Reinforcement Learning-Based Fault-Tolerant Control for Quadrotor UAVs under Actuator Fault

Xiaoxu Liu, Zike Yuan, Zhiwei Gao, Wenwei Zhang

***Abstract*— Quadrotor UAVs, renowned for their agility and versatility, are extensively utilized in a range areas. However, their inherent underactuated dynamic characteristics render them particularly vulnerable to external disturbances and systemic failures. To address this issue, our study introduces a hybrid control method tailored to combat the most prevalent types of drone failures — actuator faults. This innovative approach leverages reinforcement learning to enhance fault tolerance. Specifically, we employ reinforcement learning techniques to output compensatory control signals that bolster the core functionalities of the base controller. This integration aims to preserve the stability and continuity of mission-critical tasks even in the face of operational faults, thereby ensuring robust safety controls. We utilized the Proximal Policy Optimization (PPO) algorithm for the strategic training of our control systems. We test in both simulated environments and real-world scenarios was conducted to evaluate the efficacy of our control strategy under conditions of actuator failure. The results affirm that our method significantly enhances the safety and stability of drone operations, maintaining control integrity during rotor failures.**

***Index Terms*— Quadrotor UAV, Fault-Tolerant Control, Reinforcement Learning, PPO, Actuator Fault**

I. Introduction

QUADROTOR UAVs, renowned for their agility and speed, are extensively employed in sectors such as agriculture [1], disaster relief [2], and industrial power line inspections [3]. As interactions with drones in production and daily life become increasingly frequent, ensuring drone safety and reliability has surfaced as a pivotal research issue in this domain. Addressing safety control challenges, considerable advancements have been achieved in fault diagnosis and fault-tolerant control research [4]–[9]. These developments not only enhance system reliability and safety but also significantly contribute to safeguarding personal and property security.

Manuscript created July, 2024

This work was supported by National Natural Science Foundation of China under Grant 62003218, Stable Support Projects for Shenzhen Higher Education Institutions under Grant 20220717223051001, and Guangdong Provincial Engineering Technology Research Center for Materials for Advanced MEMS Sensor Chip under Grant 2022GCZX005.(Corresponding authors: Zhiwei Gao.)

Xiaoxu Liu, Zike Yuan, and Wenwei Zhang are with the Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, Shenzhen 518118, China (e-mail: liuxiaoxu@sztu.edu.cn; yuanzike2022@email.szu.edu.cn; zhangwenwei@sztu.edu.cn). Zhiwei Gao is with the Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Newcastle upon Tyne, NE1 8ST, U.K. (e-mail: zhiwei.gao@northumbria.ac.uk).

In recent years, researchers have extensively explored fault diagnosis and fault-tolerant control for drones [10]–[16]. Numerous fault-tolerant control methods have been implemented to solve control issues in quadrotor drones. For instance, in response to single actuator failures in quadrotors, Model Predictive Control (MPC) has been designed to sustain flight operations under fault conditions [10]. For drones with faults in two opposing rotors, a nonlinear sensor-based fault-tolerant controller has been developed to manage high-speed flight [11]. Another approach stabilizes drones with faults ranging from a single rotor up to three rotors [12]. Additionally, methods such as sliding mode control [17], adaptive control [18], and active disturbance rejection control [19] have been effectively used to stabilize quadrotors.

With the advent of reinforcement learning, which showcases adaptability and model-free advantages, learning-based drone control methods have increasingly come under scrutiny. A reinforcement learning-based quadrotor control method has been proposed, facilitating high-speed and large-angle flights through trained control strategies [20]. In the realm of fault-tolerant control, reinforcement learning eliminates the need for modeling the faulty system; instead, interaction with a simulation environment suffices for fault tolerance, making it a viable option for integrating reinforcement learning concepts into fault tolerance strategies. In cases of spoofing attack signals, a hierarchical control method has been proposed [21]. This hybrid control approach, consisting of a reinforcement learning auxiliary controller and a fundamental controller, achieves fault tolerance for the quadcopter under conditions of fault attack. [22] adopts the above-mentioned hybrid control framework and learns to adjust the PID controller parameters through reinforcement learning to achieve fault-tolerant control.The aforementioned hybrid control method effectively enhances the system's fault tolerance and disturbance rejection capabilities, yet it exhibits certain limitations. Utilizing a single strategy for fault tolerance against various complex attacks and malfunctions performs well during hovering, but its efficacy in real flight conditions remains questionable. Additionally, the discrepancy between the drone's expected and actual states, varying with the objectives of the mission, also impacts the effectiveness of the fault tolerance strategy. Another study [23] combined reinforcement learning with MPC and employed fault diagnosis to categorize actuator faults for fault-tolerant control, addressing potential stability issues in strategy output during task execution by comparing with a reference model. Although this learning-enhanced MPC method provides precise fault-tolerant control, the optimiza-

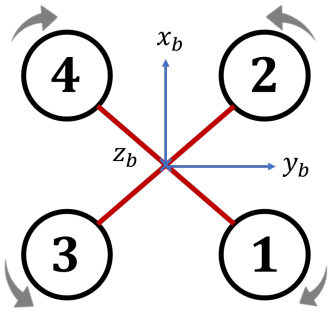

Fig. 1. UAV body coordinates and rotor number

tion costs are high, posing significant computational challenges for drones requiring high control frequencies.

To address these issues in reinforcement learning-based fault tolerance, we have optimized the approach by adopting the hierarchical control method used in [21], [22], but unlike [22], instead of optimizing controller parameters, we directly generate compensation signals through reinforcement learning strategies. Similarly, we use the deviation between the actual model state outputs and the ideal model, replacing the state error from [23] to maintain the drone's original mission as much as possible during faults. Our proposed method employs a loosely coupled, mixed control approach that ensures the execution of flight tasks through reinforcement learning. The base controller can vary according to control needs and schemes, while the RL controller outputs auxiliary fault-tolerant control strategies at a lower control frequency, ensuring safe and stable flight in the event of motor failures, while maintaining relative control precision.We employed the Proximal Policy Optimization (PPO) [24] to train our auxiliary fault tolerance strategy, deploying it in both simulated and real-world settings to validate the reliability of our approach.

The structure of our paper is as follows: Section II details the dynamic modeling of the drone used in our study, along with the modeling and analysis of faults. In Section III, we describe how to establish a reinforcement learning environment and train our strategy using the Proximal Policy Optimization (PPO) algorithm. Section IV will demonstrate the fault tolerance performance of our method in both simulated and real-world scenarios.

