Reinforcement Learning-based Intelligent Flight Control for a Fixed-wing Aircraft to Cross an Obstacle Wall

Li, Yifei; Van Kampen, Erik Jan

DOI
10.23919/ECC64448.2024.10591030

Publication date
2024

Document Version
Final published version

Published in
2024 European Control Conference, ECC 2024

2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Reinforcement Learning-based Intelligent Flight Control for a Fixed-wing Aircraft to Cross an Obstacle Wall

Yifei Li[1] and Erik-Jan van Kampen[2]

Abstract— **This paper develops an intelligent flight controller for a fixed-wing aircraft model in the longitudinal plane, using a Reinforcement Learning (RL)-based control method, namely Deep Deterministic Policy Gradient (DDPG). The neural network controller is fed the values of aircraft position, velocity, pitch angle and pitch rate, and outputs the elevator deflection. Artificial Neural Network (ANN)s are used to approximate the nonlinear state-action value function and the policy function. Simulation results show that the flight controller learns from the experienced data to fly over an obstacle wall with constrained pitch angle.**

I. INTRODUCTION

Reinforcement learning is a class of machine learning algorithms that achieves 'learning from interacting with the environment'. The basic principle of Reinforcement Learning (RL) is to maximize the the accumulated future rewards coming from interacting with the environment, by choosing an appropriate action at each time step[1]. The environment is ideally assumed to be a Markov Decision Process (MDP) with either observable or partial observable property. Random actions are taken in the training phase to explore different state transitions and at last to look for other locally maximal state/state-action values. Various reward mechanisms assist the agent to learn different behaviors in operation phase. For example, Ref [2], [3], [4], [5] set absolute value of the attitude angle tracking error at one time step as reward, the agent then learns to adjust vehicles's attitude angles to the reference. Ref [6], [7] set the quadratic value of attitude angles estimation error as reward, the agent learns to improve the estimate precision. Ref [8], [9] set the relative distances from controlled vehicles to target position as reward, the agent learns to plan approaching paths.

State-of-the-art RL algorithms can be briefly classified into two categories by training methods. The first one is value-based RL algorithms, such as Deep Q Network (DQN)[10], State-Action-Reward-State-Action (SARSA)[11], that start with learning a state value function(or state-action value function), then optimize a policy to maximize the learned state value function(or state-action value function). The disadvantage is the difficulty of exploring an infinite number of actions in a continuous action space. In contrast, policy-based RL algorithms employ the policy gradient to improve the policy, which improves the efficiency of policy search in a continuous action space. See Proximal Policy Optimization (PPO)[12], Trust Region Policy Optimization (TRPO)[13]) as commonly used policy-based RL algorithms.

Researches in recent years have combined value-based and policy-based RL to develop hybrid RL algorithms that use gradient descent in both state/state action-value function learning and policy search. Deep Deterministic Policy Gradient (DDPG) is one of these achievements and shows potential applications on controlling physical systems with continuous action spaces, such as spacecrafts[8], [14] and aerial vehicles[15], [16].

Specifically, Ref [14] uses Distributed Distributional Deep Deterministic Policy Gradient (D4PG) to train a spacecraft to dock with stationary and spinning targets. Ref [15] uses DDPG to learn obstacle-avoided paths for an Unmanned Aerial Vehicle (UAV) to move to a target destination. Ref [16] considers the control task for a compound aerial vehicle, i.e. a robotic arm suspended from a quadrotor. DDPG algorithm is used to train a control policy to make the robotic arm minimize its effect on the quadrotor dynamics, and achieve reference trajectory tracking task simultaneously. Ref [17] uses DDPG algorithm to design a flight control system for the open research civil aircraft model (RCAM). The performance of the flight controller is compared with other two robust flight controllers via Incremental Nonlinear Dynamical Inversion (INDI) and Proportional-Integral-Derivative Neural Network (PIDNN). The disadvantage of flight controllers designed by PID and INDI, lies in that a predetermined reference trajectory to be tracked is required and is difficult to design in some cases, such as the aircraft operating with constrained attitude angles and control inputs.

