ICAS 2021 SHANGHAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

LEARNING FLIGHT CONTROL SKILLS BY IMITATING EXPERT DEMONSTRATIONS

Ryoichi Takase[1]

[1]Department of Aeronautics and Astronautics, The University of Tokyo, Japan

Abstract

In the applications of reinforcement learning to flight control tasks, it is hard to design an appropriate reward function through trial and error. In order to learn policies without reward function engineering, this paper deals with the flight controller design by using inverse reinforcement learning. Two learning algorithms are investigated on the flight control tasks. The first one is generative adversarial imitation learning (GAIL). The policy is obtained by imitating expert demonstrations. The second method is adversarial inverse reinforcement learning (AIRL). In contrast to GAIL, AIRL can recover the estimated reward function, which enables us to obtain the reward function for achieving a good maneuver like human experts. The simulation results show that both methods successfully learn policies and obtain almost the same flight control performance as the human expert.

Keywords: Flight control, imitation learning, inverse reinforcement learning

1. Introduction

With the increasing demand for introducing highly autonomous systems into the aviation industry, much attention has been paid to the flight controller design using reinforcement learning, e.g. [1, 2, 3]. During the training of reinforcement learning, the agent is optimized through the interaction with the environment. The tuning parameters such as the neural networks are updated by maximizing the expected total reward. The trained policy achieves complex flight maneuvers [2] and outperforms the conventional PID control [3]. These results suggest the effectiveness of flight controller design using reinforcement learning.

In reinforcement learning, reward design is one of the key points to successfully train the agent since the policy is optimized by evaluating whether the task is achieved based on the reward from the environment. However, in practice, it is difficult to design an appropriate reward function. The difficulty is especially seen if it is required to solve a complex task and the number of observations/actions is increased. An inappropriate reward function leads to obtain the agent that acts undesired behaviors, resulting in the situation where the policy does not satisfy the criterion required for flight control systems.

In order to overcome the difficulty of reward design for flight control, this paper trains the agent by using the data of expert demonstrations. This means that it is not required to design a reward function. In this study, two methods are used for flight control tasks. The first one is generative adversarial imitation learning (GAIL) [4]. The control policy is obtained by imitating expert demonstrations. The second method is adversarial inverse reinforcement learning (AIRL) [5]. In contrast to GAIL, AIRL can recover the estimated reward function, which enables us to obtain the reward function for achieving a good maneuver like human experts.

The layout of this paper is as follows. Section 2 describes the outline of inverse reinforcement learning. Section 3 describes the flight controller design from the human expert demonstrations. Section 4 describes the results and discussion of the controller, and section 5 ends the paper with the conclusions.

2. Preliminaries

In this paper, we use generative adversarial imitation learning (GAIL) [4] and advasarial inverse reinforcement learning (AIRL) [5]. The controller $\pi : \mathbb{R}^{n_G} \to \mathbb{R}^{n_u}$ is an ℓ-layer neural network with nonlinear activation functions in hidden layers:

$$w^0(k) = s(k); \tag{1a}$$

$$w^i(k) = \phi^i(W^i w^{i-1}(k) + b^i), \quad i = 1, ..., \ell; \tag{1b}$$

$$a(k) = W^{\ell+1} w^{\ell}(k) + b^{\ell+1}, \tag{1c}$$

where $w^i \in \mathbb{R}^{n_i}$ is the output from the i^{th} layer. Symbols $W^i \in \mathbb{R}^{n_i \times n_{i-1}}$ and $b^i \in \mathbb{R}^{n_i}$ are respectively the weight matrix and the bias of the i^{th} layer in the neural network. Symbol $\phi^i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is the activation function which is applied element-wise for a given vector v:

$$\phi^i(v) := [\varphi(v_1), ..., \varphi(v_{n_i})]^T, \tag{2}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is the scalar activation function, e.g., $\tanh$. In this paper, the neural network controller defined by Eq. (1) is updated by trust region policy optimization (TRPO) [6]. In this paper, the controller π is also referred to as the policy in accordance with the field of RL researches.

