RESEARCH ARTICLE

Open Access

Uniaxial attitude control of uncrewed aerial vehicle with thrust vectoring under model variations by deep reinforcement learning and domain randomization

Atsushi Osedo[1*], Daichi Wada[1] and Shinsaku Hisada[1]

Abstract

The application of neural networks for nonlinear control has been actively studied in the field of aeronautics. Successful techniques have been demonstrated to achieve improved control performance in simulation using deep-reinforcement learning. To transfer the controller learnt in the simulation of real systems, domain randomization is an approach that encourages the adaptiveness of neural networks to changing environments through training with randomized parameters in environments. This approach applies to an extended context, with changing working environments, including model configurations. In previous studies, the adaptive performance of the domain-randomization-based controllers was studied in a comparative fashion over the model variations. To understand the practical applicability of this feature, further studies are necessary to quantitatively evaluate the learnt adaptiveness with respect to the training conditions. This study evaluates deep-reinforcement-learning and the domain-randomization-based controller, with a focus on its adaptive performance over the model variations. The model variations were designed to allow quantitative comparisons. The control performances were examined, with a specific highlight of whether the model variation ranges fell within or exceeded the randomization range in training.

Keywords Deep reinforcement learning, UAV, Control

Introduction

In aeronautics, neural network(NN)-based controllers that utilize machine learning have the potential to replace existing controllers in terms of improving the efficiency of control system construction and responding flexibly to disturbances. Supervised learning is often applied to replace the role of existing controllers and is expected to provide benefits such as more efficient gain scheduling and smoother control [1]. In contrast, deep reinforcement learning is expected to be viable when a teaching controller does not exist or to improve the performance beyond that of the existing one [2, 3].

The application of NNs for nonlinear control has been actively studied in robotics and aeronautics. Particularly, simulations have demonstrated techniques that enable complex tasks such as trajectory planning [4], aircraft landing under wind-induced disturbance [5], swarm flight [6], aerobatics [7], and fixed-wing aircraft attitude control [8] by optimizing the NN through deep reinforcement learning.

However, it is necessary to design an appropriate approach to transfer the controllers learnt in the simulations to real systems. This is because deep reinforcement learning generally optimizes the NN for a single environment represented by the simulator, and the theoretical control performance is often not achieved when the real

*Correspondence:
Atsushi Osedo
osedo.atsushi@jaxa.jp
[1] Aviation Technology Directorate, Japan Aerospace Exploration Agency, Tokyo, Japan

environment differs from the simulation (Reality gap), such as friction or time delay [9]. Sim-to-real transfer is an approach that applies a simulation-generated controller to a real-world environment without compromising its performance, and has been actively studied, specifically in the field of robotics [10]. Domain randomization is one of the approaches in the sim-to-real transfer technique. For example, domain randomization randomly changed the appearance [11, 12] and dynamics [13, 14] of objects from episode to episode in the simulation during training. NN is optimized to adaptively control the agent in these changing environments. In aeronautics, there have been examples of domain randomization in multirotor flight control [15] and attitude control using thrust vectoring [16–18].

Generally, domain randomization is used to maintain theoretical control performance in the real world; however, its feature of making controllers adaptive can be utilized to an extended application, where working environments, including model configurations, change. In previous studies, sim-to-real transfer was applied to the flight control of uncrewed aerial vehicles (UAVs), which took multiple forms of model configurations [17, 18]. The control performance of the domain-randomization-based controllers has been successfully studied in a comparative fashion over the multiple model variations with different physical characteristics [17, 18]. Moreover, it will also be essential to evaluate the "working range" to what extent the controller can apply to changes in the environment. The relationship between the range of randomization in training and the learnt tolerance to environmental variations should be examined quantitatively. These understandings would be a practical guideline for the application of domain-randomization-based controllers to a real situation, where a certain degree of uncertainties is expected, such as payload variations as well as UAV failure or damage. The aim of this research is to apply deep-reinforcement-learning and the domain-randomization- based controller to aircraft control systems. In the development of aircraft control systems, it is difficult to construct a precise model of the entire aircraft due to turbulence and increasingly complex systems, and furthermore, verification through experiments is costly and carries high safety risks. Therefore, if it becomes possible to develop control systems based on simulators without the need for precise models, there is an expectation that it could lead to cost reduction and shortened development time, among other benefits. As a first step, This study evaluates deep-reinforcement-learning and the domain-randomization-based controller, with a focus on its adaptive performance over the model variations. The model variations were designed to allow quantitative comparisons. The attitude control system for UAV based on thrust vectoring, which was constructed in our previous research [16], was utilized. The weight, inertia, and the location of the center of gravity(CoG) of the model were varied. The control performances were examined with a specific highlight of whether the model variation ranges fell within or exceeded the randomization range in training.