This paper considers the control task of a fixed-wing aircraft gliding over an obstacle wall. This task is challenging because there is no thrust for the aircraft. Therefore, the controller has to learn to generate an energy-saving trajectory, otherwise it will crash before arriving at the obstacle. In terms of control methods, traditional methods such as INDI require a predefined reference trajectory which manages to cross the obstacle. However, the specific flight environment is unpredictable so that such a reference trajectory is difficult to design. We propose to use RL-based control method to enable the controller learning an energy-saving trajectory from previous flight experience, without requiring any reference trajectories. The manually designed reward function in typical RL framework is capable of evaluating previous unsuccessful trajectories and assists tuning the controller parameters. As a result, the aircraft with an after-training controller is able to fly over an obstacle with least energy consumption.

[1]Yifei Li is with Faculty of Aerospace Engineering, Delft University of Technology, 2629HS, Delft, The Netherlands Y.Li-34@tudelft.nl

[2]Erik-Jan van Kampen is with the Faculty of Aerospace Engineering, Delft University of Technology, Delft, 2629HS, The Netherlands E.vanKampen@tudelft.nl

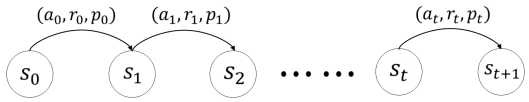

Fig. 1. Markov Decision Process (MDP)

The rest of this this paper is organized as follows. Section II introduces the Markov Decision Process model. Section III presents the basic principle of DDPG algorithm and three tricks to improve its performance. Section IV presents a simplified model of a fixed-wing aircraft in longitudinal plane. Section V provides the simulation results of the flight controller trained by DDPG algorithm in the task of crossing an obstacle wall. The conclusion of this paper is provided in Section VI.

II. Markov Decision Process

Markov Decision Process is a general dynamical model for a class of stochastic sequential decision processes, in which the cost and transition functions only depend on the current state of the system and the current action. This property is also named as Markov property of a dynamical system. A MDP model consists of four elements, i.e. state $s_t \in \mathbb{S} \subseteq \mathbb{R}^n$ is the state of system at time step t, $a_t \in \mathbb{A} \subseteq \mathbb{R}^m$ is the action at time step t taken by the agent according to a stochastic policy $\pi(s_t)$, which drives the state s_t to transfer to s_{t+1}. $p(s_{t+1}|s_t, a_t)$ is the transition probability density function from s_t to s_{t+1} with action a_t. $r(s_t, a_t)$ is the reward function that evaluates state s_t for taking action a_t. Define a set $\mathcal{D}$ of transition $(s_t, a_t, r_t, s_{t+1}, d)$, where d indicates whether state s_{t+1} is terminal.

Based on the definition of a MDP model, a RL agent tries to find a policy $\pi^*(\cdot)$ which maximizes the sum of rewards from each time step. An illustrative plot of MDP is given in Figure 1.

III. Deep Deterministic Policy Gradient Algorithm

A. Mathematical Principle of DDPG Algorithm

DDPG algorithm is an off-policy RL algorithm that is suitable for policy search in a continuous action space. Specifically, the Q-value network $Q^\pi_\psi(s_t, a_t)$ approximates the state-action value function $Q^\pi(s_t, a_t)$, which is used to calculate the expected cumulative reward for state s_t taking a specific action a_t, i.e. $Q^\pi(s_t, a_t) \triangleq \sum_{i=t}^{\infty} \mathbb{E}[\gamma^{i-t} r(s_i, a_i)]$. $r(s_t, a_t)$ is defined to be the reward of taking action a_t at state s_t. The Q-value neural network is parameterized by ψ. The policy neural network $\mu_\vartheta(s_t)$(parameterized by ϑ) is used to approximate the policy function $\pi(s_t)$.

The Bellman equation of $Q^\pi(s_t, a_t)$ is given as

$$Q^\pi(s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1} \in \mathbb{S}} [r(s_t, a_t) + \gamma \max_{a_{t+1}} Q^\pi(s_{t+1}, a_{t+1})] \quad (1)$$

where $s_{t+1} \in \mathbb{S}$ is the state in the environment at time step $t+1$. The transition from state s_t to state s_{t+1} follows the transition probability density function $p(s_{t+1}|s_t, a_t)$.