The discriminator $D_\omega(s,a)$ discriminates the expert trajectories τ_{exp} and the trajectories generated by the controller τ_π. The objective function is given as follows.

$$\max_{\omega} \mathbb{E}_{(s,a)\sim\tau_\pi}[\log(D_\omega(s,a))] + \mathbb{E}_{(s,a)\sim\tau_{exp}}[\log(1 - D_\omega(s,a))] \tag{3}$$

We supply $-\log(D_\omega(s,a))$ to policies as a reward during training. After the training, the trajectories generated by the policy will be similar to that of the experts. See [4, 5] for detailed descriptions of the algorithms.

3. Controller Design from Expert Demonstrations

3.1 Controller Design Framework

This section describes the experiment setups for designing flight controllers from expert demonstrations. The overview of the flight simulation framework is shown in Fig. 1. In this study, two programming languages are used. The first one is MATLAB/Simulink, which is used in real-time simulations in order to collect the data of time histories of aircraft states controlled by a human expert. The second programming language is Python, which is used for training policies from the expert demonstrations in desktop simulations.

The key point of the design framework is to share aircraft dynamics and problem definitions (e.g. controlled models, control objectives, and evaluation criteria) in MATLAB/Simulink and Python. In inverse reinforcement learning, the agent is trained by interacting with the environment including aircraft dynamics. If the data of the expert demonstrations are obtained in MATLAB/Simulink, the environment implemented by Python for training is required to be the same as that of MATLAB/Simulink as possible. By connecting two environments, the advantages of each programming language are provided for controller designers. The detailed description is given as follows.

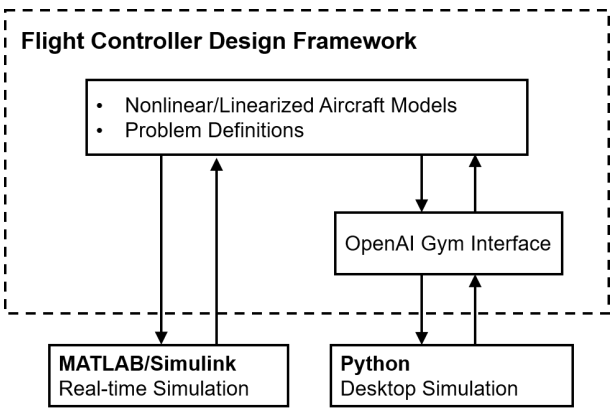


Figure 1 – Overview of flight controller design framework.

(a) MATLAB/Simulink Implementation

MATLAB/Simulink is used to carry out a high-fidelity nonlinear simulation in real-time. The main Simulink file consists of three subsystem blocks. The first subsystem is the hardware interface block which obtains pilot inputs from the wheel and pedal in the cockpit. It calculates the surface deflections and gives the output signals of the electronic commands. The second subsystem is the aircraft dynamics block. Based on the commands from the first subsystem block, the second subsystem calculates aircraft motion and gives the observation. Note that the input-output relationship of the second subsystem is corresponding to aircraft dynamics implemented by Python. The third subsystem is the visual interface for feedbacking current aircraft state to pilots, see [7] for the detailed descriptions. By using the Simulink file which is connected to the flight simulator as shown in Fig. 2, the human-in-the-loop simulations are carried out to collect the data of expert demonstrations. The collected data is used for training agents using inverse reinforcement learning algorithms implemented by Python.

Figure 2 – Fixed-based flight simulator.

