RflyMAD: A Dataset for Multicopter Fault Detection and Health Assessment

The International Journal of Robotics Research
XX(X):1–10

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Xiangli Le[1], Bo Jin[1], Gen Cui[1], Xunhua Dai[2] and Quan Quan[1]

Abstract

This paper presents an open-source dataset, RflyMAD, a Multicopter Abnormal Dataset developed by the Reliable Flight Control (Rfly) Group aiming to promote the development of research fields such as Fault Detection and Isolation (FDI) or Health Assessment (HA). The full 114 GB dataset includes 11 types of faults under 6 flight statuses which are adapted from the ADS-33 file to cover more cases where the multicopters have different levels of mobility when faults occur. In the total of 5629 flight cases, the fault time is up to 3283 minutes, and there are 2566 cases for software-in-the-loop (SIL) simulation, 2566 cases for hardware-in-the-loop (HIL) simulation, and 497 cases for real flight. As it contains simulation data based on RflySim and real flight data, it is possible to improve the quantity while increasing the quality of the data. In each case, there are ULog, Telemetry log, Flight information, and processed files for researchers to use and review. The RflyMAD dataset could be used as a benchmark for fault diagnosis methods and the support relationship between simulation data and real flight is verified by transfer learning methods. In the future, more methods will be presented as a baseline and RflyMAD will be updated with more data and types. In addition, the dataset and associated toolkit are available at https://rfly-openha.github.io/documents/4_resources/dataset.html.

Keywords

Datasets, Fault detection and isolation, Multicopter, OpenHA, RflySim, UAVs

1 Introduction

With the rapid development of the technology and related industries of Unmanned Aerial Vehicles (UAVs), multicopters have become increasingly popular and are more commonly used in daily life than ever before (Quan 2017). The increased use of multicopters has led to growing concern for their health and flight safety. Researchers have long been focusing on the problem of Fault Detection and Isolation (FDI) in UAVs, and many achievements have been made. Among these results, analytical redundancy is widely used for FDI problems and can be divided into Model-Based and Knowledge-Based methods (Fourlas and Karras 2021). Both methods, particularly the Knowledge-Based methods, require simulation or real flight datasets of UAVs. Compared to research fields such as computer vision (Deng et al. 2009), the types of datasets designed for UAVs are fewer. Moreover, there is a notable scarcity of datasets that take the failures of UAVs into account (Keipour et al. 2021).

Common datasets related to UAVs are found to focus on SLAM, localization, and image recognition, such as UAVid by Lyu et al. (2020). Antonini et al. (2020) provide the Blackbird UAV dataset, which offers high-rate IMU and image data for approximately ten hours, suitable for visual-inertial navigation and SLAM. Nguyen et al. (2022) present the NTU VIRAL dataset, which is used for simultaneous localization and mapping. NASA's Open Data Portal[1] records a series of datasets related to FDI and HA problems in aircraft, such as C-MAPSS Aircraft Engine Simulator Data and ADAPT Dataset. Although the categories of related datasets are sufficient, most of them contain merely simulation data and focus on component-level faults. Due to the high cost of hardware in real experiments, most datasets reflecting both normal and abnormal statuses of UAVs often contain simulation data while datasets recording system-level faults are scarce. ALFA is a real UAV dataset at system-level for fixed wings with abnormal statuses to overcome this problem, as reported by Keipour et al. (2021).

Although ALFA provides a real dataset with the fault status of UAVs, the amount of fault cases is still lower than expected and the dataset is only suitable for fixed-wing aircraft. The Rfly Multicopter Abnormal Data (RflyMAD) is designed to resolve the problem of insufficient data in research fields such as FDI and HA. Considering the balance between the amount of data and the real flight data, the RflySim platform is used to generate simulation data. RflySim is a model-based development (MBD) platform that enables rapid development (Quan et al. 2020; Dai et al. 2021). A nonlinear multicopter model with high accuracy and fault injection modules is built in this platform. Together with the PX4 autopilot, concepts of SIL and HIL simulation are applied to collect the SIL and HIL simulation data. Next, with the modified source code of PX4 firmware, the same fault injection modules are used to collect the real flight data. RflyMAD contains different kinds of faults in its actuators

[1]Beihang University, P.R. China
[2]Central South University, P.R. China

Corresponding author:
Quan Quan, Beihang University Reliable Flight Control Group, No.37 XueYuan Road, HaiDian District, Beijing, 100191, P.R. China.
Email: qq_buaa@buaa.edu.cn

(motors and propellers), sensors (accelerometer, gyroscope, magnetometer, barometer, and GPS), and environmental effects. Each fault type has a variety of fault parameters that indicate the degree of failure. For each case, telemetry logs from QGroundControl and ULog from the autopilot are both provided, along with a CSV file containing the specific flight information. Besides, the ground truth data with a high frequency from the RflySim platform in simulation and ROS bag files in real flight are both recorded. Lastly, basic usage methods and toolkits are provided to make RflyMAD more user-friendly. The main contributions of this paper are as follows.

- Compared with existing datasets related to UAVs, the significant feature of RflyMAD is that it has real flight abnormal data from different types of multicopters. **To the best of our knowledge, this is the first fault dataset for multicopters with multiple flight statuses and fault types, which contains both real flight and simulation data.** High-quality simulation data has a similar form to real flight data and can be used as a supplement to real flight data.
- The format of the RflyMAD dataset and the collection methods of simulation and real flight data are introduced in detail in this paper and on our website. Under the format introduced in this paper, we developed data processing tools, which can select and extract data topics in the RflyMAD. According to the data collection methods, the dataset can be extended in the future by using the provided RflySim platform in simulation or modifying PX4 firmware to add a fault injection module in real flight.
- A transfer learning method is used in this paper to verify that the SIL and HIL simulation data have an excellent support relationship with the real flight data, demonstrating that the simulation data has high quality and could replace part of the high-cost real flight data when training the diagnosis model, which can improve the utilization of real flight data. A better conclusion could be obtained from the support relationship of SIL with HIL, proving that the two have similar characteristics. The support relationships of different types of multicopters in real flight data are also verified.

The remainder of the paper is organized as follows: *Section 2* presents the properties of the RflyMAD dataset. *Section 3* introduces the hardware and software used for data collection and how to collect the data. *Section 4* verifies the support relationship between different types of data. *Section 5* outlines future work and identifies current dataset issues.

2 Properties of RflyMAD

To enhance the quality and quantity of data, RflyMAD consists of simulation and real flight data. In this section, the properties of the dataset will be introduced, including data formats, dataset hierarchy, and fault types.

