33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

META-REINFORCEMENT LEARNING-BASED FAULT TOLERANT CONTROL FOR A QUADROTOR WITH A SEVERE LOSS OF ROTOR

Seong-hun Kim[1], Yeontaek Jung[1] & Youdan Kim[1]

[1]Seoul National University

Abstract

The quadrotor has under-actuated dynamics, but its unique symmetrical rotor arrangement allows rapid attitude adjustment, making it easy to design a robust controller to external disturbances such as wind. However, if severe faults occur on one or more rotors, even hovering control is challenging to accomplish due to a reduced set of attainable force and moment. This study proposes a design scheme to obtain an optimal fault-tolerant controller to deal with a severe actuator fault by leveraging a meta-reinforcement learning (meta-RL) technique. The meta-RL trains an outer-loop network to infer the faulty situation and help an inner-loop RL process to optimize the controller quickly. A numerical simulation is provided to demonstrate the performance of the proposed design scheme and the learnt fault-tolerant controller.

Keywords: Fault tolerant control, UAV, reinforcement learning

1. Introduction

As small unmanned aerial vehicles (UAVs) begin to be actively used in various industrial fields, fault-tolerance control (FTC) techniques that allow them to be operated safely are receiving continuous attention [1]. Among various faults that threaten the safe operation of a quadrotor, which is one of the widely used types of UAVs, actuator faults occurring in the rotor, assessed by a loss of effectiveness (LoE) is one of the most common and frequent situations.

There has been a lot of studies on FTC for quadrotors to deal with LoE faults, including backstepping [2, 3], sliding mode controls [4–8], and linear matrix inequalities synthesis [9]. However, most of these studies have been shown their performance under LoE faults less than about 50%. Fault cases with severe or complete loss of rotors usually have been treated by a complete redesign of the controller [10–16]. While these methods have shown surprising performances even with the complete loss of more than one rotors, most of them heavily rely on the help of information on which rotors failed.

Recently, the field of applying reinforcement learning (RL) for quadrotor FTC is emerging [17–19]. In [17], a set of controllers were trained using integral RL for various fault cases, and a switching rule was designed based on a fault detection module. The integral RL has been also implemented for FTC of a team of quadrotors in [19]. Although, the integral RL can theoretically guarantee the stability of the learnt controller, the main drawback is the requirements of an initial stabilizing policy and a proper set of basis functions. A robust RL has been implemented to train a supervisory level controller that determines the final output of the low-level, inner PID controllers to attain the stability properties of the model-based PID controllers [18]. This concept is similar to meta-reinforcement learning (meta-RL) approaches [20–22], but has the disadvantage of requiring the model-based internal controllers.

In this study, a meta RL-based fault tolerant control method is proposed, which can be used in the severe fault situations including the complete loss. The networks in the meta-RL architecture are trained using a bunch set of flight data that can be gathered both from simulations and real flight tests. After training with a large flight dataset, the embedding network encodes a small amount of flight data into a human readable fault information and a latent vector for synthesizing initial suboptimal control

policy that can be fine-tuned in real-time. For a fast fine-tuning, a linear policy and a linear gradient Q-function network is required for an RL agent, and therefore, an equilibrium point, or a trim point, of the linear control policy is set as the human readable information part of the embedding network. A linear Q-learning method is proposed to train these networks of the RL agent and the embedding network simultaneously, which utilizes the modified Hamilton-Jacobi-Bellman (HJB) equation. A numerical simulation for a quadrotor under a severe fault situation demonstrates the validity of the proposed meta-RL scheme.

2. Problem Formulation

2.1 Quadrotor Model

The quadrotor full dynamic model is given by

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{1}$$
$$\dot{\mathbf{v}} = mg\mathbf{e}_3 - f_T R\mathbf{e}_3 + \mathbf{d}_\mathbf{v} \tag{2}$$
$$\dot{R} = R\hat{\omega}, \tag{3}$$
$$\dot{\omega} = J^{-1}(M - \omega \times J\omega + \mathbf{d}_\omega), \tag{4}$$

where $\mathbf{p} = [x,y,z]^T \in \mathbb{R}^3$ and $\mathbf{v} = [v_z, v_y, v_z]^T \in \mathbb{R}^3$ are the position and velocity vectors in the inertial coordinate system, respectively, $R \in \mathbb{R}^{3\times3}$ is the rotation matrix, $\omega \in \mathbb{R}^3$ denotes the angular velocity in the body-fixed coordinate system, and $\hat{}$ denotes the hat operator defined by $\hat{x}y = x \times y$ for all vectors $x, y \in \mathbb{R}^3$. The coordinate systems and the rotor configuration of the quadrotor model used in this study is represented in Fig. 1. The mass and the moment of inertia of the quadrotor are denoted by m and J, respectively, the scalar g is the gravitational constant, and $\mathbf{e}_3 = [0,0,1]^T$. The total force and moment, denoted by $f_T \in \mathbb{R}$ and $M = [M_1, M_2, M_3]^T \in \mathbb{R}^3$ respectively, are given by

$$\begin{bmatrix} f_T \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -r & 0 & r \\ r & 0 & -r & 0 \\ -c & c & -c & c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}, \tag{5}$$

where $l > 0$ is the length of moment arm for each rotor thrust with respect to the center of mass, $c > 0$ is the anti-torque coefficient generated by each rotor, and the j-th rotor generates the thrust amount of f_j in $-\mathbf{e}_3$ direction in the body-fixed coordinate system. The physical specifications of the quadrotor model are the same as [7], while the maximum thrust of each rotor is increased from about $15\,\mathrm{N}$ to $20\,\mathrm{N}$ to cope with severe faults occurred on rotors.

The lumped external disturbances and modelling uncertainties applied to the translational and rotational dynamics, denoted by $\mathbf{d}_\mathbf{v}$ and $\mathbf{d}_\omega$, respectively, are given as

$$\mathbf{d}_\mathbf{v} = -K_\mathbf{v}\mathbf{v}, \tag{6}$$
$$\mathbf{d}_\omega = -\|\omega\|K_\omega\omega, \tag{7}$$

where $K_\mathbf{v}$ and K_ω denote the drag coefficient of which the values are the same as in [7]. Due to the hight yaw rate in near-hovering under the severe rotor fault, the rotational drag model in (7) has a quadratic form in the magnitude of the angular velocity [12].

