_x}\right) - \tfrac{J_R}{I_x}\dot{\theta}\Omega_R + \tfrac{L}{I_x}\tau_1 + d_4\\
\ddot{\theta} &= \dot{\varphi}\dot{\psi}\left(\tfrac{I_z - I_x}{I_y}\right) + \tfrac{J_R}{I_y}\dot{\varphi}\Omega_R + \tfrac{L}{I_y}\tau_2 + d_5\\
\ddot{\psi} &= \dot{\theta}\dot{\varphi}\left(\tfrac{I_x - I_y}{I_z}\right) + \tfrac{1}{I_z}\tau_3 + d_6,
\end{aligned} \tag{1}$$

where m is the mass of the quadrotor UAV; regarding the inertia frame as reference, p_x, p_y, and p_z are the position of body frame; the attitude φ, θ, and ψ are the Euler angles describing the orientation of body frame with respect to the inertia frame; τ_1, τ_2, and τ_3 are torques on three axis, whereas τ_4 is total thrust; I_x, I_y, and I_z are the moments of inertia (MoI) on three axis of body frame; L is the length from CoM to center of each actuator; J_R notes the MoI of actuators and Ω_R donates the rotation rate of actuator; $d_1, \ldots, d_6$ are the external disturbances (wind, collision, etc.).

The toques and total thrust depend on the rotation rates of actuators. According to the system model, the relation between the thrusts(torques) τ and the rotation rate Ω can be acquired:

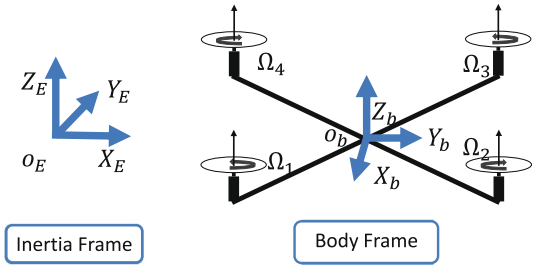


Fig. 1. Structure and frames of the quadrotor UAV.

$$\begin{aligned}\tau_1 &= C_T\left(\Omega_1^2+\Omega_2^2-\Omega_3^2-\Omega_4^2\right)\\ \tau_2 &= C_T\left(-\Omega_1^2+\Omega_2^2+\Omega_3^2-\Omega_4^2\right)\\ \tau_3 &= C_M\left(\Omega_1^2-\Omega_2^2+\Omega_3^2-\Omega_4^2\right)\\ \tau_4 &= C_T\left(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2\right),\end{aligned} \tag{2}$$

where C_T, C_M are the parameters which translate the rotation rates to the thrusts(torques). Then the dynamic of actuator is assumed as a linear one, and the relation between the rotation rate Ω and the control signals u can be shown:

$$\Omega = k_m u + b_m$$

where k_m and b_m are the motor parameters.

Considering the practical limitation of actuator, it is reliable to assume that the control signal for quadrotor UAVs is constrained with a saturated bound, i.e. $\{|u_i| \leqslant \lambda_\alpha, i = 1,\ldots,4\}$.

As a vehicle working in the air, the load and power of quadrotor UAV is very limited, therefore, it cannot carry some powerful calculation platforms. Considering this problem, in this paper, an aperiodic updating controller is designed for decreasing the computation consumption. For the proposed controller, the data from sensors are sampled continues, but the control action and controller parameters are updated on a monotonically increasing sequence $\{\delta_i|\delta_i < \delta_{i+1}, i = 0,1,\ldots,\infty\}$. δ_i is the i^{th} triggered instant.

In order to describe system more conveniently, we assume the system state as x, i.e. $x = \left\{p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z, \varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}\right\}$; note the control input as u, i.e. u=$\{u_1,\ldots,u_4\}$. Then the system can be translated into the follow form:

$$\begin{aligned}&\dot{x} = f(x) + g(x)u\\ &f(x) = \begin{bmatrix} x_2\\ d_1\\ x_4\\ d_2\\ x_6\\ -g+d_3\\ \sqrt{1-x_7^2}x_8\\ a_1x_{10}x_{12}+d_4\\ \sqrt{1-x_9^2}x_{10}\\ a_2x_8x_{12}+d_5\\ x_{12}\\ a_3x_8x_{10}+d_6 \end{bmatrix} \quad g(x) = \begin{bmatrix} 0 & 0 & 0 & 0\\ g'_X & g'_X & g'_X & g'_X\\ 0 & 0 & 0 & 0\\ g'_Y & g'_Y & g'_Y & g'_Y\\ 0 & 0 & 0 & 0\\ g'_Z & g'_Z & g'_Z & g'_Z\\ 0 & 0 & 0 & 0\\ b_1 & b_1 & -b_1 & -b_1\\ 0 & 0 & 0 & 0\\ -b_2 & b_2 & b_2 & -b_2\\ 0 & 0 & 0 & 0\\ b_3 & -b_3 & b_3 & -b_3 \end{bmatrix}\\ &u = [u_1,u_2,u_3,u_4]^T\\ &g'_X = \left(\sqrt{1-x_1^2}x_3\cos x_5 + x_1\sin x_5\right)\frac{C_T}{m}\\ &g'_Y = \left(\sqrt{1-x_1^2}x_3\sin x_5 + x_1\cos x_5\right)\frac{C_T}{m}\\ &g'_Z = \frac{c_T\sqrt{1-x_1^2}\sqrt{1-x_3^2}}{m},\end{aligned} \tag{3}$$

where $a_1 = \frac{J_{yy}-J_{zz}}{J_{xx}}, a_2 = \frac{J_{zz}-J_{xx}}{J_{yy}}, a_3 = \frac{J_{xx}-J_{yy}}{J_{zz}}, b_1 = \frac{LC_T}{J_{xx}}, b_2 = \frac{LC_T}{J_{yy}}, b_3 = \frac{C_M}{J_{zz}}$.

The triggered state is noted with x_j, i.e. $x_j = x(\delta_j)$. And the related control action is $u(\delta_j)$, which acts on system during δ_j to δ_{j+1}.

According to the state errors and control constraints, the performance index of designed controller for quadrotor UAVs is defined as follow:

$$V(x(t)) = \int_t^\infty r(x(\tau),u(\tau))\,\mathrm{d}\tau, \tag{4}$$

where $r(x(t),u(t))$ is the reward function which is defined as follow:

$$r(x(t),u(t)) = x^T(t)Wx(t) + 2\lambda\int_0^u \tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon, \tag{5}$$

where W is a symmetric positive matrices which can be used to set convergence rate; λ is a positive vector which can be used to set constrained boundary; The function used to generate the control value is called policy here. u is the control value (called action latter) generated by a policy $\mu(x(\delta_j))$, i.e. $u(t) = \mu(x_j)$ during δ_j to δ_{j+1}. Because $\tanh^{-T}(X)$ is a monotonic odd function limiting to ±1, the right item of (5) is positive and will increase rapidly when $|\upsilon/\lambda|$ is nears 1. Therefore, the reward will approach infinity when the actuator saturation happens.

Now, the objective of proposed controller is to stabilized the system (3) and minimizes the performance index (4). The detailed design will be shown in the following section.

3. Event-triggered RL controller designed with constrained control inputs

In this section, a reinforcement learning based controller is designed for quadrotor UAV. The proposed controller contains a value function (used for evaluating policy) and a policy (used for generating action), which are approximated with two neural networks. The one approximates the value function is called critic, and the other is called actor. The general scheme of proposed method is shown in Fig. 2. The content of this section is organized as follow. Firstly, considering the action is constrained by actual actuators, a tanh is utilized to limit the output of the policy. In order to reduce calculation, an event-triggered condition is designed and the controller only be updated on triggered instant. Then the networks' structure and the training algorithm for the critic and the actor are presented. Finally, the stability of proposed method is analyzed.

The infinitesimal version of (4) is provided as:

$$\begin{aligned}&V_x^{*T}(f(x)+g(x)\mu(x)) + x^TWx\\ &\quad +2\lambda\textstyle\int_0^{\mu(x)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon = 0,\end{aligned} \tag{6}$$

where $V_x^* = \partial V^*(x)/\partial x$ and $V^*(x)$ is the optimal performance index. Then a optimal policy $\mu^*(x)$ can be acquired with following equation if the system model is known.

$$\mu^*(x) = -\lambda\tanh\left(\frac{1}{2\lambda}g^T(x)V_x^*\right),$$

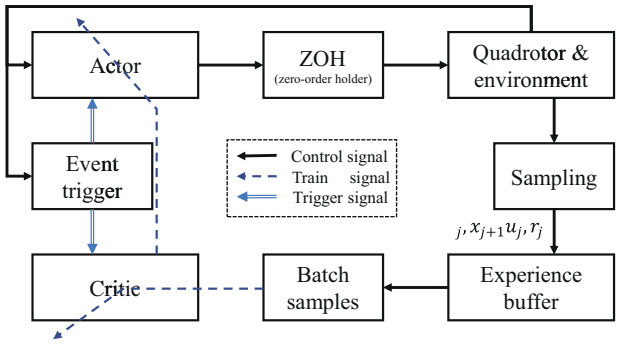


Fig. 2. Control schematic of event triggered reinforcement learning control.