(b) Python Implementation

Python is used to train agents using inverse reinforcement learning algorithms. In order to improve the convenience of using machine learning libraries, the flight controller design framework has the interface of OpenAI Gym. OpenAI Gym[1] is a toolkit for developing and comparing reinforcement learning algorithms with the aim to make a benchmark for assessing different methods [8]. The interface of OpenAI Gym defines the observation and action spaces, which can be easily used with machine learning libraries (e.g. TensorFlow [9] or PyTorch [10]) and has been used in the fields of the robotics [11] as well as the unmanned aerial vehicle [3, 1, 12]. Through the OpenAI Gym interface, the advanced learning algorithms implemented in Python can be used for the controlled models whose properties are the same as the ones in MATLAB/Simulink.

3.2 Aircraft Model

The aircraft model used in this study is a high fidelity nonlinear model of a large transport aircraft. It was developed by [13] with the aim to design auto-landing controllers (see also [14, 15]). Following a standard approach, the equations of aircraft motion are obtained from the dynamics and kinematics equations. The dynamics of the engines and the actuators are modeled as the first-order filters with magnitude and rate limits. In this aircraft model, the aircraft states X, the control input U, and the wind disturbance W are given as follows.

$$X = [p,\ q,\ r,\ \phi,\ \theta,\ \psi,\ u,\ v,\ w,\ x,\ y,\ z,\ \delta_t,\ \delta_a,\ \delta_e\ \delta_r]^T \tag{4}$$

$$U = [\delta_{tc}, \delta_{ac}, \delta_{ec}, \delta_{rc}]^T \tag{5}$$

$$W = [w_x,\ w_y,\ w_z]^T \tag{6}$$

[1] http://gym.openai.com

where p, q, r, ϕ, θ, ψ, u, v, w, x, y, z, δ_t, δ_a, δ_e, δ_r are respectively the roll rate, the pitch rate, the yaw rate, the bank angle, the pitch angle, the heading angle, the x-axis velocity, the y-axis velocity, the z-axis velocity, the X-axis position, the Y-axis position, the Z-axis position, the exhaust pressure ratio, the aileron deflection, the elevator deflection, and the rudder deflection. Symbols δ_{tc}, δ_{ac}, δ_{ec}, and δ_{rc} are respectively the exhaust pressure ratio command, the aileron deflection command, the elevator deflection command, and the rudder deflection command. Symbols w_x, w_y, w_z are respectively the wind velocities in the x-, y-, and z-axes.

The initial flight conditions in simulations are set to a level flight at an altitude of 300 m. The trim conditions are calculated by using the *MATLAB function* provided by [13]. The simulation sampling interval of the aircraft motion is set to $T_z = 0.05$ s.

3.3 Flight Control Task

The flight control task in this study is to change the pitch angle from the initial trim condition of 7.68 deg to the target pitch angle of 5.0 deg in the presence of turbulence. For simplicity, the input commands for the exhaust pressure ratio and for the lateral motion (i.e. aileron and rudder) are set to 0 in the simulations. Thus, the control input is the elevator deflection command. Regarding the observation of this flight control task, the measured outputs of the longitudinal motion are selected and supplied to the policy. The observation is summarized in Table 1.

Table 1 – Observation of the changing pitch angle task.

#	sym.	description	unit
0	θ_t	target pitch angle	rad
1	N_x	longitudinal load factor	m/s^2
2	N_z	vertical load factor	m/s^2
3	q	pitch rate	rad/s
4	θ	pitch angle	rad
5	α	angle of attack	rad
6	V_a	true airspeed	m/s
7	V_z	inertial vertical airspeed	m/s
8	H	altitude	m

3.4 Hyperparameters

The algorithms of inverse reinforcement learning are implemented by author using Python. The policy network is modeled by a fully-connected multi-layer perceptron with two hidden layers of 64 units and $\tanh$ as the activation function. The hyperparameters are selected as in Table 2 to obtain adequate flight control performance. Following a standard procedure, the exploration noise is added to actions during the training phase while deterministic actions are performed without the exploration noise during the evaluation phase. For the training, we use ten trajectories of the human expert demonstrations which are obtained in real-time simulations.