2.2 Fault Definition

In this study, we consider the loss of effectiveness (LoE) faults on rotors, which is one of the most common faults for quadrotors [7]. The loss of effectiveness factor $\lambda_j \in [0,1]$ for j-th rotor can represent the LoE fault in terms of rotor thrust as follows [1].

$$f_j = (1-\lambda_j)f_j^c, \quad \forall f_j^c \in [0, f_{j,\max}], \tag{8}$$

for all $j = 1,2,3,4$, where f_j^c denotes the commanded rotor thrust, and $f_{j,max}$ is the maximum thrust of each rotor. The healthy rotor has $\lambda_j = 0$, or 0% LoE, and the completely failed rotor has $\lambda_j = 1$, or

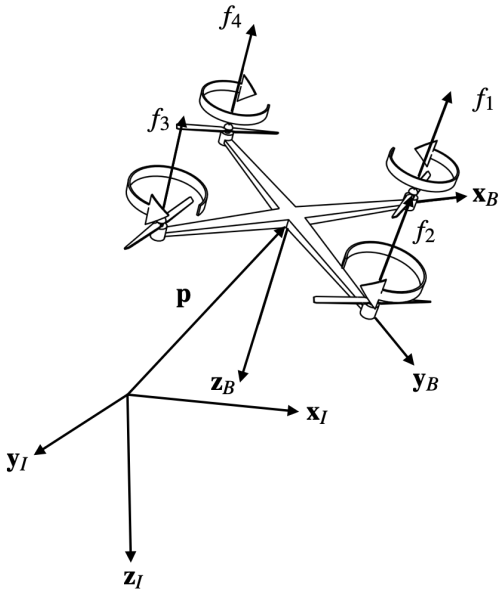

Figure 1 – Schematic of a quadrotor with the body-fixed coordinate system $(\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B)$ and the inertial coordinate system $(\mathbf{x}_I, \mathbf{y}_I, \mathbf{z}_I)$.

100% LoE. By denoting

$$\mathbf{f} := \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}, \quad \mathbf{f}_c := \begin{bmatrix} f_1^c \\ f_2^c \\ f_3^c \\ f_4^c \end{bmatrix}, \quad \Lambda := \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \lambda_3 & \\ & & & \lambda_4 \end{bmatrix}, \tag{9}$$

the equation (8) can be written as

$$\mathbf{f} = (I - \Lambda)\mathbf{f}_c, \tag{10}$$

where $I \in \mathbb{R}^{4\times 4}$ denotes the identity matrix.
Although, the fault tolerant control for quadrotor LoE faults such as (8) has been widely studied in the past decade including linear, nonlinear, and learning-based control scheme, most of them have focused on LoE less than 50% [7, 8, 23–25]. In this study, LoE greater than 90%, or equivalently $\lambda_j \geq 0.9$, as well as the complete loss is considered and defined as the severe LoE. Since the severe LoE usually involves the structural damage, the change in the mass and the moment of inertia up to 10% is also considered [25, 26]. It is defined as a fault situation when LoE, mass, or moment change occurs, and it is assumed that accurate fault information is not available during actual flight.

3. Double-Loop Control Architecture

This section propose a novel structure for the reinforcement learning-based position controller. Because the time scales of the rotational dynamics of the transform and quadrotors are well separated, most position controllers adopt a double-loop control structure, where the outer loop controls the position using attitude angles, and the inner-loop controller calculates the rotor thrust command to track the attitude command using the rotational dynamics [5, 7, 11]. However, it is difficult to keep the yaw rate at zero in severe fault situations, and therefore, a double-loop control scheme based on relaxed hover solutions has been proposed [12, 27]. In this scheme, the outer loop controls the position using the translational acceleration, and the inner-loop controller aligns the average angular velocity with the acceleration using the rotor thrusts. The authors also have shown in both of simulations and experiments that a simple linear controller for the inner-loop in this scheme is robust enough to track the position in the neighbor of the relaxed hover solution. Because a linear controller is required for fast training of the reinforcement learning in urgent severe fault situations, we adopt this relaxed hover solution-based double-loop control structure with suitable modifications for meta-RL. The overall double-loop control architecture is presented in Fig. 2.

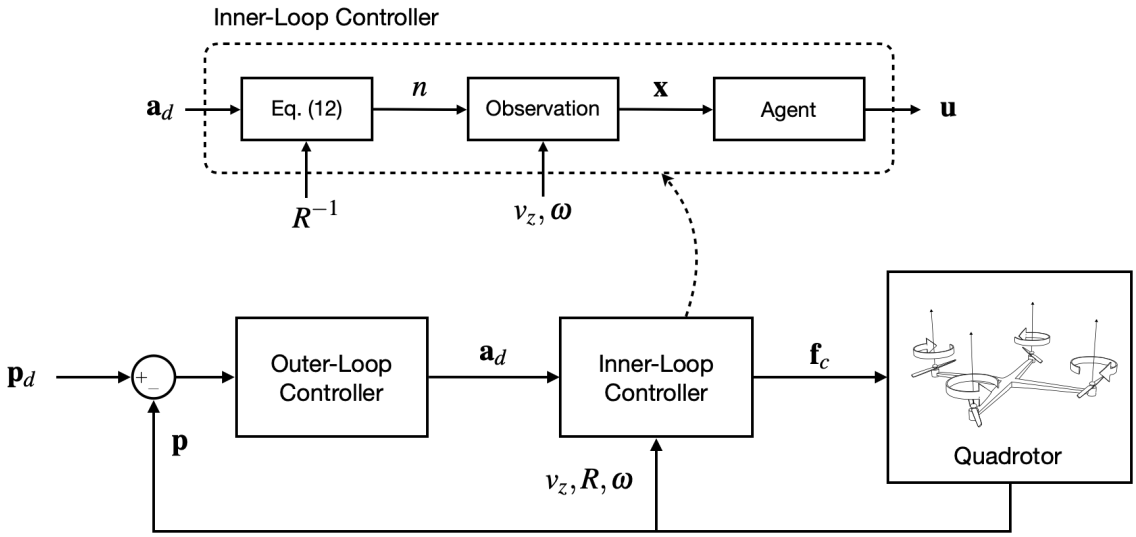

Figure 2 – Double-loop control architecture.

