Fault-tolerant Control for Unmanned Aerial Vehicle Using Deep Reinforcement Learning

1st Jiayi Zhou
School of Intelligence Science and Technology
University of Science and Technology Beijing
Beijing, 100083, China
zjyjoy0511@163.com

2nd Zizuo Zhang
School of Intelligence Science and Technology
University of Science and Technology Beijing
Beijing, 100083, China
zhangzizuo2021@163.com

3rd Yuanyuan Fei
School of Intelligence Science and Technology
University of Science and Technology Beijing
Beijing, 100083, China
yuanyuan17812@163.com

4th Yao Yu
School of Intelligence Science and Technology
University of Science and Technology Beijing
Beijing, 100083, China
yuyao@ustb.edu.cn

2024 39th Youth Academic Annual Conference of Chinese Association of Automation (YAC) | 979-8-3503-7922-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/YAC63405.2024.10598772

***Abstract*—This paper develops a fault-tolerant control strategy for quadrotor unmanned aerial vehicles using deep reinforcement learning. To Address the issue of actuator failures, which can degrade trajectory tracking and system stability, we propose a novel fault-tolerant control framework. This framework integrates neural networks to approximate the dynamics of a quadrotor through offline data, generating residuals containing fault information for reinforcement learning training. An improved Deep Deterministic Policy Gradient algorithm is employed to accelerate the learning process and improve fault-tolerant capability, enhanced by Prioritized Experience Replay to assign training priorities based on residual values. The approach significantly improves the robustness and adaptability of the controller despite actuator faults. Finally, simulation results demonstrate that the proposed method maintains tracking performance effectively even under actuator faults.**

Keywords—fault-tolerant control, reinforcement learning, UAV tracking control

I. Introduction

Unmanned aerial vehicles(UAVs) have become increasingly prevalent in various applications, ranging from military surveillance and combat [1] to civilian uses such as aerial photography, package delivery, and agricultural monitoring [2]. With the diversity and complexity increasing of the working environment faced by UAVs, it's important to guarantee the stability and safety during flight. Therefore, the control performance is increasingly demanding.

In practical applications, the occurrence of actuator failures, such as motor degradation or propeller damage, can lead to performance degradation tracking or even instability of UAVs. Actuator failures may lead to catastrophic outcomes, thus making fault-tolerant control (FTC) for UAVs a trending topic in research. Up to now, various fault-tolerant control algorithms have been investigated and applied to the actual systems. The methods for fault-tolerant control (FTC) in UAVs are primarily classified into two types: passive and active. Passive FTC does not rely on fault detection information but instead designs a fixed controller to exhibit robustness against predefined faults. A passive fault-tolerant control method based on Lyapunov analysis was proposed for nonlinear affine systems with actuator faults in [3]. Active FTC utilizes fault detection and isolation modules to identify and isolate faults within the system, then it adjusts control structures and parameters to compensates for the faults and maintain the performance of system[4]. Some researchers construct fault estimators to monitor and diagnose the health condition of the system. And common reconstruction methods in nonlinear systems include robust observers [5], sliding mode observers [6], etc. In practical scenarios, obtaining a priori knowledge of bounded faults can be challenging, which poses a difficulty in controller design. To address this issue, a fault-tolerant control scheme was introduced in [7], which is adaptive and does not rely on fault information or uncertain bounds. This scheme aims to compensate for both actuator faults and model uncertainties. Additionally, for fixed-wing UAVs with actuator faults, Reference [8] proposed a novel fault-tolerant control method based on the Nussbaum function. A control scheme focused on achieving optimal and near-optimal performance for nonlinear systems with time-varying actuator faults, time-varying disturbances, and identification errors in the neural network-based identifier was proposed[9].