4. Results and Discussion

In this paper, the following key points are discussed to evaluate the policies obtained by inverse reinforcement learning algorithms.

a). Relationships between the observation supplied to the policy during training and its final performance

b). Comparison between the behavior of the policy and that of the human expert

Regarding a), we discuss the relationship between the dimension size of observation and the difficulty of the training process. Reference [16] suggests that reducing the dimension of observation by selecting only the essential elements (e.g. the pitch angle and the pitch rate) helps the training of

Table 2 – Hyperparameters of the algorithms on the changing pitch angle task.

	GAIL	AIRL
Timesteps per iteration	2048	2048
Mini batch size	2048	2048
Discount factor	0.99	0.99
GAE discount	0.98	0.98
Policy network hidden layers	[64,64]	[64,64]
Value network hidden layers	[64,64]	[64,64]
Value network LR	3×10^{-4}	3×10^{-4}
Policy network LR	3×10^{-4}	3×10^{-4}
Num. epochs	10	10
Gradient clipping	0.5	0.5
KL-divergence limit	0.01	0.01
Damping coeff.	0.01	0.01
Backtrack coeff.	0.8	0.8
Backtrack iters	10	10
Cg iters	10	10
Disc. update iters	30	30
Disc. LR	3×10^{-4}	3×10^{-4}
Disc. network hidden layers	[100,100]	[100,100]
Disc. mini batch size	2048	2048

flight control policy. In this paper, we investigate the relationships in the case of inverse reinforcement learning algorithms. Three cases are investigated regarding the dimension of the observation. The first case is to provide the observation $obs = [\theta_t,\ q,\ \theta]$ to the policy during its training. The elements are selected in order to determine control input based on the current deviation from the desired pitch angle (i.e. $\theta_t - \theta$) and the change rate of the pitch angle. The first case is the minimum dimension size in this experiment. The second case is $obs = [\theta_t,\ N_z,\ q,\ \theta]$, which is added the vertical load factor N_z to the first case. The selection of the elements was determined by trial and error. The third case is that the elements of the observation are fully provided to the policy.

Regarding b), we discuss the time histories of aircraft states controlled by the policy. In addition to the flight control performance given by calculating an evaluation metric (e.g. mean squared error), the similarity to the expert behavior is investigated. Even if the flight control performance is almost the same as the expert demonstrations, undesired behaviors (e.g. give too large control inputs) are not acceptable in the sense of flight safety.

4.1 Results of GAIL

The learning curve of the GAIL on the pitch control task is shown in Fig. 3, where the solid line corresponds to the average and the shaded region to the minimum/maximum returns (i.e. the sum of all rewards in one episode) of evaluation rollouts over the three different random seeds. At each simulation step, the reward is give by $r = -0.5q^2 - 15.0(\theta_t - \theta)^2 - 1.0\delta_{ec}^2$ and the current episode is terminated with adding the penalty $r = -100$ if $|\theta_t - \theta| > 4.0$ deg, which means that control performance is better as reaching to $r = 0$. Note that the returns are only depicted so as to show the performance check and are not used for the training. The red line shows the behavior of the policy trained by providing the observation of $obs = [\theta_t,\ q,\ \theta]$, the blue line $obs = [\theta_t,\ N_z,\ q,\ \theta]$, and the green line all observations. The yellow solid line shows the average return of the human expert.

In the cases of $obs = [\theta_t,\ q,\ \theta]$ and $obs = [\theta_t,\ N_z,\ q,\ \theta]$, the obtained policies perform comparably to the human expert after the training of the total environment steps of 3×10^5. It is confirmed that reducing the observation size by selecting the essential elements for the pitch control task helps the training and improves the performance of the policy. On the other hand, supplying all observations for training the policy leads to poor performance and less sample efficiency.