3.1 Outer-Loop Position Controller

A simple proportional-integral-derivative (PID) controller is implemented for the outer-loop position control regarding the quadrotor as a point mass of which the position is controlled by an acceleration input. It takes the desired reference position $\mathbf{p}_d \in \mathbb{R}^3$ and the actual position $\mathbf{p}$ of the quadrotor, and returns the desired average acceleration vector $\mathbf{a}_d \in \mathbb{R}^3$ to the inner loop as follows.

$$\mathbf{a}_d = k_P(\mathbf{p}_d - \mathbf{p}) + k_I \int_0^t (\mathbf{p}_d(\tau) - \mathbf{p}(\tau))\,\mathrm{d}\tau + k_D(\dot{\mathbf{p}}_d - \dot{\mathbf{p}}), \tag{11}$$

where the positive scalars k_P, k_I, and k_D denote the PID gains. Like the position vector $\mathbf{p}$, the desired acceleration vector $\mathbf{a}_d$ is represented in the inertial coordinate system.

3.2 Inner-Loop Attitude Controller

The inner-loop attitude controller takes the desired average acceleration $\mathbf{a}_d$, the rotation matrix R, and the angular velocity ω, and returns the rotor force $\mathbf{f}_c$. The concept of a relaxed hover motion is borrowed from [12], which is a motion consistently spinning around a fixed vector in the inertial coordinate system. In this motion, the thrust vector averaged over the spinning period is parallel to the angular velocity vector and the fixed vector. Hence, the fixed vector can be represented with the desired average acceleration as $\mathbf{a}_d - g\mathbf{e}_3$, where $\mathbf{a}_d$ can be considered as a constant vector in the inner-loop controller design.

To realize the relaxed hover motion, the fixed vector in the inertial coordinate system is described in the body-fixed coordinate system with a unit vector $\mathbf{n} = [n_1, n_2, n_3]^T \in \mathbb{R}^3$ defined by

$$\mathbf{n} := R^{-1} \frac{\mathbf{a}_d - g\mathbf{e}_3}{\|\mathbf{a}_d - g\mathbf{e}_3\|}, \tag{12}$$

where $\|\cdot\|$ denotes the Euclidean norm. Then, the attitude dynamics in (3) gives the following dynamics of $\mathbf{n}$ by assuming $\|\dot{\mathbf{a}}_d\| \ll 1$ from the time-scale separation [12].

$$\dot{\mathbf{n}} = -\omega \times \mathbf{n}. \tag{13}$$

The relaxed hover motion assumes non-aggressive acceleration command, so that

$$\mathbf{n} \cdot \mathbf{e}_3 < 0. \tag{14}$$

In this case, two elements of the unit vector $\mathbf{n}$ are enough to determine $\mathbf{n}$.

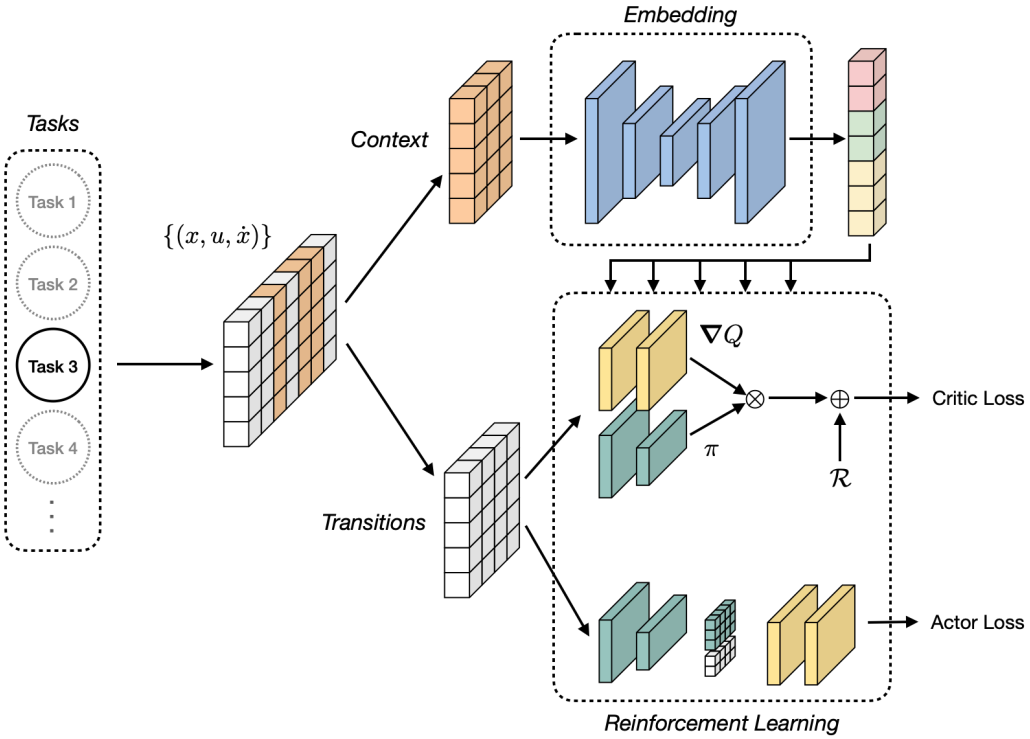

Figure 3 – Meta-training structure.

4. Meta-Reinforcement Learning Formulation

The reinforcement learning (RL) can be viewed as a method for solving optimal control problem based on the dynamic programming. Especially, the model-free RL scheme is one of the promising algorithms to find an optimal control law, when the dynamic model of the plant is unknown. Since the health level of faulty rotors, or equivalently, Λ in (10), is difficult to predict and/or estimate in real-time under urgent fault situations, it may be useful to apply the model-free RL method.

Deep neural networks for RL can effectively approximate an optimal control policy for a complex dynamics using only flight data. However, for different fault situations where the mass, the moment of inertia, and LoE are different, RL requires a lot of flight data and time to train the deep neural networks, which may be unrealistic in the urgent fault scenarios. Simple linear networks are implemented for both policy and critic networks in to be trained with a linear Q-learning method, and the deep neural network structure to represent the complex dynamics of quadrotors are implemented in an embedding network, which infers the fault situation that the quadrotor encountered. The embedding network is trained with a meta-RL scheme, and therefore, a single well-trained embedding network is enough to cope with various fault scenarios.