Although numerous fault-tolerant control (FTC) methods can deliver satisfactory control performance, it is important to recognize that the majority of FTC techniques are formulated within the context of model-based methods. Nevertheless, model-based approaches necessitate prior knowledge of the system's dynamic model. The complex dynamics and aerodynamics of UAVs make it very challenging to establish an accurate mathematical model. Reinforcement learning(RL) methods are introduced to address the challenge of precise modeling and optimal control. Researchers have already begun to apply RL algorithms to the fault-tolerant control of systems. An adaptive fault-tolerant tracking control method based on RL for affine nonlinear continuous-time systems with unknown dynamics is designed to address the optimal fault-tolerant control problem, particularly focusing on process and actuator faults, including both repetitive and non-repetitive dynamics[10]. Reference[11] proposed a fault-tolerant predictive hierarchical control method integrated with deep reinforcement learning, which ensures vehicle stability while optimizing motor power consumption. In [12], an adaptive model-free fault-tolerant control scheme based on integral RL techniques was proposed for tracking control of highly flexible aircraft with actuator faults. Besides, active FTC usually relies on a Fault Detection and Diagnosis (FDD) module to detect, isolate, and estimate the current faults, and it requires the controller to be reconfigured after a fault is detected, which demands additional time and computational resources.

This study aims to design a fault-tolerant control method based on RL for a quadrotor system, which can take

corresponding compensatory measures when faults occur. And it does not require FDD module for real-time fault detection and online controller reconfiguration compared with most active FTC methods , so it can reduce time delays and computational load. The main contributions of this paper are as follows: First, we utilize Neural Networks (NNs) to approximate the quadrotor dynamics using offline data for generating fault residuals. Second, we add residuals into the experience replay buffer to help RL controller learns fault information from the residuals. And the method of Prioritized Experience Replay is used during the training process, which assigns different training priorities to residuals based on their absolute values, thereby improving the convergence speed of the algorithm.

II. Problem Formulation and Preliminaries

A. Dynamic Model of quadrotor

The quadrotor has symmetrically arranged four rotary wings as show in Fig. 1, each driven by an independent motor. This configuration enables the quadcopter to generate lift and control its motion in various directions. And we make the following assumptions for the quadrotor: (1) The body of the quadrotor is considered as a rigid body; (2) The parameters of the four rotors are identical, including mass, thrust, and drag; (3) The centroid, center of mass, and center of gravity of the quadrotor coincide.

Fig. 1. Quadrotor UAV.

The dynamics model of the quadrotor can be expressed as [13]:

$$\begin{aligned}
\ddot{p_x} &= [cos\,\varphi sin\,\theta cos\,\psi + sin\,\varphi sin\,\psi]\frac{\tau_4}{m} + d_1 \\
\ddot{p_y} &= [cos\,\varphi sin\,\theta sin\,\psi - sin\,\varphi cos\,\psi]\frac{\tau_4}{m} + d_2 \\
\ddot{p_z} &= cos\,\theta cos\,\varphi\frac{\tau_4}{m} - g + d_3 \\
\ddot{\varphi} &= \dot{\theta}\dot{\Psi}(\tfrac{I_y - I_z}{I_x}) - \tfrac{J_R}{I_x}\dot{\theta}\Omega_R + \tfrac{L}{I_x}\tau_1 + d_4 \\
\ddot{\theta} &= \dot{\varphi}\dot{\Psi}(\tfrac{I_z - I_x}{I_y}) + \tfrac{J_R}{I_y}\dot{\varphi}\Omega_R + \tfrac{L}{I_y}\tau_2 + d_5 \\
\ddot{\psi} &= \dot{\theta}\dot{\varphi}(\tfrac{I_x - I_y}{I_z}) + \tfrac{1}{I_z}\tau_3 + d_6
\end{aligned} \tag{1}$$

where p_x, p_y, p_z are the position and φ, θ, ψ are attitude of the quadrotor. The m is the mass and I_x, I_y, I_z are the moments of inertia. L is the length from the barycenter to the center of each rotor. The J_R and Ω_R donate the moments of inertia and angular velocity of the propeller blades. $d_1, d_2, d_3, d_4, d_5, d_6$ are bounded disturbances. The τ_1, τ_2, τ_3 are thrusts in direction of roll, pitch, yaw and τ_4 is the collective thrust.

Actuators of quadrotor experience malfunctions due to various factors, such as mechanical wear, electrical faults, or external damage. The common actuator faults include Offset Fault, Multiplicative Fault and Saturation Fault. We considered multiplicative fault in the context. When it occurs, the actual output of the actuator experiences some loss compared to the expected output. The output torque τ_i of the i-th actuator under multiplicative faults can be modeled as:

$$\tau_i = \delta\tau_{ei} \tag{2}$$

where τ_{ei} are expected output torque of the i-th actuator and $\delta \in (0,1)$ denotes the magnitude of loss rate of actuator torque due to multiplicative faults.