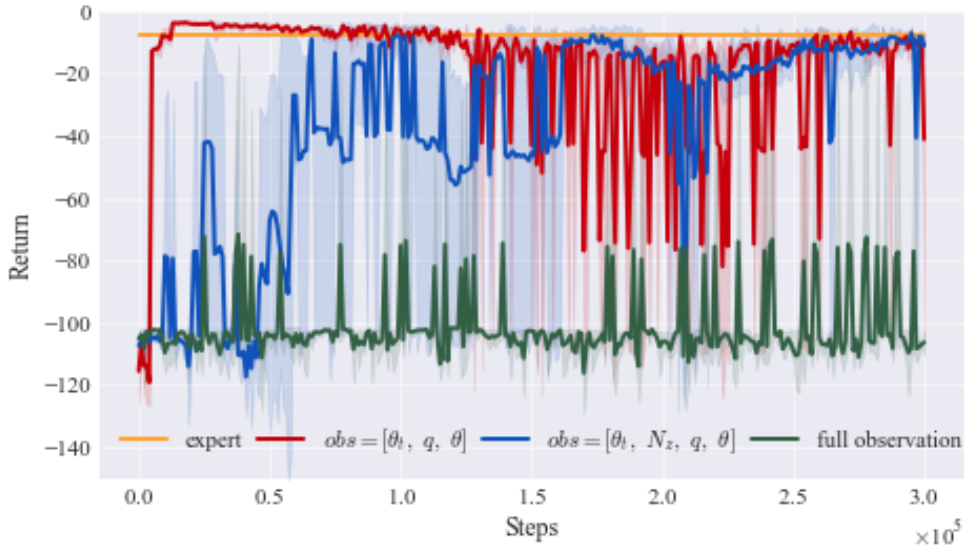


Figure 3 – Learning curve of GAIL on the changing pitch angle task.