In our event-triggered method, the action is updated on discrete instant when the set condition is satisfied. Therefore, the system states received by controller are discontinuous, i.e. x_j. And the optimal controller for our method can be described as:

$$\mu^*(x_j) = -\lambda \tanh\left(\frac{1}{2\lambda} g^T(x_j) V_{x_j}^*\right). \tag{8}$$

Set $\xi(X) = \frac{1}{2\lambda} g^T(X) V_X^*$ and apply event triggered controller (8) into (6), a new HJB equation can be obtained:

$$H(x, \mu^*(x_j), V_x^*, \xi(x), \xi(x_j)) = V_x^{*T} f(x) - 2\lambda^2 \xi^T(x) \tanh(\xi(x_j)) + x^T W x + 2\lambda \int_0^{-\lambda \tanh(\xi(x_j))} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon. \tag{9}$$

Now, in order to get a high performance controller for the quadrotor UAV, there are two key problems which should be solved. The one is to design an event-triggered condition, which regulator $\mu^*(x_j)$ stabilized the original system (3); the other is to develop matching learning algorithm, which can approximate $V^*(x_j)$ and $\mu^*(x_j)$ with neural network. The previous one is presented on 3.1, the latter on is presented on Section 3.2, and the stability of closed-loop system is analyzed on 3.3.

3.1. Event-triggered regulator designed

Before developing the event-triggered condition, the sampling error is defined as gap between sampled state and real-time state, which is noted with e_j:

$$e_j = x(\delta_j) - x(t) = x_j - x \tag{10}$$

where the latter item is a more concise format and is used in following analysis. Then several assumptions are given:

Assumption 1. $\xi(X)$ is Lipschitz continuous with respect to sampling error e_j:

$$||\xi(x_j) - \xi(x)|| \leqslant L_1 ||x_j - x||. \tag{11}$$

The event-triggered condition is designed as:

$$||e_j||^2 \leqslant \frac{\underline{\lambda}(1-\alpha^2)||x_j||^2 + r_u(x_j)}{\beta^2} \tag{12}$$

where L_1 is parameters in assumptions and α is wanted convergence rate. $r_u(x_j)$ is a function about control value, i.e.

$$r_u(x_j) = 2\lambda \int_0^{\mu(x_j)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \tag{13}$$

Defining $\beta_2 = \frac{\underline{\lambda}(1-\alpha^2)}{\beta^2}$, the event-triggered condition can be rewritten as:

$$\begin{aligned} ||e_j||^2 &\leqslant \beta_2 \left(||x_j||^2 + \frac{r_u(x_j)}{\underline{\lambda}(1-\alpha^2)} \right) \\ ||e_T||^2 &= ||x_j||^2 + \frac{r_u(x_j)}{\underline{\lambda}(1-\alpha^2)} \end{aligned} \tag{14}$$

where e_t is the event-triggered threshold. Then we can adjust the event-triggered rate with $0 < \beta_2 < 1$, which is designed for balancing performance and calculation consumption. A smaller β_2 means the more event-triggered times, heavier calculation consumption and higher performance.

Theorem 1. *If there exist a value function $V^*(x_j)$ and a policy $\mu^*(x_j)$ satisfy the HJB function (9), and the controller is updated with event-triggered condition (12), then the original system (3) is stabilized.*

Proof. Select (4) as Lyapunov function, the derivative of $V^*(x)$ with respect to time is acquired:

$$\begin{aligned} \dot{V}^*(x) &= \left(\frac{\partial V^*(x)}{\partial x}\right)^T \dot{x} \\ &= V_x^{*T}(f(x) + g(x)\mu^*(x_j)) \\ &= V_x^{*T} f(x) + V_x^{*T} g(x) \mu^*(x) - V_x^{*T} g(x) \mu^*(x) \\ &\quad + V_x^{*T} g(x) \mu^*(x_j), \end{aligned} \tag{15}$$

where $\partial V^*(x)$ is the partial derivative of the optimal performance index. Substituting (6) and (8) into (15), we obtain:

$$\begin{aligned} \dot{V}^*(x) &= -2\lambda \int_0^{-\lambda \tanh\left(\frac{1}{2\lambda} g^T(x) V_x^*\right)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &\quad - x^T W x + \lambda V_x^{*T} g(x) \tanh\left(\tfrac{1}{2\lambda} g^T(x) V_x^*\right) \\ &\quad - \lambda V_x^{*T} g(x) \tanh\left(\tfrac{1}{2\lambda} g^T(x_j) V_{x_j}^*\right) \\ &= -2\lambda \int_0^{-\lambda \tanh \xi(x)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon - x^T W x \\ &\quad + 2\lambda^2 \xi^T(x) \tanh \xi(x) - 2\lambda^2 \xi^T(x) \tanh \xi(x_j), \end{aligned} \tag{16}$$

The $r_u(x)$ can be rewritten as follow:

$$\begin{aligned} &2\lambda \int_0^{-\lambda \tanh(\xi(x))} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &= 2\lambda^2 \xi^T(x) \tanh(\xi(x)) + \lambda^2 \ln\left[1 - \tanh^2(\xi(x))\right] \end{aligned} \tag{17}$$

And, calculate the following integration:

$$\begin{aligned} &\int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} 2\lambda \xi^T(x)\, \mathrm{d}\upsilon \\ &= 2\lambda^2 \xi^T(x) \tanh(\xi(x)) - 2\lambda^2 \xi^T(x) \tanh(\xi(x_j)) \end{aligned} \tag{18}$$

Then, according to (17) and (18), we have the following equation:

$$\begin{aligned} &2\lambda \int_0^{-\lambda \tanh\left(\xi(x_j)\right)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &= 2\lambda \int_0^{-\lambda \tanh(\xi(x))} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &\quad + 2\lambda \int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &= 2\lambda^2 \xi^T(x) \tanh(\xi(x)) + \lambda^2 \ln\left[1 - \tanh^2(\xi(x))\right] \\ &\quad + 2\lambda \int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &= \int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} 2\lambda \xi^T(x)\, \mathrm{d}\upsilon + 2\lambda^2 \xi^T(x) \tanh(\xi(x_j)) \\ &\quad + \lambda^2 \ln\left[1 - \tanh^2(\xi(x))\right] + 2\lambda \int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} \tanh^{-T}(\upsilon/\lambda)\, \mathrm{d}\upsilon \\ &= 2\lambda \int_{-\lambda \tanh(\xi(x))}^{-\lambda \tanh\left(\xi(x_j)\right)} \left[\xi^T(x) + \tanh^{-T}(\upsilon/\lambda)\right] \mathrm{d}\upsilon \\ &\quad + \lambda^2 \ln\left[1 - \tanh^2(\xi(x))\right] + 2\lambda^2 \xi^T(x) \tanh(\xi(x_j)) \end{aligned} \tag{19}$$