UAV with thrust vectoring

UAV with thrust vectoring and experimental system

Figure 1 shows a developed UAV with thrust vectorings and Fig. 2 shows the system configuration. The UAV has four control outputs, which are two Electric Ducted Fans (EDFs)(JP Hobby, 120 mm and 14 CELL Motor 673 KV) and two actuators (FUTABA, HPS-A700) for thrust vectoring to deflect the thrust directions of each EDF. Two 11.1 V batteries (MATRIX, LiPo 6 s-5100 mAh 35 C) were connected in series for each of these EDFs to provide a 22.2 V power supply. The maximum thrust per EDF was 7.25 N and the maximum run time was 5 min. The operating range of thrust vectorings is ±25°.

The UAV is controlled by a Raspberry Pi 3 Model B and Pixhawk 4. The Pixhawk senses the UAV states (position, attitude, etc.) and controls the actuators. The Raspberry Pi performs control operations using NN. First, all sensor information measured by Pixhawk is sent to Raspberry Pi. Next, the NN that is implemented in the Raspberry Pi calculates the control outputs for the actuators based on the sensor information from the Pixhawk, which are sent again to the Pixhawk. The operating cycles of Pixhawk and Raspberry Pi are 200 Hz and 50 Hz, respectively.

Moreover, to enable the verification of the response of the control system to changes in the CoG, weights can be mounted outside of the fuselage. The weight, center of gravity, and moment of inertia of the UAV can be varied by changing the number and mounting position of the weights. These control devices, batteries, and control

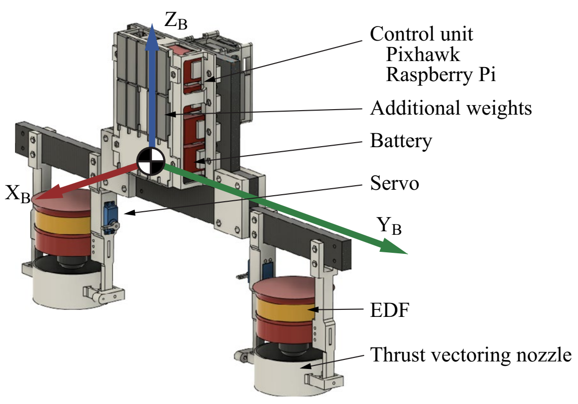

Fig. 1 Developed UAV with thrust vectoring (3D CAD model)