The optimal action a_t^* taken at time step t is solved by

$$a_t^* = \arg\max_{a_t} Q^\pi(s_t, a_t) \quad (2)$$

The Bellman equation in (1) can be used to train a Q-value network $Q^\pi_\psi(s_t, a_t)$, with a Mean Squared Bellman Error(MSBE) loss function

$$L(\psi, \mathcal{D}) = \mathop{\mathbb{E}}_{(s_t, a_t, r(t), s_{t+1}, d) \sim \mathcal{D}} \Big\{ \big[Q^\pi_\psi(s_t, a_t) - [r_t + \gamma(1 - d_t) \max_{a_{t+1}} Q^\pi_\psi(s_{t+1}, a_{t+1})] \big]^2 \Big\} \quad (3)$$

where $L(\psi, \mathcal{D})$ is a measure that describes how close the approximator $Q^\pi_\psi(s_t, a_t)$ performs with respect to $Q^\pi(s_t, a_t)$.

The training of policy network is based on the assumption that the action space is continuous so that Q value function is differentiable with respect to actor network parameters. As a result, various gradient descent methods can be used to learn a deterministic policy $\mu_\vartheta(s_t)$ which outputs the action series that maximizes $Q^\pi_\psi(s_t, \mu_\vartheta(s_t))$, i.e.

$$\vartheta^* = \max_\vartheta \mathop{\mathbb{E}}_{s_t \sim \mathcal{D}} [Q^\pi_\psi(s_t, \mu_\vartheta(s_t))] \quad (4)$$

In order for a RL agent randomly explores the action space $\mathbb{A}$ when interacting with the environment, the noise is added in action as

$$\mu'(s_t) = \mu_\vartheta(s_t) + \mathcal{N} \quad (5)$$

B. Three Tricks in DDPG Algorithm

1) Replay Buffer[18]: Replay buffer is a data pool to reserve the experienced Markov transition samples denoted with $(s_t, a_t, r_t, s_{t+1}, d)$. The samples from replay buffer are further used to update parameter sets ψ in critic network and ϑ in actor network.

2) Target Critic Network: In order to stabilize training process, target critic network is used with copied parameters from critic network. The target critic network has the same structure as critic network, but its parameters are updated differently. Define the target value(see Eq.(3)) as

$$r_t + \gamma(1 - d_t) \max_{a_{t+1}} Q^\pi_\psi(s_{t+1}, a_{t+1}) \quad (6)$$

which is a target for Q-value neural network to approach through minimizing the MSBE. Because the target depends on parameters ψ to be trained, the minimization of MSBE may be unstable. To solve this, the target network is used which has a time-delayed parameters compared to ψ. As a result, the parameter set ψ_{target} of target network is updated by

$$\psi_{\text{target}} \leftarrow \rho \psi_{\text{target}} + (1 - \rho)\psi \quad (7)$$

where ρ is a hyperparameter between 0 and 1.

Then, the MSBE loss is rewritten as

$$L(\psi,\mathcal{D}) = \underset{(s_t,a_t,r_t,s_{t+1},d)\sim\mathcal{D}}{\mathbb{E}}\Big\{\Big[Q_\psi^\pi(s_t,a_t) - [r_t + \gamma(1-d)Q_{\psi_{\text{target}}}^\pi(s_{t+1},\mu_{\vartheta(s_{t+1})})]\Big]^2\Big\} \tag{8}$$

3) Target Actor Network: The target actor network shares the same structure as the actor network. The parameter set $\vartheta_{\text{target}}$ of target actor network is updated by[18]

$$\vartheta_{\text{target}} \leftarrow \rho\vartheta_{\text{target}} + (1-\rho)\vartheta \tag{9}$$

Because the parameters of target actor are updated to slowly track the parameters of actor network, it can improve the stability of learning. Then, equation (8) can be rewritten as

$$L(\psi,\mathcal{D}) = \underset{(s_t,a_t,r_t,s_{t+1},d_t)\sim\mathcal{D}}{\mathbb{E}}\Big\{\Big[Q_\psi^\pi(s_t,a_t) - [r_t + \gamma(1-d)Q_{\psi_{\text{target}}}^\pi(s_{t+1},\mu_{\vartheta_{\text{target}}}(s_{t+1}))]\Big]^2\Big\} \tag{10}$$

As a result, Q-learning method is performed in DDPG algorithm by minimizing the MSBE loss in Eq.(10) with a gradient descent method. The pseudocode of DDPG algorithm is summerzied as follows.