Then, substituting (17), 19 and (11) to (16), we get:

$$
\begin{aligned}
\dot{V}^*(x) &= -2\lambda\int_0^{-\lambda\tanh\xi(x)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon - x^TWx \\
&\quad +2\lambda^2\xi^T(x)\tanh\xi(x) - 2\lambda^2\xi^T(x)\tanh\xi(x_j) \\
&= -x^TWx - 2\lambda\int_0^{-\lambda\tanh\xi(x_j)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon \\
&\quad +2\lambda\int_{-\lambda\tanh(\xi(x))}^{-\lambda\tanh(\xi(x_j))}\left[\xi^T(x)+\tanh^{-T}(\upsilon/\lambda)\right]\mathrm{d}\upsilon \\
&\quad +2\lambda^2\xi^T(x)\tanh(\xi(x)) + \lambda^2\ln\left[1-\tanh^2(\xi(x))\right] \\
&\quad -2\lambda\int_0^{-\lambda\tanh\xi(x)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon \\
&= -(x_j+e_j)^TW(x_j+e_j) \\
&\quad -2\lambda\int_0^{-\lambda\tanh\xi(x_j)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon \\
&\quad +2\lambda^2\int_{\xi(x)}^{\xi(x_j)}\left[\overline{\upsilon}^T-\xi^T(x)\right]\left[1-\overline{\upsilon}^T\overline{\upsilon}\right]\mathrm{d}\overline{\upsilon} \\
&\leqslant -x_j^TWx_j - 2x_j^TWe_j - e_j^TWe_j \\
&\quad -2\lambda\int_0^{-\lambda\tanh\xi(x_j)}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon \\
&\quad +\lambda^2\left(\xi(x_j)-\xi(x)\right)^T\left(\xi(x_j)-\xi(x)\right) \\
&\leqslant -(1-\alpha^2)x_j^TWx_j - (1-\tfrac{1}{\alpha^2})e_j^TWe_j \\
&\quad -2\lambda\int_0^{\mu_{x_j}}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon + L_1^2\lambda^2||e_j||^2 \\
&\leqslant -2\lambda\int_0^{\mu_{x_j}}\tanh^{-T}(\upsilon/\lambda)\,\mathrm{d}\upsilon - \underline{\lambda}(1-\alpha^2)||x_j||^2 \\
&\quad +\left(L_1^2\lambda^2+\tfrac{1}{\alpha^2}-1\right)||e_j||^2,
\end{aligned}
\tag{20}
$$

where α is a constant smaller than 1; $\underline{\lambda}$ is the minimum eigenvalue of W. Define $\beta^2 = L_1^2\lambda^2 + \frac{1}{\alpha^2} - 1$. When (12) is satisfied, $\dot{V}^*(x)$ is negative when system state leaves the equilibrium. Therefore, the original system is stable with controller $\mu^*(x_j)$ if event-triggered condition is satisfied. □

3.2. Neural-network based controller designed

In this section, neural networks are utilized to approximate $V^*(x)$ and $\mu^*(x)$ in proposed reinforcement learning structure. The network used to approximate $V^*(x)$ is called critic and the one used to approximate $\mu^*(x)$ is called actor here. The detailed algorithm process of the proposed event-triggered RL control with actuator saturation is provided in Algorithm 1.

Algorithm 1 Event-triggered reinforcement learning control for the quadrotor UAV

Initialize the weights for the actor network and the critic.
For each $i \in [1, N_{maxEpisodes}]$ do
  Initialize $x(0)$*randomly* and set $x_0 = x(0)$, $j = 0$
  Calculate $\mu_0 = \lambda\tanh(\omega_a^{*T}\phi_a(x_0))$
  for each $t \in [1, N_{maxSteps}]$ do
    Set $u(t) = \mu_j$
    Update system information with $\dot{x} = f(x) = g(x)u(t)$
    Observe the reward $r(t)$ and the new state $x(t+1)$
    if $\|x(t) - x_j\|^2 > \beta_2 e_T$ then
      Set $r_j = r(t)$,$x_{j+1} = x(t)$
      Store the transition $(x_j, \mu_j, r_j, x_{j+1})$ in experience buffer ε
      Sample a batch of transitions from ε randomly, and update the network weights with (25) and (28)
      Calculate $\mu_{j+1} = \lambda\tanh(\omega_a^{*T}\phi_a(x_j))$
      $j = j + 1$
    end If
  end for
end for

For more stable and quicker training, several technologies are applied. Firstly, experience buffer is designed for storing historical data and each data including $[\,x_j, x_{j+1}, u_j, r_j\,]$. x_j and x_{j+1} are system state in tick δ_j and δ_{j+1}; u_j is the control value generated by $\mu(x_j)$; r_j is the reward value. Then, the value function in Q-learning $Q^*(x,u)$ is utilized to replace $V^*(x)$, which are proved enjoying equal final performance in [37]. The ideal $Q^*(x,u)$ can be formulated as:

$$
Q^*(x_j) = \omega_c^{*T}\phi_c(x_j,u_j) + \varepsilon_c(\delta_j) \tag{21}
$$

where ω_c is ideal coefficient; $\varepsilon_c(\delta_j)$ is the approximation error between network and real Q^*, noting the boundary of Q^* with σ_c; $\phi_c(x_j,u_j)$ represents hidden layer neurons whose weights are generally initialised and keep constant. Then $\mu^*(x)$ can be formulated as:

$$
\mu^*(x_j) = \lambda\tanh(\omega_a^{*T}\phi_a(x_j)) \tag{22}
$$

where ω_c is ideal coefficient; $\phi_c(x_j,u_j)$ represents hidden layer neurons whose weights are generally initialised and keep constant; tanh is the activated function and utilized to constrain output. $\omega_a^{*T}\phi_a(x_j)$ is actually the $\xi(x_j)$ in previous section.

Then the critic and actor can be shown as follow:

$$
\begin{aligned}
\widehat{Q}(x_j) &= \hat{\omega}_c^T\phi_c(x_j,u_j) \\
\hat{\mu}(x_j) &= \lambda\tanh(\omega_a^{*T}\phi_a(x_j))
\end{aligned}
\tag{23}
$$

where $\hat{\omega}_c$ and $\hat{\omega}_a$ approximation coefficient for critic and actor. Define $\tilde{\omega}_c$ as error between ideal coefficient ω_c^* and approximation coefficient $\hat{\omega}_c$. Substituting (21) and (23) to HJB function (9), the following equation can be obtained:

$$
(\hat{\omega}_c^T + \tilde{\omega}_c^T)\frac{\partial\phi_c(x_j,u_j)}{\partial x}\frac{x_{j+1}-x_j}{T_j} + r(\delta_j) = 0 \tag{24}
$$

where $\dot{x}_j$ is nearby with $(x_{j+1}-x_j)/T_j$; and T_j is the sample period between δ_j and δ_{j+1}; $(x_j, x_{j+1}, u_j, r(\delta_j))$ is record data. Then, we design the update rule for critic:

$$
\dot{\omega}_c = \eta_c\left(\hat{\omega}_c^T\frac{\partial\phi_c}{\partial x}\frac{x_{j+1}-x_j}{T_j} + r(\delta_j)\right)\frac{\partial\phi_c}{\partial x}\frac{x_{j+1}-x_j}{T_j}. \tag{25}
$$

The estimation error between ω_c^* and $\hat{\omega}_c$ will decrease over time. Then, $\mu^*(x_j)$ is the optimal controller which minimizes $Q^*(x_j)$. Note ideal parameters of $\mu(x_j)$ with ω_a^*. Rewriting $\mu(x_j)$ with Tailor series on u_j, we obtain that

$$
\begin{aligned}
\mu^*(x_j) &= \lambda\tanh(\omega_a^{*T}\phi_a(x_j)) \\
&= \lambda u_j + \lambda\left(1-u_j^2\right)\left(\omega_a^{*T}\phi_a(x_j) - \mu_j\right) + \varepsilon_\mu(\delta_t) \\
&= \lambda\left(1-u_j^2\right)\omega_a^{*T}\phi_a(x_j) + \lambda u_j^3 + \varepsilon_\mu(\delta_t)
\end{aligned}
\tag{26}
$$

where $\varepsilon_\mu(t)$ is the error for abandoning Tailor high order items. Assume $\varepsilon_\mu(t) \leqslant \sigma_\mu$.

We have

$$
\begin{aligned}
&\omega_c^{*T}\phi_c\left(x_j,\mu_j(x_j)\right) = \varepsilon_a(\delta_t) \\
&\omega_c^{*T}\phi_c(x_j,\tanh(\omega_a^{*T}\phi_a(x_j))) = \varepsilon_a(\delta_t) \\
&\omega_c^{*T}\omega_{c,1}^Tx_j + \omega_c^{*T}\omega_{c,2}^T\tanh(\omega_a^{*T}\phi_a(x_j))) = \varepsilon_a(\delta_t) \\
&\omega_c^{*T}\omega_{c,2}^T\left(\lambda\left(1-\mu_j^2\right)(\hat{\omega}_a^T + \tilde{\omega}_a^T)\phi_a(x_j)\right) = \overline{\varepsilon}_a(\delta_t)
\end{aligned}
\tag{27}
$$

where ε_a is network approximation error caused by difference between μ^* and ideal action function;

$\overline{\varepsilon}_a = \varepsilon_a - \omega_c^{*T}\omega_{c,1}^T x_j - \omega_c^{*T}\omega_{c,2}^T\left(\lambda u_j^3 + \varepsilon_\mu\right)$; $\hat{\omega}_a$ is the estimation weights and $\tilde{\omega}_a$ is the estimation error. $\omega_{c,1}$ and $\omega_{c,2}$ are weights of first layer of critic. Now, we design the update rule for actor weights:

$$\dot{\omega}_a = \eta_a\left(1-\mu_j^2\right)\omega_{c,2}\omega_c^*\left(\omega_c^{*T}\omega_{c,2}^T\lambda\left(1-\mu_j^2\right)\hat{\omega}_a^T\phi_a(x_j) + \hat{\bar{\varepsilon}}_a(\delta_t)\right)\phi_a^T(x_j) \tag{28}$$

where $\hat{\bar{\varepsilon}}_a(\delta_t) = \omega_c^{*T}\omega_{c,1}^T x_j + \omega_c^{*T}\omega_{c,2}^T\lambda u_j^3$; η_a is the train step. As the control law is only updated when the event-triggered happens, the real control input can be described as follow:

$$u(t) = \begin{cases} u(x_{j-1}) & \delta_j \leqslant t < \delta_{j+1}, \\ \lambda\tanh(\hat{\omega}_a^T\phi_a(x_j)) & t = \delta_{j+1}. \end{cases} \tag{29}$$

3.3. Stability analysis of proposed method

In this section, the stability analysis for closed-loop system is presented. A Lyapunov function candidate is selected which includes state error, neural approximation error and control input.