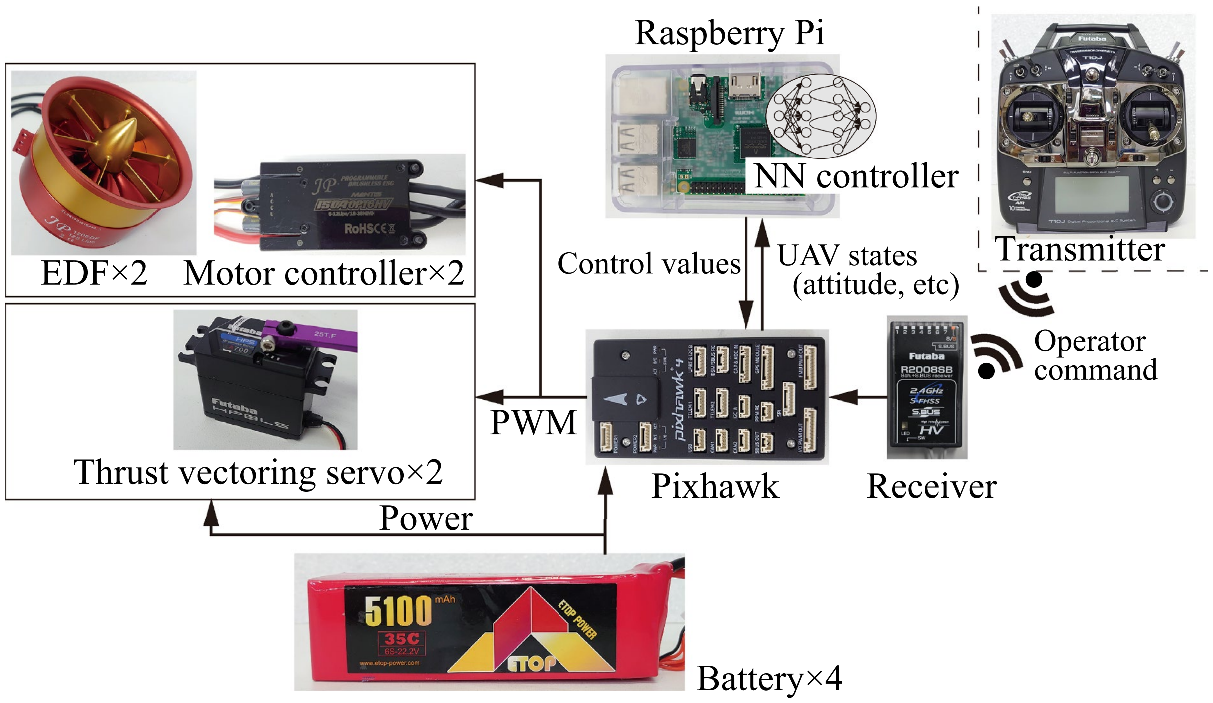

Fig. 2 System configuration

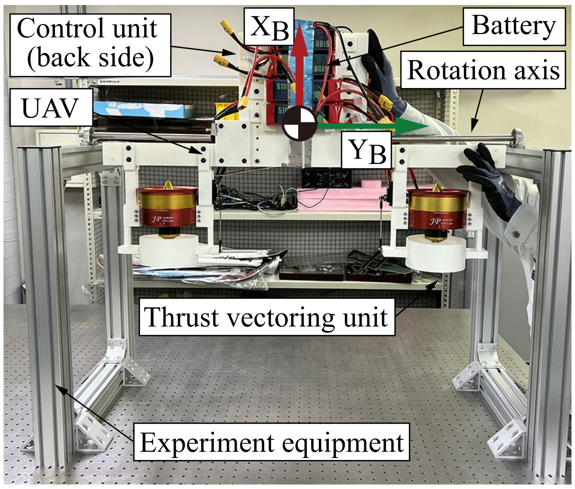

Fig. 3 Experimental setup

equipment are fixed to a plastic fuselage created by a 3D printer. Consequently, the total weight of the UAV without the additional weights is 7.0 kg.

The experimental system is shown in Fig. 3. It system consists of the UAV and a uniaxial rotation experimental device. The UAV is attached to the experimental device so that it can only rotate around the pitch (Y_b) axis. If the UAV is not weighted, the distance between the CoG and axis of rotation is +49 mm in the X-axis direction and +28 mm in the Z-axis direction. The behaviors of the two EDFs and thrust vectors are set to be the same, which sufficed the control task where the UAV only rotated around the pitch axis Therefore, the control output is effectively two variables.

Theoretical model

Since this experiment is a uniaxial rotation test without translational movement, the experimental system of the UAV is modeled in the XZ plane in the global coordinate system. The rotational motion model in the XZ plane is shown in Fig. 4, and the equation of rotational motion around the Y_b axis is shown below.

$$I_{yy}\ddot{\theta} = mgR_G \sin(\theta + \theta_G) - F_x R_T \cos\theta_T - F_z R_T \sin\theta_T, \tag{1}$$

$$R_G = \sqrt{x_G^2 + z_G^2}, \tag{2}$$

$$R_T = \sqrt{x_T^2 + z_T^2}, \tag{3}$$

$$\theta_G = \tan^{-1}\left(\frac{x_G}{z_G}\right), \tag{4}$$

$$\theta_T = \tan^{-1}\left(\frac{x_T}{z_T}\right), \tag{5}$$

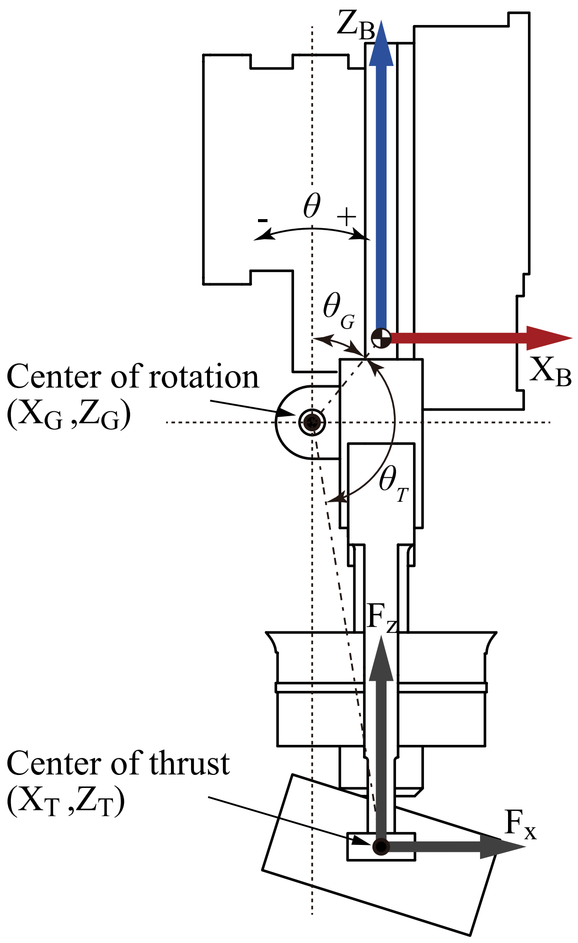

Fig. 4 Theoretical model

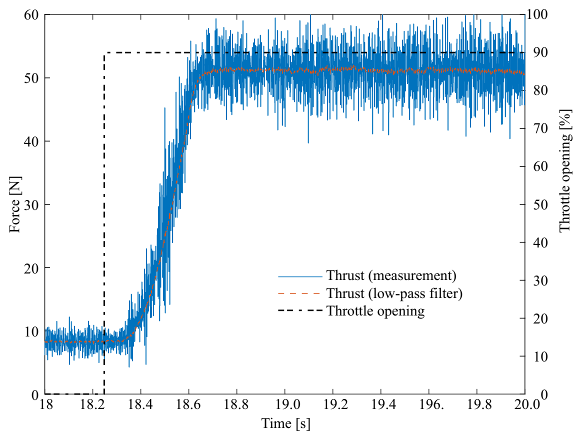

Fig. 5 Thrust response to throttle command

where I_{yy} is the fuselage moment of inertia around the Y_b axis of the aircraft body coordinates, m is the mass of the entire aircraft, g is the gravitational acceleration, F_x and F_z are the forces that divide the thrust F in the XZ plane, $G(x_g, z_g)$ is the position of the COG of the UAV with respect to the center of rotation in the XZ plane. Because the center of rotation of the experimental setup does not coincide with the center of gravity of the UAV, R_G, R_T, θ_G, and θ_T were defined and transformed as in Eqs. (1-5).