II. System Description And Problem Formulation

A. UAV Moodel Description

We conducted simulations and experiments with a quadrotor UAV configured in an X-shaped formation.Referring to [25], we model the drone. The UAV is powered by four motors that generate the required thrust. The motors receive PWM signals and produce a corresponding thrust vector $\boldsymbol{T} = [T_1\ T_2\ T_3\ T_4]$. Total thrust $T_m = T_1+T_2+T_3+T_4$ represents the sum of the thrust produced by all four motors. Under the right-hand rule, the motor numbering and coordinate orientation of the UAV's body frame are illustrated in Fig. 1. The attitude angles in the body frame are defined as $\Omega = [\phi, \theta, \psi]^T$, where ϕ, θ, and ψ represent the roll, pitch, and yaw angles, respectively. We define $\boldsymbol{\omega_b} = [p\ q\ r]^T$ as the rotational angular velocities about the body frame's three axes. The UAV's dynamic model can be expressed by the Newton-Euler equations, as shown in (1).

$$\begin{cases} m\ddot{\boldsymbol{p}}_e + mg\boldsymbol{e}_3 = \boldsymbol{F} + \boldsymbol{F}_d \\ \boldsymbol{J}\dot{\boldsymbol{\omega}}_{\boldsymbol{b}} + \boldsymbol{\omega_b} \times (\boldsymbol{J}\boldsymbol{\omega}_b) = \boldsymbol{\tau} + \boldsymbol{\tau}_d \end{cases} \tag{1}$$

where $\boldsymbol{e}_3 = [0, 0, 1]^T$ denotes the unit vector along the vertical axis in the inertial frame, and m and g represent the total mass of the UAV and the acceleration due to gravity, respectively, the UAV's position in the reference coordinate system is indicated by $\boldsymbol{p_e}$. The total thrust force $F = {}^I R_B T_m \boldsymbol{e_3}$ reflects the components of the UAV's thrust in the axes of the reference coordinate system, with ${}^I R_B$ being the transformation matrix that converts coordinates from the body frame to the reference frame.

$${}^I R_B = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}. \tag{2}$$

The inertia matrix is represented by $\boldsymbol{J} = diag\{I_{xx}, I_{yy}, I_{zz}\}$, where I_{xx}, I_{yy}, and I_{zz} are the moments of inertia about the principal axes. The vector $\boldsymbol{\tau} = [\tau_\phi, \tau_\theta, \tau_\psi]$ denotes the torques about the three axes: roll, pitch, and yaw, respectively. The mapping relationship among the generated thrust, torque, and PWM inputs can be simplified as

$$\begin{bmatrix} T_m \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} K_f & K_f & K_f & K_f \\ -d_\theta K_f & -d_\theta K_f & d_\theta K_f & d_\theta K_f \\ d_\phi K_f & -d_\phi K_f & d_\phi K_f & -d_\phi K_f \\ K_t & K_t & K_t & K_t \end{bmatrix}}_{M} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}. \tag{3}$$

The vector $\boldsymbol{u} = [u_1, u_2, u_3, u_4]^T$ represents the PWM input for each motor, where the PWM inputs satisfy $0 \le u_i \le 1$ for $i = 1, 2, 3, 4$. d_θ and d_ϕ refer to the half distance of roll and pitch rotor to rotor,respectively. K_f and K_t represent the coefficients of thrust and torque,respectively.

B. Fault Representation

UAV actuator fault can be characterized by (4) and (5). These failures are commonly caused by long-term operation leading to motor bearing overheating, stator winding faults, bearing failures, and the loss of propeller blades, among other reasons.

$$u_i^f = u_i - \Delta_i \tag{4}$$

$$u_i^f = u_i(1 - \Delta_i) \tag{5}$$

In the event of a failure, ensuring the UAV's safe and stable flight and maintaining flight quality are the focal points of our research. However, we do not consider situations where severe motor failures lead to a complete loss of control, necessitating a switch to an alternative controller for fault tolerance; thus, the failure magnitude Δ is considered to be within the range of $[0.1, 0.4]$. Typically, it is rare for multiple motors to fail simultaneously. Motor failures tend to occur sequentially, which provides the operator with the opportunity to detect and replace the faulty motors before more serious incidents can occur. Therefore, in the context of this paper, we address the fault tolerance issue with respect to random individual motor failures.

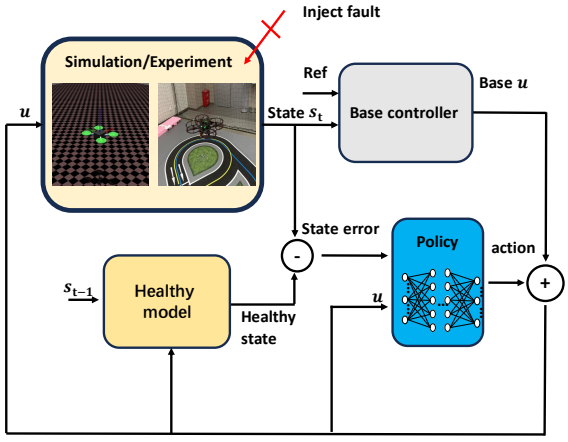

Fig. 2. The reinforcement learning-assisted fault-tolerance control framework uses a standard controller to produce the main control signals from the current state. The reinforcement learning strategy then provides a fault-tolerance compensatory control policy, which is based on the difference between the actual and the ideal model states, as well as the present throttle signal.

III. Auxiliary Fault-tolerant Control Strategy Based on Reinforcement Learning

In this section, we elaborate on the reinforcement learning-based auxiliary fault-tolerant control method that we have developed. The structure of our controller is depicted in Fig. 2. Leveraging the existing control infrastructure, our approach involves generating compensatory control signals to address motor failures using a sophisticated reinforcement learning strategy, thus ensuring the safe operation of the UAV.

Similar to the method discussed in [21], our reinforcement learning controller serves as a high-level auxiliary system, issuing control commands at a lower frequency. We utilize the Proximal Policy Optimization (PPO) algorithm to optimize our auxiliary fault-tolerant control strategy.

Upon detecting a fault, the auxiliary controller, informed by real-time UAV state data, promptly activates a fault-tolerant compensation strategy. This measure guarantees the UAV's stability and continuity of flight, safeguarding against potential disruptions caused by system failures.