Theorem 2. *Consider a nonlinear continuous-time system 3 with a bound constrained controller23. The controller is update when event trigger 12 happens and the update laws for controller are provided by 25, 28, then the system state x, neural approximation error $\tilde{\omega}_c, \tilde{\omega}_a$ are all UUB. Define the closed-loop Lyapunov function with V_{cl}, then we have:*

$$V_{cl} = V^*(x) + V^*(x_j) + \frac{1}{2\eta_c}tr(\tilde{\omega}_c^T\tilde{\omega}_c) + \frac{1}{2\eta_a}tr(\tilde{\omega}_a^T\tilde{\omega}_a) \tag{30}$$

where $\dot{V}^*(x)$ is presented in (20) and $\dot{V}^*(x_j)$ is a sample sequence of $\dot{V}^*(x)$. Define $V_{cl,\omega} = \frac{1}{2}\tilde{\omega}_c^T\tilde{\omega}_c + \frac{1}{2}\tilde{\omega}_a^T\tilde{\omega}_a$.

The proof of the stability of closed-loop system is shown in two parts including the continuous dynamic (the event is not triggered) and the jump one (the event is triggered). The first difference of the Lyapunov function is shown as follows:

$$V_{cl} = V^*(x) + V^*(x_j) + \frac{1}{2\eta_c}tr(\tilde{\omega}_c^T\tilde{\omega}_c) + \frac{1}{2\eta_a}tr(\tilde{\omega}_a^T\tilde{\omega}_a) \tag{31}$$

Firstly, we consider the continuous dynamic of system. Taking the time derivative of $V_{cl,\omega}$ and considering (25) and (28), we obtained::

$$\begin{aligned}
&\dot{V}_{cl,\omega} = \dot{\tilde{\omega}}_c^T\tilde{\omega}_c + \dot{\tilde{\omega}}_a^T\tilde{\omega}_a \\
&= \left[\left(\hat{\omega}_c^T\left(\frac{\partial\phi}{\partial x}\right)^T\frac{x_{j+1}-x_j}{T_j} + r(\delta_j)\right)\left(\frac{\partial\phi}{\partial x}\right)^T\frac{x_{j+1}-x_j}{T_j}\right]^T\tilde{\omega}_c \\
&+\eta_a\Big[\phi_a(x_j)\left(\omega_c^{*T}\omega_{c,2}^T\left(1-\mathrm{diag}\left(\mu_j^2\right)\right)\hat{\omega}_a^T\phi_a(x_j)\right. \\
&\left.+\omega_c^*\omega_{c,1}^T x_j + \omega_c^*\omega_{c,2}^T\lambda\mu_j^3)\omega_c^{*T}\omega_{c,2}^T\left(1-\mathrm{diag}\left(\mu_j^3\right)\right)\right]^T\tilde{\omega}_a \\
&= -\tilde{\omega}_c^T\left(\frac{\partial\phi}{\partial x}\right)^T\frac{x_{j+1}-x_j}{T_j}\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\tilde{\omega}_c \\
&-\eta_a\Big[\phi_a(x_j)\omega_c^{*T}\omega_{c,2}^T\left(1-\mathrm{diag}\left(\mu_j^2\right)\right)\tilde{\omega}_a^T\phi_a(x_j)\omega_c^{*T}\omega_{c,2}^T \\
&\times\left(1-\mathrm{diag}\left(\mu_j^3\right)\right)^T\tilde{\omega}_a + \eta_a\lambda\Big[\phi_a(x_j)\left(\varepsilon_a - \omega_c^{*T}\omega_{c,2}^T\varepsilon(t)\right)\Big]^T\tilde{\omega}_a \\
&= -\left[\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\tilde{\omega}_c\right]^T\left[\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\tilde{\omega}_c\right] \\
&-\eta_a\left(1-\mathrm{diag}\left(\mu_j^2\right)\right)\omega_{c,2}\omega_c^*\omega_c^{*T}\omega_{c,2}^T\left(1-\mathrm{diag}\left(\mu_j^2\right)\right) \\
&\times\tilde{\omega}_a^T\phi_a(x_j)\left[\tilde{\omega}_a^T\phi_a(x_j)\right]^T + \eta_a\lambda\left(\varepsilon_a - \omega_c^*\omega_{c,2}^T\varepsilon_\mu(t)\right)\phi_a^T(x_j)\tilde{\omega}_a
\end{aligned} \tag{32}$$

then (30) can be rewritten as:

$$\begin{aligned}
&\dot{V}_{cl} = \dot{V}^*(x) + \mathrm{tr}\left(\dot{\tilde{\omega}}_c^T\tilde{\omega}_c\right) + \mathrm{tr}\left(\dot{\tilde{\omega}}_a^T\tilde{\omega}_a\right) \\
&\leqslant -\alpha^2||x||^2 + \mathrm{tr}\left(\eta_a\lambda\left(\varepsilon_a - \omega_c^*\omega_{c,2}^T\varepsilon_\mu(t)\right)\phi_a^T(x_j)\tilde{\omega}_a\right) \\
&-\left[\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\tilde{\omega}_c\right]^T\left[\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\tilde{\omega}_c\right] \\
&-\mathrm{tr}\left(\left(1-\mathrm{diag}\left(\mu_j^2\right)\right)\omega_{c,2}\omega_c^*\omega_c^{*T}\omega_{c,2}^T\left(1-\mathrm{diag}\left(\mu_j^2\right)\right)\times\tilde{\omega}_a^T\phi_a(x_j)\left[\tilde{\omega}_a^T\phi_a(x_j)\right]^T\right) \\
&\leqslant -\alpha^2||x||^2 - \left\|\left(\frac{x_{j+1}-x_j}{T_j}\right)^T\frac{\partial\phi}{\partial x}\right\|^2\|\tilde{\omega}_c\|^2 \\
&-\left(\left\|1-\mathrm{diag}\left(\mu_j^2\right)\right\|^2\|\omega_{c,2}\omega_c^*\|^1 - K_1\|\omega_{c,2}\omega_c^*\|^2 - K_2\right) \\
&\times\|\tilde{\omega}_a^T\phi_a(x_j)\|^2 + \frac{\lambda}{4}\left(\frac{\sigma_\mu^2}{K_1} + \frac{\sigma_a^2}{K_2}\right)
\end{aligned} \tag{33}$$

where K_1 and K_2 are parameters and used for adjust the constrains boundary. And the minimum of the equation $\left\|1-\mathrm{diag}\left(\mu_j^2\right)\right\|$ is able to be decided with λ according to (23). If the following equations are satisfied,

$$\begin{aligned}
&\|\tilde{\omega}_c\| \geqslant \sqrt{\frac{\frac{\lambda}{4}\left(\frac{\sigma_\mu^2}{K_1}+\frac{\sigma_a^2}{K_2}\right)}{\left(\left(1-\lambda^2\right)^2-K_1\right)\left\|\omega_{c,2}\omega_c^*\right\|^2-K_2}} \\
&||e_j||^2 \leqslant \frac{\left(\frac{\lambda}{2}-\alpha^2\right)||x||^2 + r_u(x_j)}{L_1\lambda^2},
\end{aligned} \tag{34}$$

$\dot{V}_{cl} < 0$, i.e., the continuous dynamics of the closed-loop system are UUB.