Second, since there is a time delay between the command input and the generation of thrust force in the EDF, the thrust generated by the EDFs is also modeled. It is noted that the counter torque of the EDF is ignored because it does not affect the motion of the UAV in this experiment. Thrust response measurement experiments are conducted to evaluate the relationship between the throttle commands and thrust force. The thruster is connected to a force sensor, and the throttle command and the generated thrust are measured in time series. The measurement results are shown in Fig. 5. The first axis shows thrust and the second axis shows throttle opening, and the solid line indicates the result of measurement by the force sensor, the dashed line indicates the result of filtering the measured values, and the single-dot dashed line indicates the throttle opening. As a result of the analysis of the measurement results, the transfer function of the thruster is approximated by a first order lag model as follows,

$$R(s) = e^{-sT_d}\left(\frac{A}{\tau s + 1}\right), \tag{6}$$

where $R(S)$ is the transfer function from throttle signal to thrust force, A is the gain for converting the throttle signal to thrust, which was obtained from experimental results by system identification, T_d is the dead time, and τ is the damping time constant. In this EDF units, A is 0.58, T_d is 0.08 s and τ is 0.19 s.

Next, the response of the vectoring servo to the input was modeled. With a vectoring nozzle attached to the servo, a command (pulse width modulation signal) was applied using a step signal spanning from − 50% to +50%, and the resulting servo angle response was measured. In this servo, the servo rotates − 25 ° when the command is set to -50%, and the servo rotates +25 ° when the command is set to +50%. Furthermore, system identification was performed based on the measurement results, and the response of the servo was modeled as a first order lag system. The model of the servo is shown in Eq.(7).

$$G(s) = \frac{K_s}{\tau_s s + 1}, \tag{7}$$

where $G(s)$ represents the transfer function from the servo command signal to the servo angle (in radians), K_s denotes the gain with a value of 1.024, and τ_s represents the damping time constant with a value of 0.109.

Comparison of experimental systems and theoretical model

The differences between the experimental system and theoretical model are evaluated. The constructed theoretical model is implemented in a dynamics simulator developed in python to simulate the rotational motion of the UAV. In the two models, the models were released from the initial pitch angle and oscillated without external forces. The free vibration response is shown in Fig. 6.

Figure 6 shows that the responses of the experimental system and theoretical model are different. The solid and broken lines depict the experimental system and theoretical model vibrations, respectively. The vibration period is 6.7% faster for the simulation and the damping ratio is 32% higher for the simulator. This difference is the typical reality gap. In the domain randomization approach, rather than tuning the theoretical model towards higher fidelity, the controller is trained to be robust to the difference.

Generation of controller with deep reinforcement learning

The controller was generated with the NN structure shown in Fig. 7. The inputs (State) to the NN is "pitch angle"; "difference between the pitch angle and target pitch angle"; "pitch angle velocity"; "thrust vectoring angle"; and "thrust". Here, the two EDFs and the two thrust vectorings are set to have the same behavior; therefore, the "thrust vectoring angle," and "thrust" are one element each. Moreover, the "pitch angle" and "difference between pitch angle and target pitch angle" are separated into sine and cosine components so that they have smooth value transitions within ±1, resulting in a total of seven input components. The controller outputs are the "thrust vectoring angle" and "thrust force"; however, the NN outputs were designed as the rate of thrust vectoring and force, which are the differences between the next and current states. By this definition, the NN outputs were directly used in a penalty function as described later.

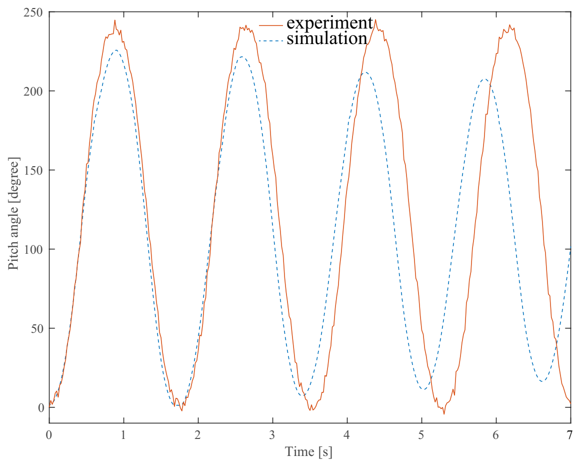

Fig. 6 Comparison of free oscillation between experimental system and theoretical model

The long short-term memory (LSTM) layer is set in the NN for effective learning when using domain randomization [13, 14, 19]. Under domain randomization, the Markov decision process is not necessarily guaranteed. Therefore, it becomes essential to use memory-augmented policies [13, 14, 20], or input time series of states to policies [21, 22]. This study employs the approach of using LSTM by following its recent success in domain randomizaiton [13, 14]. It is assumed that the LSTM layer enables adaptive control by implicitly recognizing changing dynamics from the stored histories of the system's inputs and outputs. The hidden layers other than the LSTM are with the exponential linear unit (ELU) activation function, the output layer of the policy function is with the hyperbolic tangent (tanh) activation function, and the output layer of the value function is with the linear activation function.