B. Reinforcement Learning

In RL, an agent modifies the environment by selecting actions and evaluates the quality of the chosen actions based on the reward signals received from the environment. By trying different actions and continuously adjusting its policy, the agent can gradually learn the optimal strategy to maximize the rewards.

Deep Deterministic Policy Gradient(DDPG) algorithm is an effective approach to solving reinforcement learning problems in continuous action spaces [14]. It combines the Actor-Critic framework, deep neural networks, experience replay buffer, and target networks to achieve fast convergence and good performance in high-dimensional continuous action spaces. In this paper, the quadrotor serves as the environment within the reinforcement learning framework, while the fault-tolerant compensatory controller acts as the agent.

The experience replay buffer in the DDPG algorithm is a buffer used to store the experience data. This data includes information such as states, actions, rewards, and the next state. The purpose of the experience replay buffer is to break the temporal correlation between data, improve the sample efficiency of the algorithm, and allow learning from historical experiences. During training, the algorithm learns from randomly sampled historical experiences in small batches, enhancing learning stability and effectively exploring the state-action space. The buffer continuously updates with new experiences, supporting DDPG in optimizing policies and value functions within continuous action spaces to discover optimal strategies in complex environments.

The purpose of this study is to design a new fault-tolerant compensatory controller based on RL to achieve effective tracking control of the drone under actuator fault conditions.

III. FTC Framework

In this section, we propose a novel fault-tolerant control framework based on NNs and RL. The overview of the proposed framework is shown in Fig. 2. The method depends on a baseline stabilizing controller for typical tracking control, with an additional RL controller to manage faults. We use neural networks to approximate the dynamics model of the quadrotor by training with data from fault-free operations, to predict the expected state system under normal conditions.Subsequently, residuals are calculated by computing the difference between the state predicted by the model and the actual state captured by the sensors. These residuals from each episode are then added into the experience replay buffer for further training.

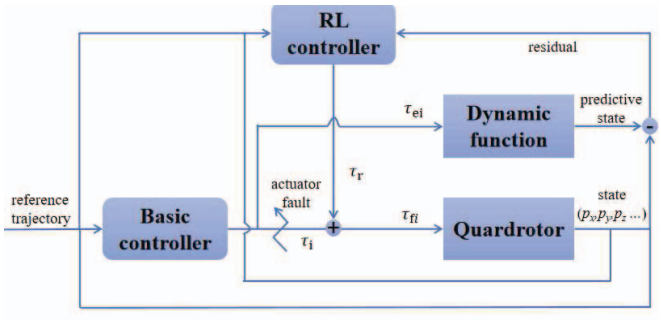

Fig. 2. FTC framework.

The basic controller τ_{ei} part refers to a pre-designed controller that provides a benchmark for subsequent improvements in control strategies or performance comparisons. The design of a basic controller is usually based on traditional control theories, such as PID control, sliding mode control, adaptive control, etc., which can achieve stable system control and performance requirements under specific operating conditions. The RL controller τ_r provides the compensatory control input to rapidly compensate for the loss of input due to actuator failures, ensuring tracking performance is maintained. And the final output torque τ_{fi} can be written as:

$$\tau_{fi} = \tau_i + \tau_r \tag{3}$$

A. Dynamic Function

Actuator faults will result in a certain error between the actual pose of the quadrotor and the expected pose, which is the residual between the expected pose and the actual pose. We conduct pre-training with a Multi-Layer Perceptron (MLP) network to predict the system dynamics of the quadrotor. Assuming that the UAV can obtain the accurate pose state s_k at the current moment k through sensors, the next state of the UAV can be predicted based on the current state s_k and the specified action torque a_k of actuator. The training objective of the MLP is to obtain an accurate predictive dynamic function $f(\omega)$, and ω is the parameter of the MLP. The relationship between the current state s_k and the predicted state $\hat{s}_{k+1}$ can be expressed as:

$$\hat{s}_{k+1} = s_k + f(\omega, s_k, a_k) \tag{4}$$

The purpose of MLP network is to approximate the dynamic function $f(\omega)$. The performance of the model is measured by calculating the Mean Squared Error(MSE) loss $e(\omega)$ between the predicted next state and the actual next state. The backpropagation algorithm is used to update the network parameters in order to minimize the loss, and the Adam optimizer is employed for parameter optimization. The loss can be expressed as:

$$e(\omega) = \frac{1}{n}\sum_{n\in D}\left\|\hat{f}(\omega) - (s_{k+1} - s_k)\right\|^2 \tag{5}$$

where n is the number of samples randomly sampled from the training dataset D. And we obtain D by collecting trajectory data of the quadrotor operation without actuator faults.

