# Using Reinforcement Learning to fly stable for Aircraft

*Note: Sub-titles are not captured in Xplore and should not be used


1st Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

2nd Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

3rd Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

4th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

5th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID

6th Given Name Surname
*dept. name of organization (of Aff.)*
*name of organization (of Aff.)*
City, Country
email address or ORCID



***Abstract*—something anasdd**



***Index Terms*—Reinforcement Learning, TD3, aircraft, Xplane, PID, upset**


## I. Introduction

Since aircraft upset incidents have remained the "highest risk to civil aviation" in the past several decades [1], it is particularly important to rescue the aircraft from an upset state. Many researchers have proposed various approaches to solve aircraft upset. Paper [2] describes a novel finite-state conditional switching structure that enables autonomous recovery for a large envelope of loss-of-control conditions. Paper [3] proposed a loss of altitude minimizing economic model predictive control strategy for deep-stall recovery. In [4], the problem of aircraft spin recovery is solved as a trajectory optimization problem using direct multiple shooting method with time and altitude-loss as cost functions to be minimized.

However, these methods are dependent on precise knowledge of the aircraft model that is difficult and resource-consuming to obtain through traditional methods. Using reinforcement learning (RL) is a good approach to overcome this issue because it allows an aircraft to learn and navigate through the changing environment without an explicit model of the environment [5].

RL has been successfully applied to various complex problems [6]–[8]. It mainly includes four elements: agent, environment, reward, and action [**?**]. RL is learning what to do and how to map situations to actions, so as to maximize some numerical reward. The learning agent is not told the correct actions; instead it explores the possible actions and remembers the reward it receives. The RL agent aims to maximise its cumulative reward over each episode. Considering parametric uncertainties and nonlinear morphing dynamics, Reinforcement learning is a promising upset recovery scheme in the sense of nonlinearity handling.


Identify applicable funding agency here. If none, delete this.


In this paper, we proposed a method based on RL to recover the aircraft back to steady level flight swiftly. When the aircraft is in an upset state, control the aircraft through RL and adjust the aircraft attitude so that the aircraft can resume smooth flight within a limited time.

The major contributions of this paper are as follows:

- A pretrained-trained RL method is proposed for random upset state recovery. Using this method, the agent can be trained successfully and quickly.
- A novel reward function is proposed to improve the performance of the training efficiency.
- We provide a comparison between PID and our proposed method for LOA(loss of altitude)-minimal recovery, and proved our method is quite more excellent than PID.

The rest of this paper is organized as follows. Section II Introduces the preliminaries of this paper, including dynamics of aircraft, classical upset states of aircraft and TD3 algorithm. In Section III, a TD3-based upset recovery method is proposed. Section IV presents the experiments. Finally, Section V summarizes this paper.

## II. Preliminaries

### A. Classical upset states of Aircraft

Upset states of the aircraft includes stall and spin [9], and spin is a nonlinear post-stall phenomena in which an aircraft develops a high rotational-rate and descends almost vertically in a helical trajectory.

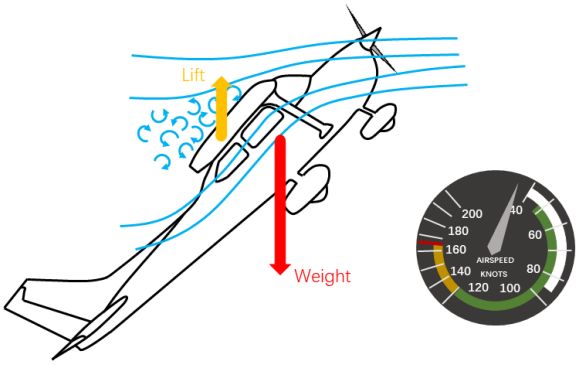


Fig. 1. The stall phenomenon of the aircraft

*a) Stall:* A stall is a condition in aerodynamics and aviation such that if the angle of attack increases beyond a certain point, then lift begins to decrease. Stalls in fixed-wing flight are often experienced as a sudden reduction in lift as the pilot increases the wing's angle of attack and exceeds its critical angle of attack. A stall does not mean that the engine have stopped working, or that the aircraft has stopped moving. The stall phenomenon is shown in Fig. 1.