During the NN training, the simulation is conducted at 50 Hz for 5 s, that is, one episode of 250 steps. Different initial conditions (initial attitude, initial angular velocity, etc.), dynamics, and data processing delay time are given for each episode. Initial conditions are generated by uniform random numbers, with ranges of $\pm25°$ for the target pitch angle, $\pm180°$ for the pitch angle, and $\pm18°$ for the pitch angle velocity, respectively. Particularly, the thrust vectoring angle and thrust force are generated in the range of $\pm25°$ and 8-66 N, corresponding to the range of motion of the actual UAV. The data processing delay time ranges from 50–300 ms, with a fixed value per episode.

The concept of the reward function is shown in Eq. 8.

$$r = -(|\Delta\theta| + c_1|\dot{\theta}| + c_2|a_F| + c_3|a_T| + c_4F), \tag{8}$$

where r is the reward. The aim of the reward function is to make the pitch follow the target angle by giving a penalty for the error $\Delta\theta$ between the pitch angle and target pitch angle. By penalizing the pitch angular velocity $\dot{\theta}$ and the control actions, which are the thrust rate a_F and vectoring rate a_T, the oscillatory motion of the actuator is suppressed to achieve smooth and lean motion. Furthermore, by penalizing the thrust F, the priority is designed so that the attitude is controlled by vectoring the operation rather than the change in the thrust. The weights were set by the coefficients c for the individual terms so that approximately 60% of the total penalty is contributed by $\Delta\theta$ and the rest was evenly shared by others.

Proximal policy optimization (PPO) is applied as the learning algorithm [3, 14, 23], where the clip threshold is 0.2 and the learning rates of the policy and value functions are 2e-4 and 5e-4, respectively. The dynamics

Table 1 Range of dynamics variables in domain randomization

Variables	Range
m	$\pm 10\% \times$ default
x_G	$\pm 20mm+$ default (49 mm)
z_G	$\pm 20mm+$ default (28 mm)
x_T	$\pm 10mm+$ default (41 mm)
z_T	$\pm 10mm+$ default (-29 mm)
I_{yy}	$\pm 20\% \times$ default
F_c	$\pm 20\% \times$ default
F_z	$\pm 20\% \times$ default
τ	$\pm 0.1s+$ default (0.2 s)
Thrust angle offset	$\pm 3.0^\circ +$ default (0.0 °)

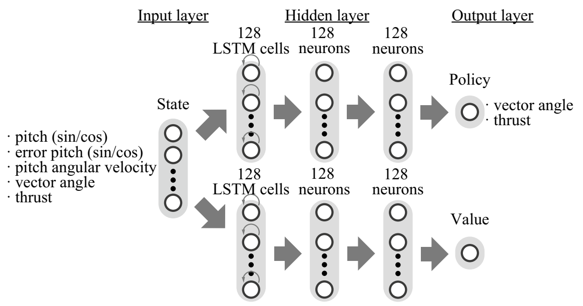

Fig. 7 Controller with neural network

variables varied within the range of Table 1 in each episode during learning, and the controller was optimized to adapt to those variables.

Experiments and evaluations

Experimental conditions

The pitch control experiments were conducted using the trained NN controller. The initial pitch angle was where the UAV was at the equilibrium by its own weight, and the target pitch angle was given at 0 °. As shown in Fig. 3, the upright state has a pitch angle of 0°, whereas the equilibrium state is tilted by approximately 110° under its own weight as shown in the photo labeled “0.1s” in Fig. 9. Four variations of the UAV model were prepared by different weights. The variations in the parameters are presented in Table 2. The same NN was used to control these models.

As shown in Fig. 3, Model 1 was the state with no weights added, namely, the state closest to the theoretical model. Weights were added to Model 1, and the weights were made heavier in the order of Model 2, 3, and 4. As a quantitative reference to the randomized range in training,, Model 2 was set at approximately 50% of the randomization range, Model 3 was set at approximately

Table 2 Model differences

Model	Weight, COG, Inertia		I_{yy}
	m	(x_G, z_G)	
Model 1	–	-	–
Model 2	+6.1 %	(+ 3.8, + 7.5) mm	+ 11%
Model 3	+10 %	(+ 6.4, + 13) mm	+ 20%
Model 4	+15 %	(+ 9.6, + 17) mm	+ 28%

100% of the range, and Model 4 exceeded the range of randomization.