4.1 Markov Decision Process Formulation

The dynamics of (R, ω) can be fully described by the dynamics of $(\mathbf{n}, \omega)$ in (13) and (4). Since the dynamics of v_z in (2) is also determined by v_z itself and R, the state tuple $(v_z, \mathbf{n}, \omega)$ and the control input $\mathbf{f}_c$ can be regarded as the observation $\mathbf{x}$ and the action $\mathbf{u}$ of a Markov decision process (MDP) as follows.

$$\mathbf{x} := \begin{bmatrix} v_z \\ n_1 \\ n_2 \\ \omega \end{bmatrix} \in \mathcal{X}, \quad \mathbf{u} := \mathbf{f}_c \in \mathcal{U}, \tag{15}$$

where only two elements of $\mathbf{n}$ are used as discussed above. The RL agent in Fig. 2 has a policy $\pi : \mathcal{X} \to \mathcal{U}$ that maps an observation $\mathbf{x}$ to an action $\mathbf{u}$.

As discussed above, the purpose of the inner-loop controller is to make ω parallel to $\mathbf{n}$, which implies $\dot{\mathbf{n}} = 0$ in (13). It has been shown in [12] that the dynamic equations (13) and (4) have an equilibrium

point, denoted by $(\mathbf{n}^e, \omega^e, \mathbf{f}_c^e)$, and the linear quadratic regulator (LQR) designed based on the corresponding linearized model can control the entire system to the relaxed hover solution, where $v_z^e = 0$. Therefore, the reward of the MDP is defined by the negative LQR cost as

$$r = -(\mathbf{x} - \mathbf{x}^e)^T \mathbf{Q} (\mathbf{x} - \mathbf{x}^e) - (\mathbf{u} - \mathbf{u}^e)^T \mathbf{R} (\mathbf{u} - \mathbf{u}^e), \tag{16}$$

where $\mathbf{Q} \in \mathbb{R}^{6\times 6}$ and $\mathbf{R} \in \mathbb{R}^{4\times 4}$ denote the LQR gains, and $\mathbf{x}^e$ and $\mathbf{u}^e$ are the equilibrium defined by

$$\mathbf{x}^e := \begin{bmatrix} 0 \\ n_1^e \\ n_2^e \\ \omega^e \end{bmatrix} \in \mathscr{X}^e, \quad \mathbf{u}^e := \mathbf{f}_c^e \in \mathscr{U}^e, \tag{17}$$

where $\mathscr{X}^e \subset \mathscr{X}$ and $\mathscr{U}^e \subset \mathscr{U}$ denote the set of equilibrium observations and actions, respectively, and $\mathbf{n}^e =: [n_1, n_2, n_3]^T$.

4.2 Near-Hover Solutions

The dynamics of the observation $\mathbf{x}$ in (15) can be written as

$$\dot{\mathbf{x}} = F(\mathbf{x}) + B\mathbf{u}, \tag{18}$$

where the function $F : \mathbb{R}^6 \to \mathbb{R}^6$ and the matrix $B \in \mathbb{R}^{6\times 4}$ are assumed to be unknown. Given $\mathbf{x}_0$ and $\mathbf{u}_0$, for $\delta\mathbf{x} := \mathbf{x} - \mathbf{x}_0$ and $\delta\mathbf{u} := \mathbf{u} - \mathbf{u}_0$ such that $\|\delta\mathbf{x}\| \ll 1$ and $\|\delta\mathbf{u}\| \ll 1$, the dynamic equation in (18) can be approximated by

$$\dot{\mathbf{x}} \simeq F(\mathbf{x}_0) + B\mathbf{u}_0 + A\delta\mathbf{x} + B\delta\mathbf{u} =: F_0(\mathbf{x}_0, \mathbf{u}_0) + F_1(\delta\mathbf{x}, \delta\mathbf{u}), \tag{19}$$

where $A := \nabla_{\mathbf{x}} F(\mathbf{x}_0)$, the function F_1 denotes the linearized part of the dynamics, and $F_0(\mathbf{x}_0, \mathbf{u}_0) := F(\mathbf{x}_0) + B\mathbf{u}_0$. Note that if a point $(\mathbf{x}_0, \mathbf{u}_0)$ is one of the equilibrium of (18), for examples, $(\mathbf{x}^e, \mathbf{u}^e)$ in (17), then $F_0(\mathbf{x}_0, \mathbf{u}_0) = 0$. These points are, indeed, the same as the relaxed hover solutions in (17).

Proposition 1. *If* $\mathbf{a}_d \equiv 0$ *and* $F_0(\mathbf{x}_0, \mathbf{u}_0) = 0$*, then* $(\mathbf{x}_0, \mathbf{u}_0) \in \mathscr{X}^e \times \mathscr{U}^e$*.*

Since $\mathbf{n}^e$ is a unit vector parallel to ω^e, and from the constraint in (14), a relaxed hover solution can be fully represented by

$$\mathbf{h}^e := (\omega^e, \mathbf{u}^e) \tag{20}$$

which is defined as a near-hover solution. The proposed meta-RL scheme trains the embedding network and the RL agent to find a near-hover solution appropriate to the current fault situation.

4.3 Task Definition

In each fault scenario, the quadrotor may have different dynamics, which makes it plausible to assign a task to each fault case. However, the linearized model in (19) is accurate only in a small neighborhood of the linearization point $(\mathbf{x}_0, \mathbf{u}_0) \in \mathscr{X} \times \mathscr{U}$, and therefore, the space $\mathscr{X} \times \mathscr{U}$ should be divided into several regions, and each region should be defined as a single task combined with the fault.
An i-th task is denoted by $\mathscr{T}_i \in \mathscr{T}$, where $\mathscr{T}$ is a set of all tasks, as in the meta-learning literature. For each task $\mathscr{T}_i$, the quadrotor may have different linearized dynamics as discussed above, but a set of transitions $\Xi_i := \{\xi_i\}$, where the transition is defined by

$$\xi_i(t) := \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \\ \dot{\mathbf{x}}(t) \end{bmatrix}, \tag{21}$$

may contain enough information of the linear model of task $\mathscr{T}_i$.