*b) Spin:* Spin (as shown in Fig. 2) is one of the most complex aircraft maneuver and has been a subject of numerous research projects, tests and investigations since the early years of aviation. A spin is defined as an aggravated stall, which results in autorotation of an aircraft while descending in a helical pattern about the vertical spin axis. In an aggravated stall, one wing is stalled more than the other. The more stalled wing experiences less lift and more drag as compared to other and this imbalance of forces initiates autorotation and subsequent rapid decent of the aircraft [**?**]

### *B. Twin Delayed Deep Deterministic Policy Gradient Algorithm*

Reinforcement learning considers the paradigm of an agent interacting with its environment with the aim of learning reward-maximizing behavior. At each time step $t$, with a given state $s \in \mathcal{S}$, the agent selects actions $a \in \mathcal{A}$ with respect to its policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, receiving a reward $r$ and the new state of the environment $s'$. The return is defined as the discounted sum of rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, where $\gamma$ is a discount factor determining the priority of short-term rewards.

Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [10] is a RL algorithm proposed for agents with continuous states and actions. It is based on actor-critic architecture which considers the interplay between function approximation error in both policy and value updates. TD3 trains a deterministic policy in an off-policy way. In TD3, it mainly used three critical tricks to address over estimating the Q values of the critic network:

- Using a pair of critic networks

spin.png

Fig. 2. The spin phenomenon of the aircraft

TD3 learns two Q-functions instead of one, $Q_{\phi_1}$, $Q_{\phi_2}$, and uses the smaller of the two Q-values to form the targets in the Bellman error loss functions.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,targ}}(s', a') \qquad (1)$$

Where $d = 0$ *or* $1$. Then the parameters of both Q-value function $\phi_1$ and $\phi_1$ are updated by one step of gradient descent using:

$$\nabla_{\phi_i} \frac{1}{\mathcal{B}} \sum_{(s,a,s',r,d) \in \mathcal{B}} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \qquad (2)$$

where $i = 1, 2$ and $\mathcal{B}$ is a mini-batch sampled from the replay buffer $D$. Using the smaller Q-value for the target, and regressing towards that, helps decrease overestimation in the Q-function.

- Delayed policy updates
  TD3 updates the policy (and target networks) less frequently than the Q-function, and we define the delayed frequency as $p_d$. The parameter of the policy network $\pi_\theta$ is updated by one step of gradient ascent to maximize the Q-value using:

$$\nabla_\theta \frac{1}{\mathcal{B}} \sum_{s \in \mathcal{B}} Q_{\phi_1}(s, \pi_\theta(s)) \qquad (3)$$

- Target policy smoothing
  In order to reduce the variance caused by over-fitting, TD3 uses a regularisation technique known as target policy smoothing. Ideally there would be no variance between target values, with similar actions receiving similar values. TD3 reduces this variance by adding a

small amount of random noise to the target and averaging over mini batches. The range of noise is clipped in order to keep the target value close to the original action. The target actions are thus:

$$a'(s') = clip(\pi_{\theta_{targ}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High}) \tag{4}$$

where $\pi_{\theta_{targ}}$ is the target policy, action $a$ satisfy $a_{Low} \leq a \leq a_{High}$, $\epsilon \in \mathcal{N}(0, \sigma)$.

## III. TD3-Based Upset Recovery Method

Under a upset initial condition, our goal is to train the agent to recover the aircraft back to steady level flight swiftly. TD3 algorithm is used to design the controller.

### A. Aircraft Control

As mentioned in Section I, if the classical methods usually provide simple and robust ways to control a system, most of the time they require a good knowledge of its dynamics and perform poorly when the level of uncertainty raises. Since the dynamics of the plane can get quite complex and difficult to model, a controller based on RL will be designed for its capacity to capture the complex behavior of physical system without the need to specify the laws of motion explicitly.

In order to control the aircraft using RL, we need to define the state space and action space of the agent.