C. Pseudocode

Algorithm 1: Deep Deterministic Policy Gradient

1: Initialization: policy network parameter set ϑ, Q network parameter set ψ, blank replay buffer $\mathcal{D}$
2: Initialization: target networks parameter sets $\vartheta_{\text{target}} \leftarrow \vartheta, \psi_{\text{target}} \leftarrow \psi$
3: **Repeat**
4: Observe state s_t and select action $a_t = \text{clip}(\mu_\vartheta(s_t) + \epsilon, a_t^{\text{Low}}, a_t^{\text{High}})$, where $\epsilon \sim \mathcal{N}$
5: Take action a_t to the present environment
6: Observe next state s_{t+1}, reward r_t, and done signal d_t to indicate whether s_{t+1} is terminal
7: Store $(s_t, a_t, r_t, s_{t+1}, d_t)$ in replay buffer $\mathcal{D}$
8: If s_{t+1} indicates terminal signal, reset all environment states.
9: **If** training is required **then**
10: **for** training steps **do**
11: From buffer $\mathcal{D}$, sample a batch of transitions with batch size n $\mathcal{B}_n = (s,a,r,s',d)^*$
12: Calculate targets with samples $(s,a,r,s',d) \in \mathcal{B}_n$ $z(r,s',d) = r + \gamma(1-d)Q_{\psi_{\text{target}}}^\pi(s', \mu_{\vartheta_{\text{target}}}(s'))$
13: Update Q-value network parameter set ψ by one-step gradient $\nabla_\psi \frac{1}{|\mathcal{B}_n|}\sum_{(s,a,r,s',d)\in\mathcal{B}_n}[Q_\psi^\pi(s,a) - z(r,s',d)]^2$
14: Update policy network parameter set ϑ by one-step gradient $\nabla_\vartheta \frac{1}{|\mathcal{B}_n|}\sum_{s\in\mathcal{B}_n} Q_\psi^\pi(s, \mu_\vartheta(s))$
15: Update target critic/actor network parameter sets $\psi_{\text{target}}, \vartheta_{\text{target}}$ $\psi_{\text{target}} \leftarrow \rho\psi_{\text{target}} + (1-\rho)\psi$ $\vartheta_{\text{target}} \leftarrow \rho\vartheta_{\text{target}} + (1-\rho)\vartheta$
16: **end for**
17: **end if**
18: **until** convergence

* (s,a,r,s',d) is a general representation of state transitions collected in replay buffer.

IV. Fixed-wing Aircraft Dynamical Model

The aircraft model considered in this paper is a fixed-wing perching UAV composed of foam and weights approximately 2 lb, which is capable of providing long-distance flights[19]. Model coefficients are provided in Table I.

$$\begin{aligned} m\ddot{x} =& -F_f\sin\theta - L_w\sin\gamma_w - D_w\cos\gamma_w - L_s\sin\gamma_s \\ & - D_s\cos\gamma_s \end{aligned} \tag{11}$$

$$\begin{aligned} m\ddot{y} =& F_f\cos\theta + L_w\cos\gamma_w - D_w\sin\gamma_w + L_s\cos\gamma_s \\ & - D_s\sin\gamma_s - mg \end{aligned} \tag{12}$$

$$\begin{aligned} I\ddot{\theta} =& -F_f d_f + d_{\text{cp}}L_w\cos(\theta-\gamma_w) + d_{\text{cp}}D_w\sin(\theta-\gamma_w) \\ & - d_sL_s\cos(\theta-\gamma_s) - d_sD_s\sin(\theta-\gamma_s) \end{aligned} \tag{13}$$

where I is the moment of inertia about center of gravity, θ is aircraft pitch angle, F_f is fuselage aerodynamic force, d_{cp} is center of pressure moment arm, L_s, D_s are stabilator lift and drag forces. d_f is fuselage moment arm, d_s is stabilator moment arm. γ_w is angle made by wing velocity and wing horizontal velocity, γ_s is angle made by stabilator velocity and stabilator horizontal velocity. γ_w, γ_s are defined as

$$\gamma_w = \arctan\frac{\dot{y}_w}{\dot{x}_w} \tag{14}$$

$$\gamma_s = \arctan\frac{\dot{y}_s}{\dot{x}_s} \tag{15}$$

and (x_w, y_w) and (x_s, y_s) are positions of wing and stabilator surface area centroids in a longitudinal body axis[13]:

$$(x_w, y_w) = (x + d_w\cos\theta, y + d_w\sin\theta) \tag{16}$$

$$(x_s, y_s) = (x - d_s\cos\theta, y - d_s\sin\theta) \tag{17}$$

The angles of attack of the wing (α_w), the stabilator (α_s), and the fuselage (α_f) are defined as:

$$\alpha_w = \theta - \gamma_w + \alpha_i \tag{18}$$

$$\alpha_s = \theta - \gamma_s + \phi \tag{19}$$

$$\alpha_f = \theta - \arctan(\frac{\dot{y}_f}{\dot{x}_f}) \tag{20}$$

where α_i is the wing incidence angle measured relative to the longitudinal body axis, ϕ is the stabilator deflection, and (x_f, y_f) is the position of the fuselage centroid.

In Ref[19], the detailed models are developed for wing lift force L_w and drag force D_w, stabilator lift force L_s and drag force D_s, the aerodynamic force F_f, and center of pressure, which are used in the following numerical simulation.

V. Simulation Results

This section provides the simulation results of the flight controller trained by DDPG algorithm for the task of crossing an obstacle wall. In the training phase, the initial states of the aircraft are set to be $[x_0, \dot{x}_0, y_0, \dot{y}_0, \theta_0, \dot{\theta}_0] = [10m, 8m/s, 10m, -0.5m/s, 0°, 57.29°/s]$, where θ represents the pitch angle. The aircraft tries to cross an obstacle wall with the height 10.2m and position $x_{\text{obstacle}} = 17m$.

TABLE I

AIRCRAFT PHYSICAL COEFFICIENTS

Parameter	Value
Mass m	0.88kg
Moment of intertia I	0.30 kg$\cdot$ m^2
Fuselage length L	0.38m
Stabilator span S_s	0.40m
Stabilator moment arm d_s	0.49m
Fuselage moment arm d_f	0.53m
Wingspan S_w	1.41m
Wing chord c_w	0.21m
Wing centroid to center of gravity d_w	0.29m

A. Reward Shaping

The destination of reward shaping is to reward the states and actions that are helpful to finish the final task of crossing the obstacle wall, and penalize the states and actions that hinder the final task. Intuitively, to cross the obstacle wall with the height $10.2m$, the aircraft is encouraged to fly over a certain height. Therefore, the reward function is designed to reward the aircraft altitude over $10m$ before the obstacle, which is the initial altitude of the aircraft. To reward the successful crossing maneuver, the altitude of aircraft at $17m$ is multiplied with a parameter a to emphasize its importance. Meanwhile, to penalize the crossing maneuver with large pitch angles, the cost is designed to be a quadratic form of pitch angle, multiplied with an importance parameter b. Then the reward function is designed as

$$\text{reward} = \begin{cases} y - 10, & x \neq 17 \\ a(y - 10.2) - b|\theta|^2, & x = 17 \end{cases} \tag{21}$$

B. Neural Network Implementation

The implementation of DDPG algorithm is based on the usage of 4 artificial neural networks, namely critic network, target critic network, actor network and target actor network, the nonlinear activation function is selected to be ReLU function. In the first and second hidden layers, the numbers of neuron are 400, 300, respectively. The schematics of critic and actor networks are provided in Figure 3 and Figure 4, respectively. The target critic(actor) network shares the same structure as the critic(actor) network. The optimizer used to achieve stochastic gradient descent for critic and actor networks is Adaptive Moment Estimation(Adam)[23]. The hyperparameters are provided in Table II.

TABLE II

HYPERPARAMETERS OF DDPG AGENT

Parameters	Value
learning rate(critic)	0.00001
learning rate(actor)	0.00001
batch size	64
ρ	0.99
γ	0.99
a	10000
b	500