B. Design of State Space and Action Space

In this paper, we use an improved DDPG method as our RL algorithm. The learning process is often viewed as the solution process of a Markov Decision Process(MDP). It consists of a set of states, actions that can be executed in each state, state transition probabilities, and a reward function.

The training process in our study is considered as a standard MDP with state space $S \subseteq \mathrm{R}^{12}$ and action space $A \in \mathrm{R}^4$. The state $s \in S$ is designed as:

$$s = \left[P_x, P_y, P_z, \dot{p}_x, \dot{p}_y, \dot{p}_z, \varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}\right]^T \tag{6}$$

where $\dot{p}_x, \dot{p}_y, \dot{p}_z$ are the velocities in the three coordinate axis directions and $\dot{\varphi}, \dot{\theta}, \dot{\psi}$ are the angular velocities of attitude. Action $a \in A$ follows:

$$a = \tau_{fi} \tag{7}$$

The reward function is designed as:

$$\mathrm{r} = -|P - P_r| - |\psi - \psi_r| - \alpha|\theta + \varphi| - A_c \tag{8}$$

where $P \in \mathrm{R}^3$ is current position of quadrotor and $P_r \in \mathrm{R}^3$ is the reference position. ψ_r is expected yaw angle and A_c represents the cost of the input action a, which is used to evaluate the quality of a specific action.

C. RL training

During training process, we randomly introduce actuator faults within the predetermined range of fault probability, without needing to know the exact loss rate for each data, and the location of the fault randomly occurs among the four rotors. This study does not consider the extreme case of complete actuator failure. RL controller maintains functionality during both operational and fault scenarios. This type of training configuration enables the agent to adjust to changing environments and acquire the skills necessary for recovery from unexpected failures.

When dealing with faults in quadrotor systems, we incorporate fault residuals into the experience replay buffer. In this way, the reinforcement learning controller can draw data from the experience replay pool and learn the system's behavioral patterns during fault occurrences. By analyzing these residuals, the controller can identify specific patterns and characteristics that emerge when faults occur, enabling it to respond swiftly to similar situations in the future. To further enhance the learning efficiency, we employ the method of Prioritized Experience Replay. This approach not only stores ordinary experience data but also assigns a priority level to each piece of experience, which is typically related to the magnitude or significance of the experience. At the beginning of training, residuals with smaller absolute values are given higher priority to ensure network can converge and improve the speed of convergence. Once the model becomes stable, residuals with larger absolute values, which indicate significant degradation in system performance, are assigned higher priority. This means that experiences carrying critical fault information are more likely to be selected by the controller for training, enabling the controller to learn how to handle faults that cause significant performance loss more rapidly.

Through this allocation of priority, our reinforcement learning controller focuses more on faults that have the greatest impact on system performance, thereby accelerating the learning process and enhancing the robustness and adaptability of the controller in the face of actual faults. This method not only increases the convergence speed of the algorithm but also strengthens the reliability and effectiveness of the controller in practical operations, providing strong

support for the stable operation of quadrotors. The detailed implementation of the designed DDPG algorithm is shown in Algorithm 1.

Algorithm 1 Improved DDPG Algorithm

Input: current state *s*, reference trajectory
Output: actuator torque *a*

Initialize critic network, target networks and actor with weights
Initialize replay buffer *R* and exploration noise process *N*
for episode = 1 to *M* **do**
    Initialize a random process
  Receive initial observation state *s_1*
  for t = 1 to *T* **do**
    Select action *a_t* according to the current policy
    Execute action *a_t* and observe reward *r_t* and new state *s_(t+1)*
    Compute the residual *re_t* between the observed and the predicted state
    Store experience tuple (*s_t, a_t, r_t, s_(t+1),re_t*) in *R* based on priority
    Sample a minibatch of *N* transitions from *R* by Prioritized Experience Replay
    Calculate target value *y*
    Update critic
    Update the actor policy
      Update the target networks
  end for
end for
return *a*

IV. STIMULATION RESULTS

To further verify the reliability and effectiveness of the proposed solution, simulations of tracking control were conducted on the Python platform. The control objective is to make the quadrotor track a desired sinusoidal trajectory P_{ref}= [sin(πt/500), sin(πt/500), 1] and a reference yaw angle of 0°, ensuring that the tracking control process maintains a small tracking error. We randomly introduce actuator multiplicative faults at a probability of 0.2 and set $\delta \in [0.3,0.8]$ during RL training.