4.4 Embedding Network

The embedding network infers the task and can give some information to the inner RL agent to have a (sub-)optimal policy immediately. This information is encoded with a latent variable $\mathbf{z}$. The probabilistic embeddings for actor-critic RL (PEARL) [21], where $\mathbf{z}$ is a probabilistic variable, has been widely used for meta-RL synthesis, because it has shown superior performances in general dynamic systems. Unlike the original PEARL method, the embedding network in this study returns $\mathbf{z}$ deterministically, because the fault scenario is uncertain but less probabilistic.
The embedding network μ is a deep neural network with parameter ϕ, which infers $\mathbf{z}$ as follows.

$$\mathbf{z} \simeq \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{u}_0 \\ \mathbf{e} \end{bmatrix} = \mu(C_i; \phi), \tag{22}$$

where $\mathbf{e}$ denotes the embedding vector which will be discussed in the next section. The context C_i is a matrix defined by

$$C_i = \begin{bmatrix} \mathbf{x}(t_1) & \cdots & \mathbf{x}(t_N) \\ \mathbf{u}(t_1) & \cdots & \mathbf{u}(t_N) \\ \dot{\mathbf{x}}(t_1) & \cdots & \dot{\mathbf{x}}(t_N) \end{bmatrix} = \begin{bmatrix} \xi_i(t_1) & \cdots & \xi_i(t_N) \end{bmatrix}, \tag{23}$$

with randomly sampled transitions $\{\xi_i\} \subset \Xi_i$.

5. Training Algorithms

5.1 Meta-Training

The policy of the RL agent is designed as the following linear feedback control law.

$$\pi(\mathbf{x}, \mathbf{z}) = \mathbf{u}_0 - K(\mathbf{e}; \psi)(\mathbf{x} - \mathbf{x}_0), \tag{24}$$

where K is a function of the embedding vector $\mathbf{e}$ and the policy network parameter ψ such that $K(\mathbf{e}; \psi) \in \mathbb{R}^{4\times 6}$. The objective of meta-RL is to infer a tuple $(\mathbf{x}_0, \mathbf{u}_0) \in \mathscr{X} \times \mathscr{U}$ such that $F_0(\mathbf{x}_0, \mathbf{u}_0) = 0$, which implies that the tuple is a relaxed hover solution, as well as to estimate an optimal gain $K(\mathbf{e}; \psi)$. Therefore, the synthesized meta-RL can be viewed as an automatic gain scheduling method where the scheduling variable is the latent variable $\mathbf{z}$ which encodes the information of task $\mathscr{T}_i$.
The optimal policy π can be obtained through an off-policy RL method which is modified for linear systems to ensure the stability of the policy using a Q-function, denoted by $\mathscr{Q}$. Restricting the structure of the Q-function to have a quadratic form of $(\mathbf{x}, \mathbf{u})$, a linear network can represent the gradient of the Q-function as follows.

$$\nabla \mathscr{Q}(\mathbf{x}, \mathbf{u}, \mathbf{z}) := \begin{bmatrix} \nabla_{\mathbf{x}} \mathscr{Q}(\mathbf{x}, \mathbf{u}, \mathbf{z}) \\ \nabla_{\mathbf{u}} \mathscr{Q}(\mathbf{x}, \mathbf{u}, \mathbf{z}) \end{bmatrix} \simeq H(\mathbf{e}; \theta)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}_0 \\ \mathbf{u} - \mathbf{u}_0 \end{bmatrix}, \tag{25}$$

where H is a function of the embedding vector $\mathbf{e}$ and the critic parameter θ such that $H(\mathbf{e}; \theta) \in \mathbb{R}^{10\times 10}$. We propose a learning framework called a linear Q-learning to train the above linear networks for the policy and the gradient Q-function using the following modified Hamilton-Jacobi-Bellman (HJB) equation.

$$0 = [\nabla \mathscr{Q}(\mathbf{x}, \mathbf{u}, \mathbf{z})]^T \begin{bmatrix} \dot{\mathbf{x}} \\ -K\dot{\mathbf{x}} - s(\mathbf{u} - \pi(\mathbf{x}, \mathbf{z})) \end{bmatrix} + \mathscr{R}(\delta\mathbf{x}, \delta\mathbf{u}) =: \mathscr{H}(\xi, \mathbf{z}), \tag{26}$$

$$0 = \nabla_{\mathbf{u}} \mathscr{Q}(\mathbf{x}, \pi, \mathbf{z}), \tag{27}$$

where the scalar $s > 0$ is the design parameters of the linear Q-learning. The quadratic cost function $\mathscr{R}$ is given by

$$\mathscr{R}(\delta\mathbf{x}, \delta\mathbf{u}) := \delta\mathbf{x}^T \mathbf{Q} \delta\mathbf{x} + \delta\mathbf{u}^T \mathbf{R} \delta\mathbf{u}, \tag{28}$$

where Q and R are the positive semi-definite and positive definite symmetric matrices, respectively. The critic and actor losses for a task $\mathscr{T}_i$ are defined as follows.

$$\mathscr{L}_{\text{critic}}^{\mathscr{T}_i}(\phi, \theta) = |\mathscr{H}(\xi, \mathbf{z})| + \max(\|\mathbf{e}\| - 1,\ 0), \tag{29}$$

$$\mathscr{L}_{\text{actor}}^{\mathscr{T}_i}(\psi) = \|\nabla_{\mathbf{u}} \mathscr{Q}(\mathbf{x}, \pi, \mathbf{z})\|^2, \tag{30}$$

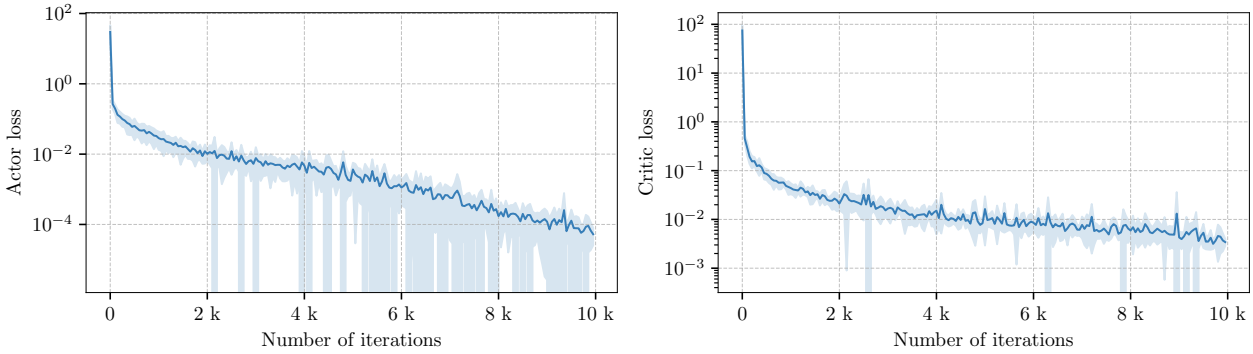

Figure 4 – Traning history of meta-RL.