2.1 Data formats

2.1.1 Data types in one flight Each flight within the dataset comprises four types of raw data along with corresponding processed files, which could be described as follows.

- **Flight Information**. It contains the flight command (e.g., takeoff, landing, and moving to a target position), fault types, and fault parameters. The data is provided in comma-separated values (CSV) format named *TestInfo.CSV* within the dataset.
- **ULog**. It is used to log uORB topics as messages, including device inputs (e.g., sensors, RC inputs), internal state (e.g., attitude, EKF states), and string messages. The file could be converted into CSV conveniently. This data is provided in original format and processed CSV format.
- **Telemetry Log**. TLog is recorded by the ground station, and the main content is the information communicated between a multicopter and its corresponding QGroundControl[2] (QGC). Thus frequency of transmission is decided by the communication quality in real flight or the performance of the simulation computer. This data is provided in two formats: tlog(original) and CSV (processed).
- **Ground Truth Data**. It is generated by the RflySim platform during the simulation and recorded at approximately 120Hz. It contains the kinematics information, fault states, and motor speeds. This data is abbreviated as "GTData" in the subsequent text and is provided in CSV format.
- **BAG**. It is generated by the ROS system during each real flight. It contains the position, attitude, and control commands of a multicopter. This data is provided in two forms: the raw, unprocessed data, and the processed BAG data converted to CSV files.

It is worth noting that the GTData only exists in simulation data, while the BAG files only exist in real flight data. Therefore, each flight is associated with four types of data. However, due to the widespread use of ROS in the robotics community and the upgrade of the simulation tools, which will be introduced in *Section 2.4*, the BAG or ROS bag data will be added to the simulation data in the subsequent data updates so that the users can have a wider range of choices.

2.1.2 Choose your most suitable data type The RflyMAD dataset provides different types of data, as introduced in *Section 2.1.1*, users ought to select the proper data type according to task requirements and their own usage habits. Here a brief introduction of usage scenarios for different data types is given as a simple guidance.

Flight Information data can be used as a preview of a flight case, which includes control commands in flight, fault types, fault duration and etc. As mentioned above, the quality of TLog data depends on the quality of the communication, so this type of data can be used to diagnose remote control related faults. GTData only exists in simulation sub-dataset, for it comes from the simulation platform. GTData contains the true value from the simulation physical model, even the real speed of motors, which is usually difficult to obtain in the real flight data. Due to the simplicity of the CSV format, users who are not familiar with ULog and BAG can also use this data easily.

ULog data is the most common data format when using open source autopilot, such as PX4, and can be be supplemented by flight enthusiasts in the future. Fault diagnosis algorithms developed using this format can be easily deployed on UAVs without an onboard computer. This data format can also be easily preprocessed by using official

or our developed preprocessing tools. In addition, the timing information record of ULog data is more standardized.

BAG, also called ROS bag data, is currently widely used in the robotics community for debug or data collection. ROS bag data can realize functions such as data playback and format conversion by simple commands. Real-time online or offline fault diagnosis algorithms can be developed by using ROS bag data, and can be easily deployed to onboard computers of UAVs. ROS uses the Unix timing strategy, which also has advantages in time-dependent data processing in practice.

All of the aforementioned data can be transformed, stored, and preprocessed automatically by the toolkits we developed. More importantly, the toolkits we developed can preprocess different data types, such as BAG and ULog, into a single processed file, making the development of FDI algorithm more convenient.

2.2 Scale

RflyMAD is a large-scale dataset in the research field of UAV abnormal data, for it contains a great amount of simulation data which could be divided into SIL and HIL simulation data in detail. Together with the real flight data, RflyMAD comprises three sub-datasets, whereas the fault types and flight statuses in each sub-dataset are still similar.

Table 1. Fault types and flight numbers in RflyMAD.

Type of Faults	Type of Sub-dataset		
	SIL Sim	***HIL Sim***	***Real Flight***
Motor (1-4)	921	921	231
Propeller (1-4)	435	435	×
Low Voltage	20	20	×
Wind	150	150	×
Load Lose	150	150	×
Sensors' Noise	50	50	82
Accelerometer	128	128	20
Gyroscope	128	128	20
Magnetometer	128	128	20
Barometer	128	128	20
GPS	128	128	20
No Fault	200	200	84
Total	2566	2566	497

Note:× represents this item does not exist in sub-dataset. Motor(1-4) represents the number of failure motors is in range of 1 to 4.

TABLE 1 displays the number of flights in each sub-dataset associated with different fault types. There are 11 fault types in simulation data and 7 in real flight data to cover the common faults that may occur in a multicopter. Notably, data about different combinations of failure units of motor or propeller are collected. Most importantly, faults from the power system, sensors, and multicopter's frame structure to the external environment are considered. We firmly believe that the inclusion of more fault types will result in a more extensive dataset.

In TABLE 1, the total number of flight cases amounts to 5629, so the total size of RflyMAD is about 114.6 GB. In each flight of a sub-dataset, the fault parameters which represent the degree of failure are different even if they have the same fault types. Due to the high cost of a real flight with injected faults, the real flight dataset contains fewer instances with high failure degrees compared to the simulation dataset.

Table 2. Flight statuses in RflyMAD.

Flight Status	Type of Sub-dataset		
	SIL Sim	***HIL Sim***	***Real Flight***
Hover	✓	✓	✓
Waypoints	✓	✓	✓
Velocity Control	✓	✓	✓
Circling	✓	✓	✓
Acceleration	✓	✓	✓
Deceleration	✓	✓	×

Note: ✓represents this item existing in sub-dataset and × represents not.

Considering when the fault occurs, the multicopter may be at different flight statuses and thus have different performance. To fully study the mobility of the multicopter, 6 flight statuses are designed during the time when the fault occurs. TABLE 2 shows the flight statuses in each sub-dataset. The selected flight status refers to the Mission-Task-Elements in the ADS-33 file, which is a specification document for U.S. military rotorcraft flight quality (Baskett and Daniel 2000). As this document is mainly used for helicopters, some basic tasks are selected and modified to be suitable for multicopters. In each flight status, the RflyMAD dataset contains all fault types with different fault parameters in order to satisfy various conditions for multicopters.

2.3 Hierarchy

The hierarchy of the RflyMAD dataset is illustrated in Figure 1, featuring five layers delineating its internal structure. The outermost layer comprises three sub-datasets. The second layer contains flight status information, with 6 types for simulation data and 5 types for real flight data. The third layer represents fault type, while the fourth layer consists of a series of cases with the same flight status and fault, but with varying fault parameters. The fifth layer consists of four basic files for each flight. The reason why RflyMAD is designed as such a complicated structure is to make data of each flight have a distinct classification. This organizational approach facilitates the work of researchers in designing model-based or data-driven algorithms. Additionally, the fault type, fault parameters and fault injection time could be found in the Flight information, ULog, and BAG data in each flight.