In the first experiment, we use a PID controller as the basic controller. The simulation result is shown in Fig. 3 when no actuator faults occur. During the tracking control process, actuator multiplicative faults ranging from 0.3 to 0.5 are randomly introduced with a probability of 0.1. The results of the simulation experiment are shown in Fig. 4 and Fig. 5, which respectively depict the tracking trajectories of

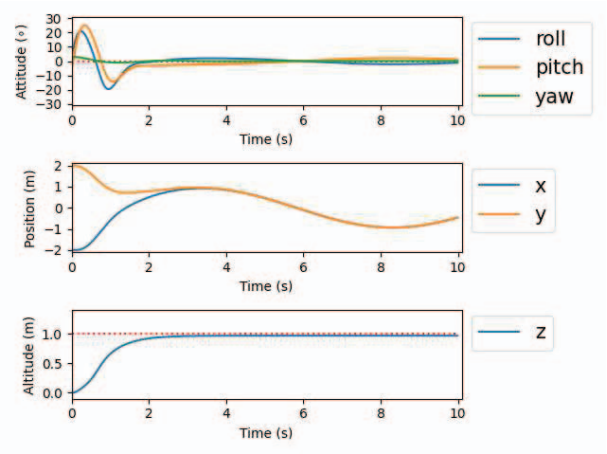

Fig. 3. Tracking trajectory using the PID with no actuator faults.

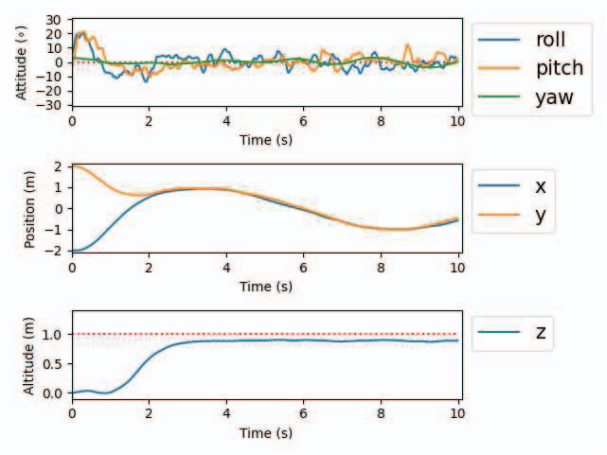

Fig. 4. Tracking trajectory using the PID with actuator faults.

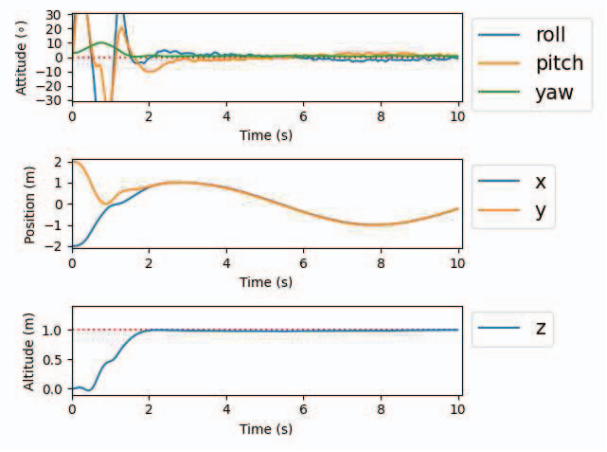

Fig. 5. Tracking trajectory using the proposed FTC with PID.

the quadrotor using only the PID controller and the fault-tolerant control method proposed in this paper. We conducted multiple simulation experiments and calculated the average error in the z-axis direction and the average error of the yaw angle during the process of trajectory tracking. We find that when actuator faults occur, using only the PID controller fails to track the reference signal with a z-axis average error of 0.1562 meters and a yaw angle average error of 0.1377 degrees, whereas our proposed FTC method can still follow the reference signal accurately with a z-axis average error of

0.1146 meters and a yaw angle average error of 0.1270 degrees.