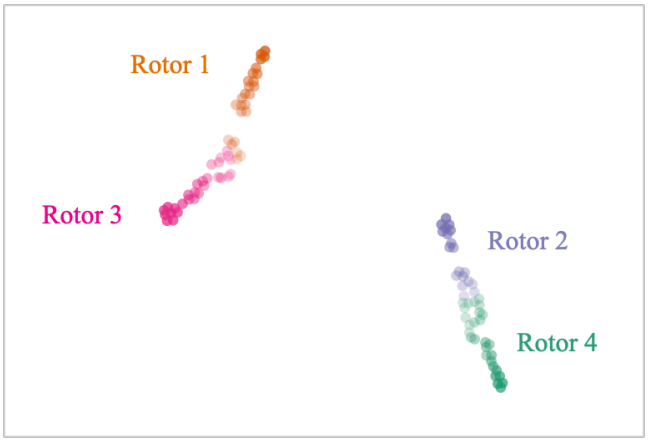

Figure 5 – The latent variable from the trained embedding network represented using t-SNE.

where the second term in (29) penalizes the embedding vector of which the norm is greater than 1. Note that the embedding network parameter ϕ is updated using the critic loss $\mathscr{L}^{\mathscr{T}_i}_{\text{critic}}$, because $\mathscr{L}^{\mathscr{T}_i}_{\text{critic}} \equiv 0$ implies that $F_0(\mathbf{x}_0, \mathbf{u}_0) \equiv 0$. Therefore, if the meta-RL networks are trained completely for a set of all tasks $\mathscr{T}$ and the optimal parameters $(\phi^*, \theta^*, \psi^*)$ are found that satisfy the following equations

$$\mathbb{E}_{\mathscr{T}_i \sim \mathscr{T}} \left[\mathscr{L}^{\mathscr{T}_i}_{\text{critic}}(\phi, \theta) \right] = 0, \tag{31}$$

$$\mathbb{E}_{\mathscr{T}_i \sim \mathscr{T}} \left[\mathscr{L}^{\mathscr{T}_i}_{\text{actor}}(\psi) \right] = 0, \tag{32}$$

the embedding network will return the optimal latent vector $\mathbf{z}^*$ in (22), which can be used to obtain the human readable near-hover equilibrium $(\mathbf{x}^e, \mathbf{u}^e)$ and the optimal gain K^* for the encountered task. In the meta-training stage, a large set of tasks should be collected. Due to the nature of embedding network and model-free RL, both simulation and real flight data can be used. Using this task set, all the parameters of the embedding network as well as the networks of RL agent are trained offline.

5.2 Fine-Tuning

After the meta-training, the networks will be implemented into the flight control computer of the quadrotor, and the fine-tuning will be performed in online. Using a small amount of transition data obtained during the flight, the trained embedding network infers the near-hover equilibrium $(\mathbf{x}_0, \mathbf{u}_0)$ and the embedding vector $\mathbf{e}$ corresponding to the current situation. The policy and the gradient Q-function are modified as

$$\pi(\mathbf{x}; \hat{K}) = \mathbf{u}_0 - \hat{K}(\mathbf{x} - \mathbf{x}_0), \tag{33}$$

$$\nabla \mathscr{Q}(\mathbf{x}, \mathbf{u}; \hat{H}) = \hat{H}^T \begin{bmatrix} \mathbf{x} - \mathbf{x}_0 \\ \mathbf{u} - \mathbf{u}_0 \end{bmatrix}, \tag{34}$$

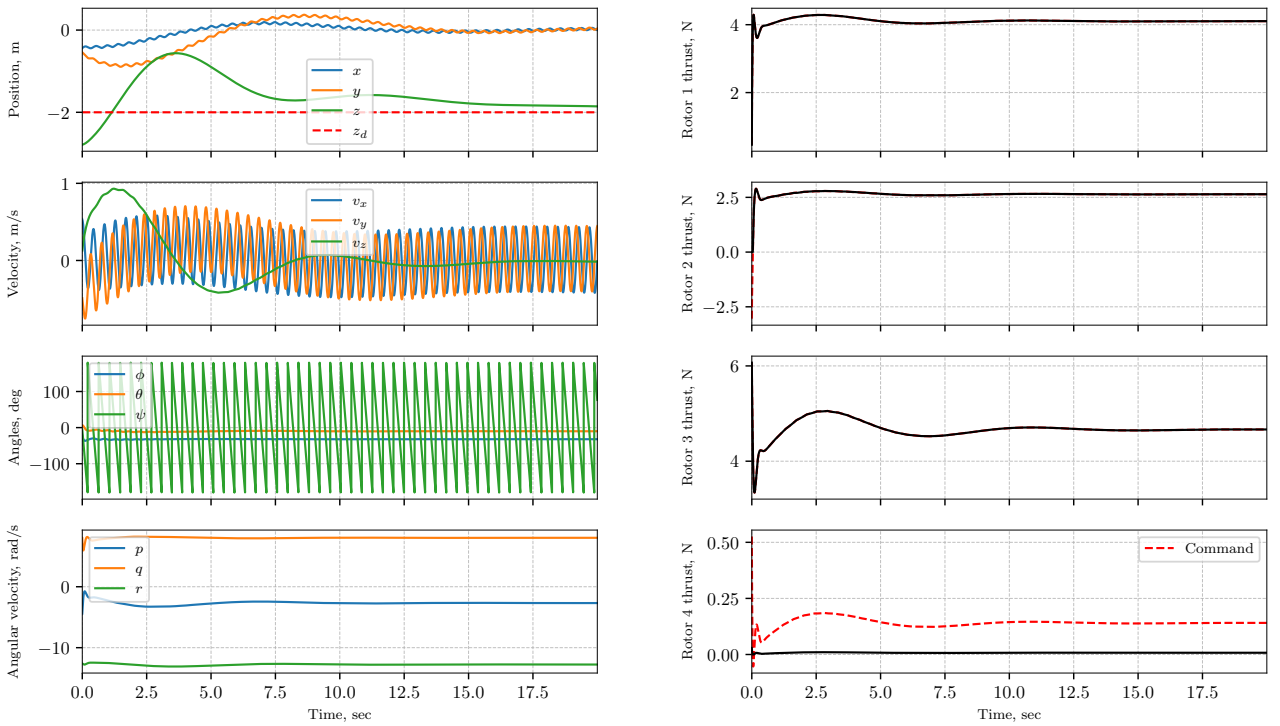

Figure 6 – State and control input responses for the task with 95% LoE on the rotor 4.