2.4 Convenient platform and extensibility

2.4.1 RflySim in data collection This paper utilizes the RflySim platform to design and realize simulation experiments, and simultaneously carry out real flight experiments. A detailed introduction to the RflySim platform and how to use it to collect fault data is given in this section.

RflySim is a model-based platform for unmanned system control and safety testing. It uses MATLAB/Simulink as the core component of the platform to model UAVs and design controllers in autopilot, and uses Python to implement top-level visual or swarm algorithms, trajectory planning, and safety assessment functions. Development based on RflySim usually includes the following five phases: modeling phase, controller design phase, SIL

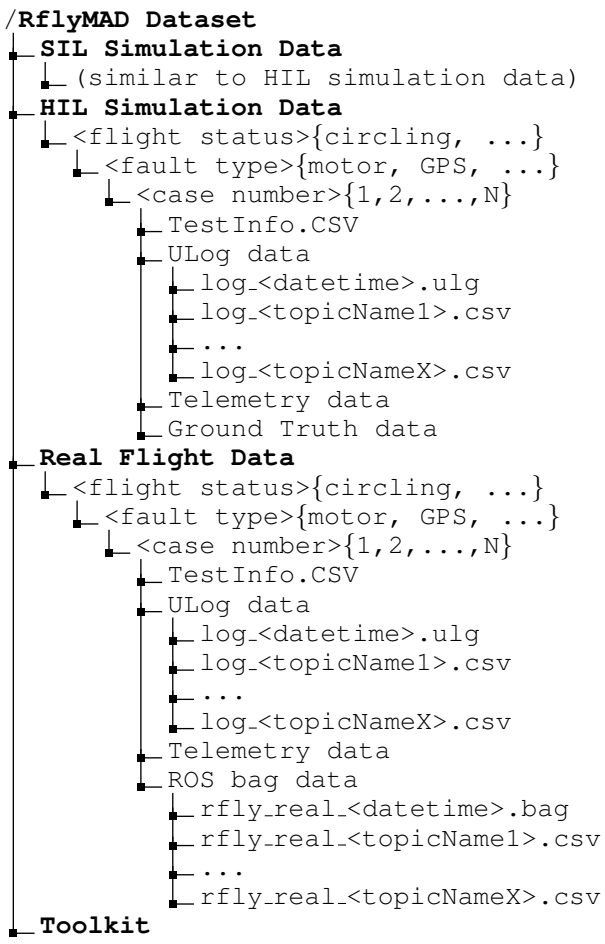


Figure 1. RflyMAD file hierarchy.

simulation phase, HIL simulation phase and real flight test phase. Through the automatic code generation technology of MATLAB/Simulink, the controller can be easily and automatically downloaded to the hardware for HIL simulation and real flight test. In addition, complex algorithms that are not suitable for deployment on autopilot can be easily immigrated to ROS on the onboard computer of UAVs. Therefore, in the latest version of RflySim, SIL and HIL simulation also support to simulate together with ROS.

When the simulation starts running, RflySim will start multiple programs. The RflySim3D program displays the UAVs and the surrounding environment, the QGC program is used as the ground station, and CopterSim is the core program of the platform, which can realize the transmission of internal platform messages and external mavlink, visual and other messages. The more detailed information about RflySim can be found in Dai et al. (2021) and RflySim platform website: https://rflysim.com/doc/en/.

As for using RflySim for fault data collection, some additional design is required. Firstly, from the perspective of facilitating the construction of the dataset, we construct control sequence for flight statuses and fault types of the UAVs. The control sequence includes the flight trajectory, speed, fault type, and fault parameters which represented the magnitude of the fault. Secondly, when constructing the multicopter dynamic model, a fault injection module that can receive fault parameters is added. This module can be used not only in simulation but also in real flight. Finally, an automatic testing Python script is construct, which can automatically complete the startup of the RflySim platform, the initialization of the multicopter, the reading and sending of flight statuses and fault types commands, and the recording and saving of the required data to construct the dataset. In SIL and HIL simualtion, the automatic testing script can repeatedly read control sequences from a set of sequences for testing, and complete the shutdown and restart of the RflySim platform, improving the efficiency of tests. In real flight experiments, this function is not added for safety consideration. And how the simulation and real flight data are collected exactly will be introduced in *Section 3*.

With the above introduction, the reason why we use RflySim as a fault injection experiment and data collection platform instead of using existing simulation platforms such as Gazebo and AirSim is that RflySim has the following advantages:

- (1) **Model-based development platform.** By using MBD methods, RflySim can achieve detailed modeling and physical parameters setting of the power system, sensors, and structures of the multicopter. However, the modification of physical parameters in Gazebo is complicated, and the number of parameters that can be modified in AirSim is quite limited.
- (2) **Fault injection module.** RflySim contains a complete fault injection module. The fault injection module is built into the dynamic model of the multicopter, and fault injection can be quickly realized through external commands. This method is suitable for both simulation and real flight. Other simulation platforms have not developed corresponding functional modules.

2.4.2 Data extensibility Based on the convenient simulation platform RflySim, if the current data available in RflyMAD is insufficient for the application or research needs, users can use this platform to design and conduct their own experiments and collect simulation data. Access to the RflySim platform [3] is freely available, and users are encouraged to further explore its capabilities. With this platform, users only need to write the control sequence in advance to arrange the flight statuses and fault types to be injected. The data will be collected automatically by programs during the simulation.

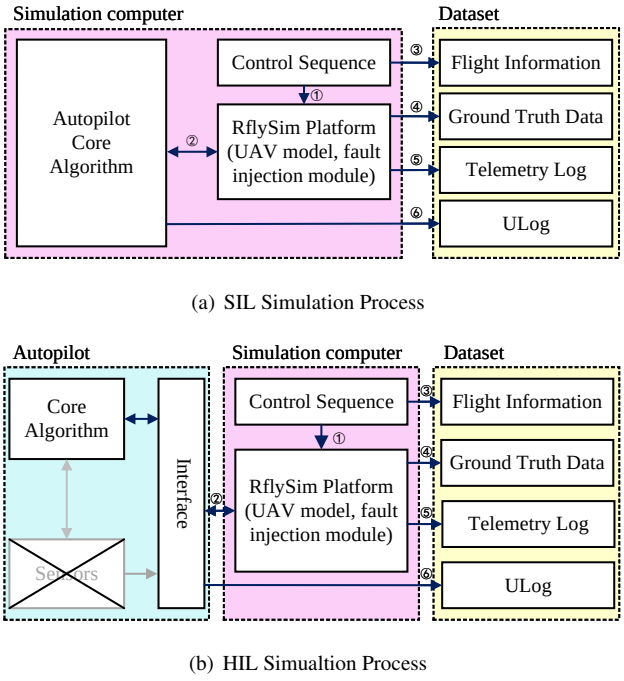
3 Dataset Construction

The RflyMAD dataset is divided into three parts, each with its own unique collection method. This section will provide a detailed introduction to these methods and then explain how to construct the dataset.