Then, the jump dynamic part will be analyzed.

$$\begin{aligned}
\Delta V_{cl} &= V^*\left(x\left(\delta_{j+1}^+\right)\right) - V^*(x(\delta_{j+1})) + V^*(x(\delta_{j+1})) - V^*(x(\delta_j)) \\
&+\frac{1}{2\eta_c}tr(\tilde{\omega}_c^T(\delta_{j+1})\tilde{\omega}_c(\delta_{j+1})) - \frac{1}{2\eta_c}tr(\tilde{\omega}_c^T(\delta_j)\tilde{\omega}_c(\delta_j)) \\
&+\frac{1}{2\eta_a}tr(\tilde{\omega}_a^T(\delta_{j+1})\tilde{\omega}_a(\delta_{j+1})) - \frac{1}{2\eta_a}tr(\tilde{\omega}_a^T(\delta_j)\tilde{\omega}_a(\delta_j))
\end{aligned} \tag{35}$$

where δ_{j+1} and δ_j are event-triggered instants. According to the proof on the first part, the network estimation error are UUB, therefore, there exists $tr(\tilde{\omega}_c^T(\delta_{j+1})\tilde{\omega}_c(\delta_{j+1})) \leqslant tr(\tilde{\omega}_c^T(\delta_j)\tilde{\omega}_c(\delta_j))$ and $tr(\tilde{\omega}_a^T(\delta_{j+1})\tilde{\omega}_a(\delta_{j+1})) \leqslant tr(\tilde{\omega}_a^T(\delta_j)\tilde{\omega}_a(\delta_j))$at jump instant. Meanwhile, at the event-triggered instant, $x\left(\delta_{j+1}^+\right) = x(\delta_{j+1})$. Because the continuous dynamic are UUB, there exists $V^*(x(\delta_{j+1})) \leqslant V^*(x(\delta_{j+1})-1) \leqslant V^*(x(\delta_{j+1})-2) \leqslant \ldots \leqslant V^*(x(\delta_j))$. Therefore, $\Delta V_{cl} \leq 0$, the closed system is UUB when the event is triggered.

3.4. Minimum inter-event period

The inter-event period is the time between two adjacent ET tick, e.g, t_k, t_{k+1}. The ET controller will degenerate into a continuous one when the inter-event period equaling zero, which is called Zeno behavior. The minimum inter-event period is analyzed in this section.

Considering the practical limit, we make the follow assumption:

$$\begin{aligned}
&\|f(x)\| \leqslant L_f\|x\| \\
&\|g(x)\| \leqslant L_g
\end{aligned} \tag{36}$$

According to the ET condition (12), we have the following equation when the ET happen:

$$\begin{aligned}
&\beta^2||e_j||^2 \geqslant \underline{\lambda}(1-\alpha^2)||x_j||^2 + r_u(x_j)\\
&\geqslant \underline{\lambda}(1-\alpha^2)\|x+e_j\|^2\\
&= \underline{\lambda}(1-\alpha^2)\left(\tfrac{1}{2}\|x\|^2 - \|e_j\|^2\right)\\
&+\tfrac{1}{2}\|x\|^2 + 2\|xe_j\| + 2\|e_j\|^2\\
&\left[\underline{\lambda}(1-\alpha^2)+\beta^2\right]\|e_j\|^2 \geqslant \tfrac{1}{2}\underline{\lambda}(1-\alpha^2)\|x\|^2
\end{aligned} \tag{37}$$

when the ET happens, $||e_j|| = 0$. During the inter-event period, $\frac{||e_j||}{||x||}$ varies from 0 to Γ, i.e. $0 \leqslant \frac{||e_j||}{||x||} \leqslant \Gamma$, where Γ is defined as follow:

$$\Gamma = \sqrt{\frac{\frac{1}{2}\underline{\lambda}(1-\alpha^2)}{\underline{\lambda}(1-\alpha^2)+\beta^2}} \tag{38}$$

Base on the above analysis, we know that the feature of $\frac{\mathrm{d}}{\mathrm{d}t}\frac{||e_j||}{||x||}$ affects the minimum inter-event period. According to the assumption in the start of this section, we have the follow equation:

$$\begin{aligned}
&\|\dot{e}_j\| = \|\dot{x}_j - \dot{x}\| = \|\dot{x}\|\\
&= \|f(x) + g(x)\mu(x_j)\|\\
&\leqslant L_f||x|| + L_gL_1||x_j||\\
&= (L_f + L_gL_1)||x|| + L_gL_1||e_j||\\
&\leqslant L_2(||x|| + ||e_j||)
\end{aligned} \tag{39}$$

where $L_2 = L_f + L_gL_1$.

Then, submitting (39) to $\frac{\mathrm{d}}{\mathrm{d}t}\frac{||e_j||}{||x||}$, the follow equation can be gotten:

$$\begin{aligned}
&\tfrac{\mathrm{d}}{\mathrm{d}t}\tfrac{||e_j||}{||x||} = \tfrac{\|\dot{e}_j\|||x|| - ||e_j||||\dot{x}||}{||x||^2}\\
&\leqslant \tfrac{L_2(||x||+||e_j||)(||x||-||e_j||)}{||x||^2}\\
&\leqslant \tfrac{L_2(||x||+||e_j||)^2}{||x||^2}\\
&\leqslant L_2(1+\Gamma)^2
\end{aligned} \tag{40}$$

Therefore, the minimum inter-event period (note with t_M) can be bounded as follow equation:

$$t_m = \frac{\Gamma}{\frac{\mathrm{d}}{\mathrm{d}t}\frac{||e_j||}{||x||}} \geqslant \frac{\Gamma}{L_2(1+\Gamma)^2} \tag{41}$$

From above analysis, the minimum inter-event period exists and the proposed method is Zeno free behavior.

4. Simulations

4.1. Simulation setup

In order to verify the proposed method more conveniently, we developed a software framework for quadrotor UAV simulation called QuadrotorFly, which was open-source on the GITHUB. In contrast to the existing frameworks [38,39], Quadrotorfly is whole written with Python (like Matlab) and few extend lib is necessary, aimed at a friendly and powerful tool for researchers. The software provided abundant features, i.e. nonlinear quadrotor UAV dynamic, sensor system with noise, logging, plotting, 3d animation for visualization and organization with multi-UAVs, etc. The following simulations were all implemented on the QuadrotorFly, and the main parameters of PC used for simulation include CPU(i7-8086), memory(DDR4, 64 GB) and hard disk(SSD, 512 GB).

Before simulation, we need to define the simulation parameters. The quadrotor UAV used for simulations is a 'x' type one, whose physical parameters are listed in Table 1. The reference signals for the quadrotor UAV are position in XYZ and attitude in yaw. The control signals for actuators should be between (0, 1). Actually, the control signal for the quadrotor acts on motor, which is the percentage of the throttle of the motor. Therefore, the 0 means 0% throttle while the 1 means 100% throttle. The initial weights of critic and actor are chosen randomly within (−1, 1) and the initial state of the quadrotor UAV is selected randomly too. The neurons' number of hidden layer in the actor is 100, and the one in the critic is 300. The activated function of actor is tanh for input constrained.

During simulations, the process of one step contains five sections: 1) Sensor Reading, the current state of quadrotor UAV can be read directly here; 2) Action Calculation, calcuate the action with actor according current state; 3) Dynamic Update, update the state based on dynamic and action; 4) Policy Train, train the critic and actor with historical data; 5) Data Collection, record the simulation data for policy train and presentation. The sample period Dynamic Update is set as 0.01 s and the update function is the classical Runge-Kutta. The Action Calculation and Policy Train are only executed when the event-triggered condition is satisfied, while other sections are runing all steps.

Confirming which section takes the main duty for calculation consumption is meanful for decreasing cost in quadrotor UAV control. For analysing this problem, we execute 10000 steps without event-triggered each step and calculate the average time-costs of each section. Then, in order to verify the practicability of proposed method, the same benchmark is completed on the Raspberry platform, which is lightweight embeded computing platform and widely applied on small vehiles like the quadrotor UAV. The result is shown on Fig. 3. From the left sub-figure, we get the time cost for Action Calculation is 3.38 micro-seconds, which means the quadrotor UAV can be controlled real-time(250 Hz) if the Policy Train is off-line. From the right sub-figure, we find the Policy Train occupies the most consumption(77.02%) and the Action Calculation takes 8.38%. These two operations utilize over 85% calculation consumption of the simulation. Therefore, declining the calculation cost through event-triggered is reasonable, and the Policy Train may be able to run on-line when enough calculation consumption is omitted.