Experimental results and discussion

Figure 8 shows the experimental results and Figs. 9, 10, 11, 12 show the behavior of the UAVs during the experiment for each model. In Fig. 8, from left to right, the pitch angle, vectoring angle, and total thrust of Models 1, 2, 3, and 4 are shown in time series, respectively. The time series of the experimental results for 6 s (8 s for Model 4 only) is shown, with experimental time $t = 0$ s at 0.1 s before the start of control.

Figures 8a and 9 showed that in Model 1, the pitch smoothly transitioned to the target pitch angle of 0 ° as soon as the control started, and that the pitch remained static after the target attitude was reached. The thrust increased to approximately 80 N at the initial start-up, but it remained at approximately 50 N in the static state. However, the vectoring angle was manipulated rapidly in the range from − 25 ° to 20 ° during the initial start-up, and it was adjusted agile during the subsequent static state. This behavior of maneuvering with vectoring more than thrust indicated the successful effect of the thrust penalty. Additionally, it was reasonable because the thrust operation included the uncertainty in the time constant, which should discourage the quick control of the thrust force.

In Models 2 and 3, the attitude control is as successful as in Model 1, despite the addition of more weights than in the Model 1 situation. The thrust and vectoring angle responses behaved similarly to Model 1; however, comparing the maximum thrust, the rise in thrust at the start of the experiment increased to 80, 100, and 130N as the weight increased for Models 1, 2, and 3. These experimental results suggest that the NN adapts to changes in the model and adjusts its output during control.

In Model 4, a case that exceeded the training range, the pitch angle did not reach the target angle, as shown in Fig. 12. Figure 8d showed a repetition of the action of trying to raise the pitch to the target attitude but failing to reach it and falling back. This behavior is an example of control failure when the physical model deviates from the domain randomization range. The thrust did