3.1 Simulation data

Figure 2(a) and Figure 2(b) demonstrate the collection of simulation data. The hardware required for SIL simulation data consists simply of a high-performance computer. Additionally, a PixHawk autopilot is required for HIL simulation (Quan et al. 2020; Dai et al. 2021; Wang et al. 2021).

In SIL simulation, RflySim serves as our core platform for data collection. A nonlinear multicopter dynamic model is built in RflySim, which has high accuracy compared to



(a) SIL Simulation Process

(b) HIL Simualtion Process

Figure 2. Methods to collect simulation data. Different colors represent different software or hardware equipment. The lines with arrows demonstrate the direction of signal transmission and also indicate the source of each data type in the dataset.

a real multicopter. The model includes an actuator module, battery module, GPS module, IMU module, and environment module that considers air drag and wind effects. The physical parameters of each module are obtained or fitted from real-world data. Besides, a fault injection module is designed to pass the fault parameters into the model. During the simulation, the fault parameters take effect by multiplying with throttle signals of motors or sensor signals, or by being added with them as noise or deviation values. The control sequence comprises a series of flight commands and fault information, which determines the flight statuses, fault types, fault parameters of the multicopter, and the start and end times of the fault. In the end, the control sequence will be recorded in the dataset as Flight Information. The core algorithm uses PX4 firmware to control the multicopter model of RflySim during simulation.

When the simulation begins, the RflySim platform will run the multicopter model and call the QGC program, and then execute the control sequence sequentially. After sending the fault injection commands, the fault parameters take effect and the corresponding module loses its function partially or completely. Simultaneously, the performance of the multicopter becomes abnormal, indicating the success of fault injection. Upon completion of the simulation, TLog, ULog, and GTData will be obtained from the RflySim platform and subsequently stored in the dataset.

In HIL simulation, in addition to the simulation computer, a PX4 autopilot is required to work together with the RflySim platform. Unlike SIL simulation, HIL simulation keeps core algorithms on autopilot and establishes communication with the RflySim platform through interfaces like MAVLink communication protocol. Since sensor modules are embedded in the multicopter model of the RflySim platform, the sensor signal from the PX4 autopilot needs to be shielded. Consequently, only the core algorithms of PX4 autopilot are used to control the model and generate ULog in autopilot's SD card. Other parts are similar to SIL simulation.

3.2 Real flight data

(a) Droneyee X200 (b) Droneyee X450 (c) Droneyee X680

Figure 3. Quadcopters used in real flight. They have different diagonal sizes: (a) 200mm (1.054kg), (b) 450mm (2.084kg) and (c) 680mm (4.068kg).

Due to the potential impact of the size and weight of multicopter on the diversity of the RflyMAD dataset and the performance of diagnosis methods, real flight data collection employs three types of quadcopters manufactured by Droneyee company. The quadcopters used in real flight can be found in Figure 3. All quadcopters have the same components and sensors, including a PX4 autopilot, a GPS module, an onboard computer, and a radio receiver.

Figure 4 presents one trajectory of the quadcopter in our experiment ground, located in Huailai County, Zhangjiakou City, Hebei Province, P. R. China. In our experiments, the quadcopter follows commands from the control sequence, executing different flight statuses when faults are injected.

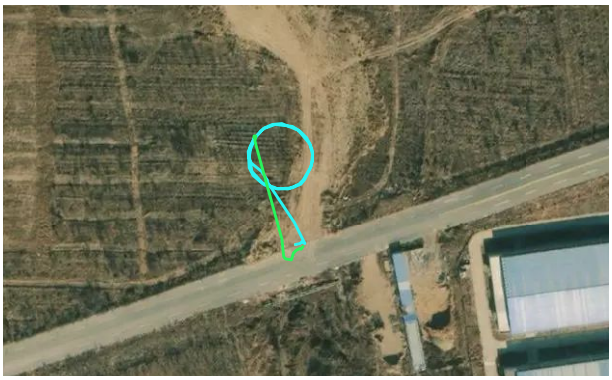


Figure 4. Trajectory of one flight in experiments and the location of data collection experiments.

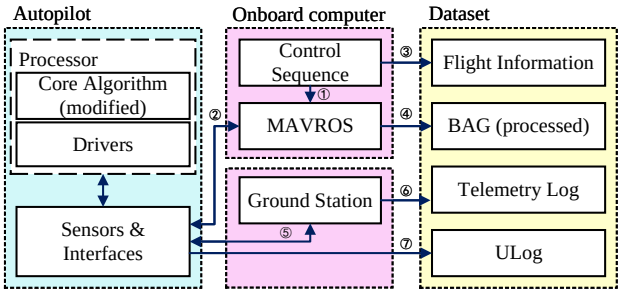


Figure 5. Methods to collect real flight data. Different from simulation, MAVROS is used to send commands and then generate BAG data.

Detailed information about flight statuses and fault types is introduced in *Section 2.2*.

Figure 5 illustrates the construction process of real flight data, which is quite different from that of simulation data. Firstly, the whole of the PX4 autopilot is used without shielding the sensor signals. In real flight, the fault injection module is redeveloped by modifying the core algorithm of PX4. The firmware version of PX4 used is 1.12.3. The modified methods can be summarized in three steps as follows:

- Create a *ctrl.msg* file for the uORB topic. This file includes timestamp, control flag, mode flag, and control signals.
- Subscribe to the *ctrl* signals in each module that need to be modified. Copy the fault parameters into each module.
- Update firmware signals with fault parameters. If the corresponding fault occurs, the fault parameters will be added or multiplied to the original signals.

Secondly, the onboard computer utilizes Robot Operating System (ROS) Noetic Ninjemys with Ubuntu 20.04 system to collect information on flight states, sensor signals, and input control commands through the MAVROS package, also known as "MAVLink for ROS". It could transform ROS topic messages into the format of MAVLink and send them to PX4 autopilot. Then the messages will be converted into the format of uORB and published in PX4. Using this method, the control sequence we have written could be sent into the core algorithm of PX4 and enable fault injection. Similar to simulation data, the control sequence will be collected as Flight Information in the real flight data.