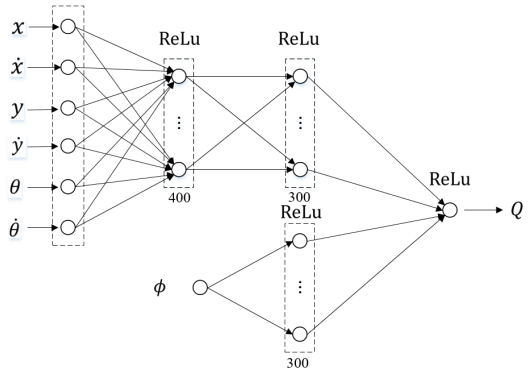

Fig. 2. Critic Neural Network

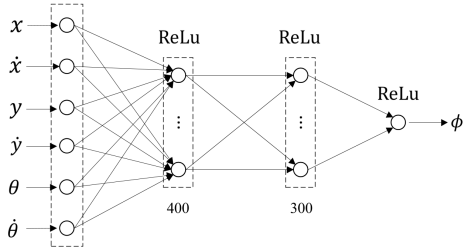

Fig. 3. Actor Neural Network

C. Online Training Phase

This subsection provides the training results in the task of crossing an obstacle wall. Figure 5 plots the score history of 100 training episodes. The first episode in which the crossing task is successful is episode 66 with a score of 200.24. By adding a randomized noise to the stabilator deflection before each step of the agent interacting with the environment, the agent learns to find other locally maximal Q values in episodes 72, 79 and 87.

D. Online Operation Phase

This subsection verifies the online operation performance of the flight controller trained in the previous online training phase. The flight condition considered is the initial positions with uncertainties. This is common in practical cases because the simulated trajectory is in the terminal phase of the aircraft and its initial positions may be affected by the flight conditions in previous flight phases. Specifically, it is assumes that the initial position variables x_0, y_0 follow Guassian distributions as $x_0 \sim \mathcal{N}(10m, 0.5m)$, $y_0 \sim \mathcal{N}(10m, 0.5m)$.

Figure 6 provides the score history of 100 episodes. The high scores does not necessarily equal successful crossings because the initial position variables are different. By observing the flight data collected in replay buffer, the successful rate is 57%, which means that in 57 of 100 simulated online operations the trained flight controller controls the aircraft to cross the obstacle wall.

Figures 7, 8 provide the averaged reward history in various time periods(plotted with the solid line), as well as its max-min bounds that make the shaded region. The max-min bounds in [85,105] time steps are large (see the first sub-

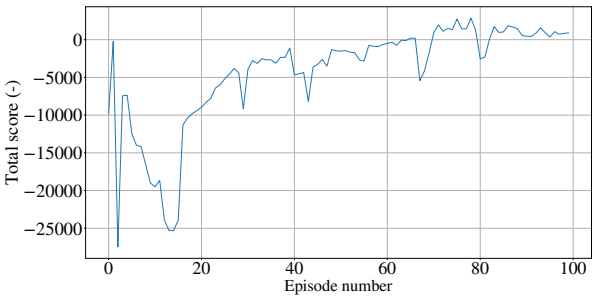

Fig. 4. Score history of 100 episodes in online training phase.

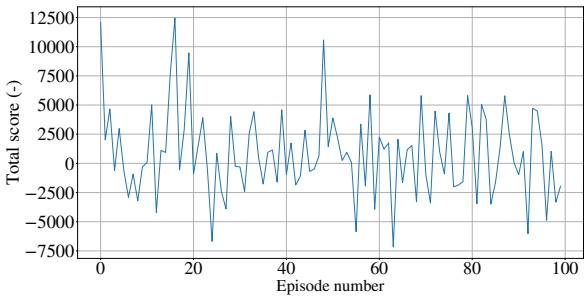

Fig. 5. Score history of 100 episodes in online operation phase. The task successful rate is 57%.

figure in Figure 7) because the parameters a, b are used to amplify the reward made by altitude and the cost made by pitch angle above the wall. In simulation, the obstacle wall is set to be a line without width, but the large max-min bounds last for 20 time steps(0.2s). This is a result of different arriving times to the obstacle wall in 100 episodes.

Figures 9, 10, 11 provide the histories of averaged states, $\overline{x}(t), \overline{y}(t), \overline{\dot{x}}(t), \overline{\dot{y}}(t), \overline{V}(t)$ and the history of averaged action $\overline{\phi}(t)$(plotted with the solid line). In the second sub-figure of Figure 9, the shaded region over 10.2m indicates the successful crossing over the wall.