A. PPO Based Auxiliary Controller Design

We employ the PPO algorithm [24] from reinforcement learning to optimize our strategy. The reinforcement learning approach is often conceptualized as a Markov Decision Process (MDP) [26], which can be formulated as $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \gamma, r)$, where $\mathbf{S}$ denotes the state space, $\mathbf{A}$ represents the action space, and $\mathbf{P}$ is the state transition probability function. The system commences from an initial state and progresses for T time steps, selecting an action $a \in \mathbf{A}$ at each step and receiving a corresponding reward r, resulting in a trajectory $(s_{1:T+1}, a_{1:T}, r_{1:T})$. Our objective is to derive a parameterized policy π_θ, where θ is referred to as the policy parameter, with the aim of maximizing the average value as much as possible. This objective can be achieved by (6).

$$J(\pi_\theta) = \mathbb{E}_S\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k}\right] = \int_S d^{\pi_\theta}(s)V^{\pi_\theta}(s)ds, \quad (6)$$

where

$$V^{\pi_\theta}(s) = \mathbb{E}_S\left[\sum_{t=t}^{\infty}\gamma^t r^t | s_t = s, \pi_\theta\right]. \quad (7)$$

PPO algorithm uses the Actor-Critic network architecture. In this setup, the Actor policy network determines the actions to be taken given the current state, while the Critic network evaluates the quality of these actions under the current policy. The Critic continually updates to provide a more accurate estimate of the value returns, and the Actor adjusts its policy parameters based on feedback from the Critic, aiming to maximize long-term rewards.The PPO algorithm incorporates an improved objective function for task optimization, with the form of this objective function presented as shown in (8).

$$L(\theta) = \hat{\mathbb{E}}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \quad (8)$$

Wherein, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ denotes the probability ratio between the new policy and the old policy, and $\hat{A}_t$ is the estimate of the advantage functionwitch can be written as (9).

$$\begin{aligned}\hat{A}_t &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &\approx r + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\end{aligned} \quad (9)$$

The advantage function represents the relative value benefit of taking a certain action compared to the current policy action. The term ϵ is a clipping factor used to limit large discrepancies between the new and old policies, thereby preventing excessively large strategy updates. The essence of PPO lies in constraining the extent of policy updates to avoid substantial policy shifts, which in turn enhances the stability of training.

The Critic network employs the Temporal Difference (TD) method for updating weights. Our goal is to reduce the error in value estimation. The loss function can be expressed as:

$$L(\phi) = \hat{\mathbb{E}}\left[\left(r_t + \gamma V(s_{t+1};\phi) - V(s_t;\phi)\right)^2\right] \quad (10)$$

This formula represents the expected value of the squared difference between the predicted value of the current state and the reward plus the discounted value of the next state, thereby quantifying the error in the value estimation made by the network.

To prevent premature convergence of the agent, which could result in insufficient exploration of policies, we incorporate Entropy(π_θ) into the loss function. Consequently, the final expression for the loss function is:

$$\text{loss} = -\alpha L(\theta) + \beta L(\phi) - \eta \cdot \text{Entropy}(\pi_\theta) \quad (11)$$

This formulation ensures that the entropy term encourages policy diversity, hence promoting more thorough exploration during the learning process.

The pseudo code of the PPO algorithm is shown in Algorithm 1.

The selection of state and action is in accordance with the formula presented in (12). Where $\hat{x}$ represents the state output

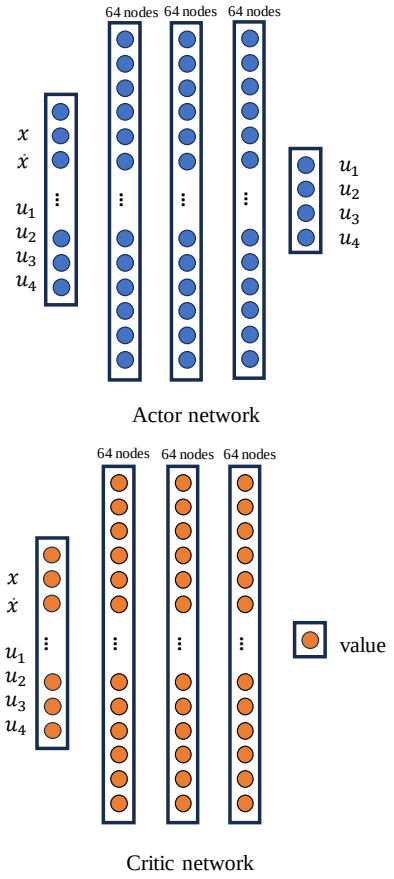

Fig. 3. Network structure of the Actor and Critic. The state input passes through a three-layer fully connected neural network, subsequently outputting the action and the corresponding value estimate

at the next control cycle under the current control signal, according to the reference healthy model. $\boldsymbol{e_k}$ represents the difference between the actual state and the reference state, and together with the actual control input $\boldsymbol{u}$, it forms the state input for the strategy. As for the actions in the reinforcement learning context, we output control signals for the four motors with each motor output a_i constrained within the range of $[0, 0.4]$ for $i = 1, 2, 3, 4$.

$$\begin{aligned} \hat{\boldsymbol{x}} &= f_h(\boldsymbol{x_{k-1}}, \boldsymbol{u_{base}} + \boldsymbol{u_{rl}}) \\ \boldsymbol{e_k} &= \boldsymbol{x} - \hat{\boldsymbol{x}} \\ \boldsymbol{s_k} &= [\boldsymbol{e_k}, \boldsymbol{u}], \boldsymbol{s_k} \in \mathbf{S}, S \subset \mathbb{R}^{16} \\ \boldsymbol{a_k} &= [a_1, a_2, a_3, a_4] \end{aligned} \tag{12}$$

The UAV's reward structure, as indicated in (13), is primarily divided into two parts. The term r_{err} is designed to encourage the UAV to maintain its original trajectory tracking quality as closely as possible even in the presence of failures. The purpose of r_f is to ensure that the UAV's policy actions are optimally adjusted to compensate for the impact of any failures.

$$r = r_{err} + r_f \tag{13}$$

where

$$\begin{aligned} r_{err} &= ||x_k - x_{des}||_\Lambda \\ r_f &= c||\boldsymbol{u_{rl}} - \boldsymbol{f}|| \end{aligned} \tag{14}$$

Algorithm 1 PPO Algorithm

Initialize policy parameters θ, old policy parameters θ_{old}, value function parameters ϕ
Initialize environment initial state s_0
for iteration $= 1, 2, \ldots, N$ **do**
  for $t = 0, 1, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi(a|s_t; \theta_{old})$
    Execute action a_t in the environment
    Observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1})
  end for
  Compute advantage estimates $\hat{A}_t$ using the current value function V_ϕ
  Update θ by optimizing the PPO-Clip objective:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

  Update ϕ by fitting V_ϕ to the observed returns
  Update $\theta_{old} \leftarrow \theta$
end for

where Λ, c represent the corresponding coefficients, $\boldsymbol{u_{rl}}$ represents the output of the reinforcement learning policy, while $\boldsymbol{f}$ denotes the actuator fault.