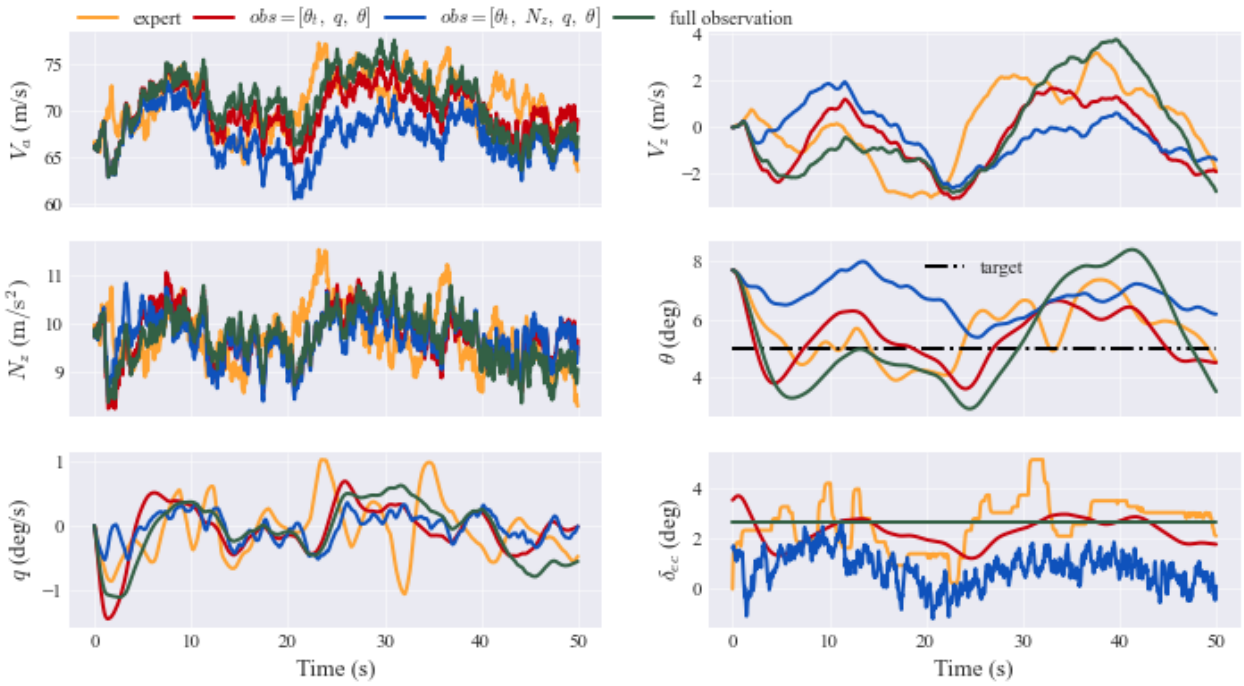


Figure 4 – Time histories of aircraft states of GAIL on the changing pitch angle task.

Figure 4 shows the time history of aircraft states on the changing pitch angle task. The expert demonstration, whose simulation is carried out in MATLAB/Simulink, is also plotted for reference. In the subplot regarding the pitch angle, the target pitch angle of $\theta_t = 5.0$ deg is plotted with the black dash-dot line. The difference of the behaviors between $obs = [\theta_t,\ q,\ \theta]$ and $obs = [\theta_t,\ N_z,\ q,\ \theta]$ is seen in the time histories of the elevator deflection commands. In the case of $obs = [\theta_t,\ q,\ \theta]$, the pitch angle is kept to the target one with acceptable elevator deflection commands. On the other hand, in the case of $obs = [\theta_t,\ N_z,\ q,\ \theta]$, the unsmoothed elevator deflection commands are provided to control the pitch angle. This is considered to be due to the disturbance since the control input is determined by the vertical load factor affected by turbulence. In the case of providing all elements of the observation for the policy training, the behavior is not acceptable since the elevator deflection

command is fixed at about the average of the expert command.

4.2 Results of AIRL

The learning curve of AIRL is shown in Fig. 5, whose plot layout is the same as in Fig. 3. It is confirmed that the performance and the sample efficiency are similar to those of GAIL. Selecting the elements of the observation helps the policy training and improves the performance. Figure 6 shows the time histories of the aircraft states controlled by AIRL. In the case of $obs = [\theta_t,\ q,\ \theta]$, the behavior is acceptable and the flight control task is successfully achieved.

Figure 7 shows the heat map of the estimated reward function in the case of $obs = [\theta_t,\ q,\ \theta]$. The heat map shows larger values at the pitch angle of around 5.0 to 6.0 deg and at the pitch rate of around -1.0 to 0.0 deg/s. The learned reward function encourages the pitch angle to keep 5.0 deg with a smaller pitch rate, which corresponds to the flight control task to be achieved. Thus, as the reward function, the flight control skill is estimated from the human expert demonstrations. By estimating the reward function during the training, AIRL performs comparably to the human expert.

5. Conclusion

In this paper, we have investigated the application of inverse reinforcement learning to flight control tasks. By selecting only the essential elements such as the pitch angle and the pitch rate, GAIL and AIRL successfully obtain the control policies from the human expert demonstrations. The performance of both algorithms is almost the same as the human expert, whose behaviors are similar to the expert demonstrations. In order to obtain more comprehensive results, our future works will include simulations for both longitudinal and lateral directional motions at the same time with more human expert demonstrations and more maneuvers.

References

[1] W. Koch, R. Mancuso, and A. Bestavros, "Neuroflight: Next generation flight control firmware," *arXiv preprint arXiv:1901.06553*, 2019.

[2] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[3] 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. 1–21, 2019.

[4] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4572–4580, 2016.