The MAVROS can receive messages from the autopilot and generate a data file called ROS bag through ROS during each flight. This data is automatically collected and stored as a BAG file in the dataset. ULog and TLog data are collected separately from the autopilot and the QGC.

3.3 Fault injection method

As mentioned in *Section 3.1* and *Section 3.2*, during the collection process of simulation and real flight data, a similar fault injection method is used to transfer fault types and fault parameters to the simulation model or PX4 autopilot in real flight through the MAVLink interface. Then the fault parameters can be applied to the original signal to realize fault injection. The fault injection methods of motors and sensors will be introduced in detail. For a single motor of a multicopter, it can be modeled as

$$\begin{aligned} T &= c_T \varpi^2 \\ \varpi &= \frac{\varpi_{ss}}{T_m s + 1} \\ \varpi_{ss} &= f(k_m(t)\sigma) \end{aligned} \tag{1}$$

where T represents the pulling force generated by the actuator, which includes the motor and propeller. c_T is the pulling force coefficient of the propeller; ϖ is the speed of the motor, ϖ_{ss} is the steady state speed of the motor decided by the Electronic Speed Control (ESC) under a given throttle command σ, which is a PWM signal. And $k_m(t) \in [0, 1]$ is the fault parameter we injected, function $f(\cdot)$ means the mapping from the throttle command σ to the motor steady state speed ϖ_{ss}, which generally satisfies the form of a linear or quadratic function. In Equation (1), the dynamic process of the brushless motor in the simulation is simplified into a first order inertial element, and T_m represents the time constant of the dynamic response.

According to the common situations of motor faults, it can be divided into three types as follows:

- (a) Sudden Faults, also called Abrupt Faults, which usually occur instantaneously. After fault injection time t_f, the fault parameter $k_m(t)$ becomes a constant $C \in [0, 1)$.
- (b) Gradual Faults. These types of faults represent slow changes in system parameters and can be expressed as a gradual degradation of the motor over time.
- (c) Periodic Faults. These faults appear due to the partial failure in the system. In our method, these kinds of faults are defined as occurring in period T_0, so the set represents when faults occur periodically is $\mathcal{F} = \{t \in \mathbb{R} \mid t_f + nT_0 \le t \le t_f + (n+1)T_0, n = 0, 2, 4, \dots\}$

$$k_m(t) = \begin{cases} 1, & t \notin \mathcal{F} \\ C, & t \in \mathcal{F} \end{cases} . \tag{2}$$

As for sensor faults, taking an accelerometer as an example, the measured values can be expressed as

$$\begin{aligned} {}^{\mathrm{b}}\mathbf{a}_{imeas} &= {}^{\mathrm{b}}\mathbf{a} + {}^{\mathrm{b}}\boldsymbol{\omega} \times ({}^{\mathrm{b}}\boldsymbol{\omega} \times \mathbf{d}) + {}^{\mathrm{b}}\dot{\boldsymbol{\omega}} \times \mathbf{d} - g\mathbf{R}_{\mathrm{be}}^{\mathrm{T}}\mathbf{e}_3 \\ {}^{\mathrm{b}}\mathbf{a}_m &= \mathbf{K}_a \cdot {}^{\mathrm{b}}\mathbf{a}_{imeas} + \mathbf{b}_a + \mathbf{n}_a \\ \dot{\mathbf{b}_a} &= \mathbf{n}_{\mathbf{b}_a} \end{aligned} \tag{3}$$

where ${}^{\mathrm{b}}\mathbf{a}_{imeas}$ represents the ideal measured accelerations, ${}^{\mathrm{b}}\mathbf{a}_m$ are measured accelerations, ${}^{\mathrm{b}}\mathbf{a}$ are true values in body axes, ${}^{\mathrm{b}}\boldsymbol{\omega}$ are body-fixed angular rates, $\mathbf{d}$ is the distance from the accelerometer to the center of gravity of the multicopter, $\mathbf{R}_{\mathrm{be}}^{\mathrm{T}}$ is the rotation matrix from the body-fixed frame to the inertial frame; $\mathbf{K}_a$ is the scale factor, $\mathbf{b}_a$ represents the drift noise of the sensor, $\mathbf{n}_a$ and $\mathbf{n}_{\mathbf{b}_a}$ are Gaussian white noise.

It is obvious from Equation (3) that the scale factor $\mathbf{K}_a$ and noises such as $\mathbf{b}_a$ and $\mathbf{n}_a$ are usually the main reasons for the error between the measured value and true value of the sensor. Sensor fault injection is able to realize by adjusting the two items to the following form

$${}^{\mathrm{b}}\mathbf{a}_m = (\mathbf{k}_s(t) \cdot \mathbf{K}_a) {}^{\mathrm{b}}\mathbf{a}_{imeas} + \mathbf{b}_a + \mathbf{n}_a + \mathbf{k}_{sn}(t) \tag{4}$$

where $\mathbf{k}_s(t)$ is the fault parameter to change the scale factor and $\mathbf{k}_{sn}(t)$ is the fault parameter for sensor noises. The fault injection methods of other sensors, such as gyroscope, barometer, magnetometer, and GPS are similar to Equation (4) and are ignored for the limited space.

4 Data Validation

In this section, the data validation is carried out from two aspects: (1) the support relationship between simulation data and real flight data, and the support relationship between different types of multicopters in real flight data, which are verified by transfer learning method; (2) the validity of fault data, which is verified by a statistical anomaly detection method. All the experiments are conducted on a personal computer equipped with an Intel(R) Core(TM) i7-12700H CPU, NVIDIA GeForce RTX 3060 Laptop GPU, 16 GB memory, and Windows 11 64-bit system.

4.1 Why need data validation

Compared with the real flight data, the acquisition cost of simulation data is relatively low, and the simulation data and real flight data are expected to be consistent so that the real flight data can be replaced by the simulation data to some extent. Besides, as mentioned in *Section 3.2*, the different sizes and weights of the multicopter may have an impact on the performance of diagnosis methods. Since it is impossible for us to collect fault data of all types of multicopters, the real flight data of existing types of multicopters in the dataset are expected to have common characteristics, thus they can be substituted with each other to a certain extent. Validating this kind of support relationship presents a challenge. A transfer learning method (Azad et al. 2024; Hakim et al. 2022) is used to check the transfer ability of two kinds of data, so as to judge the equivalency between them.