The network structure used in this study is illustrated in Fig. 3. Both of the Actor and Critic network involves the processing of state inputs through three layers of fully connected networks, which respectively output the action and the current value assessment. After training is complete, inference is conducted solely via the Actor network, which outputs the UAV's auxiliary fault-tolerant control strategy. This method streamlines the deployment process, allowing for efficient implementation of the learned strategies in real-world UAV operations.

B. Stability analysis

In this subsection, we provide a preliminary analysis of the stability of the proposed hybrid fault-tolerant control system for quadcopter.

Assumption 1: Consider a quadcopter system modeled by the equation:

$$x = f(x, u) + \omega \tag{15}$$

where u is the control input and ω represents system disturbances. We assume the system to be stable under a specified control strategy and there exists a Lyapunov function $V_c(x)$ such that:

- $V_c(0) = 0$,
- $V_c(x) > 0$ for all $x \neq 0$,
- $\dot{V}_c(x) < 0$ for all $x \neq 0$.

Additionally, the magnitude of the disturbance must satisfy the condition:

$$\|\omega\| + \delta < \|\omega_{\max}\| \tag{16}$$

where $\omega_{\max}$ is the critical threshold beyond which the controller's capability to ensure stability is compromised. Here, δ represents a margin that indicates the difference between the system's disturbance and the maximum allowable disturbance.

Under condition (16), We hope that x and ω meet the following robustness performance criterion:

$$\|x\| < \gamma\|\omega\| \tag{17}$$

This condition suggests that the system is asymptotically stable and meets the necessary robustness condition.

Assumption 2: For the reinforcement learning strategy, it is assumed that there always exists an optimal policy $\pi^*(a|s;\theta)$ such that:

$$\|\pi^*(a|s;\theta) + f\| = 0 \tag{18}$$

representing that the auxiliary fault-tolerant control strategy can effectively compensate for the system's faults. For the actual policy $\pi(a|s;\theta)$ trained via reinforcement learning, the optimality of the policy can be described by:

$$\|\pi - \pi^*\| \leq o \tag{19}$$

where o depends on the training tasks, reward structure, and hyperparameters. When these parameters are appropriately set, the actual policy π closely approximates π^*, that is, o is very small, indicating that the current fault-tolerant strategy can compensate for system faults.

Theorem 1: For a system satisfying Assumption 1, and the system is faulty. There exist an appropriate reinforcement learning strategy with reasonable hyperparameters and rewards,let the difference between the trained policy π satisfies:

$$\|\pi - \pi^*\| < \delta \tag{20}$$

Then, the system, when augmented with an auxiliary fault-tolerant policy, achieves asymptotic stability and meets the robust performance criteria specified in (21).

$$\|x\| < \gamma\|\hat{\omega}\| \tag{21}$$

where $\|\hat{\omega}\| \triangleq \|\pi - \pi^*\| + \|\omega\|$.

Proof. For a dynamic system with actuator fault:

$$\dot{x} = f(x,u) + f + \pi(a|s;\theta) + w \tag{22}$$

where the controller u is designed for the health system, external disturbances ω satisfies the robustness performance criterion $\|\omega\| < \|\omega_{\max}\|$. According to Assumption 1, we define the Lyapunov function as:

$$V(x) = V_c(x) \tag{23}$$

The system is asymptotically stable when there are no faults and disturbances. By incorporating Assumption 2, the system dynamics can be rewritten as:

$$\begin{aligned}\dot{x} =& f(x,u) + f + \pi(x,a;\theta) + \pi^*(a|s;\theta) \\ & - \pi^*(a|s;\theta) + \omega \\ =& f(x,u) + \omega + \omega_\pi\end{aligned} \tag{24}$$

where $\omega_\pi = \pi(a|s;\theta) - \pi^*(a|s;\theta)$ denotes the deviation of the actual policy π from the optimal policy π^*. With appropriately chosen hyperparameters and reward settings, it can be established that:

$$\|\omega_\pi\| \leq o < \delta \tag{25}$$

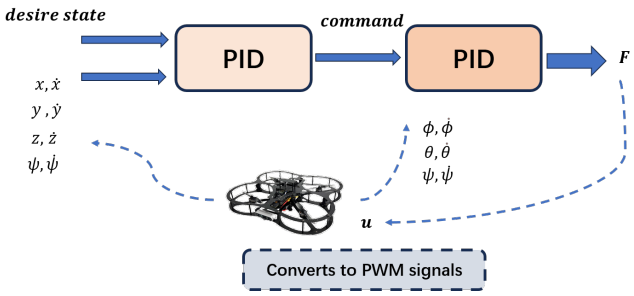

Fig. 4. The UAV employs a cascaded PID control scheme. The outer loop generates the total thrust and desired attitudes based on the expected states and the observed states of the UAV's x, y, z, ψ. The inner loop PID then produces the corresponding three-axis torques based on the desired attitudes and the UAV's actual rotational angles and angular velocities.

At this time, the system's disturbance has been updated from ω to $\hat{\omega}$, meeting the robustness performance criterion specified in (21). According to Assumptions 1 and 2, the system remains stable when when $\hat{\omega}$ is within the permissible disturbance range of the system.

C. Simulation Setup

We have utilized the MujoCo [27] physics engine to construct our simulation environment. In the simulation, the UAV's relevant parameters are set based on the actual QDrone specifications. The specific parameters of the UAV are detailed in Table I. Within this environment, the UAV's parameters are carefully configured based on the specifications of the actual QDrone, with these specific parameters detailed in Table I. We set the reward coefficient as $\Lambda = \text{diag}\{0.5, 0.5, 0.5, 0.1, 0.1, 0.1\}$ and $c = 1$. Our simulation incorporates a cascaded PID controller, identical to the one utilized in the actual hardware setup, as illustrated in Fig. 4. This controller operates at a control frequency of 500 Hz, providing fundamental stability and control of the UAV.

In addition to the basic control setup, our simulation features a fault-tolerant control strategy developed through reinforcement learning, which operates at an output frequency of 20Hz. This strategy is specifically designed to enhance fault tolerance. During simulations, the UAV is tasked with following a predefined desired state. To test the robustness of our control strategy, we introduce faults of varying magnitudes, ranging from 0 to 0.4, into any of the motors. To enhance the diversity of state sampling for the reinforcement learning algorithm, we periodically switch the affected motor within the same tracking control sequence, thereby providing a more comprehensive representation of the UAV's behavior under various failure scenarios.