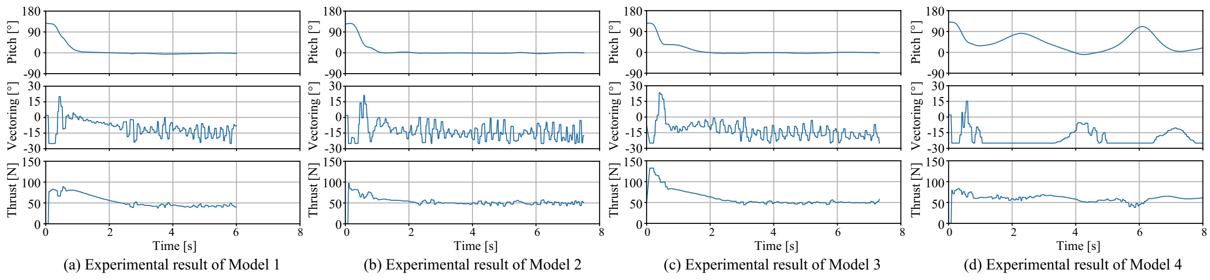

Fig. 8 Experimental result of Model 1-4

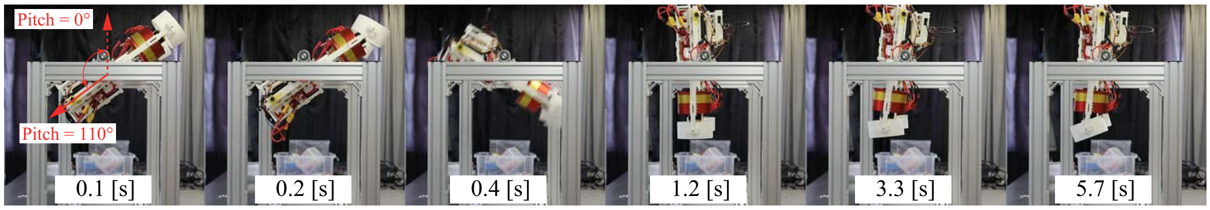

Fig. 9 Sequential photo of the experiment of Model 1

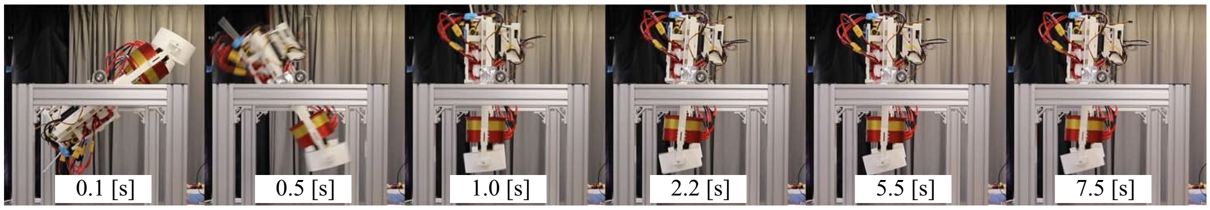

Fig. 10 Sequential photo of the experiment of Model 2

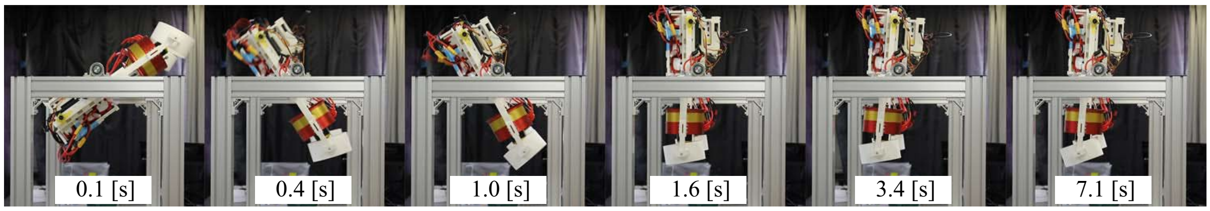

Fig. 11 Sequential photo of the experiment of Model 3

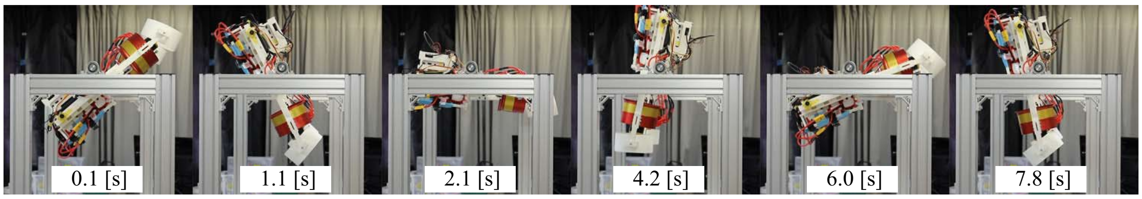

Fig. 12 Sequential photo of the experiment of Model 4

not reach the maximum nor even the value observed in Model 3. It was reckoned that the thrust penalty discouraged the controller to explore the control scenario using higher thrust in training. It was noted that even though the thrust was not sufficient, the vectoring was controlled to maximize the pitching moment, which was reasonable to reach the target pitch.

These experiments showed that by including the uncertainties of the model in training, it was possible to generate an NN controller that could work in the real environment. The successful control was observed for the model variations when the variations fell in the range assumed in the training. These results indicate the possibility of providing a robust control system for pilots in aircraft with a high ratio of fuel to airframe weight, such as small aircraft and flying cars, even if the aircraft's COG changes due to the amount of remaining fuel, resulting in changes in control performance. Notably, as a nature of deep reinforcement learning, the lessons learnt from the results tend to be case-specific, specifically for the failure behavior for Model 4. The results show the possibility of building a controller that can use the limits of the control devices (control surfaces and thrusts) to bring the current value close to the target value without breaking control, even for system that are difficult to control due to their physical characteristics.(In other words, like humans, they could have demonstrated the possibility of avoiding a crash by using available control devices in response to airframe damage or controller failure.) More detailed evaluations of the relationship between the randomization range, the model variations and the control performance are awaited to gain a generalizable understanding.

Conclusion

In this study, deep reinforcement learning utilizing domain randomization was applied to the pitch control of the UAV to demonstrate a control technique that adapts to changes in the physical model. First, the UAV equipped with EDFs and thrust vectorings was developed, and the theoretical model was constructed. Secondly, the NN with the LSTM layer was designed and optimized by deep reinforcement learning with domain randomization. Finally, the control tests were conducted using the same NN on multiple models with different COG positions, inertia and weights. These experimental results showed that control succeeded for model variations within the randomization range in training. The control failed for the model of which the variation was exceeded the randomization range. Through the results, the work range of the controller trained by domain randomization was successfully depicted.

In the future, we aim to expand the control target from one axis of pitch to six degrees-of-freedom. In such an extended control task, in addition to the variation of mass, other factors such as delay and thrusts uncertainties should possibly be investigated to verify the control performance in reality. Furthermore, the future research could look into a control technique for UAVs to perform complex tasks by controlling thrust vectoring and multiple control surfaces. The external disturbance such as wind could also be a factor to be considered.

Abbreviations

UAV(s)	Uncrewed aerial vehicle(s)
NN	Neural networks
EDF(s)	Electric ducted fan(s)
CoG	Center of gravity
PPO	Proximal policy optimization
LSTM	Long short term memory
ELU	Exponential linear unit

Acknowledgements
This work was supported by Grant-in-Aid for Japan society for the promotion of science fellows [grant number 20K14707].

Author contributions
AO, DW, and HS developed the system and carried out the experiments. AO and DW constructed the study concept and drafted the manuscript. All members verified the content of their contributions.

Funding
Not applicable.