In addition to the supporting relationships within the dataset, RflyMAD, as a fault dataset, should contain faults that can be diagnosed and have a certain degree of challenge in order to promote subsequent research. Therefore, a statistical anomaly detection method (Keipour et al. 2019) and related evaluation metrics proposed in Keipour et al. (2021) are used to test our RflyMAD dataset. While verifying the validity of the fault data in RflyMAD, it provides a benchmark for subsequent research.

4.2 Support relationship between simulation data and real flight data

In the application of fault diagnosis based on data-driven methods, transfer learning can be used to replace part of the real flight data with high-quality simulation data to complete the training of the model, thereby reducing the demand and dependence on real flight data.

4.2.1 Experimental data and preprocessing For experiments, ULog data from real flight and GTData from the simulation are used to diagnose single-motor fault during hovering flight. The fault state refers to the situation that the pulling force of a single motor falls from 100% to 85% at a specific moment. The diagonal sizes of the simulated aircraft and the actual aircraft are both 450mm, as shown in Figure 3(b).

By using the toolkits we developed, the ULog and GTData are converted into CSV format and ordered by timestamps. Although the different sensor data in the same flight case have different starting working times and sampling frequencies, it is necessary to synchronize the clocks of these sensors and unify the sensor data acquisition time accuracy to 10ms. Then interpolation processing is performed on the sensor sampling data whose accuracy is not satisfied. Finally, each sample contains timestamps and 12 characteristic information such as velocity, angular velocity, acceleration, and Euler angles in three dimensions.

Selecting 600 SIL data samples, 600 HIL data samples, 600 real flight data samples, and the normal and fault data account for 50% of the samples. In the experiment, 400 pieces are randomly sampled from the SIL simulation data, HIL simulation data, and the real flight data respectively, described as D_{SIL}, D_{HIL}, and D_R. 140 pieces of HIL simulation data and 140 pieces of real flight data are used as two different test sets, denoted as $D_{TestHIL}$ and D_{TestR}.

Table 3. Model parameters in transfer learning method.

No.	Layers	Output
1	Convolution1D	12×64
2	Batch Normalization	12×64
3	Convolution1D	12×32
4	Batch Normalization	12×32
5	Dense	1×64
6	Batch Normalization	1×64
7	Dense	1×32
8	Batch Normalization	1×32
9	Dense	1×2

Table 4. Average accuracy of 13 experiments with different support relationship and adaptive algorithm.

No.	The support relationship	Training set	Test set	Adaptive algorithm	Average accuracy
1	SIL and real data	10%D_R ~90%D_R	D_{TestR}	None	45.7%~100%
2		D_{SIL}		None	63%
3		D_{SIL}∪10%D_R		None	96%
4		D_{SIL}∪10%D_R		TrAdaBoost	97%
5		D_{SIL}∪10%D_R		AdaBN	98.2%
6	HIL and real data	D_{HIL}		None	55.9%
7		D_{HIL}∪10%D_R		None	95.8%
8		D_{HIL}∪10%D_R		TrAdaBoost	99.1%
9		D_{HIL}∪10%D_R		AdaBN	99.1%
10	SIL and HIL data	D_{SIL}	$D_{TestHIL}$	None	97%
11		D_{SIL}∪10%D_{HIL}		None	97.8%
12		D_{SIL}∪10%D_{HIL}		TrAdaBoost	97.8%
13		D_{SIL}∪10%D_{HIL}		AdaBN	97.8%

Note: The average accuracy in Experiment 1 is explained in Figure 6.

Accordingly, 60 pieces of real flight data and 60 pieces of HIL simulation data are selected as two different verification sets.

4.2.2 Model and domain adaptation algorithms The model used in the experiment is an improved model based on the classic LeNet-5 model (LeCun et al. 1998), and the model parameters are shown in Table 3. The classic TrAdaBoost (Dai et al. 2007) and AdaBN (Li et al. 2018) domain adaptation algorithms in transfer learning are used to verify the support relationship between simulation data and real flight data. By varying the number of samples from D_R in the training set and the domain adaptation algorithm, the results of different experiments are compared and analyzed.

Experiments are implemented using Google's TensorFlow toolbox[4]. To minimize the loss function, the Adam optimization algorithm is used to train the model. Epochs are set to 10 in different experiments, and the average prediction accuracy is calculated after multiple training. Assume that the total amount of data pieces in the test set is N_T, and the amount of pieces in the test set that can be predicted correctly is n, including True Positive (TP) and True Negative (TN) in confusion matrix, so the accuracy can be defined as

$$Acc = \frac{n}{N_T} \tag{5}$$

The experimental results are shown in Table 4 and Figure 6. The average accuracy in the Table 4 means the experiment is conducted 10 times for each test set in order to avoid accidental.

4.2.3 Interpretation of experimental results Based on the above experiments, the following conclusions could be drawn.

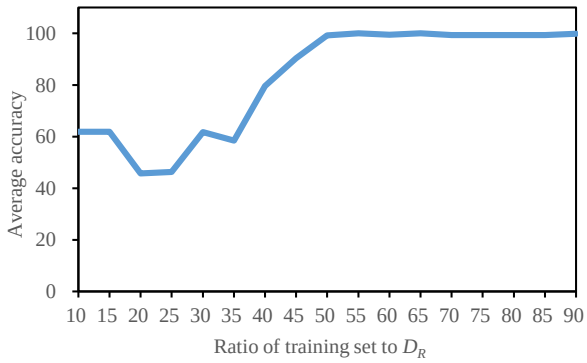


Figure 6. Results of average accuracy by using different ratios of D_R to form the training set and using D_{TestR} as the test set.