The flight status can be obtained from sensors in the aircraft. Although we cannot directly obtain the flight status from sensors, we can obtain some related data from the master pilot's perspective. We noticed that our goal had no dependence on its location, so latitude and longitude can be ignored. Under comprehensive consideration, we define our state space $S$ as a set of states $S_1, S_2, ..., S_n \in S$ as a collection of values corresponding to the following:$\{\omega, \kappa, \xi, p, q, r, h, v\}$:

$$S = \{s | s = [\omega, \kappa, \xi, p, q, r, h, v]\} \tag{5}$$

All of them are described in Table I:

Table I. The state data fed into the reinforcement learning model

| Field | Description | Range |
|---|---|---|
| $\omega$ | pitch | $[-180°, 180°]$ |
| $\kappa$ | roll | $[-180°, 180°]$ |
| $\xi$ | heading/yaw | $[0°, 360°]$ |
| p | rotational velocity of pitch | $[-60°/s, 60°/s]$ |
| q | rotational velocity of roll | $[-60°/s, 60°/s]$ |
| r | rotational velocity of heading | $[-60°/s, 60°/s]$ |
| h | altitude | $[0\ m, 5000\ m]$ |
| v | indicated airspeed of the aircraft | $[0\ m/s, 150\ m/s]$ |

As for action space, we define it as a set of actions in 4 dimensions based on the high level controls available to a pilot: elevator, ailerons, rudder and throttle. Among them, elevator, ailerons, rudder are in the continuous range $[-1, 1]$, and the engine throttle is in $[0, 1]$. Thus, the action space can be defined as:

$$A = \{a | a = \delta_e, \delta_a, \delta_r, \delta_t\} \in [-1, 1]^3 \times [0, 1] \tag{6}$$

where $\delta_e, \delta_a, \delta_r, \delta_t$ denote elevator, ailerons, rudder, throttle, respectively. Fig. 3 shows the main control surfaces.

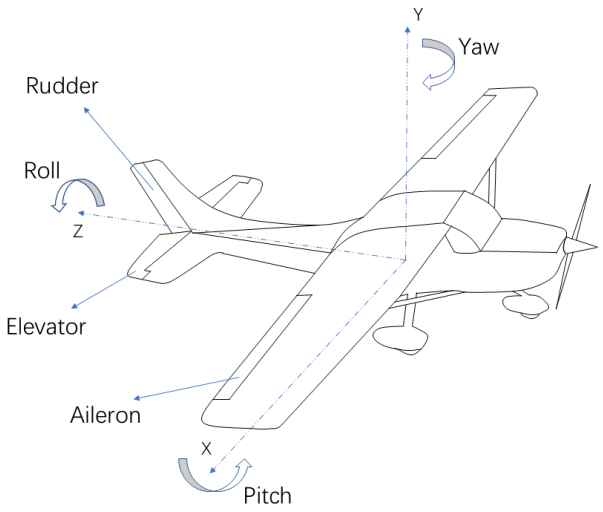


Fig. 3. Aircraft control surfaces

### B. Reward Function

In RL, reward is an environmental feedback that is given to the agent for every action it takes. The design of reward function is crucial as it governs not only the convergence time but also the quality of the convergent point of the learning model. In this work, the reward function is designed such that the agent earns highest reward for suggesting a maneuverable resolution that successfully recover the aircraft back to steady level flight under a upset initial condition.

Since our end goal is to recover the aircraft back to steady level flight which is a sparse reward problem, leading to the slow convergence rate of training. We designed the reward function with the idea of rewarding shaping [11].

The reward function is defined as follows:

$$R(s_t, a_t, s_{t+1}) = T(s_{t+1}) + F(s_t, a_t, s_{t+1}) \tag{7}$$

where $T(s_t)$ is the termination reward and $F(s_t, a_t, s_{t+1})$ is a bounded real-value function.

The termination reward is the reward obtained when the next state is extreme or is successful to recover. It is defined as:

$$T(s_{t+1}) = \begin{cases} -1000, & \text{if the aircraft is crushed} \\ +2000, & \text{if the aircraft is successful to recover} \\ 0, & else \end{cases} \tag{8}$$

$F(s_t, a_t, s_{t+1})$ has the form:

$$F(s_t, a_t, s_{t+1}) = \Gamma\Phi(s_{t+1}) - \Phi(s_t) \tag{9}$$

where $\Phi(s_t)$ is a real-valued function over states, $\Gamma$ is a constant. $\Phi(s_t)$ is defined as:

$$\Phi(s_t) = C(s_t) + P(s_t) \tag{10}$$

where $C(s_t)$ is the continuous action reward, it is related to the elevator and ailerons; and $P(s_t)$ is the aircraft attitude reward, it is given by (11):

$$P(s_t) = \sum_{i=1}^{8} p_i |s_{norm}[i] - s_{targ,norm}[i]| \tag{11}$$

where $p_i (i = 1, ..., 8)$ are negative weight coefficients, $s_{norm}$ is the normalized data of state $s$, $s_{targ,norm}$ is the normalized data of target state $s_{targ}$.