To simulate real-world operational conditions more accurately, we introduce random Gaussian noise into both the motor outputs and the observed states of the UAV. This addition aims to test the robustness and fault tolerance of our strategy under conditions that more closely mimic real-world uncertainties and disturbances.We present the relevant hyperparameters for our algorithm in Table II. The lr_p and lr_v

TABLE I
PARAMETERS OS THE QUARDROTOR UAV

Parameter	Value
m	1.121 kg
d_θ	0.1758 m
d_ϕ	0.2136 m
J_{xx}	0.01 kgm^2
J_{yy}	0.0082 kgm^2
J_{zz}	0.0148 kgm^2

TABLE II
PARAMETERS OF PPO

Parameter	Value
Policy Network Learning Rate lr_p	0.0001
Value Network Learning Rate lr_v	0.001
Max Steps per Episode	600
Max Action Std σ_{max}	0.6
Min Action Std σ_{min}	0.1
PPO Epochs	80
Clip Range ϵ	0.2
Discount Factor	0.99

determine the update speeds for the policy and value networks, respectively. The maximum steps per episode define the limit on the number of actions the agent can take within a single episode. σ_{max} and σ_{min} control the exploration noise during training, with σ_{max} used at the start and σ_{min} towards the end. PPO epochs specify the number of times the policy is updated per batch of data. The clip range ϵ helps maintain stability by preventing large policy changes, and the discount factor balances the importance of immediate versus future rewards.

IV. SIMULATION AND REALWORLD EXPERIMENT

A. Simulation Result

The training of our fault-tolerance control strategy is depicted in Fig. 5, showing convergence after approximately 4000 episodes. We evaluate our trained fault-tolerant control strategy through hover tests and trajectory flight tests. As shown in Fig. 6, we first conducted hover tests where a bias fault of magnitude 0.3 was introduced to one of the motors. The controller without fault-tolerant compensation exhibited biases in x, y, and yaw angle ψ. In contrast, our fault-tolerant control strategy effectively compensated for the actuator faultsmake the drone converge to the desired state.

We conducted tests on our trained control strategy across various trajectory tracking tasks, specifically evaluating its performance in the presence of bias faults and actuator failures. Fig.7-Fig.10 illustrate the comparative fault tolerance effects under different trajectories.

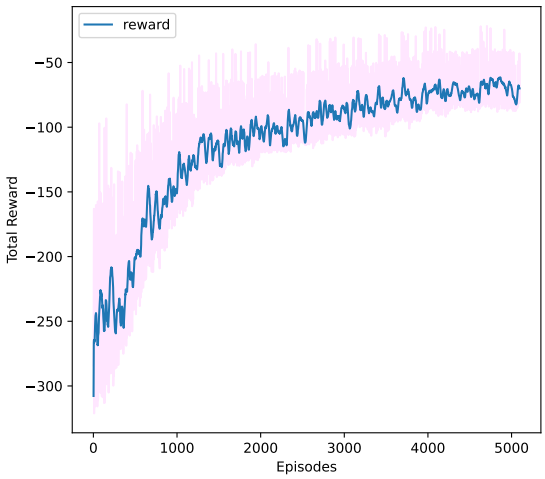

Fig. 5. Reward changes during training

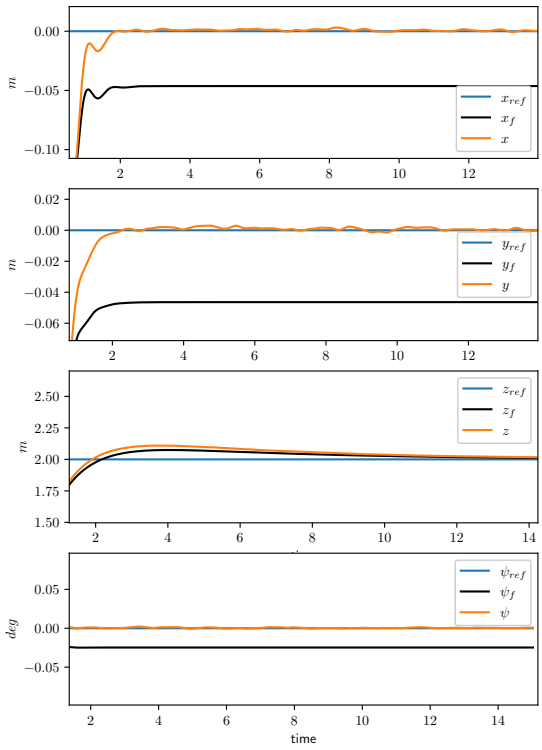

Fig. 6. Hover test in simulation

Fig. 7 and Fig. 8 depict the trajectory and state comparisons under bias fault conditions. The magnitude of the bias fault is 0.3. Fig. 9 and Fig. 10 show the trajectory and state comparisons when partial actuator failures occur. The magnitude of the actuator failure is also 0.3.

We use Root Mean Square Error (RMSE) and maximum tracking error to assess the quality of our proposed method for trajectory tracking. Tables III and IV present the numerical errors under two fault trajectory tracking scenarios. We quantitatively compare our method against both trajectory tracking in a healthy state and trajectory tracking in the presence of a

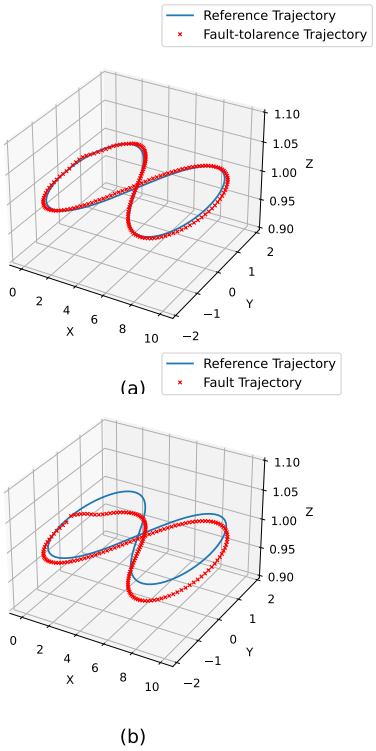

Fig. 7. Trajectory comparison between fault-tolerant and nonfault-tolerant states under actuator bias fault conditions. (a) Trajectory comparison with fault-tolerant controller. (b) Trajectory comparison without fault-tolerant controller