- Differences among D_{SIL}, D_{HIL}, and D_R can be obtained by using a single source of data as the training set for the transfer learning method. Using D_{TestR} as the test set, only a sufficient number of D_R can achieve higher accuracy. In *Experiment 1*, with the gradual increase of the data sample size of the D_R in the training set, the model prediction accuracy basically exhibits an increasing trend as shown in Figure 6. In *Experiments 2 and 6*, when there is no domain adaptation algorithm and dataset D_R, the model could not apply the feature recognition method learned by D_{SIL} or D_{HIL}, resulting in low prediction accuracy, which also indicated that the data distribution gap between the SIL or HIL simulation data (source domain) and the real flight data (target domain) is large.
- By adding real flight data to the source domain that is dominated by simulation data, the quality of the simulation data can be verified. The combination of a small amount of real flight data and simulation data can achieve better prediction results. As shown in Figure 6, using 10% D_R can only achieve an accuracy of about 60%, which is similar to the results of simulation data in *Experiments 2 and 6*. Through *Experiments 3 and 7*, it could be found that after adding target domain data to the training set, the gap between overall data distribution and target domain is reduced, thus improving the prediction accuracy under the same model. Therefore, it could be shown that the SIL and HIL simulation data quality is high.
- By adding domain adaptation algorithms, the transfer learning capabilities of the model can be further improved. In *Experiments 4, 5, 8, and 9*, the prediction accuracy of the model could reach the same level when 50% of the training is performed, which means the classical domain adaptation method combined with enough simulation data could approximately replace 40% of the real flight data.
- In addition to verifying the support relationship between simulation data and real flight data, it is verified that SIL simulation data could effectively support HIL simulation data in a similar way through *Experiments 10-13*. Even without HIL simulation data and domain adaptation algorithm, high detection accuracy could still be achieved.
- By using transfer learning methods, the average accuracy of *Experiments 1, 3-5 and 7-13* can reach a high level. This is because only the simple fault type of single motor fault during the hovering flight data in the dataset are used for training and testing, in order to illustrate that our data quality is high. When the fault types are mixed together, or more complex types of fault under other flight statuses are tested, such as a milder fault parameters or all motor fails to some extent, it will not be easy to achieve such a high accuracy. More experimental results under different fault states and flight statuses could be seen at the supporting documentation about transfer learning on RflyMAD dataset website [5].

Using the classic transfer learning method, the experimental verification of the support relationship between the simulation data and real flight data is carried out through the flight log file of the multicopter, and the support relationship between them is quantitatively analyzed, which provided a basis for the subsequent real flight data collection, simulation data quality evaluation, and fault diagnosis algorithm analysis.

4.3 Support relationship between different types of multicopters in real flight data

In this section, additional experiments are conducted to verify the transfer ability of the real flight data from one type of multicopter to new types. The experimental environment is consistent with the above experiments in *Section 4.2*.

After data preprocessing, the training set (source domain) data in the experiment comes from the real flight data of the multicopters with diagonal sizes of 450mm, described as D_{450R}, and the test set (target domain) data comes from the real flight data of the multicopters with diagonal sizes of 200mm and 680mm, denoted as $D_{Test200R}$ and $D_{Test680R}$ respectively. So these multicopters in the source and target domain are different. In the experiment, a single motor fault is taken as an example, the experiment is conducted 10 times for each test set, and the average prediction accuracy is calculated as shown in Table 5. The average accuracy defined here is similar to that in *Section 4.2.2*. In order to get closer to real conditions, the same fault type is used in the source domain and the target domain, but with different fault parameters which represent the degrees of failures. The fault parameters are described as follows:

- Source domain failure: single motor fails from 100% to 80% at a specific moment.
- Target domain failure: single motor fails from 100% to 90% at a specific moment.

Table 5. Average accuracy of transfer learning experiments with different types of multicopters.

No.	The support relationship	Training set	Test set	Adaptive algorithm	Average accuracy
1	Different types of multicopters	D_{450R}	$D_{Test200R}$	AdaBN	96.49%
2			$D_{Test680R}$		87.28%

It can be concluded from the above experiments that the real flight data from the multicopter with a diagonal size of 450mm can effectively predict the failures of two other different types of multicopters, with an average prediction accuracy of 91.88%, indicating that the data collected in the source domain includes the common fault characteristics in single motor failures of multicopters with different diagonal size and weight. The results also demonstrate that the use of

transfer learning or other similar methods can help extract common fault features, thereby predicting the same faults that do not exist in the training set. In conclusion, the experiments verify that the real flight data collected in RflyMAD is of high quality and as long as the common characteristics of faults can be extracted, the fault prediction methods can be applied to other multicopter types.

4.4 The validity of fault data in RflyMAD dataset

In this section, a statistical anomaly detection , which is a traditional fault detection method, is used to diagnose motor faults in RflyMAD dataset.

This method uses the Recursive Least Squares (RLS) to detect abnormal behaviors of UAVs, due to the characteristics of RLS, this method can be easily deployed on UAVs and realize real-time online fault detection. The core idea of this method is to use RLS to estimate the relationship between a pair of correlated input and output signals, such as the commanded signal and the actual measured signal of roll/pitch, then an estimated model is obtained. Next step is to use the estimated model to predict the outputs and compare them with the measured signals, thereby obtaining the error term. In the last step, while designing the criteria for anomaly detection, Welford's recursive method is used to calculate the variance of the error term, and finally the Z-Score is obtained to judge whether there is an anomaly.

In our manuscript, we used a similar method and input signals like those in article (Keipour et al. 2019) to conduct experiments. Therefore the commanded roll and pitch signals u_{roll} and u_{pitch}, and the measured roll and pitch signals y_{roll} and y_{pitch} are used as the input pairs. Specifically for RflyMAD dataset, signals used in experiments are obtained from processed ULog files, which are shown in the Table 6.

Table 6. List of signals used in experiments and where these signals can be obtained in processed ULog files.

Name	Source File in ULog	Usage
u_{roll}, u_{pitch}	actuator controls 0 0.csv	RLS method
y_{roll}, y_{pitch}	sensor combined 0.csv	RLS method
Fault state	rfly ctrl lxl 0.csv	Fault ground truth
Ground contact	vehicle land detected 0.csv	Get flight time

Similar to *Section 4.2*, ULog data are used to diagnose motor faults. But there are some differences, the data selected here contain various types of motor faults, such as one, two, three or four motor faults under different flight statuses. By using the toolkits we developed, the processed ULog data are unified into the same timestamp and adjusted to a uniform frequency of 20 Hz. In our experiments, we also used the inputs from the last second (20 samples) for the estimation of the new output. The experimental results are shown in the Table 7.

The accuracy defined here is quite different, for this RLS based fault diagnosed method requires a segment of data with a continuous time series for fault detection. Therefore, a whole flight data is regarded as a sample, and as long as the fault is detected at the time when the fault is injected, the sample is considered to be successfully diagnosed. If there is no fault in a flight and the diagnosis algorithm also indicates that no fault occurs throughout the flight, it is also considered to be a successful diagnosis.

Table 7. Test results for validity of fault data with different types of sub-dataset.

Type of Sub-dataset	# of tests	Flight Times(s)	Avg. Detection Times(s)	Max Detection Times(s)	Accuracy (%)
SIL Sim	14	467.25	0.13	0.3	85.71
HIL Sim	14	424.65	0.11	0.54	64.29
Real Flight	14	1492.35	0.84	1.25	64.29
Total	42	2384.25	0.28	1.25	71.43

Based on the experimental results and comparison with the experimental results in Keipour et al. (2019), the following conclusions could be drawn.