The folloing simulation contain two parts: the stability task and the tracking task. These task is executed after 200 episodes of pre-training. An episode pre-training is a process that the quadrotor UAV starts from a random state (with variances diag $\left[2,2,2,0,0,0,8.72e^{-2},8.72e^{-2},8.72e^{-2},0,0,0\right]^T$) and then run with a series of actions generated by actor. The episodes ends when the UAV goes out of allowable space or arrives the max episode time(10 s here). Then the experience data captured during these

Table 1
Parameters of the Quadrotor UAV.

Symbol	Description	Value	Units
m	Mass	1.5	kg
g	Acceleration of gravity	9.8	m/s^2
L	Radius of quadrotor	4.5	$10^{-1}\ m$
J_{xx}	Moment of quadrotor inertia	1.75	$10^{-2}\ N\cdot s^2\cdot rad^{-1}$
J_{yy}	Moment of quadrotor inertia	1.75	$10^{-2}\ N\cdot s^2\cdot rad^{-1}$
J_{zz}	Moment of quadrotor inertia	3.18	$10^{-2}\ N\cdot s^2\cdot rad^{-1}$
J_R	Moment of one rotor	9.90	$10^{-5}\ N\cdot s^2\cdot rad^{-1}$
C_T	Thrust coefficient	1.11	$10^{-5}\ N\cdot (rad/s)^{-2}$
C_M	Torque coefficient	1.49	$10^{-7}\ N\cdot m\cdot (rad/s)^{-2}$
k_m	Motor speed coefficient	6.46	$10^2\ rad/s$
b_m	Motor speed bias	1.66	$10^2\ rad/s$

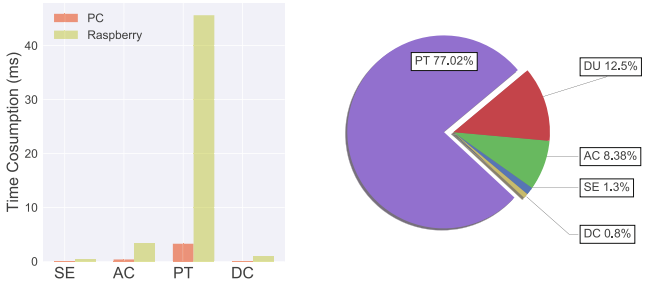


Fig. 3. Time consumption every process of proposed method running on different platform, where SE is sensor reading, AC is action calculation, DU is dynamic update, PT is policy train and DC is data collection.

episodes are utilized to train the critic and actor, completing the pre-training. The system noises d_i are set as random variances with expect zero and variance five for extending exploration. The event trigger is by-pass during pre-training for speeding up the rate of sample-capture.

4.2. Stability task

In this section, a stable control task is presented to verify the control performance of proposed method. During this simulation, the quadrotor UAV starts from the initial state $X = [-5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$ and then is guided by actions generated by the actor. The reference signals is set as a constant vector $X = [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$. The update algorithms for action, weights of actor, weights of critic are executed when the event-triggered happen. A random system noisy with variances 1 is added for increasing practicability, i.e.,

$$d_i(t) = w_i(t), i = 1, \ldots, 6 \tag{42}$$

where d_i are disturbances in the original system [1]; w_i are random variables with expects zeros and variances 1.

After the simulation, we find the quadrotor UAV is regulated to the set reference quickly and holds on the equilibrium, which are shown in Figs. 4 and 5. Meanwhile, there is no overshoot after the quadrotor UAV reaches the set point first time. From Fig. 6, we find that the actions for quadrotor UAV go stable after 2.5 s, keeping small changing for rejecting disturbances. The actions are updated discontinuous when the event trigger is satisfied, which saves calculation resources although causes chatters in angular rate in Fig. 5. The inter-event time between two consecutive triggered is shown in Fig. 7 and the ratio of triggered accounts

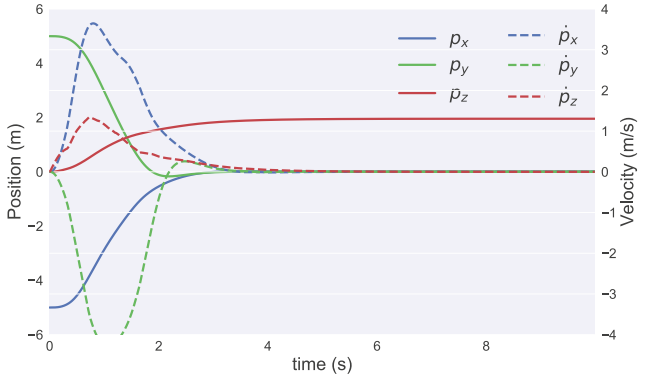


Fig. 4. Control performance about position and velocity of quadrotor UAV for stability task using proposed method.

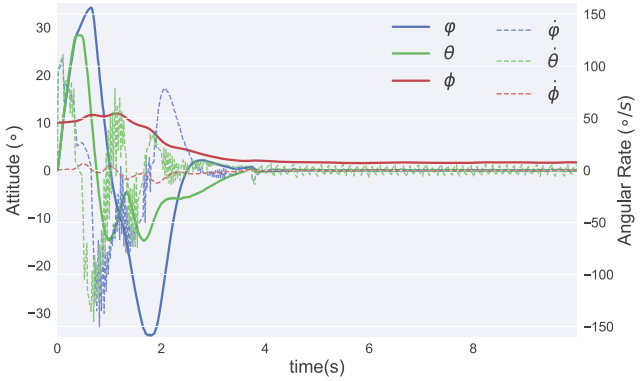


Fig. 5. Control performance about attitude and angular rate of quadrotor UAV for stability task using proposed method.

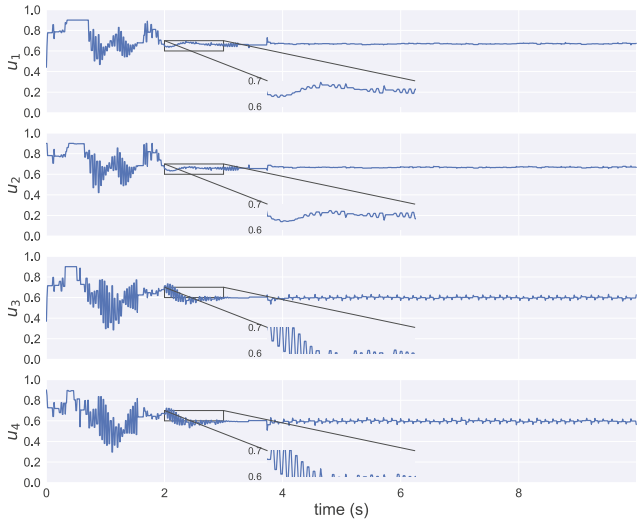


Fig. 6. Trajectory of the action for stability task using proposed method.

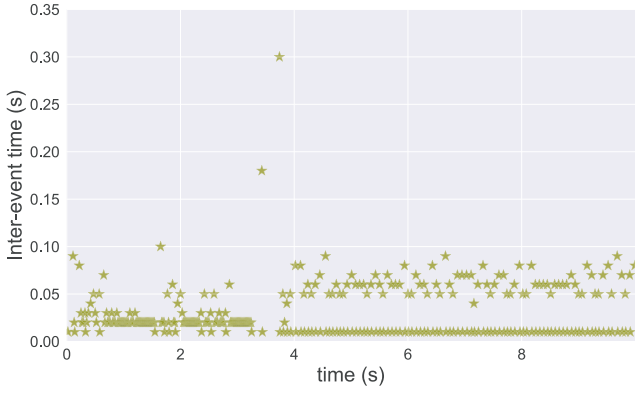


Fig. 7. Inter-event time during the stability task using proposed method.

to sampling accounts is 0.307 in this task. The proposed method can save 58.9%((1–0.307)*0.85) consumption of calculation at least, which is meaningful for declining cost for the on-board computer. The relationship between the gap $||e_j||$ and the threshold $||e_T||$ is shown in Fig. 8. It can be obtained that the gap is smaller than the threshold all the time, guaranteeing the stable of closed system. With the analysis above, the proposed method is able to

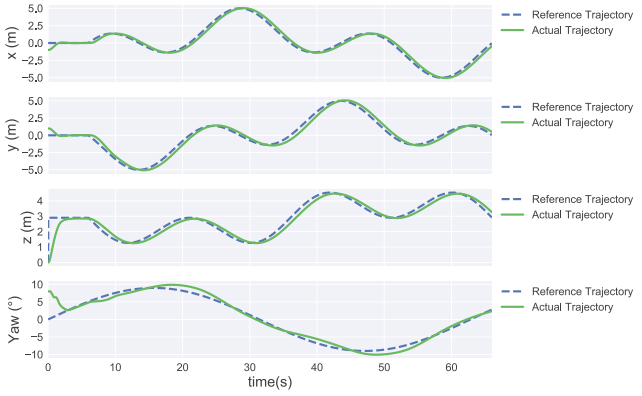


Fig. 8. Triggered error $||e_j||$ and triggered threshold $||e_T||$ during stability task.