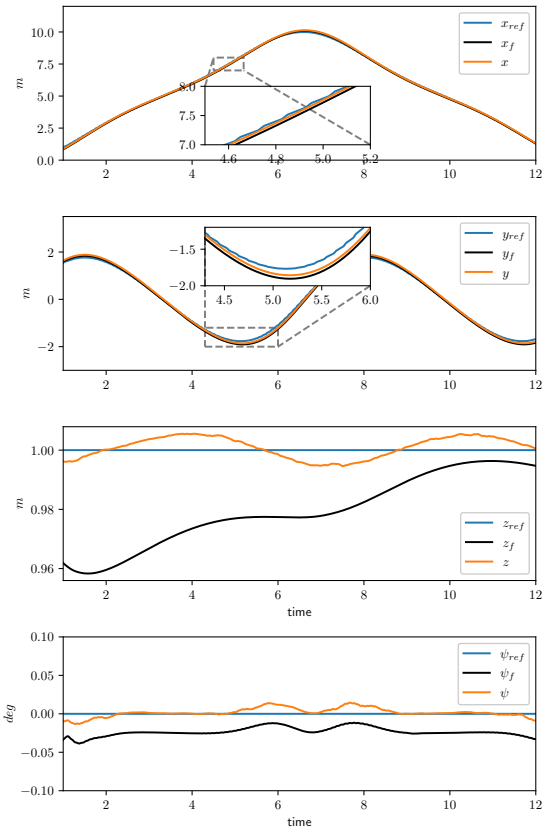

Fig. 8. State comparison between fault-tolerant and non-fault-tolerant conditions under actuator bias fault.

fault without compensation.

The results clearly demonstrate that in the presence of faults—whether they are bias faults or actuator failures—the original controller struggles to accurately track the desired system states. However, with the implementation of our auxiliary fault-tolerant control strategy, the system is able to more closely follow the desired states, showcasing the effectiveness of our approach in mitigating the impact of faults.

TABLE III
ERROR COMPARISON IN CONDITION 1

Metric	Healthy	Our Method	No Fault-Tolerance
RMSE(m)	0.058	0.059	0.072
Max Error(m)	0.157	0.158	0.187

TABLE IV
ERROR COMPARISON IN CONDITION 2

Metric	Healthy	Our Method	No Fault-Tolerance
RMSE(m)	0.068	0.069	0.081
Max Error(m)	0.122	0.123	0.168

B. Real-world Experiment

Similarly, we deployed and validated the trained strategy on a physical platform using Quanser's QDrone to test our proposed method. The framework of the algorithm on the actual system is illustrated in Fig. 11. We utilized a motion capture system along with the UAV's onboard IMU to obtain pose information. The controller was designed, and fault signals were introduced on a workstation. Approximately 15 seconds into the experiment, an actuator bias fault was introduced to motor 1. Around 20 seconds, we switched to our fault-tolerant control strategy.

The UAV's state after the output of the auxiliary control signal from our fault-tolerant control is displayed in Fig. 12, which documents the hovering test results under these conditions. After the fault was introduced, the UAV immediately deviated from its original state and displayed a significant error relative to the desired posture. This error was eliminated after the introduction of the auxiliary fault-tolerant strategy.

Additionally, the physical experiments highlighted the robustness of our proposed method. The real-time adaptation of the fault-tolerant control strategy ensured that the UAV could recover from unexpected actuator faults promptly. The effectiveness of the strategy in maintaining stability and accuracy, despite the fault, underscores its potential for practical applications in UAV operations. The results demonstrate that our fault-tolerant control mechanism can significantly mitigate the impact of actuator faults, ensuring the UAV's performance remains within acceptable limits.

V. CONCLUSION

This paper introduces an innovative auxiliary fault-tolerant control approach based on reinforcement learning to ensure the safe control of quadrotors in the event of actuator failures. The

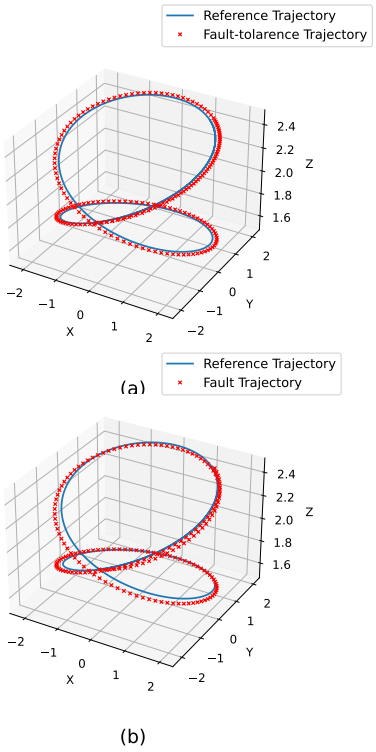

Fig. 9. Trajectory comparison between fault-tolerant and nonfaulttolerant conditions under actuator failure. (a) Trajectory comparison with fault-tolerant controller. (b) Trajectory comparison with nonfault-tolerant controller.

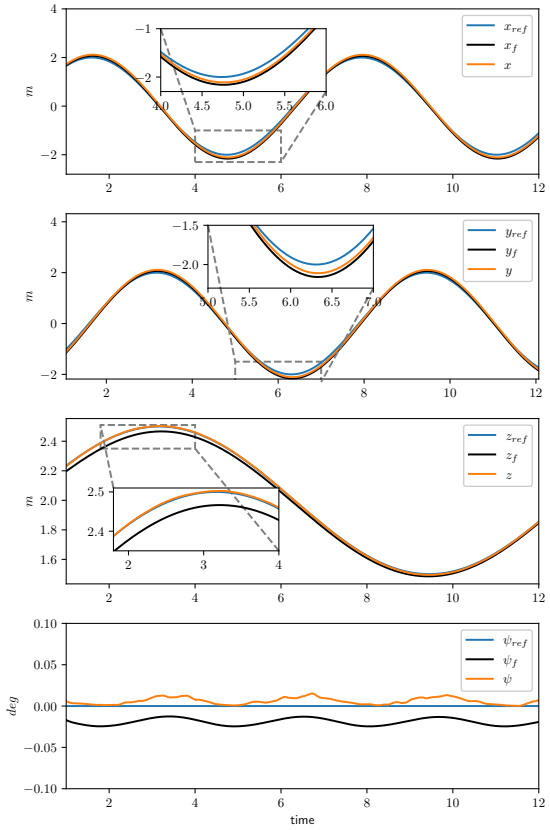

Fig. 10. State comparison between fault-tolerant and non-fault-tolerant conditions under actuator failure.