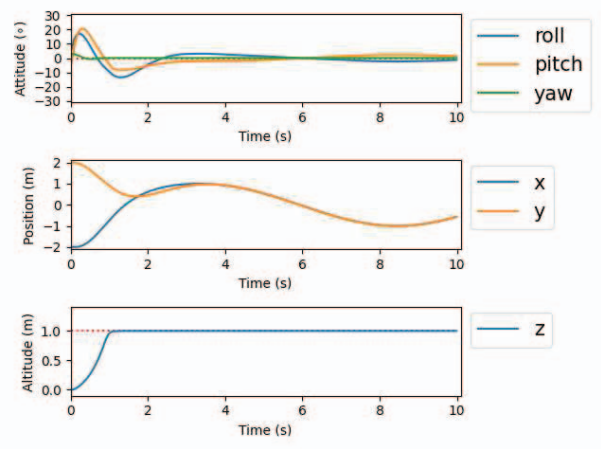

Fig. 6. Tracking trajectory using the SMC with no actuator faults.

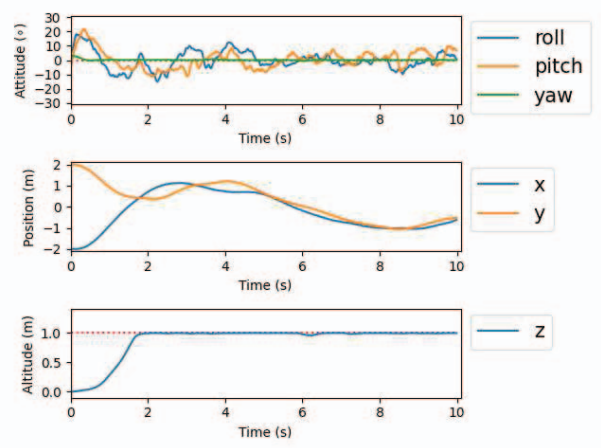

Fig. 7. Tracking trajectory using the SMC with actuator faults.

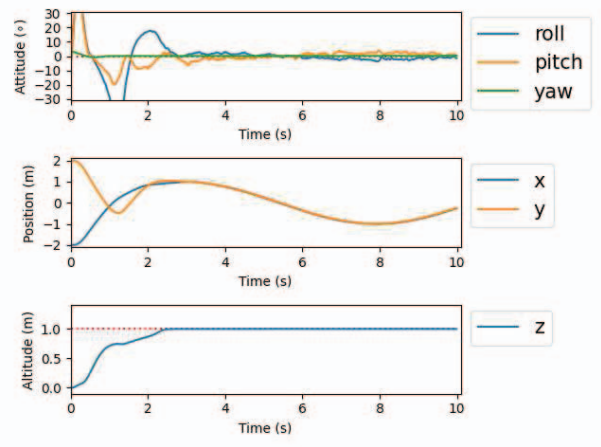

Fig. 8. Tracking trajectory using the proposed FTC with SMC.

In the second experiment, we use a sliding mode controller(SMC) as the basic controller. Tracking trajectory of quadrotor under normal conditions is shown in Fig. 6. The simulation results with actuator faults are shown in Fig. 7 and Fig. 8. Similar to the results of the first experiment, when using only the sliding mode controller, it is impossible to achieve reference trajectory tracking with a z-axis average error of 0.1494 meters and a yaw angle average error of 0.0682 degrees when actuator faults happen. Our proposed method can effectively compensate for the loss due to actuator faults with a z-axis average error of 0.0963 meters and a yaw angle average error of 0.0671 degrees. However, compared to situations without actuator faults, it still requires a certain amount of time for controller to improve control precision and accurately track the reference signal. To shorten the time to track the reference trajectory with actuator faults, we can adjust and optimize the reward function in the reinforcement learning algorithm to place more emphasis on response speed, which ensures that quickly recovering trajectory tracking receives higher rewards. Alternatively, utilizing incremental learning methods may allow the system to react quickly to new failures. The effectiveness of these solutions needs to be verified through further experiments.