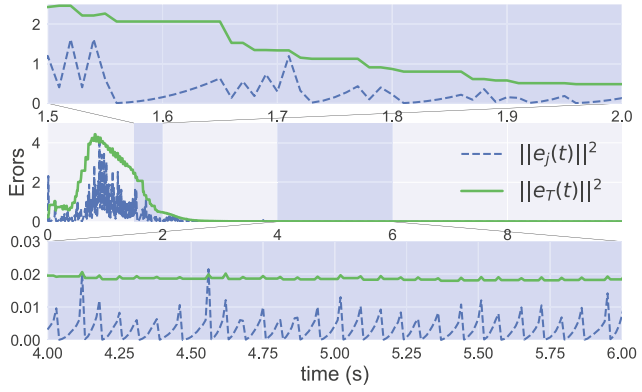


Fig. 9. Control performance about position and velocity of quadrotor UAV for tracking task using proposed method.

reduce the calculation consumption with expected control performance in the stabilized task.

4.3. Tracking task

In this section, the tracking performance of proposed method is studied. Firstly, the system state x_j supplied for actor and critic is necessary to modify, because these networks are pre-trained only with stabilized task. The system state used for action calculation should be errors between system state and reference signals:

$$x_0 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0.14(8^\circ) \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}, x_j' = \begin{bmatrix} x_{j,1} - r_x \\ x_{j,2} - r_y \\ x_{j,3} - r_z \\ x_{j,4} \\ x_{j,5} \\ x_{j,6} - r_\psi \\ x_{j,7} \\ \dots \\ x_{j,12} \end{bmatrix}, \tag{43}$$

where x_0 is initial system state and x_j' is the modified state supplied for actor and critic.

Then, a nonlinear trajectory with shape of clover is selected as reference signal, whose equations can be presented as:

$$\begin{aligned} &r_x, r_y, r_z, r_\psi = [0, 0, 2.9, 0], t \leqslant 5; \\ &r_x = -5\cos(0.209t + 0.785)\cos(0.105t + 0.785) \\ &r_y = 5\cos(0.209t + 0.785)\sin(0.105t + 0.785) \\ &r_z = 2 + 1.5\cos(0.209t + 0.785)\sin(0.105t + 0.785) \\ &-1.5\cos(0.209t + 0.785)\cos(0.105t + 0.785)) \\ &r_\psi = 0.157\sin(0.1t), t > 5, \end{aligned} \tag{44}$$

where r_x, r_y, r_z, r_ψ are the reference signals.

A 3D tracking result is presented intuitively on Fig. 12. Combined the Fig. 12 with Fig. 9, we known that the quadrotor UAV tracks the reference signal after taking off and keeps low tracking errors until the trajectory finishes. The attitude signals and control signals are shown in Fig. 10. We find that the attitude angles react fast to track the reference signals and keep smooth. According to sampling error and triggering threshold shown in Fig. 11, we get that the threshold is maintained during inter-event interval, which

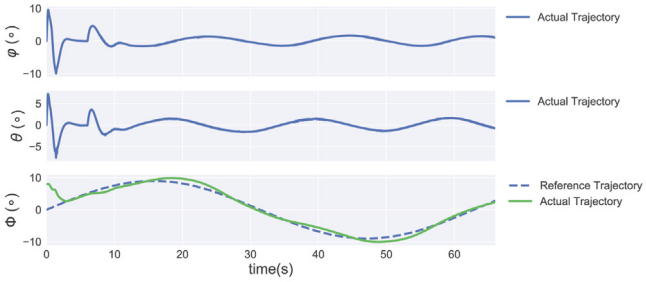


Fig. 10. Control performance about attitude and angular rate of quadrotor UAV for tacking task using proposed method.

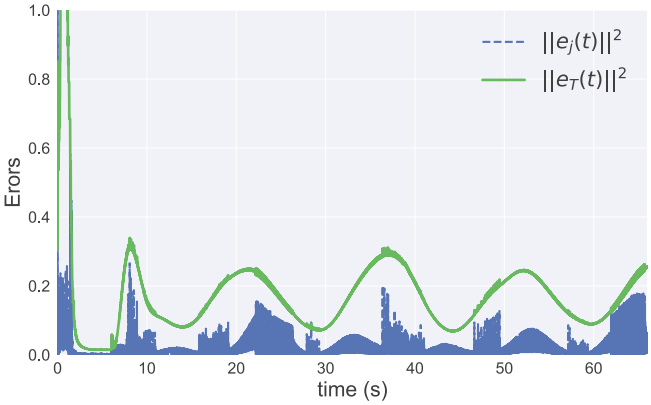


Fig. 11. Triggered error $||e_j|| = ||x_j - x(t)||$ and triggered threshold $||e_T||$ during tracking task.

is always larger than sampling error. Fig. 13 shows the inter-event time between two continuous tick. We find that the inter-event time exists and is uniform distributed. The ratio of triggered accounts to sampling accounts is 0.474 in this task, which is larger than the stabilized task for the reason that the actor has to adjust the control signals frequently to track the reference trajectory. The proposed method is able to regulate the quadrotor UAV to track a reference trajectory, conserving 44.7% calculation consumption meanwhile.

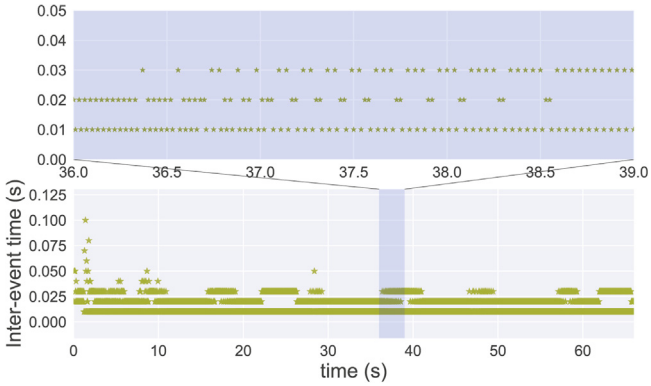


Fig. 12. 3d-trajectory of position during tracking task using proposed method.

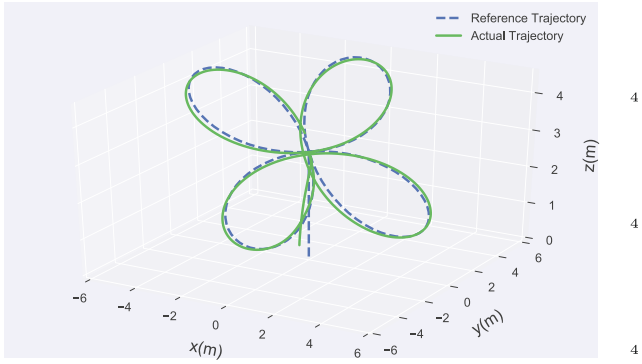


Fig. 13. Inter-event time during the tracking task using proposed method.

5. Conclusion

An ET-RL controller was proposed in this paper for the quadrotor UAV with actuator saturation. Two neural networks were utilized to approximate the performance index and the desired control policy. The ET technology was applied to reduce computation consumption, which can guarantee the real-time calculation on board to be possible. The actuator saturation was rejected with the modified reward function and actor. The stability for both closed system and NN convergence were analyzed by Lyapunov. A stabilized task and a tracking task were executed on simulation to show the effective of proposed method. Future research should be devoted to the development of the event-triggered containment RL controller for more complex system, for example, the multiple quadrotor UAVs[40,41].

CRediT author statement

Xiaobo Lin: Writing – original draft. **Jian Liu:** Writing – review & editing. **Yao Yu:** Supervision. **Changyin Sun:** Conceptualization.