VI. Conclusion

The aircraft flight controller trained by DDPG algorithm achieves the task of crossing the obstacle wall with a 57%

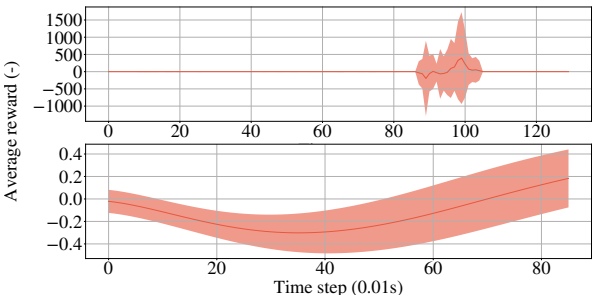

Fig. 6. The curve of reward in two periods of time: [0,130] and [0,85] time steps. The solid line is the averaged evaluation of reward, the shaded region is made by its max-min bounds within 100-episode data.

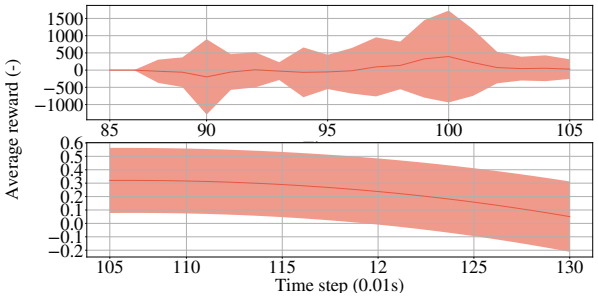

Fig. 7. The curve of reward in two periods of time: [85,105] and [105,130] time steps. The solid line is the averaged evaluation of reward, the shaded region is made by its max-min bound within 100-episode data.

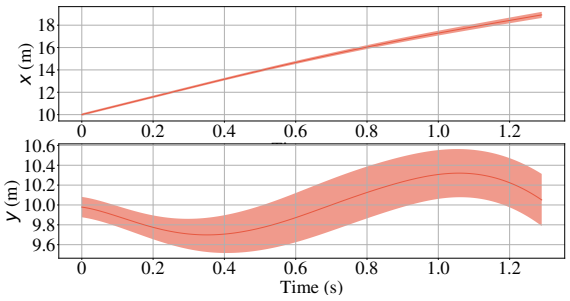

Fig. 8. The histories of averaged $x(t), y(t)$. The shaded region is made by their max-min bounds.

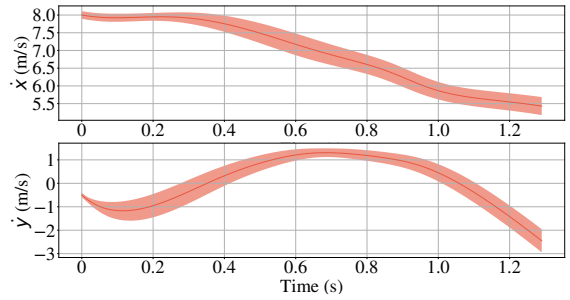

Fig. 9. The histories of averaged $\dot{x}(t), \dot{y}(t)$. The shaded region is made by their max-min bounds.

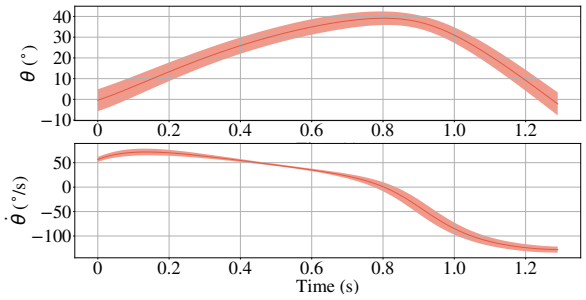

Fig. 10. The histories of averaged $V(t), \phi(t)$. The shaded region is made by their max-min bounds.

successful rate, under the condition of uncertain initial position variables x_0, y_0. Designing the reward function carefully leads to a flight trajectory that balances between the reward terms and the cost terms.

REFERENCES

[1] R.S.Sutton, A.G.Barto, Reinforcement learning: An introduction. Cambridge, Massachussetts: The MIT Press, 2018, ch1.

[2] W.Koch, R.Mancuso, R.West, and A.Bestavros, Reinforcement learning for UAV attitude control, ACM Transactions on Cyber-Physical Systems, vol.3, no.2, pp.1-22, Feb, 2019.