- By using statistical anomaly detection method, fault conditions in SIL and HIL simulation and real flight data can be successfully detected, which illustrates the validity of the fault data.
- The total accuracy of experiments in this manuscript is a little lower than that of the same method using ALFA dataset. This is because, on the one hand, the ALFA dataset contains faults such as 'Engine full power loss' or 'Elevator stuck at zero', while RflyMAD contains fault states where the motors are not completely failed. On the other hand, the roll and pitch channels may have more obvious characteristics when the fixed wing fails than the multicopters. So for multicopters, other input pairs might be a better choice. This result also reflects from another aspect that the RflyMAD dataset is still challenging.
- The fault diagnosis accuracy of SIL simulation data is higher than that of HIL simulation and real flight data. These results show that although the model in SIL simulation has ideal assumptions, SIL data is still valuable. This is because SIL data can still be used to learn the characteristics of faults, as introduced in transfer learning and the above experimental results. However, the accuracy of HIL simulation and real flight data is basically the same, which shows that HIL simulation data is closer to real flight data than SIL. An example is that when a single motor fails from 100% to 90% at a certain moment, the HIL and real flight data cannot be diagnosed, but the SIL data can still be successfully diagnosed. Although the fault cannot be diagnosed with the HIL simulation and real flight data, the fault indicator, or the Z-Score, in the method still has an obvious change when the fault happened. This also indicates the validity of fault data in RflyMAD dataset.

5 Discussion and Future Work

5.1 Discussion

To the best of our knowledge, there are few high-quality datasets with abnormality data for multicopters, or the high-quality datasets are not open-source for some reason. The RflyMAD dataset is an open-source dataset that contains both normal and abnormal data with the simulation and real flight for multicopters. The

introduction video of the RflyMAD dataset is available at https://youtu.be/MZby4mOPRu4.

Although RflyMAD includes numerous fault types and flight statuses, it is still insufficient. Firstly, flight cases for each fault type are inadequate in the real flight sub-dataset. We hope other researchers and enthusiasts would join us and contribute their data with different faults and hardwares by using our collection standards and dataset formats. Besides, the types of data should be diverse. For example, the RflyMAD dataset mainly consists of flight information, ULog, Telemetry log, and processed files from the RflySim platform or ROS system. The addition of more sensors, such as motor encoders, and capturing videos from multiple perspectives during the flight could expand the applicable fields of the RflyMAD dataset.

5.2 Future work

We hope that the RflyMAD dataset will become an important dataset for multicopter research. So we will continue supporting the development of RflyMAD, providing some basic codes to facilitate the use of the dataset and to design some model-based and data-driven methods as benchmarks for other researchers to compare and improve their algorithms. As mentioned above, we will add fault types and flight cases in simulation and real flight. The whole dataset and the succeeding data will be publicly available and can be accessed from our website. Besides, efficient toolkits will also be developed to improve the quality of the dataset.

Acknowledgements

Thanks to Droneyee company[6] for supporting equipment and UAVs in real flight experiments. Also, the authors would like to thank Yifan Liu, Haoyu Wei, Xinquan Chen, and Ziqiang Wen for their help during months of real flight on hot summer days.

Notes

1. NASA's Open Data Portal website: https://data.nasa.gov/
2. The QGroundControl website: http://qgroundcontrol.com/
3. RflySim platform website: https://rflysim.com/
4. TensorFlow toolbox: https://www.tensorflow.org/
5. The support document of RflyMAD dataset: https://rfly-openha.github.io/documents/4_resources/transfer_leanring.html
6. Droneyee Company website: http://www.feisilab.com/

References

Antonini A, Guerra W, Murali V, Sayre-McCord T and Karaman S (2020) The blackbird UAV dataset. *The International Journal of Robotics Research* 39(10-11): 1346–1364.

Azad MM, Kim S, Cheon YB and Kim HS (2024) Intelligent structural health monitoring of composite structures using machine learning, deep learning, and transfer learning: a review. *Advanced Composite Materials* 33(2): 162–188.

Baskett BJ and Daniel DLO (2000) Aeronautical design standard performance specification handling qualities requirements for military rotorcraft. *United States Army Aviation and Missile Command* : 25–48.

Dai WY, Yang Q, Xue GR and Yu Y (2007) Boosting for transfer learning. In: *Proceedings of the 24th International Conference on Machine Learning*. pp. 193–200.

Dai XH, Ke CX, Quan Q and Cai K-Y (2021) RflySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations. *Aerospace Science and Technology* 114: 106727.

Deng J, Dong W, Socher R, Li LJ, Li K and Li FF (2009) Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255.

Fourlas GK and Karras GC (2021) A survey on fault diagnosis and fault-tolerant control methods for unmanned aerial vehicles. *Machines* 9(9): 197.

Hakim M, Omran AAB, Ahmed AN, Al-Waily M and Abdellatif A (2022) A systematic review of rolling bearing fault diagnoses based on deep learning and transfer learning: Taxonomy, overview, application, open challenges, weaknesses and recommendations. *Ain Shams Engineering Journal* : 101945.

Keipour A, Mousaei M and Scherer S (2019) Automatic real-time anomaly detection for autonomous aerial vehicles. In: *2019 International Conference on Robotics and Automation (ICRA)*. pp. 5679–5685. DOI:10.1109/ICRA.2019.8794286.

Keipour A, Mousaei M and Scherer S (2021) ALFA: A dataset for UAV fault and anomaly detection. *The International Journal of Robotics Research* 40(2-3): 515–520.

LeCun Y, Bottou L, Bengio Y and Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Li Y, Wang N, Shi J, Hou X and Liu J (2018) Adaptive batch normalization for practical domain adaptation. *Pattern Recognition* 80: 109–117.

Lyu Y, Vosselman G, Xia GS, Yilmaz A and Yang MY (2020) UAVid: A semantic segmentation dataset for UAV imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* 165: 108–119.

Nguyen TM, Yuan S, Cao M, Lyu Y, Nguyen TH and Xie L (2022) NTU VIRAL: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *The International Journal of Robotics Research* 41(3): 270–280.

Quan Q (2017) *Introduction to Multicopter Design and Control*. Springer, Singapore.

Quan Q, Dai XH and Wang S (2020) *Multicopter Design and Control Practice: A Series Experiments based on MATLAB and Pixhawk*. Springer, Singapore.

Wang S, Dai XH, Ke CX and Quan Q (2021) RflySim: A rapid multicopter development platform for education and research based on Pixhawk and MATLAB. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 1587–1594.