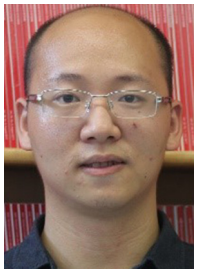
Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S.P. Lai, M.L. Lan, Y.X. Li, B.M. Chen, Safe navigation of quadrotors with jerk limited trajectory, Front Inf. Technol. Electron. Eng. 20 (1) (2019) 107–119.
[2] K. Chang, Y. Xia, K. Huang, D. Ma, Obstacle avoidance and active disturbance rejection control for a quadrotor, Neurocomputing 190 (2016) 60–69.
[3] M. Geisert, N. Mansard, Trajectory generation for quadrotor based systems using numerical optimal control, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 2958–2964.
[4] J. Liu, Y. Zhang, Y. Yu, C. Sun, Fixed-time event-triggered consensus for nonlinear multiagent systems without continuous communications, IEEE Trans. Syst. Man Cybern.: Syst. 49 (11) (2019) 2221–2229.
[5] A. L. Salih, M. Moghavvemi, H.A. Mohamed, K.S. Gaeid, Modelling and pid controller design for a quadrotor unmanned air vehicle, in: 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), vol. 1, IEEE, 2010, pp. 1–5.
[6] C. Powers, D. Mellinger, V. Kumar, Quadrotor kinematics and dynamics, in: Handbook of Unmanned Aerial Vehicles, Springer, 2015, pp. 307–328.
[7] A. Das, F. Lewis, K. Subbarao, Backstepping approach for controlling a quadrotor using lagrange form dynamics, J. Intell. Rob. Syst. 56 (1–2) (2009) 127–151.
[8] K. Peng, F. Lin, S.K. Phang, B.M. Chen, Nonlinear flight control design for maneuvering flight of quadrotors in high speed and large acceleration, in: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2018, pp. 212–221.
[9] F. Muñoz, I. González-Hernández, S. Salazar, E.S. Espinoza, R. Lozano, Second order sliding mode controllers for altitude control of a quadrotor uas: real-time implementation in outdoor environments, Neurocomputing 233 (2017) 61–71.
[10] H. Liu, Y. Tian, F.L. Lewis, Y. Wan, K.P. Valavanis, Robust formation tracking control for multiple quadrotors under aggressive maneuvers, Automatica 105 (2019) 179–185.
[11] X. Lin, Y. Yu, C.Y. Sun, A decoupling control for quadrotor uav using dynamic surface control and sliding mode disturbance observer, Nonlinear Dyn. 97 (2019) 781–795.
[12] M. Chen, S. Xiong, Q. Wu, Tracking flight control of quadrotor based on disturbance observer, IEEE Trans. Syst. Man Cybern.: Syst.
[13] S. Li, Y. Wang, J. Tan, Y. Zheng, Adaptive rbfnns/integral sliding mode control for a quadrotor aircraft, Neurocomputing 216 (2016) 126–134.
[14] Z. Li, X. Ma, Y. Li, Robust tracking control strategy for a quadrotor using rpd-smc and rise, Neurocomputing 331 (2019) 312–322.
[15] S. Bouabdallah, A. Noth, R. Siegwart, PID vs LQ control techniques applied to an indoor micro quadrotor, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), vol. 3, IEEE, 2004, pp. 2451–2456.
[16] R. Bellman, A markovian decision process, J. Math. Mech. (1957) 679–684.
[17] R. A. Howard, Optimization techniques in general markov decision programming, in: Dynamic Programming and Markov Processes, John Wiley, 1960, pp. 10–15.
[18] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Trans. Syst. Man Cybern. SMC-13(5) (1983) 834–846.
[19] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971.
[20] X. Zhong, Z. Ni, H. He, Gr-gdhp: a new architecture for globalized dual heuristic dynamic programming, IEEE Trans. Cybern. 47 (10) (2016) 3318–3330.
[21] C. Mu, D. Wang, H. He, Novel iterative neural dynamic programming for data-based approximate optimal control design, Automatica 81 (2017) 240–252.
[22] S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, arXiv preprint arXiv:1802.09477.
[23] J. Hwangbo, I. Sa, R. Siegwart, M. Hutter, Control of a quadrotor with reinforcement learning, IEEE Robot. Autom. Lett. 2 (4) (2017) 2096–2103.
[24] C. Mu, Y. Zhang, Learning-based robust tracking control of quadrotor with time-varying and coupling uncertainties, IEEE Trans. Neural Networks Learn. Syst. (2019) 1–15.
[25] X. Lin, Y. Yu, C.S. Sun, Supplementary reinforcement learning controller designed for quadrotor UAVs, IEEE Access 7 (2019) 26422–26431.
[26] P. Tabuada, Event-triggered real-time scheduling of stabilizing control tasks, IEEE Trans. Autom. Control 52 (9) (2007) 1680–1685.
[27] X. Zhong, H. He, An event-triggered ADP control approach for continuous-time system with unknown internal states, IEEE Trans. Cybern. 47 (3) (2017) 683–694.
[28] L. Dong, Y. Tang, H. He, C. Sun, An event-triggered approach for load frequency control with supplementary ADP, IEEE Trans. Power Syst. 32 (1) (2017) 581–589.
[29] J. Zhang, Z. Wang, H. Zhang, Data-based optimal control of multiagent systems: a reinforcement learning design approach, IEEE Trans. Cybern. 49 (12) (2018) 4441–4449.
[30] J. Zhang, H. Zhang, T. Feng, Distributed optimal consensus control for nonlinear multiagent system with unknown dynamic, IEEE Trans. Neural Networks Learn. Syst. 29 (8) (2017) 3339–3348.
[31] N. Szanto, V. Narayanan, S. Jagannathan, Event-sampled control of quadrotor unmanned aerial vehicle using neural networks, in: 2017 American Control Conference (ACC), IEEE, 2017, pp. 2956–2961.
[32] M. Abu-Khalaf, F.L. Lewis, Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach, Automatica 41 (5) (2005) 779–791.
[33] X. Yang, Y. Huang, D. Liu, Neural-network-based online optimal control for uncertain non-linear continuous-time systems with control constraints, IET Control Theory Appl. 7 (17) (2013) 2037–2047.

[34] Y. Zhu, D. Zhao, H. He, J. Ji, Event-triggered optimal control for partially unknown constrained-input systems via adaptive dynamic programming, IEEE Trans. Industr. Electron. 64 (5) (2017) 4101–4109.
[35] B. Convens, K. Merckaert, M.M. Nicotra, R. Naldi, E. Garone, Control of fully actuated unmanned aerial vehicles with actuator saturation, IFAC-PapersOnLine 50 (1) (2017) 12715–12720.
[36] Z.T. Dydek, A.M. Annaswamy, E. Lavretsky, Adaptive control of quadrotor uavs: a design trade study with flight evaluations, IEEE Trans. Control Syst. Technol. 21 (4) (2012) 1400–1406.
[37] B. Luo, D. Liu, T. Huang, Q-learning for optimal control of continuous-time systems, arXiv preprint arXiv:1410.2954.
[38] S. Shah, D. Dey, C. Lovett, A. Kapoor, Airsim: high-fidelity visual and physical simulation for autonomous vehicles, in: Field and Service Robotics, Springer, 2018, pp. 621–635.
[39] M. Roberts, Flashlight: a python library for analyzing and solving quadrotor control problems (2016).
[40] J. Liu, Y. Zhang, Y. Yu, C. Sun, Fixed-time leader-follower consensus of networked nonlinear systems via event/self-triggered control, IEEE Trans. Neural Networks Learn. Syst. early access Jan 6 (2020), https://doi.org/10.1109/TNNLS.2019.2957069.
[41] H. Liang, L. Zhang, Y. Sun, T. Huang, Containment control of semi-markovian multiagent systems with switching topologies, IEEE Trans. Syst. Man Cybern.: Syst.



Xiaobo Lin received his B.S. degree in automation from University of Science and Technology Beijing (USTB), China, in 2015. Now, He is a Ph.D. candidate in School of Automation and Electrical Engineering, University of Science and Technology Beijing. His current research interests include nonlinear control for quadrotor UAVs, attitude estimation, and reinforcement learning.

Jian Liu received his B.S. and Ph.D. degree from School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing, China, in 2015 and 2020. Currently, he is a postdoc with School of Automation, Southeast University, Nanjing, China. From September 2017 to September 2018, he was a joint training student with the Department of Mathematics, Dartmouth College, Hanover, NH 03755, USA. His current research interests include multi-agent systems, nonlinear control, event-triggered control

Yao Yu received the B.S. degree in control science and engineering from Huazhong University of Science and Technology, China, in 2004, and Ph.D. degree in control science and engineering from Academy of Tsinghua University in 2010. Now, he is an Associate Professor in School of Automation and Electrical Engineering, University of Science and Technology Beijing. His current research interests are robust control, multi-agent, and machine learning.

Chang-Yin Sun (M'17) received his B.S. degree in College of Mathematics, Sichuan University, Chengdu, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from the Southeast University, Nanjing, China, in 2001 and 2003, respectively. He is a Professor in School of Automation, Southeast University, Nanjing, China. He is the Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, Neural Processing Letters, IEEE/CAA Journal of Automatica Sinica. His current research interests include intelligent control, flight control, pattern recognition, and optimal theory.