[3] K.Dally, E.van Kampen, Soft actor-critic deep reinforcement learning for fault-tolerant flight control, In AIAA SciTech 2022 Forum, San Diego, Jan, 2022, pp. 2022-2078.

[4] K.Shayan, E.van Kampen, Online actor-critic-based adaptive control for a tailless aircraft with innovative control effectors, In AIAA SciTech 2021 Forum, Virtual Event, Jan, 2021, pp. 2021-0884.

[5] S.Heyer, D.Kroezen, E.van Kampen, Online adaptive incremental reinforcement learning flight control for a CS-25 class aircraft, In AIAA SciTech 2020 Forum, Orlando, Jan, 2020, pp. 2020-1844.

[6] Y.Tang, L.Hu, Q.Zhang, and W.Pan, Reinforcement learning compensated extended kalman filter for attitude estimation, IEEE/RSL International Conference on Intelligent Robots and Systems(IROS), Prague, Sept, 2021, pp. 6854-6859.

[7] L.Hu, Y.Tang, Z.Zhou, and W.Pan, Reinforcement learning for orientation estimation using inertial sensors with performance guarantee, IEEE International Conference on Robotics and Automation(ICRA), Xi'an, May, 2021, pp.10243-10249.

[8] K.Hovell, S.Ulrich, Deep reinforcement learning for spacecraft proximity operations guidance, Journal of Spacecraft and Rockets, vol.58, no.2, pp.254-264, Mar, 2021.

[9] A.Zavoli, L.Federici, Reinforcement learning for robust trajectory design of interplanetary missions, Journal of Guidance, Control and Dynamics, vol.44, no.8, pp.1440-1453, Aug, 2021.

[10] V.Mnih, K.Kavukcuoglu, D.Silver, et.al, Human-level control through deep reinforcement learning, Nature, vol.518, pp.529-540, Feb, 2015.

[11] R.S.Sutton, Generalization in reinforcement learning: successful example using sparse coarse coding, Advances in Neural Information Processing Systems(NIPS), The MIT Press, 1996, pp.1038-1044.

[12] J.Schulman, F.Wolski, P.Dhariwal, A.Radford, O.Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347, 2017.

[13] J.Schulman, S.Levine, P.Moritz, M.Jordan, P.Abbeel, M.Jordan, and P.Moritz, Trust region policy optimization, 32st International Conference on Machine Learning(ICML), Lille, Jul, 2015, pp.1889-1897.

[14] K.Hovell, S.Ulrich, On reinofrcement learning for spacecraft guidance, In AIAA SciTech Forum, Orlando, Jan, 2020, pp.2020-1600.

[15] O.Bouhamed, H.Ghazzai, et.al, Autonomous UAV navigation: a ddpg-based deep reinforcement learning approach, IEEE International Symposium on Circuits and Systems, Seville, October, 2020.

[16] Y.Liu, C.Huang, DDPG-based adaptive robust tracking control for aerial manipulators with decoupling approach, IEEE Transactions on Cybernetics, vol.52, no.8, pp.8258-8271, Jul, 2022.

[17] D.Milz, G.Looye, Design and evaluation of advanced intelligent flight controllers, In AIAA SciTech 2020 Forum, Orlando, Jan, 2020, pp. 2020-1846.

[18] T.P.Lillicrap, J.J.Hunt,et.al. Continuous control with deep reinforcement learning, 4th International Conference on Learning Representation(ICLR), San Juan, May, 2016.

[19] M.Puopolo, J.D.Jacob, Model for longitudinal perch maneuvers of a fixed-wing unmanned aerial vehicle, Journal of Aircraft, vol.52, no.6, pp.1-11, Nov, 2015.

[20] S.A.Brandt, R.J.Stiles, J.J.Bertin and R.Whitford, Introduction to Aeronautics: A Design Perspective. American Institute of Aeronautics and Astronautics, 2004, pp.118-123.

[21] E.N.Jacobs, K.E.Ward, and R.M.Pinkerton, The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel. NACA Report-460, 1933.

[22] D.J.Tritton. Physical Fluid Dynamics, 2nd ed., New York, US: Oxford University Press Inc., 1988, pp.32-33.

[23] D.P.Kingma, J.L.Ba, Adam: a method for stochastic optimization, 3rd International Conference on Learning Representations(ICLR), San Diego, May, 2015, pp.1-15.