Availability of data and materials
The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 20 November 2022 Accepted: 15 July 2023
Published online: 07 August 2023

References
1. Gu W, Valavanis KP, Rutherford MJ, Rizzo A (2019) A survey of artificial neural networks with model-based control techniques for flight control of unmanned aerial vehicles. In: 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 362–371. https://doi.org/10.1109/ICUAS.2019.8797853
2. Koch W, Mancuso R, West R, Bestavros A (2019) Reinforcement learning for UAV attitude control. ACM Trans Cyber-Phys Syst 3(22):1–21. https://doi.org/10.1145/3301273
3. Hodge VJ, Hawkins R, Alexander R (2021) Deep reinforcement learning for drone navigation using sensor data. Neural Comput Appl 33:2015–2033. https://doi.org/10.1007/s00521-020-05097-x
4. Xi C, Liu X (2020) Unmanned aerial vehicle trajectory planning via staged reinforcement learning. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS). Athens, Greece, 1-4 September; pp 246–255. https://doi.org/10.1109/ICUAS48674.2020.9213983
5. Tang C, Lai Y-C (2020) Deep reinforcement learning automatic landing control of fixed-wing aircraft using deep deterministic policy gradient. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1-4 September; pp 1–9. https://doi.org/10.1109/ICUAS48674.2020.9213987

6. Yan C, Xiang X, Wang C (2020) Fixed-wing UAVs flocking in continuous spaces: a deep reinforcement learning approach. Rob Auton Syst 131:103594
7. Clarke SG, Hwang I (2020) Deep reinforcement learning control for aerobatic maneuvering of agile fixed-wing aircraft. In: AIAA SciTech Forum. Florida, US, 6-10 January. https://doi.org/10.2514/6.2020-0136
8. Bøhn E, Coates EM, Moe S, Johansen TA (2019) Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. In: Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS). Atlanta, US, 11-14 June, pp 523–533
9. Muratore F, Gienger M, Peters J (2021) Assessing transferability from simulation to reality for rein-forcement learning. IEEE Trans Pattern Anal Mach Intell 43, pp 1172–1183
10. James S, Wohlhart P, Kalakrishnan M, Kalashnikov D, Irpan A, Ibarz J, Levine S, Hadsell R, Bousmalis K (2019) Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). California, US, 16-20 June, pp 12627–12637. https://doi.org/10.1109/CVPR.2019.01291
11. Sadeghi F, Levine S (2017) Cad2rl: Real single-image flight without a single real image. In: Robotics: Science and Systems. Massachusetts, US, 12-16 July. https://doi.org/10.15607/RSS.2017.XIII.034.
12. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vancouver, BC, Canada, 24-28 September, pp 23–30. https://doi.org/10.1109/IROS.2017.8202133
13. Peng XB, Andrychowicz M, Zaremba W, Abbeel P (2018) Sim-to-real transfer of robotic control with dynamics randomization. In: 2018 IEEE International Conference on Robotics and Automation. Brisbane, Australia, 21-26 May, pp 3803–3810
14. OpenAI Andrychowicz M, Baker B, Chociej M, Jozefowicz R, McGrew B, Pachocki J, Petron A, Plappert M, Powell G, Ray A, Schneider J, Sidor S, Tobin J, Welinder P, Weng L, Zaremba W (2020) Learning dexterous in-hand manipulation. Int J Robot Res 39:3–20. https://doi.org/10.1177/0278364919887447
15. Xiao C, Lu P, He Q (2021) Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and sim2real. IEEE Transactions on Neural Networks and Learning Systems 34(5), pp 2701–2708. https://doi.org/10.1109/TNNLS.2021.3107742
16. Oosedo A, Wada D, Hisada S (2022) Pitch control with thrust vectoring by using deep reinforcement learning for unmanned aerial vehicle. In: Robotics and Mechatronics Conference 2022 in Sapporo. Sapporo, Japan, 1-4 Jun 2022. (in Japanese)
17. Wada D, Araujo-Estrada S, Windsor S (2022) Sim-to-real transfer for fixed-wing uncrewed aerial vehicle: Pitch control by high-fidelity modelling and domain randomization. IEEE Robot Autom Lett 7(4):11735–11742. https://doi.org/10.1109/LRA.2022.3205442
18. Xu J, Du T, Foshey M, Li B, Zhu B, Schulz A, Matusik W (2019) Learning to fly: computational controller design for hybrid UAVs with reinforcement learning. ACM Trans Graph 3838:1–12. https://doi.org/10.1145/3306346.3322940
19. Chen X, Hu J, Jin C, Li L, Wang L (2022) Understanding domain randomization for sim-to-real transfer. In: Tenth International Conference on Learning Representations. Virtual. https://doi.org/10.48550/arXiv.2110.03239
20. Dey R, Salem FM (2017) Gate-variants of gated recurrent unit (gru) neural networks. In: IEEE 60th International Midwest Symposium on Circuits and Systems, pp 1597–1600. https://doi.org/10.1109/MWSCAS.2017.8053243
21. Yu W, Kumar VCV, Turk G, Liu CK (2019) Sim-to-real transfer for biped locomotion. In: 2019 IEEE/RSJInternational Conference on Intelligent Robots and Systems (IROS), pp. 3503–3510. https://doi.org/10.1109/IROS40897.2019.8968053
22. Peng XB, Coumans E, Zhang T, Lee T-WE, Tan J, Levine S (2020) Learning agile robotic locomotion skills by imitating animals. arXiv preprint arXiv:2004.00784
23. Kaufmann E, Bauersfeld L, Scaramuzza D (2022) A benchmark comparison of learned control policies for agile quadrotor flight. In: Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), pp 10504–10510. https://doi.org/10.1109/ICRA46639.2022.9811