The training speed can be improved using the reward function proposed above. For different tasks in realistic situations, training efficiency can be improved by adjusting one or more reward functions or coefficients.

### C. Pre-trained TD3-based training framework

For complex scenarios, successfully training an agent is very challenging. What's more, even if the training is successful, the convergence speed may be very slow. Therefore, it is necessary to propose an improved algorithm. Considering the state and episode reward, a pre-trained TD3-based algorithm is proposed. Firstly, since the elements in the state are quite different, we need to normalize the state. The whole process has two phases: Model pre-training and training. We apply the original TD3 algorithm to pre-train a guidance agent and then obtain a guidance policy. Both critic network and policy network are initialized by the pre-training phase. Based on this, it can greatly speed up the training process and save a lot of time. After the pre-training process, a modified TD3 algorithm is used to train the agent. This algorithm is more efficient than purely using TD3 in our scenarios. The overall framework of training algorithm is shown in Fig. 4.

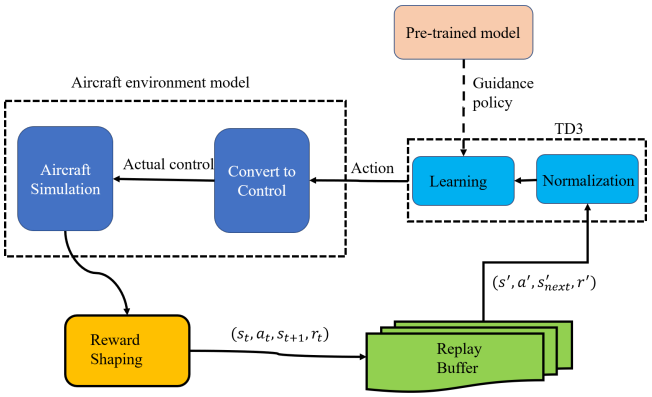


Fig. 4. The overall framework of training process.

The pseudocode is shown in Algorithm 1.

**Algorithm 1:** Pre-trained TD3-based upset recovery algorithm

**Input:** initial policy parameters $\theta$, Q-function parameters $\phi_1$, $\phi_2$, empty replay buffer $\mathcal{D}$

Set target parameters equal to main parameters $\theta_{targ} \leftarrow \theta$, $\phi_{targ,1} \leftarrow \phi_1$, $\phi_{targ,2} \leftarrow \phi_2$;
Set Pre-trained aircraft model parameters and environment parameters;
Pre-train a guidance agent using original TD3;
Set the guidance agent as the baseline agent;
Reset aircraft model parameters;
**for** *episode=1 to M* **do**
  Receive initial observation state $s_1$ and normalize it;
  **for** *t=1 to T* **do**
    Select action $a_t$ according to the guidance policy;
    Convert the action into the aircraft control;
    Execute action $a_t$ in aircraft environment, and obtain reward $r_t$ and obtain new state $s_{t+1}$;
    Normalize the state $s_{t+1}$;
    Get $r_t^{rs}$ using $r_t^{rs} = T(s_{t+1}) + F(s_t, a_t, s_{t+1})$;
    Store transition $(s_t, a_t, r_t^{rs}, s_{t+1})$ in $\mathcal{D}$;
    Sample a mini-batch of transitions, $\mathcal{B} = \{(s, a, r^{rs}, s', d)\}$ from $\mathcal{D}$;
    Compute target actions $a'(s') = clip(\pi_{\theta_{targ}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High})$;
    Compute targets $y(r^{rs}, s', d) = r^{rs} + \gamma(1-d)\min_{i=1,2} Q_{\phi_{i,targ}}(s', a')$;
    Update Q-functions using $\nabla_{\phi_i} \frac{1}{\mathcal{B}} \sum_{(s,a,s',r^{rs},d)\in\mathcal{B}} (Q_{\phi_i}(s,a) - y(r^{rs}, s', d))^2$;
    **if** *t mod $p_d$=0* **then**
      Update policy using $\nabla_\theta \frac{1}{\mathcal{B}} \sum_{s\in\mathcal{B}} Q_{\phi_1}(s, \pi_\theta(s))$;
      Update the target network: $\phi_{targ,i} \leftarrow \tau\phi_{targ,i} + (1-\tau)\phi_i$, $\theta_{targ} \leftarrow \tau\theta_{targ} + (1-\tau)\theta$