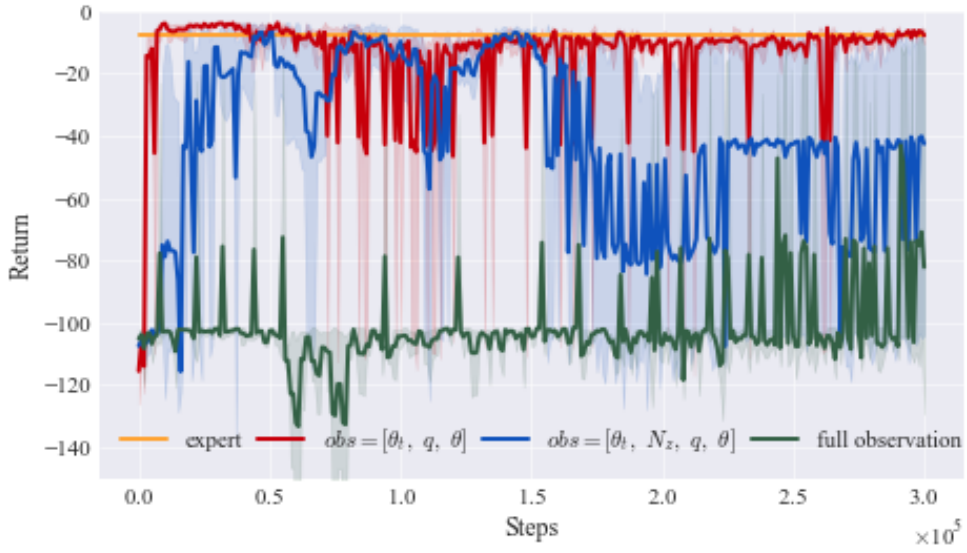


Figure 5 – Learning curve of AIRL on the changing pitch angle task.