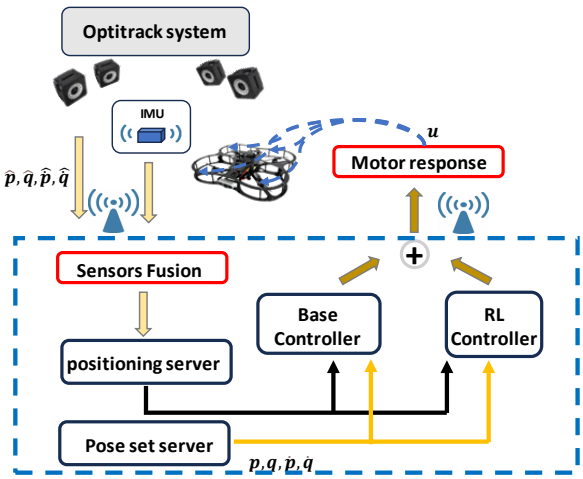

Fig. 11. Data $\hat{p}, \hat{q}, \hat{\dot{p}}, \hat{\dot{q}}$ are obtained through the fusion of information from the OptiTrack motion capture system and the UAV's onboard IMU. The variables $p, q, \dot{p}, \dot{q}$ represent the desired pose and velocity commands issued to the UAV. These commands are processed by the base controller and the RL controller, resulting in motor control signals that implement fault-tolerant control in the actual QDrone system.

approach employs a data-driven hybrid control strategy, which allows for end-to-end fault-tolerant control without the need for fault detection or modeling. The PPO algorithm was used to train the policy, utilizing the required quadrotor states and throttle information to output auxiliary fault-tolerant control signals.

During the simulation phase, the policy was trained and validated. The UAV executed various tasks for state sampling, and random noise was introduced to simulate real-world conditions and enhance the robustness of the strategy. Post-training, tests were conducted through trajectory tracking and hover tasks with induced faults, comparing the performance with and without the implementation of fault tolerance to validate the effectiveness of the approach.

Empirical experiments were also conducted using Quanser's QDrone as a test platform. During task execution, faults were introduced into one of the drone's motors. The reinforcement learning-based fault-tolerant strategy demonstrated its capability in handling faults, ensuring safe flight.

Future research will continue exploring the application of reinforcement learning in UAV safety control and planning to achieve fault-tolerant control and safe flight under more complex failure scenarios.

REFERENCES

[1] J. Su, D. Yi, B. Su, Z. Mi, C. Liu, X. Hu, X. Xu, L. Guo, and W.-H. Chen, "Aerial visual perception in smart farming: Field study of wheat yellow rust monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, DOI 10.1109/TII.2020.2979237, no. 3, pp. 2242–2249, 2021.

[2] W. Wang, C. Fang, and T. Liu, "Multiperiod unmanned aerial vehicles path planning with dynamic emergency priorities for geohazards monitoring," *IEEE Transactions on Industrial Informatics*, vol. 18, DOI 10.1109/TII.2022.3153031, no. 12, pp. 8851–8859, 2022.

[3] J. Fu, A. Núñez, and B. De Schutter, "Real-time uav routing strategy for monitoring and inspection for postdisaster restoration of distribution networks," *IEEE Transactions on Industrial Informatics*, vol. 18, DOI

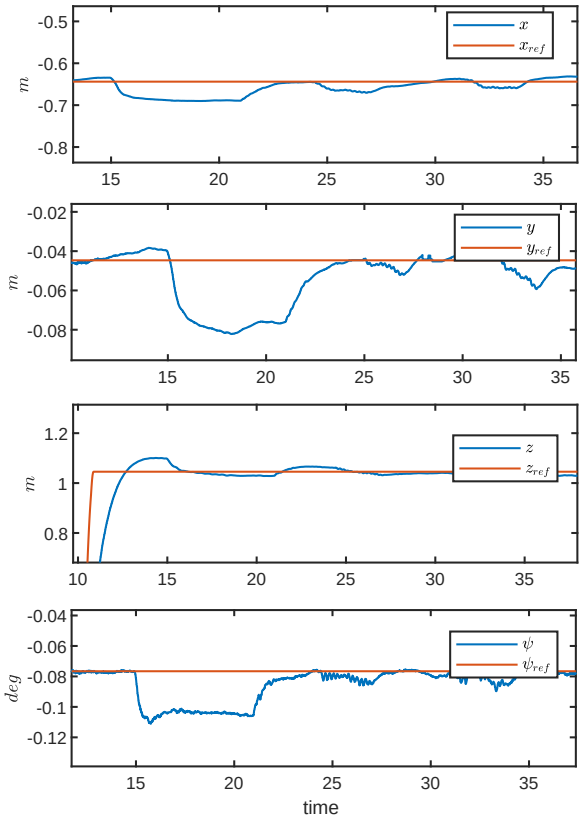

Fig. 12. Illustration of fault-tolerant effects in the physical hover experiment.

10.1109/TII.2021.3098506, no. 4, pp. 2582–2592, 2022.

[4] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[5] X. Liu, Z. Gao, and M. Z. Q. Chen, "Takagi–sugeno fuzzy model based fault estimation and signal compensation with application to wind turbines," *IEEE Transactions on Industrial Electronics*, vol. 64, DOI 10.1109/TIE.2017.2677327, no. 7, pp. 5678–5689, 2017.

[6] H. Li, H. Liu, H. Gao, and P. Shi, "Reliable fuzzy control for active suspension systems with actuator delay and fault," *IEEE Transactions on Fuzzy Systems*, vol. 20, DOI 10.1109/TFUZZ.2011.2174244, no. 2, pp. 342–357, 2012.

[7] S. Yin, B. Xiao, S. X. Ding, and D. Zhou, "A review on recent development of spacecraft attitude fault tolerant control system," *IEEE Transactions on Industrial Electronics*, vol. 63, DOI 10.1109/TIE.2016.2530789, no. 5, pp. 3311–3320, 2016.

[8] F. Zhang, L. Chen, Y. Dai, L. Kou, P. Ji, and Y. Liu, "Bearing fault diagnosis based on convolution neural network with logistic chaotic map," *Advanced Theory and Simulations*, p. 2301090, 2024.

[9] S. Yin, H. Luo, and S. X. Ding, "Real-time implementation of fault-tolerant control systems with performance optimization," *IEEE Transactions on Industrial Electronics*, vol. 61, DOI 10.1109/TIE.2013.2273477, no. 5, pp. 2402–2411, 2014.

[10] F. Nan, S. Sun, P. Foehn, and D. Scaramuzza, "Nonlinear mpc for quadrotor fault-tolerant control," *IEEE Robotics and Automation Letters*, vol. 7, DOI 10.1109/LRA.2022.3154033, no. 2, pp. 5047–5054, 2022.