and the parameters $\hat{K}$ and $\hat{H}$ are initialized with the trained models and the embedding vector as follows.

$$\hat{K} = K(\mathbf{e}; \psi), \quad \hat{H} = H(\mathbf{e}; \theta). \tag{35}$$

Then, the parameters are trained online with the modified HJB equation in (26). This training process is fast and data-efficient since it is an off-policy algorithm. Moreover, the stability of the learnt policy can be guaranteed in the neighborhood of the near-hover equilibrium.

6. Numerical Simulation

This setion demonstrates the validity of the proposd meta-RL basd FTC for a quadrotor through a numerical simulation. The simulation time step is 0.001 seconds, and the derivative $\dot{\mathbf{x}}$ is obtained from two subsequent observations and the Euler method, while the simulation itself uses Runge-Kutta 4th-order method for integration. A task dataset is prepared for meta-training stage with 100 randomly generated tasks and 1,500 transitions for each task. A task is defined with a random $\lambda_j \in [0.8, 1]$ occurred in a random rotor single rotor j and random losses in mass and moment of inertia up to 5%. To ensure all the transitions for each task are contained in a small region which posesses an equilibrium point of the task, one relaxed hover solution is first obtained by assuming the faulty rotor of the task is completely failed, and the transitions near the relaxed hover solution will only be stored.

In the meta-training stage, the networks are updated using Adam optimizer [28] with a learning rate 0.001. The embedding network has four fully connected linear layers of (64, 128, 128, 64) neurons and the batch normalization and the ReLU activation functions are used for each layers, except for the output layer. The dimenstion of the embedding vector is set to 50, and also 50 transitions are randomly sampled from the transition set of each task to construct the context C_i. The LQR gain matrices $\mathbf{Q}$ and $\mathbf{R}$ and the design parameter s for the modified HJB equation in (26) are set to

$$\mathbf{Q} = \mathrm{diag}(1, 20, 20, 0, 0, 0), \quad \mathbf{R} = I_4, \quad s = 1. \tag{36}$$

The critic loss and the actor loss during the meta-training stage is presented in Fig. 4, where the average and the standard deviation are obtained from 10 runs. It can be seen that losses decrease consistently during the training.

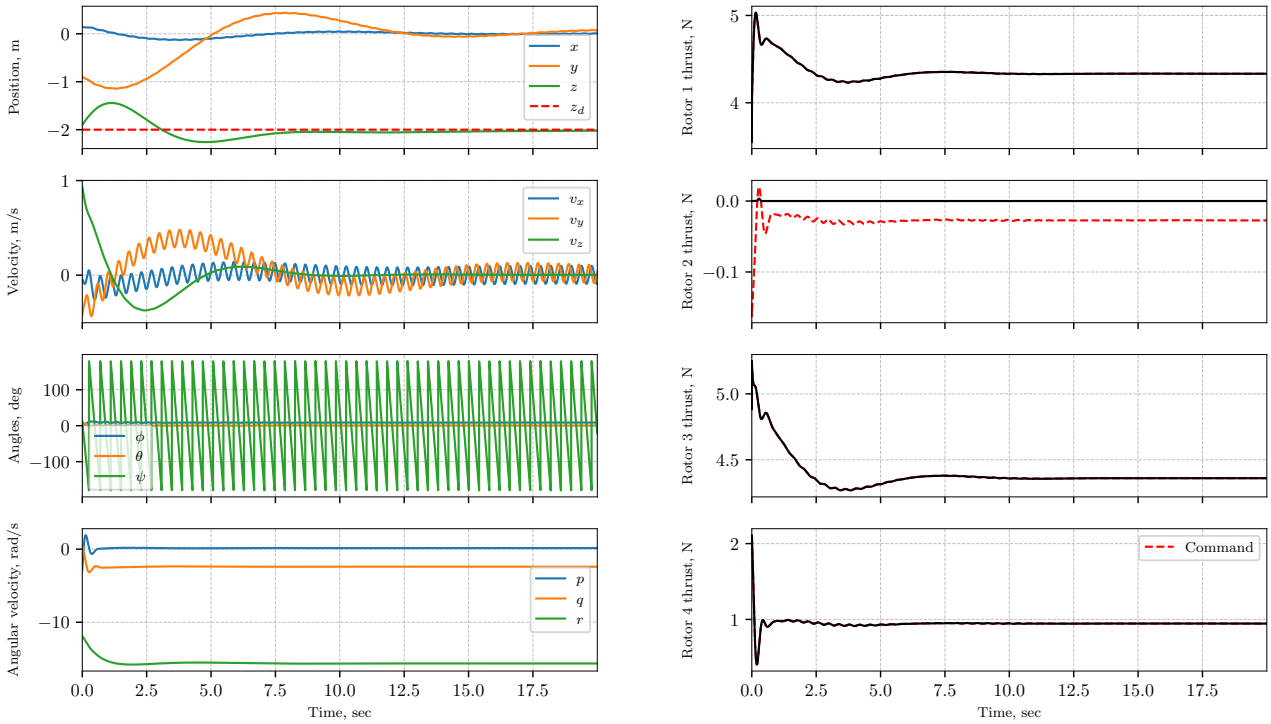

Figure 7 – State and control input responses for the task with 83% LoE on the rotor 2.

After the meta-training stage, the output latent variable of the embedding network is demonstrated in Fig. 5, where t-SNE is used for downsizing the dimenstion from 60 to 2. Different colors indicate the different faulty rotors. Since various equilibriums exist for each rotor fault case, the latent variable is distributed in some range. The transparent scatter points in Fig. 5 imply the near-hover solutions not using the opposite rotor of the faulty rotor, and therefore, it can be seen that for the opposite faulty rotors have latent variables located close to each other.
In the fine-tuning stage, only 1,000 transitions are used to train the final control policy, which can be gathered in 1 second in this setup. Two different tasks that have different faulty rotors are tested with a single meta-training result to show the performance of the proposed meta RL-based FTC. The state and control input responses for these tasks are shown in Fig. 6 and 7.