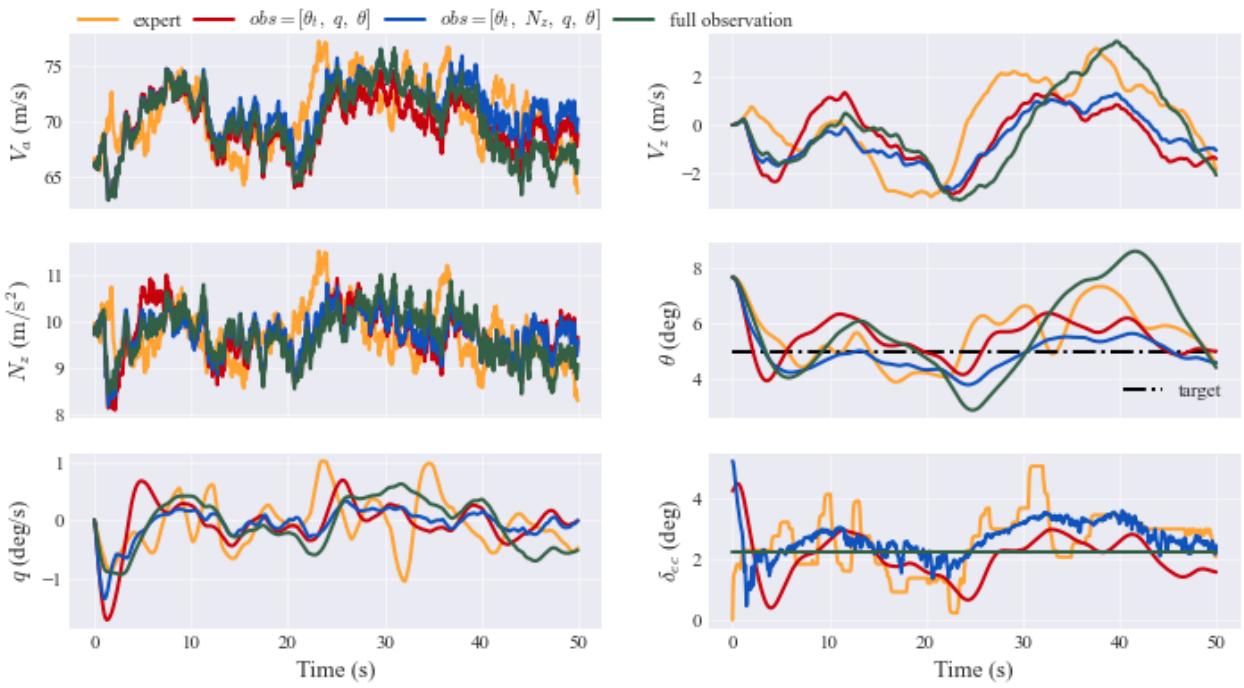
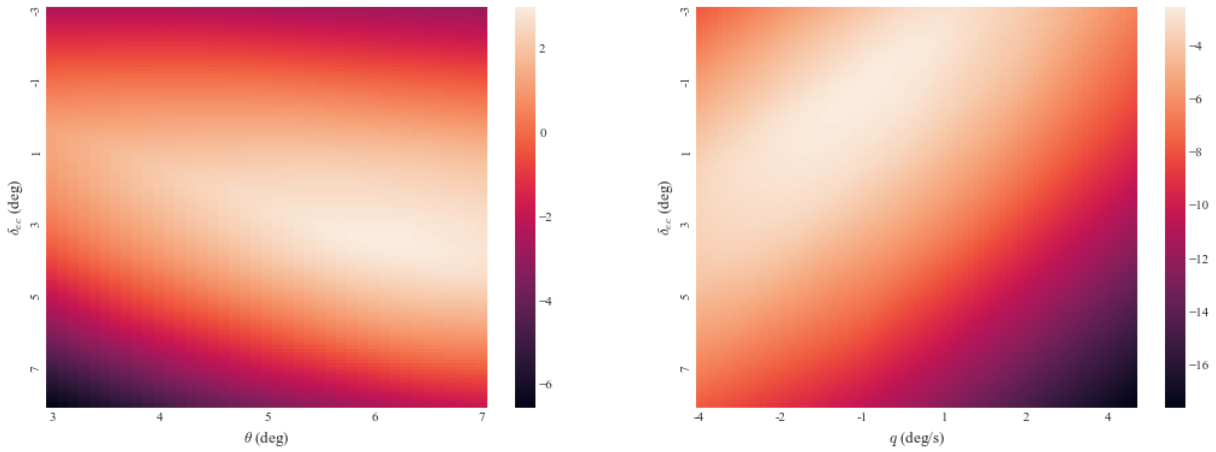


Figure 6 – Time histories of aircraft states of AIRL on the changing pitch angle task.



(a) Heat map with θ_t and δ_{ec}.

(b) Heat map with q and δ_{ec}.

Figure 7 – Reward function learned by AIRL on the changing pitch angle task.

[5] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.

[7] J. O. Entzinger and T. Tsuchiya, "Construction of a research flight simulator and example applications for control evaluation (japanese)," in *58th Aircraft Symposium of JSASS*, 2020.

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[9] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[11] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint arXiv:1608.05742*, 2016.

[12] S. Krishnan, B. Borojerdian, W. Fu, A. Faust, and V. J. Reddi, "Air learning: An ai research platform for algorithm-hardware benchmarking of autonomous aerial robots," *arXiv preprint arXiv:1906.00421*, 2019.

[13] O. AIRBUS, "A civilian aircraft landing challenge," 2016.

[14] J.-M. Biannic and C. Roos, "Flare control law design via multi-channel $\mathscr{H}_\infty$ synthesis: Illustration on a freely available nonlinear aircraft benchmark," in *2015 American Control Conference (ACC)*, pp. 1303–1308, IEEE, 2015.

[15] J. Theis, D. Ossmann, and H. Pfifer, "Robust autopilot design for crosswind landing," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3977–3982, 2017.

[16] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 523–533, IEEE, 2019.

Contact Author Email Address

mailto: takase-aero@g.ecc.u-tokyo.ac.jp

Copyright Statement