## IV. Experiments

### A. Experiment Set Up

We tested our reinforcement learning agent in X-Plane which is a flight simulation software. As a professional flight simulation software, X-Plane includes a lot of light aircraft, commercial aircraft, and military aircraft, as well as almost global landscapes. Moreover, the software is very extensible, allowing developers or players to expand functions arbitrarily, such as adding their own designed airplanes or landscapes. As X-Plane can provide sophisticated dynamic models of various types of aircraft with a blade-element approach along with a realistic flying environment, it can emulate the flight conditions relatively reliably and is thus chosen as the flight simulator for the algorithm test in this study.

The integrated simulation system, utilizing X-Plane 11.0 and Tensorflow-gpu 2.4, runs under 64-bit Windows 10 on a PC with 32G RAM, 3.4GHz Frequency, and a RTX 3070 GPU for training of neural networks.

The test aircraft is N172SP, as shown in Fig. 5. Since our goal is to test recovery performance of our RL method, N1172SP needs to be deliberately upset. We mainly test two classical upset scenarios as described in Section II: stall and inverted flight.


Fig. 5. The test aircraft: N172SP

*a) Stall:* The stall state is set up by initializing the aircraft state as: $s = [\omega, \kappa, \xi, p, q, r, h, v] = [90, 90, 90, 60, 60, 60, 2500, 0]$.

*b) Spin:* We initialize the aircraft state as: $s = [\omega, \kappa, \xi, p, q, r, h, v] = [90, 90, 90, 60, 60, 60, 2500, 0]$, followed by a continuous excitation $a = [\delta_e, \delta_a, \delta_r, \delta_t] = [1, 1, 1, 0]$.

To study the effects of wind on upset recovery, simulations are also carried out with wind conditions. In X-Plane, we can easily set the wind condition via the settings panel.

For real simulation, all data of states are not precise data, they are obtained from panel data that the pilot can see. This means that the data obtained is inaccurate, but this is in line with reality.

As for the algorithm set up, the learning rate is $1.0 \times 10^{-4}$. The actor has three hidden layers with 64, 64, and 32 units, respectively. The critic has two hidden layers with 64, 64units, respectively. The time step $T$ is 500 and the minibatch size $\mathcal{B}$ is 64. The number of episodes $M$ is 400 if no winds and it is 800 if winds.

### *B. Results*

We tested a total of windy and no wind conditions using the pid policy, and windy and non-wind conditions using our RL policy.

Under no wind conditions, we trained our agent, the average reward during training process is shown in Fig. 6. We can see the algorithm converges after about 200 episodes. To test the upset recovery effect of our improved algorithm, we use the upset recovery time and height loss as evaluation indicators to compare with PID. The corresponding attitude changes are shown in Fig. 7, and the altitude loss is shown in Fig. 8. Fig. 9 shows the angular rate.

In Windy effects,

## V. Conclusion

## References

[1] Christine M Belcastro, John V Foster, Gautam H Shah, Irene M Gregory, David E Cox, Dennis A Crider, Loren Groff, Richard L Newman,