7. Conclusion

A fault-tolerant control architecture for a quadrotor is proposed using a meta-RL framework. An embedding network and a linear Q-learning agent is trained using a modifed Hamilton-Jacobi-Bellman equation with a bunch set of tasks. Once the networks are trained, it can infer a near-hover solution appropriate to the current state of the quadrotor, and the inner RL agent with a linear policy can be fine-tuned quickly in online. The effectiveness of the proposed framework is demonstrated by a numerical simulation.

8. Acknowledgment

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2019R1A2C2083946).

9. Contact Author Email Address

Mailto: kshoon92@gmail.com

10. Copyright Statement

References

[1] A. A. Amin and K. M. Hasan, "A review of fault tolerant control systems: advancements and applications," *Measurement*, vol. 143, pp. 58–68, 2019.

[2] X. Zhang, Y. Zhang, C.-Y. Su, and Y. Feng, "Fault-tolerant control for quadrotor UAV via backstepping approach," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. Orlando, Florida: American Institute of Aeronautics and Astronautics, Jan. 2010.

[3] S. Zeghlache, A. Djerioui, L. Benyettou, T. Benslimane, H. Mekki, and A. Bouguerra, "Fault tolerant control for modified quadrotor via adaptive type-2 fuzzy backstepping subject to actuator faults," *ISA Transactions*, vol. 95, pp. 330–345, 2019.

[4] A.-R. Merheb, H. Noura, and F. Bateman, "Design of passive fault-tolerant controllers of a quadrotor based on sliding mode theory," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 3, pp. 561–576, 2015.

[5] S. Barghandan, M. A. Badamchizadeh, and M. R. Jahed-Motlagh, "Improved adaptive fuzzy sliding mode controller for robust fault tolerant of a quadrotor," *International Journal of Control, Automation and Systems*, vol. 15, no. 1, pp. 427–441, 2017.

[6] B. Wang and Y. Zhang, "An adaptive fault-tolerant sliding mode control allocation scheme for multirotor helicopter subject to simultaneous actuator faults," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4227–4236, 2018.

[7] S. Mallavalli and A. Fekih, "A fault tolerant tracking control for a quadrotor UAV subject to simultaneous actuator faults and exogenous disturbances," *International Journal of Control*, vol. 93, no. 3, pp. 655–668, 2020.

[8] P. Tang, D. Lin, D. Zheng, S. Fan, and J. Ye, "Observer based finite-time fault tolerant quadrotor attitude control with actuator faults," *Aerospace Science and Technology*, vol. 104, p. 105968, 2020.

[9] H. Başak, E. Kemer, and E. Prempain, "A passive fault-tolerant switched control approach for linear multivariable systems: application to a quadcopter unmanned aerial vehicle model," *Journal of Dynamic Systems, Measurement, and Control*, vol. 142, no. 3, p. 031004, 2020.

[10] A. Freddi, A. Lanzon, and S. Longhi, "A feedback linearization approach to fault tolerance in quadrotor vehicles," in *Proceedings of the 18th IFAC World Congress*, Milano, Italy, Aug. 2011, pp. 5413–5418.

[11] A. Lanzon, A. Freddi, and S. Longhi, "Flight control of a quadrotor vehicle subsequent to a rotor failure," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 580–591, 2014.

[12] M. W. Mueller and R. D'Andrea, "Relaxed hover solutions for multicopters: application to algorithmic redundancy and novel vehicles," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 873–889, 2016.

[13] S. Sun, L. Sijbers, X. Wang, and C. de Visser, "High-speed flight of quadrotor despite loss of single rotor," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3201–3207, 2018.

[14] N. Nguyen and S. Hong, "Fault diagnosis and fault-tolerant control scheme for quadcopter UAVs with a total loss of actuator," *Energies*, vol. 12, no. 6, p. 1139, 2019.

[15] Z. Hou, P. Lu, and Z. Tu, "Nonsingular terminal sliding mode control for a quadrotor UAV with a total rotor failure," *Aerospace Science and Technology*, vol. 98, p. 105716, 2020.

[16] Y. Wu, K. Hu, X.-M. Sun, and Y. Ma, "Nonlinear control of quadrotor for fault tolerance: a total failure of one actuator," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 5, pp. 2810–2820, 2021.

[17] H.-J. Ma, L.-X. Xu, and G.-H. Yang, "Multiple environment integral reinforcement learning-based fault-tolerant control for affine nonlinear systems," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1913–1928, 2021.

[18] Y. Sohege, M. Quinones-Grueiro, and G. Provan, "A novel hybrid approach for fault-tolerant control of uavs based on robust reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 10 719–10 725.

[19] W. Zhao, H. Liu, and F. L. Lewis, "Data-driven fault-tolerant control for attitude synchronization of nonlinear quadrotors," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5584–5591, 2021.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. Sydney, Australia: PMLR, Aug. 2017, pp. 1126–1135.

[21] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97. Long Beach, California: PMLR, Jun. 2019, pp. 5331–5340.

[22] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-learning in neural networks: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022.

[23] Z. Liu, C. Yuan, Y. Zhang, and J. Luo, "A learning-based fault tolerant tracking control of an unmanned quadrotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 145–162, 2016.

[24] Z. Song, S. Ling, and K. Sun, "Adaptive fault tolerant attitude tracking control for miniature rotorcrafts under actuator saturation," *Aerospace Science and Technology*, vol. 69, pp. 27–38, 2017.

[25] K. Yan, M. Chen, and Q. Wu, "Neural network-based adaptive fault tolerant tracking control for unmanned autonomous helicopters with prescribed performance," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 12, pp. 4350–4362, 2019.

[26] N. P. Nguyen, N. X. Mung, L. N. N. Thanh Ha, T. T. Huynh, and S. K. Hong, "Finite-time attitude fault tolerant control of quadcopter system via neural networks," *Mathematics*, vol. 8, no. 9, p. 1541, 2020.

[27] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *2014 IEEE International Conference on Robotics & Automation (ICRA)*, Hong Kong, China, May 2014, pp. 45–52.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, May 2015, pp. 1–15.