[11] S. Sun, X. Wang, Q. Chu, and C. d. Visser, "Incremental nonlinear fault-tolerant control of a quadrotor with complete loss of two opposing rotors," *IEEE Transactions on Robotics*, vol. 37, DOI 10.1109/TRO.2020.3010626, no. 1, pp. 116–130, 2021.

[12] C. Ke, K.-Y. Cai, and Q. Quan, "Uniform passive fault-tolerant control of a quadcopter with one, two, or three rotor failure," *IEEE Transactions on Robotics*, vol. 39, DOI 10.1109/TRO.2023.3297048, no. 6, pp. 4297–4311, 2023.

[13] K. Guo, W. Zhang, Y. Zhu, J. Jia, X. Yu, and Y. Zhang, "Safety control for quadrotor uav against ground effect and blade damage," *IEEE Transactions on Industrial Electronics*, vol. 69, DOI 10.1109/TIE.2022.3140494, no. 12, pp. 13 373–13 383, 2022.

[14] X. Yu, X. Zhou, K. Guo, J. Jia, L. Guo, and Y. Zhang, "Safety flight control for a quadrotor uav using differential flatness and dual-loop observers," *IEEE Transactions on Industrial Electronics*, vol. 69, DOI 10.1109/TIE.2021.3135640, no. 12, pp. 13 326–13 336, 2022.

[15] R. C. Avram, X. Zhang, and J. Muse, "Quadrotor actuator fault diagnosis and accommodation using nonlinear adaptive estimators," *IEEE Transactions on Control Systems Technology*, vol. 25, DOI 10.1109/TCST.2016.2640941, no. 6, pp. 2219–2226, 2017.

[16] B. Wang, Y. Shen, and Y. Zhang, "Active fault-tolerant control for a quadrotor helicopter against actuator faults and model uncertainties," *Aerospace Science and Technology*, vol. 99, DOI https://doi.org/10.1016/j.ast.2020.105745, p. 105745, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963819308454

[17] 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, DOI https://doi.org/10.1016/j.ast.2020.105716, p. 105716, 2020.

[18] D. Xu, J. F. Whidborne, and A. Cooke, "Fault tolerant control of a quadrotor using c 1 adaptive control," *International Journal of Intelligent Unmanned Systems*, vol. 4, no. 1, pp. 43–66, 2016.

[19] Y. Guo, B. Jiang, and Y. Zhang, "A novel robust attitude control for quadrotor aircraft subject to actuator faults and wind gusts," *IEEE/CAA Journal of Automatica sinica*, vol. 5, no. 1, pp. 292–300, 2017.

[20] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, DOI 10.1109/LRA.2017.2720851, no. 4, pp. 2096–2103, 2017.

[21] F. Fei, Z. Tu, D. Xu, and X. Deng, "Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, DOI 10.1109/ICRA40945.2020.9196611, pp. 7358–7364, 2020.

[22] Y. Sohège, M. Quiñones-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)*, DOI 10.1109/ICRA48506.2021.9562097, pp. 10 719–10 725, 2021.

[23] H. Jiang, F. Xu, X. Wang, and S. Wang, "Active fault-tolerant control based on mpc and reinforcement learning for quadcopter with actuator faults," *IFAC-PapersOnLine*, vol. 56, DOI https://doi.org/10.1016/j.ifacol.2023.10.589, no. 2, pp. 11 853–11 860, 2023, 22nd IFAC World Congress.

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

[25] S. Wang, A. Polyakov, and G. Zheng, "On generalized homogenization of linear quadrotor controller," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6190–6195. IEEE, 2020.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Xiaoxu Liu rreceived the B.S. degree in information and computing sciences and the M.S. degree in operations research and cybernetics from Northeastern University, Shenyang, China, in 2012 and 2014, respectively, and the Ph.D. degree in electrical engineering from the University of Northumbria at Newcastle, Newcastle upon Tyne, U.K., in 2018.

She is taking an Associate Professor Position at the Sino-German College of Intelligent Manufacturing, Shenzhen Technology University, China. She is an Associate Editor of IEEE Transaction on Industrial Informatics. Her research interests include robust fault diagnosis, fault-tolerant control, non linear systems, stochastic systems, fuzzy modeling, distributed systems and data-driven methods.

Zike Yuan received his B.S. degree in Automation from the Department of Automation, Zhejiang University of Technology, Hangzhou, Zhejiang, China, in 2022. He is currently working towards his M.S. degree in Mechanical Engineering at Shenzhen University, Shenzhen, Guangdong, China.

His research interests focus on the application of reinforcement learning, motion planning and fault-tolerant control in robotics.

Zhiwei Gao (Fellow, IEEE) received the B.Eng. degree in industrial automation and the M.Eng. and Ph.D. degrees in systems engineering from Tianjin University, Tianjin, China, in 1987, 1993, and 1996, respectively.

His research interests include diagnosis and control, digital twins, artificial intelligence, machine learning, data-driven approaches, biometrics, wind energy systems, wave converter, electric vehicles, and offshore energy.

Dr. Gao was the recipient of the Alexandervon Humboldt Fellowship in 2004. For contributions to real-time diagnosis and control for wind turbines, he was elevated to IEEE Fellow in 2023. He is the Co-Editor-in-Chief of IEEE Transactions on Industrial Informatics, Senior Editor of IEEE Access, Area Editor of Journal of Ambient Intelligence and Humanized Computing, and Associate Editor of IEEE Transactions on Systems Man and Cybernetics: Systems. He was the Associate Editor of IEEE Transactions on Industrial Electronics, IEEE Transactions on Industrial Informatics, IEEE Transactions on Automatic Control, and IEEE Transactions on Control Systems Technology.

Wenwei Zhang received the B.S. and M.S. degrees in precision instruments and the Ph.D. degree in measurement technology and instrumentation from Tianjin University, Tianjin, China, in 1989, 1994, and 1997, respectively, and the MBA degree from the University of Glasgow, Glasgow, U.K., in 2000.

He is currently the Director of the Advanced MEMS Sensor Chip Engineering Technology Research Center at Guangdong Province Universities, a Distinguished Professor at Shenzhen Technology University, Chair of the Professor Committee, Vice-Chair of the Academic Committee at Shenzhen Technology University, and the Dean of the Sino-German College of Intelligent Manufacturing at the same university. Previously, he served as R&D Director at Honeywell UK, Tyco UK, and Micronas Germany, accumulating over 15 years of experience in the R&D and management of magnetic sensor SOC chips. His expertise lies in core technologies for automotive magnetic sensor chips, with extensive practical experience in device physics, device materials, and the design and optimization of mixed-signal integrated circuits.