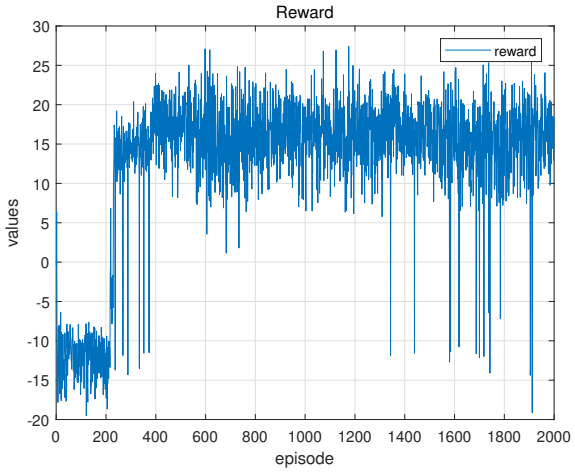


Fig. 6. The average reward during training process

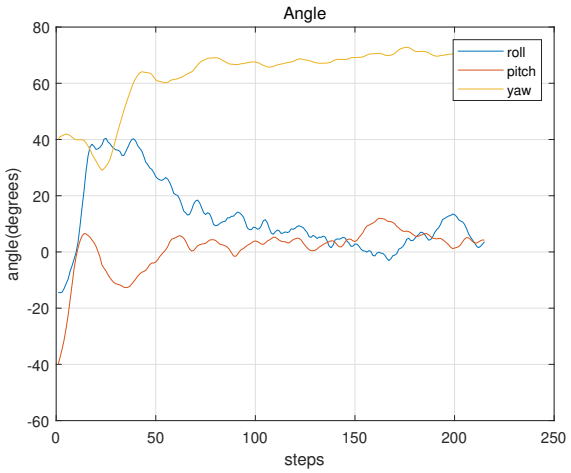


Fig. 7. The attitude angles changes

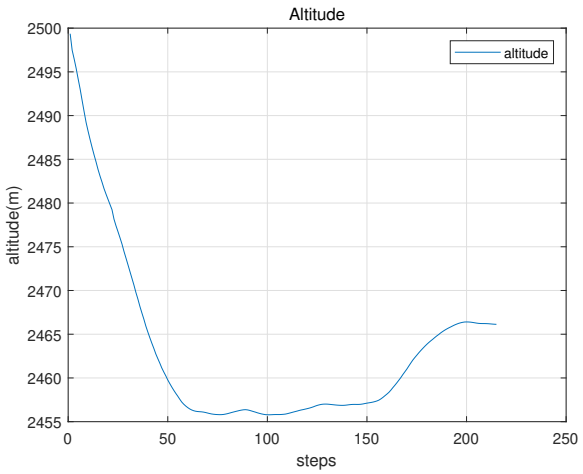


Fig. 8. The attitude angles changes

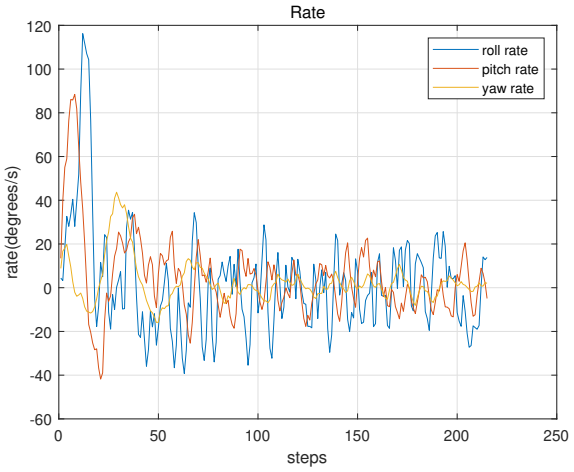


Fig. 9. The angular rate changes

and David H Klyde. Aircraft Loss of Control Problem Analysis and Research Toward a Holistic Solution. *Journal of Guidance, Control, and Dynamics*, 40(4):733–775, 4 2017.

[2] Anil Yildiz, M Ugur Akcal, Batuhan Hostas, and N Kemal Ure. Switching Control Architecture with Parametric Optimization for Aircraft Upset Recovery. *Journal of Guidance, Control, and Dynamics*, 42(9):2055–2068, 4 2019.

[3] T Cunis, D Liao-McPherson, J Condomines, L Burlion, and I Kolmanovsky. Economic Model-Predictive Control Strategies for Aircraft Deep-stall Recovery with Stability Guarantees. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 157–162, 2019.

[4] D.M.K.K. Venkateswara Rao and Tiauw H Go. Optimization of aircraft spin recovery maneuvers. *Aerospace Science and Technology*, 90:222–232, 2019.

[5] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in Deep Reinforcement Learning: A survey. *arXiv*, pages 1–22, 2020.

[6] X Lin, Y Yu, and C Sun. Supplementary Reinforcement Learning Controller Designed for Quadrotor UAVs. *IEEE Access*, 7:26422–26431, 2019.

[7] C Tang and Y C. Lai. Deep Reinforcement Learning Automatic Landing Control of Fixed-Wing Aircraft Using Deep Deterministic Policy Gradient. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1–9, 2020.

[8] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[9] Bilal Malik, Jehanzeb Masud, and Suhail Akhtar. A review and historical development of analytical techniques to predict aircraft spin and recovery characteristics. *Aircraft Engineering and Aerospace Technology*, 92(8):1195–1206, 1 2020.

[10] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2018.

[11] Andrew Y Ng, Daishi Harada, and Stuart J Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.