V. CONCLUSIONS

We proposes a RL-based FTC framework. The dynamics model approximated is used to predict the expected pose of system, and by comparing it with the actual pose, fault residual data is obtained. The residuals are incorporated into the experience replay buffer of the improved DDPG algorithm as training data. During the reinforcement learning training process, the Prioritized Experience Replay algorithm is used, assigning different priorities to different residual data. This ensures rapid convergence of the system model while enabling the learning of more fault information, thereby enhancing the tracking performance of the quadrotor under actuator fault conditions. The RL algorithm primarily learns how to rapidly adjust control strategies to compensate for failures by undergoing a continuous training process. When actuator faults occurs, the control strategy designed mainly handles the failure through a RL compensatory controller. After inputting the reference trajectory and actual state to the RL controller, it outputs a control torque, which together with the output of the basic controller, acts on the quadrotor system to complete the tracking control task. We compared our proposed method with the other methods in simulation experiment.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (grant numbers 61931020 and 62033010).

REFERENCES

[1] Z. Xiaoning, "Analysis of military application of UAV swarm technology," 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China, 2020, pp. 1200-1204.

[2] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," Computer Networks, vol. 172, 2020.

[3] M. Benosman and K. . -Y. Lum, "Passive Actuators' Fault-Tolerant Control for Affine Nonlinear Systems," in IEEE Transactions on Control Systems Technology, vol. 18, no. 1, pp. 152-163, Jan. 2010.

[4] D. H. Zhou and P. M. Frank, "Fault diagnostics and fault tolerant control," in IEEE Transactions on Aerospace and Electronic Systems, vol. 34, no. 2, pp. 420-427, April 1998.

[5] M. -Y. Lee and B. -S. Chen, "Robust H∞ Fuzzy Observer-Based Fault-Tolerant Tracking Control for Nonlinear Stochastic System: A Sum of Square Approach," in IEEE Access, vol. 10, pp. 127667-127684, 2022.

[6] E. Shahzad et al., "Sensor Fault-Tolerant Control of Microgrid Using Robust Sliding-Mode Observer," Sensors, vol. 22, no. 7, 2022.

[7] B. Wang, D. Zhu, L. Han, H. Gao, Z. Gao, and Y. Zhang, "Adaptive Fault-Tolerant Control of a Hybrid Canard Rotor/Wing UAV Under Transition Flight Subject to Actuator Faults and Model Uncertainties," IEEE Transactions on Aerospace and Electronic Systems, vol. 59, no. 4, pp. 4559-4574, 2023.

[8] Z. Yu et al., "Nussbaum-based finite-time fractional-order backstepping fault-tolerant flight control of fixed-wing UAV against input saturation with hardware-in-the-loop validation," Mechanical Systems and Signal Processing, vol. 153, 2021.

[9] Y. Du, B. Jiang, and Y. Ma, "Adaptive optimal sliding-mode fault-tolerant control for nonlinear systems with disturbances and estimation errors," Complex & Intelligent Systems, vol. 10, no. 1, pp. 1087-1101, 2023.

[10] S. Roshanravan and S. Shamaghdari, "Adaptive Fault-Tolerant Tracking Control for Affine Nonlinear Systems With Unknown Dynamics via Reinforcement Learning," in IEEE Transactions on Automation Science and Engineering, vol. 21, no. 1, pp. 569-580, Jan. 2024.

[11] H. Deng, Y. Zhao, A. -T. Nguyen and C. Huang, "Fault-Tolerant Predictive Control With Deep-Reinforcement-Learning-Based Torque Distribution for Four In-Wheel Motor Drive Electric Vehicles," in IEEE/ASME Transactions on Mechatronics, vol. 28, no. 2, pp. 668-680, April 2023.

[12] J. Ma and C. Peng, "Adaptive model-free fault-tolerant control based on integral reinforcement learning for a highly flexible aircraft with actuator faults," Aerospace Science and Technology, vol. 119, 2021.

[13] Z. T. Dydek, A. M. Annaswamy and E. Lavretsky, "Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluations," in IEEE Transactions on Control Systems Technology, vol. 21, no. 4, pp. 1400-1406, July 2013.

[14] H. Tan, "Reinforcement Learning with Deep Deterministic Policy Gradient," 2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA), Xi'an, China, 2021, pp. 82-85.